Convex Optimization and Applications

Instructors:	Anh-Huy Phan and Andrzej Cichocki
Teaching Assistants:	Salman Ahmadi Asl, Valentin Leplat
	Dmitry Ermilov, Sergei Gostilovich, Airat Kotliar-Shapirov
	and Anastasia Sozykina

31st October, 2023

Optimization in a Nutshell

- **Optimization** is concerned with finding the best solution of a given problem among a set of candidate solutions
- Need ingredients
  - A measure of the goodness of a candidate solution:
    Usually measured by a real-valued function (Objective Function)
  - A description of the set of candidate solutions:
    Usually expressed by some functional relations (Constraints)

Optimization Problem

- General optimization problem in standard form:

$$\begin{array}{ll} \text{minimize}_x & f_0(x) \\ \text{subject to} & f_i(x) \leq 0 \quad i = 1, 2, \ldots, m \\ & h_i(x) = 0 \quad i = 1, 2, \ldots, p \end{array}$$

  where
  $x = (x_1, \ldots, x_n)$ is the optimization variable
  $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the objective function
  $f_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, \ldots, m$ are inequality constraint functions
  $h_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, \ldots, p$ are equality constraint functions.
- **Goal:** find an optimal solution $x^\star$ that minimizes f_0 while satisfying all the constraints.

History

- Newton-Raphson algorithm (1669) for finding root $f(x) = 0$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

  x_0 is the initial guess for the root of $f(x)$, $f'(x_k)$: the function's derivative
  x_{k+1} gives a better approximation of the root than x_k.

- In **Optimization**: local minima and maxima of $f(x)$ can be found by applying Newton's method to the derivative, i.e., equating the gradient to zero

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

  ($f(x)$ is twice differentiable, and its second derivative is positive at the minimum or negative at the maximum)

Optimization: Heart in Machine learning and Signal Processing

- Optimization is important in modern applications, heart of most machine learning algorithms
  - Signal processing: signal filtering, tracking, estimation, reconstruction
  - Image processing: deblurring, compressive sensing, blind separation
  - Finance: portflio design, index tracking
  - Energy
  - Digital economy: search engine, recommender systems
  - Deep neural networks and AI
  - Electronic: circuit design
  - Communication systems (e.g., transceiver design, beamforming design, power control in wireless)
  - System control
- **Bad news**: cannot solve all optimization problems

Gradients I

Gradient for a function $f(x)$ of n variables evaluated at the point $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ is

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^T$$

where $\frac{\partial f}{\partial x_i}$ is the partial derivative of f w.r.t x_i, i.e., the gradient combines the rates of change of f along all directions.
Directional derivative of f at a point $\boldsymbol{x}$ in the direction of a (unit) vector $\boldsymbol{v}$ is the rate of change of f at x along $\boldsymbol{v}$.

$$D_{\boldsymbol{v}} f(\boldsymbol{x}) = \lim_{h \to 0} \frac{f(\boldsymbol{x} + h\boldsymbol{v}) - f(\boldsymbol{x})}{h}.$$

Directional derivative can be computed as

$$D_v f(x) = \nabla f(x) \cdot v = \|\nabla f(\boldsymbol{x})\| \, \|\boldsymbol{v}\| \cos\theta = \|\nabla f(\boldsymbol{x})\| \cos\theta$$

Gradients II

where θ is the angle between the gradient and $\boldsymbol{v}$.
Maximum value of the directional derivative occurs when $\theta = 0$, i.e., $\boldsymbol{v}$ points in the same direction as the gradient,
Minimum value of the directional derivative occurs when $\theta = \pi$ or that $\boldsymbol{v}$ points in the opposite direction of the gradient

the function decreases the most along the direction opposite to the gradient, and its rate of change is equal to the negative magnitude of the gradient.

Therefore, the gradient can be used to find the maximum or minimum value of a function, i.e.,

$$\nabla f(x) = 0$$

This condition is necessary but not sufficient for an extremum.
To determine whether an extremum is a maximum or a minimum, we need to look at the second derivatives of f

Optimization in Signal Processing: Denoising I

- A noisy signal, y, is the observation of the true signal x corrupted by some noise n. The noise can be additive, multiplicative, or nonlinear, depending on the nature of the signal and the measurement process.
- A fidelity measure $F(x, y)$, quantifies how close the signal x is to the noisy observation y, e.g., the mean squared error, the Kullback-Leibler divergence.
- A smoothness measure $S(x)$, which is a function that quantifies how smooth the signal x is, e.g., based on the total variation, the gradient, the wavelet coefficients of x.

$$TV(x) = \sum_{i=1}^{N-1} |x_{i+1} - x_i|$$

Optimization in Signal Processing: Denoising II

The optimization problem can then be written as:

$$\min_x \quad F(x, y) + \lambda S(x)$$

subject to some optional constraints on x, such as bounds, sparsity, or non-negativity where the regularization parameter $\lambda > 0$ balances the trade-off between the smoothness and the fidelity of the signal.

Optimization in ML I

- Dimensionality reduction

$$\min_{\mathbf{U},\mathbf{V}} \quad \|\mathbf{Y} - \mathbf{U}\mathbf{V}^T\|_F^2$$

  where $\mathbf{Y}$ is a data matrix of size $I \times J$, approximated by $\mathbf{X} = \mathbf{U}\mathbf{V}^T$, U of size $I \times R$ and $\mathbf{V}$ of size $J \times R$ where $R \ll (I, J)$.
  The decomposition is not unique. Additional constraints can be imposed to $\mathbf{U}$ and/or $\mathbf{V}$.

