Non-Convex Optimization: Matrix Sensing & Factored Model

Enayat Ullah[1]
Guide: Prof Raman Arora[2]

September 24, 2016

[1]Indian Institute of Technology Kanpur
[2]Johns Hopkins University

Table of Contents

Introduction

Convex Optimization

$$\min_{x \in \mathcal{C}} f(x)$$

$f : \mathbb{R}^d \to \mathbb{R}$ is a **convex** function

$\mathcal{C} \subset \mathbb{R}^d$ is a **convex** set

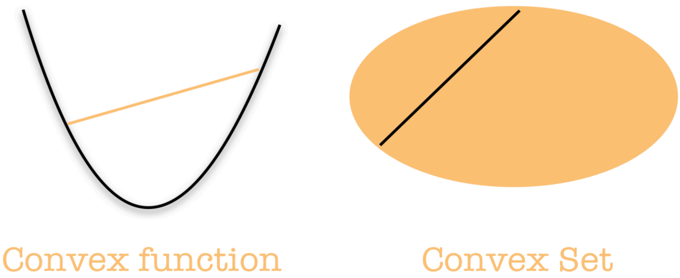

Figure 1: Convex function and Domain

Convex Optimization

Examples

- Linear Programming
- Quadratic Programming

Techniques

- Projected Subgradient Methods
- Interior Point Methods

Non-Convex Optimization

Non-Convex Optimization

Non-convex objective or Non-convex Domain (or both).

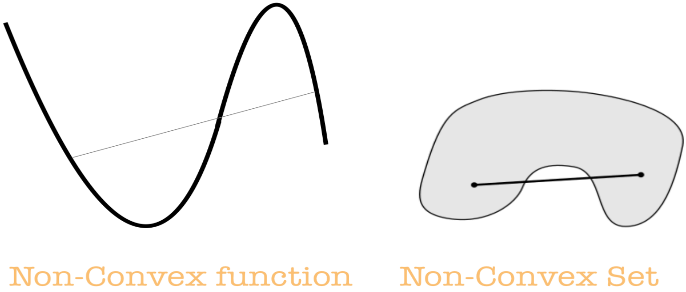

Figure 2: Non-convex function and Domain

Non-Convex Optimization

Challenges

- Exact minimization is NP-Hard.
- Proliferation of Saddle points

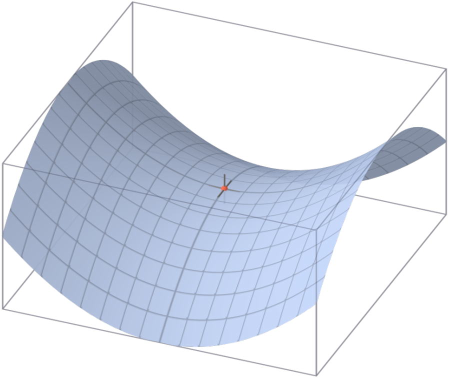

Figure 3: Saddle Point

Strict Saddle Property

Strict Saddle

A twice differentiable function $f(w)$ is strict saddle, if all its local minima have $\nabla^2 f(w) > 0$ and all its other stationary points satisfy $\lambda_{min}\left(\nabla^2 f(w)\right) < 0$.

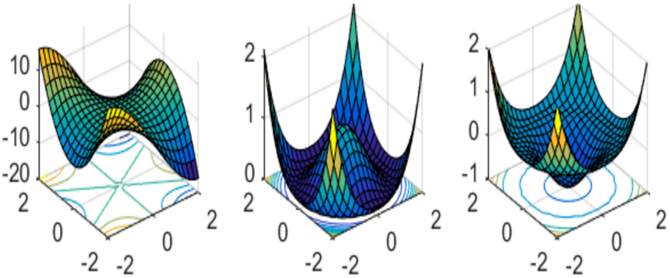

Figure 4: Non-strict-saddle Saddle Points

Source: **?**

Matrix Sensing

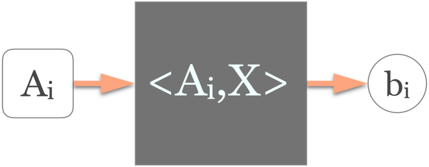

Figure 5: Matrix Sensing

Optimization problem

$$\min_{U,V} f(U,V) = \frac{1}{N}\sum_{i=1}^{N} \left(\langle \mathcal{A}_i, UV^T\rangle - \langle \mathcal{A}_i, X\rangle\right)$$

Expected Optimization problem

$$\min_{U,V} \mathop{\mathbb{E}}_{[\mathcal{A}_i]_{jk}\sim\mathcal{N}(0,1)} [f(U,V)] = \left\|UV^T - X\right\|_{\mathcal{F}}^2$$