- Dimensionality reduction with minimal rank

$$\begin{aligned} \min_{\mathbf{X}} \quad & \mathrm{rank}(\mathbf{X}) \\ \text{subject to} \quad & \|\mathbf{Y} - \mathbf{X}\|_F^2 \leq \delta \end{aligned}$$

Optimization in ML II

- Linear regression

$$\min_{\boldsymbol{w}} \quad \|\boldsymbol{y} - \mathbf{X}\boldsymbol{w}\|_2^2$$

- Logistic regression
  For an i-th observation $\boldsymbol{x}_i$, the label y_i takes either 0 or 1.
  The probability of a binary outcome y_i for a given $\boldsymbol{x}_i$

$$p(y_i|\boldsymbol{x}_i) = \frac{1}{1 + e^{-\boldsymbol{x}_i^T \boldsymbol{w}}}$$

  where $\boldsymbol{w}$ is the weight vector.
  Find $\boldsymbol{w}$ that maximizes the likelihood function, the product of the probabilities of observations, $\boldsymbol{x}_i$

Optimization in ML III

$$L(\boldsymbol{w}) = \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i)^{y_i}(1 - p(y_i|\boldsymbol{x}_i))^{1-y_i}$$

The log-likelihood function

$$\begin{aligned}\log L(\boldsymbol{w}) &= -\sum_{i=1}^{n} y_i \log p(y_i|\boldsymbol{x}_i) + (1 - y_i)\log(1 - p(y_i|\boldsymbol{x}_i)) \\ &= \sum_{i=1}^{n} y_i(\boldsymbol{x}_i^T \boldsymbol{w}) - \log(1 + e^{\boldsymbol{x}_i^T \boldsymbol{w}})\end{aligned}$$

Optimization in ML IV

- Support Vector Machine (SVM)

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & \xi_i \geq 1 - y_i \boldsymbol{x}_i^T \boldsymbol{w}, \xi_i \geq 0. \end{aligned}$$

- Collaborative filtering (used in recommender systems):

$$\min_{\boldsymbol{w}} \quad \sum_{i<j} \log(1 + \exp(\boldsymbol{x}_i^T \boldsymbol{w} - \boldsymbol{x}_j^T \boldsymbol{w}))$$

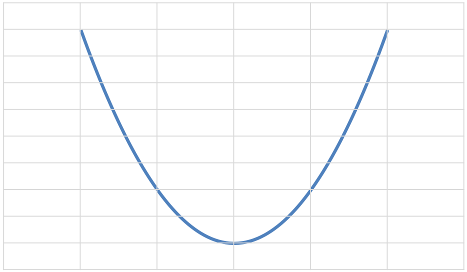
Optimization in ML V

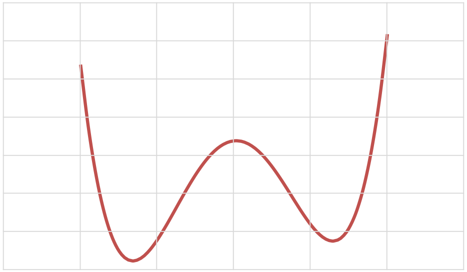
- K-means clustering: partition samples (points) into k clusters $C_1, \ldots, C_k$ whose centroids are $\boldsymbol{\mu}_j$

$$\min_{C_1,\ldots,C_k} \quad \sum_{j=1}^{k} \frac{1}{|C_j|} \sum_{i,l \in C_j} \|\boldsymbol{x}_i - \boldsymbol{x}_l\|_2^2$$

$$\min_{\substack{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k \\ C_1,\ldots,C_k}} \quad \sum_{j=1}^{k} \sum_{i \in C_j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|_2^2$$

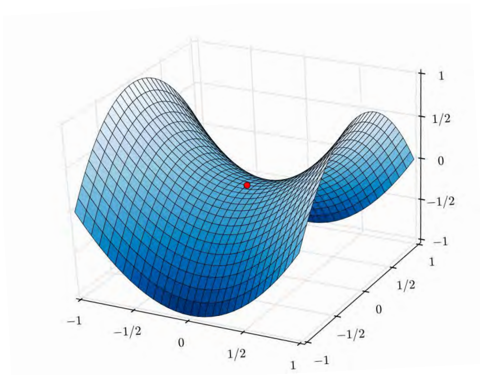
Convex and Non-Convex Optimization





What makes non-convex optimization hard?

- Potentially many local minima
    - Saddle points (a stationary point but not a local extremum)
    - Very flat regions
    - Widely varying curvature (2nd derivative)
- Can be extremely difficult
- Initial guess can be critical and can greatly affect the objective value of the local solution obtained.
- Often sensitive to algorithm parameter values, which may need to be adjusted

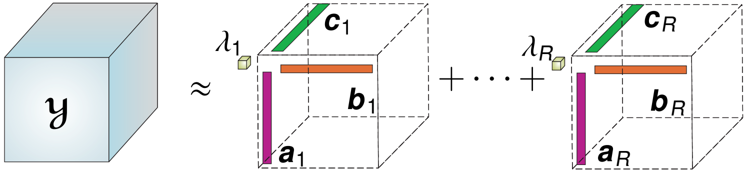


- Optimal solution cannot be found in reasonable time or with reasonable processing power

Example: Minimize a function of 10 variables over the unit box can take more than 30 million years to find an approximate solution!

Examples of Non-convex Problems I

- Low-rank models and tensor decomposition



$$\min_{\substack{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_R\\ \boldsymbol{b}_1,\ldots,\boldsymbol{b}_r\\ \boldsymbol{c}_1,\ldots,\boldsymbol{c}_R}} \quad \|\mathcal{Y} - \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r\|_F^2$$

Examples of Non-convex Problems II

- **Maximum likelihood estimation** with hidden variables usually non-convex
  Estimation of mean and covariance of random variables which follow the tensor Gaussian distribution
  - Consider real-valued tensors, $\mathcal{X}_t$, of size $I_1 \times I_2 \times \cdots \times I_N$, $t = 1, \ldots T$, drawn from a separable Gaussian distribution with
    - mean, $\boldsymbol{\mu}_n \in \mathbb{R}^{I_n}$ and covariance, $\boldsymbol{\Theta}_n \in \mathbf{S}_{I_n}$ for each mode-n for $n = 1, \ldots, N$
  - Distribution of vectorization $\boldsymbol{x}_t = \mathrm{vec}(\mathcal{X}_t) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Theta})$ is known to be normal where mean $\boldsymbol{\mu} = \boldsymbol{\mu}_N \otimes \cdots \otimes \boldsymbol{\mu}_2 \otimes \boldsymbol{\mu}_1$ and covariance matrix $\boldsymbol{\Theta} = \boldsymbol{\Theta}_N \otimes \cdots \otimes \boldsymbol{\Theta}_2 \otimes \boldsymbol{\Theta}_1$, and the probability density function given by

$$p(\mathcal{X}_t) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\mu})^T \left(\bigotimes_{n=N}^{1} \boldsymbol{\Theta}_n\right)^{-1} (\boldsymbol{x}_t - \boldsymbol{\mu})\right)}{(2\pi)^{\frac{K}{2}} \det^{\frac{1}{2}} \left(\bigotimes_{n=N}^{1} \boldsymbol{\Theta}^{(n)}\right)}$$

    where $K = I_1 I_2 \cdots I_N$ is the total number of elements of the tensor $\mathcal{X}_t$.

Examples of Non-convex Problems III

- Estimation of the mean $\boldsymbol{\mu}_n$ and covariance matrices $\boldsymbol{\Theta}_n$ can be achieved by maximizing the log-likelihood of T samples, $\mathcal{X}(t)$,

$$\mathcal{L}(\{\boldsymbol{\mu}_n\}_{n=1}^N, \{\boldsymbol{\Theta}_n\}_{n=1}^N) = \sum_{t=1}^{T} \ln p\left(\mathcal{X}(t) \middle| \{\boldsymbol{\mu}_n\}_{n=1}^N, \{\boldsymbol{\Theta}_n\}_{n=1}^N\right)$$

$$= -\frac{TK}{2}\ln(2\pi) - \frac{T}{2}\ln\det\left(\bigotimes_{n=N}^{1}\boldsymbol{\Theta}^{(n)}\right) - \frac{1}{2}\sum_{t=1}^{T}(\boldsymbol{x}_t - \boldsymbol{\mu})^T\left(\bigotimes_{n=N}^{1}\boldsymbol{\Theta}_n\right)^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}) \ .$$

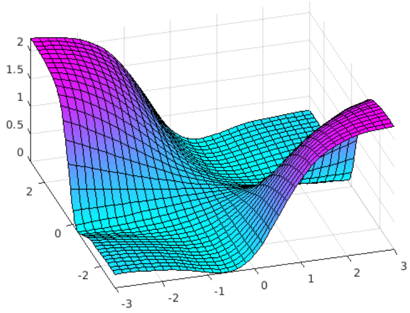
Deep neural networks: Non-convex loss I

- **Composition of convex functions is not convex**
- Example

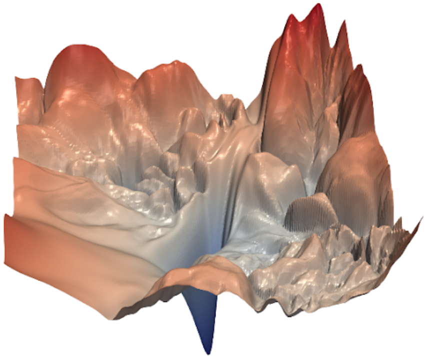
Consider the multi-layer perceptron

$$\hat{y} = \tanh(w_1 \tanh(w_2 x))$$

x is scalar input, $\hat{y}$ is scalar output
w_1 and w_2 are two parameters
$\tanh()$ is the hyperbolic tangent function
The loss function minimizes the error
$f(w_1, w_2) = |y - \hat{y}|^2$



Deep neural networks: Non-convex loss II

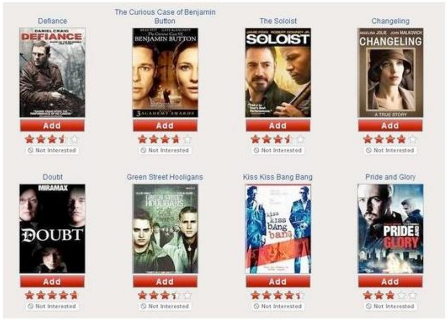


Loss surfaces of ResNet-56 without skip (H.Li, Z.Xu, G.Taylor, C.Studer and T. Goldstein).

Deep networks have more non-convexity and irregularity in their loss surfaces, more hills and valleys in the loss surface
which makes finding a good minimizer of the loss function harder and slower.