Matrix Sensing

Gradient

$$\begin{bmatrix} \nabla_\alpha \\ \nabla_\beta \end{bmatrix} = \begin{bmatrix} 2\,\mathrm{vec}\left(\left(UV^T - X\right) V\right) \\ 2\,\mathrm{vec}\left(\left(UV^T - X\right)^T U\right) \end{bmatrix}$$

Hessian

$$\begin{bmatrix} \nabla^2_{\alpha\alpha} & \nabla^2_{\beta\alpha} \\ \nabla^2_{\alpha\beta} & \nabla^2_{\beta\beta} \end{bmatrix} = \begin{bmatrix} V^T V \otimes \mathbb{I}_m & (V^T \otimes U)\mathcal{K}(n,k) + (\mathbb{I}_k \otimes (UV^T - X)) \\ (U^T \otimes V)\mathcal{K}(m,k) + (\mathbb{I}_k \otimes (UV^T - X)^T) & U^T U \otimes \mathbb{I}_n \end{bmatrix}$$

Critical Points

Gradient = 0

- $U = 0, V = 0$

Strict Saddle

Critical Points

Gradient = 0

- $U = 0, V = 0$ **Strict Saddle**
- $UV^T = X$ **Global Minima**

Critical Points

Gradient = 0

- $U = 0, V = 0$ **Strict Saddle**
- $UV^T = X$ **Global Minima**
- $U = 0, V \neq 0, XV = 0$ **Strict Saddle**

Critical Points

Gradient = 0

- $U = 0, V = 0$ **Strict Saddle**
- $UV^T = X$ **Global Minima**
- $U = 0, V \neq 0, XV = 0$ **Strict Saddle**
- $V = 0, U \neq 0, X^T U = 0$ **Strict Saddle**

Critical Points

Gradient = 0

- $U = 0, V = 0$ **Strict Saddle**
- $UV^T = X$ **Global Minima**
- $U = 0, V \neq 0, XV = 0$ **Strict Saddle**
- $V = 0, U \neq 0, X^T U = 0$ **Strict Saddle**
- $U \neq 0, V \neq 0, UV^T \neq X, \left(UV^T - X\right) V = 0, \left(UV^T - X\right)^T U = 0$ **Not Possible**

Some Non-Convex Optimization Problems

Low Rank Matrix Regression

Data

$X_i \in \mathbb{R}^{m \times n}$

$y_i \in \mathbb{R}$

Model

$y_i = \langle X_i, W \rangle + \epsilon_i$ or

$y_i = \mathcal{B}\left(\sigma\left(\langle X_i, W \rangle\right)\right)$

$W = UV^T$

$U \in \mathbb{R}^{m \times k}$

$V \in \mathbb{R}^{m \times k}$

where $k \leq \min(m, n)$

Low rank Matrices (*Non-Convex Set*)

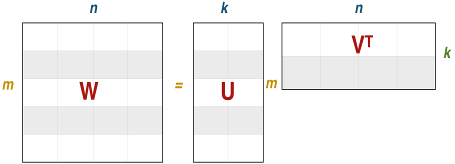

Figure 6: Low rank Matrix Decomposition

Low Rank Matrix Regression

Why Low rank?

- Low Intrinsic Dimensionality.
- eg: Factor Analysis, Collaborative Filtering.

Optimization Problem

$$\min_W f(W) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(W, X_i, y_i) \text{ s.t rank (W)} \leq k$$

Convex Relaxation

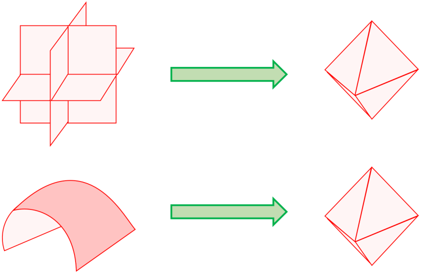

Figure 7: Convex Relaxation

Convex relaxation

$$\min_{W} f(W) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(W, X_i, y_i) + \lambda \left\| W \right\|_*^2$$

Convex Relaxation

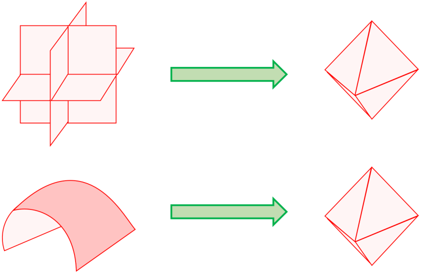

Figure 8: Convex Relaxation

Convex relaxation

$$\min_{W} f(W) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(W, X_i, y_i) + \lambda \|W\|_*^2$$ SLOW (SVD)

Non-Convex Reparametrization

Non-Convex Reparametrization

$$\min_{U} f(U, V) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(UV^T, X_i, y_i) + \frac{\lambda}{2}\left(\|U\|_{\mathcal{F}}^2 + \|V\|_{\mathcal{F}}^2\right)$$

Advantages

- Equivalent to nuclear norm minimization.
- Convex in one variable when the other is kept fixed.