Examples of Non-convex Problems: The Netflix Prize

- Collaborative filtering: predict un-rated video based on incomplete "movie-user"
  - 48K raters and 18K movies
  - Around 98.7% missing ratings
    Some users had only reviewed 9 movies, gave the contestants very little information
    Some had rated over 900 films.
    "Batman Begins" have hundreds of thousands of star-ratings, but some obscure movies only had 20 ratings.
  - $1,000,000 cash prize



Why Convexity Matters

- *"...in fact, the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity."*
  - R. Tyrrell Rockafellar, in SIAM Review, 1993
- **Global optimal solutions** can be achieved
- **Efficient algorithms** to common convex optimization problems
- **Duality:** solve a dual maximization problem with the same optimal value
  Optimality Certificate: check that a solution to an optimization problem is correct
- **Typical optimization:** Linear optimization, quadratic optimization, semidefinite optimization, geometric optimization, sum-of-norm optimization, etc.

Least Squares I

$$\min_{\boldsymbol{x}} \quad \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$$

- Analytical solution $\boldsymbol{x}^\star = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\boldsymbol{b}$
- Reliable and efficient algorithms and software
- Computation time proportional to $n^2 p$ for $\mathbf{A} \in \mathbb{R}^{p \times n}$
- Widely used e.g, in linear regression, tensor decomposition

Using least-squares

- Least-squares problems are easy to recognize
- Can be combined with various constraints (weights, regularization, . . .)

Least Squares II

Examples

- $$\min_{\boldsymbol{x}} \quad \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \|\mathbf{L}\boldsymbol{x}\|_2^2$$
- $$\min_{\boldsymbol{x}} \quad \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad \text{s.t.} \quad \mathbf{U}^T\boldsymbol{x} = 0$$
- $$\min_{\boldsymbol{x}} \quad \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad \text{s.t.} \quad \boldsymbol{x} \geq 0$$
- $$\min_{\boldsymbol{x}} \quad \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_2^2 = 1$$

Linear Programming I

$$\begin{array}{ll} \min_{\boldsymbol{x}} & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} & \boldsymbol{a}_i^T \boldsymbol{x} \leq \boldsymbol{b}_i, \quad i = 1, \ldots, m \end{array}$$

- no analytical formula for solution; extensive theory
- reliable and efficient algorithms and software
- computation time proportional to $n^2 m$ if $m \geq n$; less with structure
- a widely used technology (1947 G. B. Dantzig)

Using linear programming

- not as easy to recognize as least-squares problems
- a few standard tricks used to convert problems into linear programs (e.g., problems involving ℓ_1- or ℓ_∞-norms, piecewise-linear functions)

Optimal Transport I

The problem of finding the most efficient way of moving a certain amount of mass from one location to another, subject to some constraints.

- Suppose there are three factories that produce goods and four markets that demand goods.
- Each factory has a certain supply of goods, and each market has a certain demand for goods.
- The cost of transporting one unit of goods from factory i to market j is given by a matrix C. The problem is to find the optimal allocation of goods from factories to markets that minimizes the total transportation cost, while satisfying the supply and demand constraints.

Optimal Transport II

Let x_{ij} the amount of goods transported from factory i to market j, for $i = 1, 2, 3$ and $j = 1, 2, 3, 4$.
The objective function is to minimize the total transportation cost, which is given by the sum of the products of x_{ij} and C_{ij}

$$\min \sum_{i=1}^{3} \sum_{j=1}^{4} C_{ij}\, x_{ij}$$

Constraints:

- **Supply constraints**: for each factory i, the total amount of goods transported from factory i must be equal to the supply of factory i

$$\sum_{j=1}^{4} x_{ij} = s_i \quad \text{for } i = 1, 2, 3$$

Optimal Transport III

- **Demand constraints** for each market j, the total amount of goods transported to market j must be equal to the demand of market j,

$$\sum_{i=1}^{3} x_{ij} = d_j \quad \text{for } j = 1, 2, 3, 4$$

- **Nonnegativity constraints** for each pair (i, j), the amount of goods transported from factory i to market j must be nonnegative

$$x_{ij} \geq 0 \quad \text{for } i = 1, 2, 3 \text{ and } j = 1, 2, 3, 4$$

Optimal Transport IV

Optimal transport problem for two distributions for two variables x and y can be formulated as follows:
Given two probability measures $p(x)$ and $q(y)$ over the same metric space S, we want to find a joint distribution (x, y) that minimizes the expected cost of transporting x to y, where the cost is given by a function $c(x, y)$ that measures the distance between x and y. That is,

$$W_c(p, q) = \inf_{\gamma \in \Gamma(p,q)} \int_{S \times S} c(x, y) d\gamma(x, y)$$

where $W_c(p, q)$ is the optimal transport distance between p and q, and $\Gamma(p, q)$ is the set of all joint distributions with marginals p and q.
We can preserve the mass of p and q by requiring marginal constraints

$$p(x) = \int_S \gamma(x, y) dy$$

Optimal Transport V

$$q(y) = \int_S \gamma(x, y) dx$$

$p(x)$ and $q(y)$ are known probability measures
The joint distribution (x, y) represents the optimal transport plan that minimizes the expected cost of transporting x to y.

Quadratic Programming I

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{x}^T \mathbf{Q} \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \mathbf{A}^T \boldsymbol{x} \leq \boldsymbol{b} \end{aligned}$$

- Generalization of both least-squares and linear programming problems
- Popular in many areas
- Markowitz (student of Dantzig) introduced in 50’s, and won the Nobel prize in Enonomics in 1990 mainly for this work

Convex optimization problem I

$$\begin{aligned} \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0 \quad i = 1, \ldots, p \end{aligned}$$

$h_i(x)$ is affine (composition of a linear function and a constant)

- Objective and constraint functions are convex

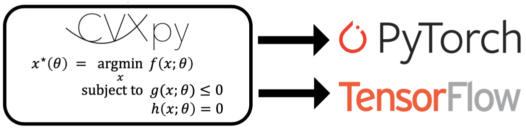
$$f_i(\theta x + (1-\theta)y) \leq \theta f_i(x) + (1-\theta) f_i(y)$$

  for all x, y, $0 \leq \theta \leq 1$

- Least-squares, linear and quadratic programming are special cases

CVX Software I

- CVX solvers: CVX for Matlab and CVXPY Python
- Cvxpylayers: Differentiable Convex Optimization Layers
  formulate layers in NN as convex optimization problems
- Other solvers: Gurobi, CPLEX, MOSEK (not free)
  Matlab (license for Skoltech students)


CVXpy
$x^\star(\theta) = \underset{x}{\mathrm{argmin}}\ f(x;\theta)$
subject to $g(x;\theta) \le 0$
$h(x;\theta) = 0$
PyTorch
TensorFlow

CVX Software II

Examples

$\min \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \quad s.t. \quad l \leq \boldsymbol{x} \leq u$

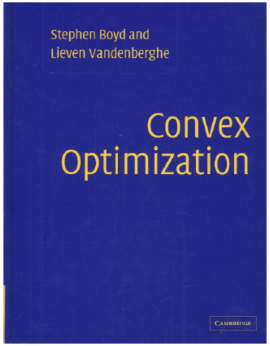
```
cvx_begin
   variable x(n)
   minimize (norm(A*x-b))
   subject to  l ≤ x ≤ u;
cvx_end
```

$\min \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_1$

```
cvx_begin
   variable x(n)
   minimize(norm(A*x-b,1))
cvx_end
```

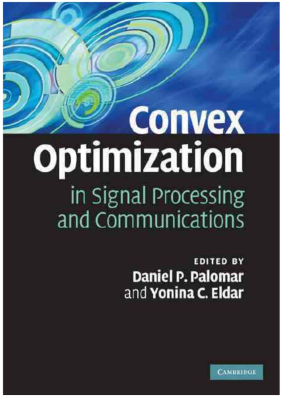
```
import cvxpy as cp
import numpy as np

x = cp.Variable(n)
cost = cp.sum_squares(A @ x - b)
constraints = [l ≤ x, x ≤ u]
prob = cp.Problem(cp.Minimize(cost),constraints)
prob.solve()
```

Books


Stephen Boyd and
Lieven Vandenberghe

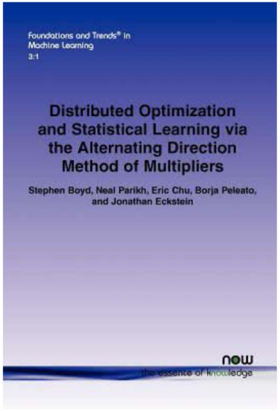
Convex
Optimization

CAMBRIDGE


Convex
Optimization
in Signal Processing
and Communications

EDITED BY
Daniel P. Palomar
and Yonina C. Eldar

CAMBRIDGE


Foundations and Trends® in
Machine Learning
3:1

Distributed Optimization
and Statistical Learning via
the Alternating Direction
Method of Multipliers

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato,
and Jonathan Eckstein

now
the essence of knowledge

Low-rank Approximation I

Problem

Find the best representation of a data (matrix) by another matrix of low-rank

$$\mathbf{Y} \approx \mathbf{U}\mathbf{V}^T$$

where $\mathbf{Y}$ is (often) big, incomplete (sparse).

Theorem (Eckart-Young-Mirsky)

The low-rank matrix approximation problem

$$\min_{\mathbf{X}} \quad \|\mathbf{Y} - \mathbf{X}\|_F^2 \quad \textit{s.t.} \quad \textit{rank}(\mathbf{X}) \leq R$$

has analytical solution $\mathbf{X} = \mathbf{U}\,\mathrm{diag}(\sigma_1, \ldots, \sigma_R)\mathbf{V}^T$, which is rank-$R$ truncated singular value decomposition of the data matrix $\mathbf{Y}$.

Low-rank Approximation II

Theorem

The low-rank matrix approximation problem

$$\min_{\mathbf{X}} \quad \textit{rank}(\mathbf{X}) \quad \textit{s.t.} \quad \|\mathbf{Y} - \mathbf{X}\|_F^2 \leq \delta$$

has analytical solution, $\mathbf{X} = \mathbf{U}\,\mathrm{diag}(\sigma_1, \ldots, \sigma_R)\mathbf{V}^T$*, which is rank-*$R$ *truncated singular value decomposition of the data matrix* $\mathbf{Y}$ *such that*

$$\sum_{r=1}^{R-1} \sigma_r^2 < \|\mathbf{Y}\|_F^2 - \delta \leq \sum_{r=1}^{R} \sigma_r^2$$

The Netflix Prize: The problem of Matrix Completion I

Problem

How to find a good low-rank approximation of an incomplete matrix $\mathbf{Y}$?