Alternating Minimization

$$\min_{U,V} f(U,V)$$

$$\text{s.t U} \in \mathcal{C}_1, V \in \mathcal{C}_2$$

Algorithm 1 Alternating Minimization

1: Initialize U_0, V_0
2: **for** $t = 0$ to $T - 1$ **do**
3: $\quad U_t = \arg\min_{U \in \mathcal{C}_1} f(U, V_{t-1})$
4: $\quad V_t = \arg\min_{V \in \mathcal{C}_2} f(U_t, V)$
5: **end for**

Matrix Linear Regression

Model

$y_i = \langle W, X_i \rangle + \epsilon_i,$ where $W = UV^T$

Optimization problem

$$\min_{U,V} f(U,V) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left\langle UV^T, X_i \right\rangle\right)^2 + \frac{\lambda}{2}\left(\|U\|_{\mathcal{F}}^2 + \|V\|_{\mathcal{F}}^2\right)$$

`Alternate closed form solutions.`

Matrix Linear Regression - Algorithm

Alternating Exact Minimization!

Algorithm 2 Matrix Linear Regression

1: Initialize U, V
2: **for** $t = 1, 2, \ldots$ **do**
3: $\quad \mathrm{vec}\,(U) = \left(N\lambda \mathbb{I}_{mk} + \sum_{i=1}^{N} \mathrm{vec}\,(X_i)\,\mathrm{vec}\,(X_i V)^T \right)^{-1} \sum_{i=1}^{N} y_i\, \mathrm{vec}\,(X_i V)$
4: $\quad \mathrm{vec}\,(V) = \left(N\lambda \mathbb{I}_{nk} + \sum_{i=1}^{N} \mathrm{vec}\,\left(X_i^T U\right) \mathrm{vec}\,\left(X_i^T U\right)^T \right)^{-1} \sum_{i=1}^{N} y_i\, \mathrm{vec}\,\left(X_i^T U\right)$
5: **end for**

Matrix Logistic Regression

Model

$y_i = \mathcal{B}\left(\sigma(\langle W, X_i \rangle)\right)$, where $W = UV^T$

Optimization problem

$$\min_{U,V} f(U,V) = \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)\right) + \frac{\lambda}{2}\left(\|U\|_{\mathcal{F}}^2 + \|V\|_{\mathcal{F}}^2\right)$$

`No (alternate) closed form solutions.`

Matrix Logistic Regression (with Gradient Descent)

Alternating Minimization with gradient descent!

Algorithm 3 Matrix Logistic Regression with Gradient Descent

Initialize U, V
for $t = 1, 2, \dots$ **do**
$\quad \nabla_U = \lambda U - \frac{1}{N}\sum_{i=1}^{N} \frac{y_i \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)(X_i V)}{1 + \exp\left(y_i \left\langle UV^T, X_i \right\rangle\right)}$
$\quad \alpha_{opt} = backtrack(X, Y, U, V)$
$\quad U = U - \alpha_{opt} \nabla_U$
$\quad \nabla_V = \lambda V - \frac{1}{N}\sum_{i=1}^{N} \frac{y_i \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)(X_i U)}{1 + \exp\left(y_i \left\langle UV^T, X_i \right\rangle\right)}$
$\quad \alpha_{opt} = backtrack(X, Y, U, V)$
$\quad V = V - \alpha_{opt} \nabla_V$
end for

Newton's Method

Second-Order Taylor Approximation of f

$$f(U,V) = f(U_0,V_0) + \begin{bmatrix} \text{vec}(V-V_0)^T & \text{vec}(U-U_0)^T \end{bmatrix} \begin{bmatrix} \nabla_\alpha f(U_0,V_0) \\ \nabla_\beta f(U_0,V_0) \end{bmatrix} +$$

$$\begin{bmatrix} \text{vec}(V-V_0)^T & \text{vec}(U-U_0)^T \end{bmatrix} \begin{bmatrix} \nabla_{\alpha\alpha} f(U_0,V_0) & \nabla_{\beta\alpha} f(U_0,V_0) \\ \nabla_{\alpha\beta} f(U_0,V_0) & \nabla_{\beta\beta} f(U_0,V_0) \end{bmatrix} \begin{bmatrix} \text{vec}(U-U_0) \\ \text{vec}(V-V_0) \end{bmatrix}$$