- Define a binary matrix $\mathbf{W}$ whose elements are 0s for missing entries of $\mathbf{Y}$, and 1s elsewhere.
- Predicting un-rated video is similar to recovering missing entries of $\mathbf{Y}$
  This can be achieved by finding an estimate of $\mathbf{Y}$ with minimal rank, i.e.,

$$\min_{\mathbf{X}} \quad \text{rank}(\mathbf{X}) \qquad \text{s.t.} \quad \mathbf{W} \circledast \mathbf{Y} = \mathbf{W} \circledast \mathbf{X}$$

  or for the inexact case

$$\min_{\mathbf{X}} \quad \text{rank}(\mathbf{X}) \qquad \text{s.t.} \quad \|\mathbf{W} \circledast (\mathbf{Y} - \mathbf{X})\|_F^2 \leq \delta$$

  where '$\circledast$' is the elementwise multiplication.

The Netflix Prize: The problem of Matrix Completion II

- $\mathrm{rank}(\mathbf{X})$ is a discontinuous and non-convex function
- $\mathrm{rank}(\mathbf{X}) = \|\boldsymbol{\sigma}\|_0$, where $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots]$: singular values of the matrix $\mathbf{X}$
- Nuclear norm of $\mathbf{X}$: $\|\mathbf{X}\|_* = \|\boldsymbol{\sigma}\|_1$ is convex surrogate of the rank function.

Problem (Solving the Netflix)

$$\min_{\mathbf{X}} \quad \|\mathbf{X}\|_* \qquad \textit{s.t.} \quad \|\mathbf{W} \circledast (\mathbf{Y} - \mathbf{X})\|_F^2 \leq \delta^2$$

Solving Matrix Completion

```
cvx_begin
    variable X(size(Y))     % declare $\bX$ of the same size with $\bY$
    minimize norm_nuc(X)    % objective function
    subject to
        sum_square(W(:).*(X(:)-Y(:)))≤ Δ^2; % constraint function
cvx_end
```

- CVX is useful to check the algorithm for data of small scale
- Processing large data needs a faster algorithm

Example: Matrix Completion without CVX Solver I

Question: How to solve the constrained nuclear norm minimization more efficiently without using the CVX toolbox

$$\min_{\mathbf{X}} \quad \|\mathbf{X}\|_* \qquad \text{s.t.} \quad \|\mathbf{W} \circledast (\mathbf{Y} - \mathbf{X})\|_F^2 \leq \delta^2$$

- Introduce new variable $\mathbf{Z}$ and a new equality constraint

$$\begin{aligned} \min_{\mathbf{X},\mathbf{Z}} \quad & \|\mathbf{X}\|_* \\ \text{s.t.} \quad & \|\mathbf{W} \circledast (\mathbf{Y} - \mathbf{Z})\|_F^2 \leq \delta^2 \\ & \mathbf{X} = \mathbf{Z} \end{aligned}$$

Example: Matrix Completion without CVX Solver II

- Define $\mathcal{D}$ a set of matrices, $\mathbf{Z}$, which satisfy the first inequality condition, i.e., $\mathcal{D} = \{\mathbf{Z} \mid \|\mathbf{W} \circledast (\mathbf{Y} - \mathbf{Z})\|_F^2 \leq \delta^2\}$ and an indicator function $i_{\mathcal{D}}(\mathbf{Z})$ associated with the set $\mathcal{D}$, i.e., $i_{\mathcal{D}}(\mathbf{Z}) = 0$ if $\mathbf{Z} \in D$, otherwise ∞.
- Rewrite the optimization problem as

$$\begin{aligned} \min_{\mathbf{X},\mathbf{Z}} \quad & \|\mathbf{X}\|_* + i_{\mathcal{D}}(\mathbf{Z}) \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{Z} \end{aligned}$$

- **How to solve the above optimization problem?**

Example: Matrix Completion without CVX Solver III

- Write the augmented Lagrangian function

$$\mathcal{L}(\mathbf{X}, \mathbf{Z}, \mathbf{T}) = \|\mathbf{X}\|_* + i_{\mathcal{D}}(\mathbf{Z}) + \frac{\lambda}{2}(\|\mathbf{X} - \mathbf{Z} - \mathbf{T}\|_F^2 - \|\mathbf{T}\|_F^2)$$

- Next we update parameters $\mathbf{Z}$, $\mathbf{X}$ and $\mathbf{T}$ sequentially

$$\begin{aligned} \mathbf{X} &= \arg\min_{\mathbf{X}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \mathbf{T}) = \arg\min_{\mathbf{X}} \ \|\mathbf{X}\|_* + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{Z} - \mathbf{T}\|_F^2 \\ \mathbf{Z} &= \arg\min_{\mathbf{Z}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \mathbf{T}) = \arg\min_{\mathbf{Z}} \ i_{\mathcal{D}}(\mathbf{Z}) + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{Z} - \mathbf{T}\|_F^2 \\ \mathbf{T} &\leftarrow \mathbf{T} + \mathbf{Z} - \mathbf{X} \end{aligned}$$

- **Can we simplify the update rules for $\mathbf{Z}$ and $\mathbf{X}$?**

Example: Matrix Completion without CVX Solver IV

- Update $\mathbf{X}$ is achieved by SVD of $\mathbf{Z} + \mathbf{T} = \mathbf{U}\,\mathrm{diag}(\boldsymbol{\sigma})\mathbf{V}^T$

$$\mathbf{X}^* = \mathbf{U}\,\mathrm{diag}(\max(\boldsymbol{\sigma} - \frac{1}{\lambda}, 0))\mathbf{V}^T$$

Question: can we derive the above analytical solution?

- Update $\mathbf{Z}$ as projection of $\mathbf{X} - \mathbf{T}$ to the set $\mathcal{D}$, i.e.,

$$\begin{aligned}\mathbf{Z} &= \arg\min \|\mathbf{X} - \mathbf{T} - \mathbf{Z}\|_F^2 \quad \text{s.t.} \quad \|\mathbf{W} \circledast (\mathbf{Y} - \mathbf{Z})\|_F^2 \leq \delta^2 \\ &= \mathbf{X} - \mathbf{T} + (1 - \alpha)\mathbf{H}\end{aligned}$$

where $\mathbf{H} = \mathbf{W} \circledast (\mathbf{Y} - \mathbf{X} + \mathbf{T})$ and $\alpha = \frac{\delta}{\|\mathbf{H}\|_F}$

Example: Matrix Completion without CVX Solver V

```
 ...
for ki = 1: ...

    % Update Z
    H = W.*(Y - X + T);
    alpha = Δ/norm(H(:));
    Z = X-T+H*(1-alpha);

    % update X
    [u,s,v] = svd(Z+T);
    s = max(diag(s) - 1/lambda,0);

    X = u*diag(s)*v';
    % update T
    T = T + Z - X;
    ...
end
```

Example: Least Squares with Spherical constraint I

$$\min_{\boldsymbol{x}} \quad \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_2^2 = 1$$

Sphere $\|\boldsymbol{x}\|_2^2 = 1$ is not a convex set, but the norm ball is convex $\|\boldsymbol{x}\|_2^2 \leq 1$

$$\min_{\boldsymbol{x}} \quad \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_2^2 \leq 1$$

```
cvx_begin
    variable x(d,1)
    minimize sum_square(A*x - b)
    subject to norm(x)≤ 1;
cvx_end
```

Example: Least Squares with Spherical constraint II

Problem (Can we utilize the closed-form solution for LS?)

- Introduce a new variable z and a new equality constraint

$$\begin{aligned} \min_{\boldsymbol{x},z} \quad & \frac{1}{2}\|\mathbf{AX} - \boldsymbol{b}\|_2^2 + i_B(z) \\ \text{s.t.} \quad & \boldsymbol{x} = z \end{aligned}$$

  where indicator function $i_B(z)$ is for the unit ball
  $i_B(z) = 0$ if $\|z\| = 1$, and ∞ elsewhere.

- Write the augmented Lagrangian function

$$\mathcal{L}(\boldsymbol{x}, z, \boldsymbol{t}) = \frac{1}{2}\|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + i_B(z) + \frac{\lambda}{2}(\|\boldsymbol{x} - z - \boldsymbol{t}\|_F^2 - \|\boldsymbol{t}\|_F^2)$$

Example: Least Squares with Spherical constraint III

- Next we update parameters $\boldsymbol{x}$, $\boldsymbol{z}$ and $\boldsymbol{y}$ sequentially

$$
\begin{aligned}
\boldsymbol{x} &= \arg\min_{\mathbf{X}} \ \frac{1}{2}\|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \boxed{\tfrac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{z} - \boldsymbol{t}\|_F^2} \\
&= (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}(\mathbf{A}^T\boldsymbol{b} + \lambda(\boldsymbol{z} + \boldsymbol{t})) \\
\boldsymbol{z} &= \arg\min_{\boldsymbol{z}} \ i_B(\boldsymbol{z}) + \boxed{\tfrac{\lambda}{2}\|\boldsymbol{x} - \boldsymbol{z} - \boldsymbol{t}\|_F^2} \\
&= \text{project}_B(\boldsymbol{x} - \boldsymbol{t}) = \frac{\boldsymbol{x} - \boldsymbol{t}}{\|\boldsymbol{x} - \boldsymbol{t}\|_2} \\
\boldsymbol{t} &\leftarrow \boldsymbol{t} + \boldsymbol{z} - \boldsymbol{t}
\end{aligned}
$$

Example: Least Squares with Spherical constraint IV

```
% precomputing Q matrix
Q = A'*A+lambda * eye(size(A,2));
iQ = inv(Q); h = A'*b;

% main algorithm
for ki = 1: ...

    % update x
    x = iQ*(h + lambda*(z + t));

    % update z
    z = x-t; z = z/norm(z);

    % update dual variable
    t = t + x - z;
end
```

CVX in Deep Neural Networks: Optimization as a layer I

- Most layers in DNN are expressed as functions that map inputs of the layers to outputs in an mathematical form defined precisely

$$y = \phi(\mathbf{W}\boldsymbol{x} + \boldsymbol{b})$$

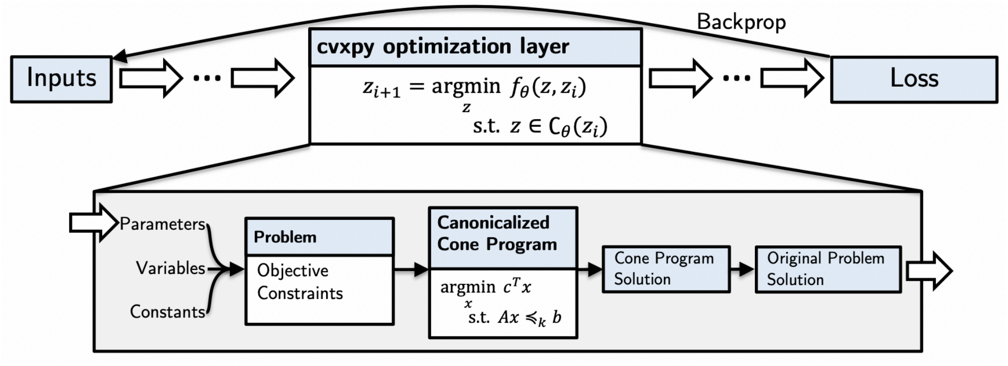
  where ϕ can be linear activation, ReLU, Hyperbolic tangent, sigmoid, ...
  Such representation limits neural network learning complex constraints,
  e.g., the kernels or outcomes can be orthogonal, statistically independent, lie on manifold of low-rank tensors.
- **Optimization Layer** Optimization as a layer, expresses *output of the layer as outcomes of an optimization problem*