Matrix Logistic Regression - Newton's Method

∇_α
∇_β

Gradient

$\nabla^2_{\alpha\alpha}$	$\nabla^2_{\beta\alpha}$
$\nabla^2_{\alpha\beta}$	$\nabla^2_{\beta\beta}$

Hessian

Updates

$$\mathrm{vec}(U) = \mathrm{vec}(U_0) - \left(\nabla^2_{\alpha\alpha}\right)^{-1} \left(\nabla^2_{\beta\alpha}\,\mathrm{vec}(V - V_0) + \nabla_\alpha\right)$$

$$\mathrm{vec}(V) = \mathrm{vec}(V_0) - \left(\nabla^2_{\beta\beta}\right)^{-1} \left(\nabla^2_{\alpha\beta}\,\mathrm{vec}(V - V_0) + \nabla_\beta\right)$$

$\alpha = \mathrm{vec}(U)$, $\beta = \mathrm{vec}(V)$

Matrix Logistic Regression - Newton's Method

Alternating Minimization with Newton Method!

Algorithm 3 Matrix Logistic Regression with Newton Method

Initialize U, V

for $t = 1, 2, \ldots$ **do**

$$\nabla_{\alpha} = \lambda \operatorname{vec}(U) - \frac{1}{N}\sum_{i=1}^{N} \frac{y_i \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right) \operatorname{vec}(X_i V)}{1 + \exp\left(y_i \left\langle UV^T, X_i \right\rangle\right)}$$

$$\nabla_{\beta} = \lambda \operatorname{vec}(V) - \frac{1}{N}\sum_{i=1}^{N} \frac{y_i \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right) \operatorname{vec}(X_i U)}{1 + \exp\left(y_i \left\langle UV^T, X_i \right\rangle\right)}$$

$$\nabla^2_{\alpha\alpha} = \lambda \mathbb{I}_{mk} + \frac{1}{N}\sum_{i=1}^{N} \frac{y_i^2 \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right) \operatorname{vec}(X_i V) \operatorname{vec}(X_i V)^T}{\left(1 + \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)^2\right)}$$

$$\nabla^2_{\alpha\beta} = \frac{1}{N}\sum_{i=1}^{N} \frac{y_i \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)}{\left(1 + \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)\right)} \left(\frac{y_i \operatorname{vec}\left(X_i^T U\right) \operatorname{vec}(X_i V)^T}{1 + \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)} - \left(\mathbb{I}_k \otimes X_i^T\right) \right)$$

$$\nabla^2_{\beta\alpha} = \frac{1}{N}\sum_{i=1}^{N} \frac{y_i \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)}{\left(1 + \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)\right)} \left(\frac{y_i \operatorname{vec}(X_i V) \operatorname{vec}\left(X_i^T U\right)^T}{1 + \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)} - \left(\mathbb{I}_k \otimes X_i\right) \right)$$

$$\nabla^2_{\beta\beta} = \lambda \mathbb{I}_{nk} + \frac{1}{N}\sum_{i=1}^{N} \frac{y_i^2 \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right) \operatorname{vec}\left(X_i^T U\right) \operatorname{vec}\left(X_i^T U\right)^T}{\left(1 + \exp\left(-y_i \left\langle UV^T, X_i \right\rangle\right)^2\right)}$$

$$\operatorname{vec}(U) = \operatorname{vec}(U_0) - \nabla^{-1}_{\alpha\alpha}\left(\nabla_{\beta\alpha} \operatorname{vec}(V - V_0) + \nabla_{\alpha}\right)$$

$$\operatorname{vec}(V) = \operatorname{vec}(V_0) - \nabla^{-1}_{\beta\beta}\left(\nabla_{\alpha\beta} \operatorname{vec}(V - V_0) + \nabla_{\beta}\right)$$

$U_0 = U$

$V_0 = V$

end for

Summary

- Convex vs Non-Convex Optimization.
- Convex Relaxation based methods.
- Non-convex Reparametrization.
- Alternating Newton method vs Alternate Gradient Descent.
- Implementations of Matrix Linear Regression, Matrix Logistic Regression and Robust PCA.

Acknowledgements

I thank Prof Raman Arora, Tuo Zhao, Xingguo Li, Poorya Mianjy, Anirbit Mukherjee and Bhuvesh Kumar for their continued support and guidance throughtout the project.

Questions?