CVX in Deep Neural Networks: Optimization as a layer II

- Many NN layers solve convex and constrained optimization problems

Layer	Expression	Optimization problem
ReLU	$f(\boldsymbol{x}) = \max\{0, \boldsymbol{x}\}$	$f(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}} \quad \frac{1}{2}\lVert \boldsymbol{x} - \boldsymbol{y}\rVert_2^2 \quad$ s.t. $\boldsymbol{y} \geq 0$
Sigmoid	$f(\boldsymbol{x}) = (1 + e^{-\boldsymbol{x}})^{-1}$	$f(\boldsymbol{x}) = \arg\min_{0<\boldsymbol{y}<1} \quad -\boldsymbol{x}^T\boldsymbol{y} - H_b(\boldsymbol{y})$ where $H_b(\boldsymbol{y})$ is the binary entropy function $H_b(\boldsymbol{y}) = -(\sum_i y_i \log y_i + (1 - y_i)\log(1 - y_i))$
Softmax	$f(\boldsymbol{x})_j = e^{x_j} / \sum_i e^{x_i}$	$f(\boldsymbol{x}) = \arg\min_{0<\boldsymbol{y}<1} \quad -\boldsymbol{x}^T\boldsymbol{y} - H(\boldsymbol{y}) \quad$ s.t. $\mathbf{1}^T\boldsymbol{y} = 1$ where $H()$ is the entropy function $H(\boldsymbol{y}) = -\sum_i y_i \log y_i$

CVX in Deep Neural Networks: Optimization as a layer III


Backprop
Inputs
cvxpy optimization layer
$z_{i+1} = \underset{z}{\operatorname{argmin}} \ f_\theta(z, z_i)$
s.t. $z \in C_\theta(z_i)$
Loss
Parameters
Variables
Constants
Problem
Objective
Constraints
Canonicalized Cone Program
$\underset{x}{\operatorname{argmin}} \ c^T x$
s.t. $Ax \preccurlyeq_k b$
Cone Program Solution
Original Problem Solution

CVXPyLayer

ReLu layer $z = \max(Wx + b, 0)$ can be written as a convex optimization problem

$$\begin{array}{ll} \text{minimize} & \|z - \tilde{W}x - b\|_2^2 \\ \text{subject to} & z \geq 0, \\ & \tilde{W} = W, \end{array} \tag{1}$$

Topics I

Lecture 1 - Introduction to Convex Optimization
Lecture 2 - Convex sets and convex functions
Lecture 3 - Convex optimization problems
Lecture 4 - Linear Programming and norm minimization
Lecture 5 - Sparse Approximation and ell-1 Trend Filtering
Lecture 6 - Application: Convex nonnegative matrix factorization
Lecture 7 - Constrained Quadratic Programming
Lecture 8 - Gradient method
Lecture 9 - Convex optimization solvers and CVX
Lecture 10 - Application: Image denoising
Lecture 11 - Lagrangian duality theory
Lecture 12 - Alternating Direction Method of Multipliers
Lecture 13 - Primal-Dual method
Lecture 14 - CVXPyLayers: differentiable convex optimization layers
Lecture 15 - Application to Machine learning: regression, classification and SVM
Lecture 16 - Optimal Transport problem
Lecture 17 - Application: low-rank optimization problems

Grades I

- Homework - 40%
    - Homework 1-2 on theory (convex sets and convex functions)
    - Homework 3-4 on applications and computation (using CVX)
- Mid-term exam - 20%
    - On theory and applications
- Project - (35+5)%
    - Study a research problem and give a presentation
- Bonus:
    - In-class problems
    - Early-submission bonus: 20% or 40% of the full score before the due date one or 2 days, respectively.
      The bonus is awarded only to the correct and complete answers.

Course Goals

- Introduce convex optimization theory and to illustrate its use with many recent applications in image processing, blind separation, machine learning, etc.
- Recognize convex optimization problems
- Use tools to solve convex optimization problems
- Solve non-convex optimization problems by Convex Relaxation methods

Project Evaluation

- Scientific work (15 points)
  - Review of Literature + Complexity of the problem and its difficulty
  - Implementation + Comparison with existing methods for the similar problems + Teamwork + Creative
- Presentation (10 points)
  - Motivation of the project + Importance and contribution of the study methods or scientific papers + Application of the methods or problems investigated + How to solve the problem + Discussion
- Demonstration (10 points)
  - Show examples to demonstrate the method
  - Quality of the implementation
- Audience assessment (5 points)
  - Appropriateness to problem investigated + Clarity, conciseness + Significance of the project and possible contribution + Review of Literature (thoroughness and comprehensiveness) + Quality of the examples

People tend to look and search for something where it is easiest

THIS IS WHERE YOU LOST YOUR WALLET?
NO, I LOST IT IN THE PARK. BUT THIS IS WHERE THE LIGHT IS.