Classification and Representation Learning

Course 2 : Linear Algebra and Optimization

Hoel Le Capitaine

Academic year 2017-2018

Why going back ?

- Linear algebra notation is used extensively in machine learning, optimization, power systems
- This lecture reviews some basic notation and methods, and introduces matrix calculus used in class
- Optimization is at the heart of most machine learning algorithms

Linear system

Let us consider a set of linear equations (here two equations, two unknown variables)

$$\begin{aligned} 4x_1 - 5x_2 &= -13 \\ -2x_1 + 3x_2 &= 9 \end{aligned}$$

Linear system

Let us consider a set of linear equations (here two equations, two unknown variables)

$$
\begin{aligned}
4x_1 - 5x_2 &= -13 \\
-2x_1 + 3x_2 &= 9
\end{aligned}
$$

This set can be compactly represented using matrix notation $Ax = b$, where

$$
A = \begin{bmatrix} 4 & -5 \\ -2 & 3 \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ and } b = \begin{bmatrix} -13 \\ 9 \end{bmatrix}
$$

Notation

Matrices and vectors

- A matrix with real-valued entries, m rows, and n columns $A \in \mathbb{R}^{m \times n}$
- A_{ij} denotes the entry in the ith row and jth column
- A (column) vector with n real-valued entries $\mathbf{x} \in \mathbb{R}^n$, x_i denotes the ith entry

The transpose operation

- The transpose operator A^T switches rows and columns of a matrix : $A_{ij} = (A^T)_{ji}$
- For a vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^T \in \mathbb{R}^{1 \times n}$ is a row vector

Matrix operations

Addition

- For two matrices of the same size $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times n}$, their addition is the pointwise addition of elements $A + B = C \in \mathbb{R}^{m \times n}$, where $C_{ij} = A_{ij} + B_{ij}$
- This operation is not defined if $size(A) \neq size(B)$

Multiplication

- For two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, their product is $AB = C \in \mathbb{R}^{m \times p}$, where $C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$
- This operation is not defined if $ncol(A) \neq nrow(B)$

Matrix multiplication

Special cases

- inner product, for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$$\mathbf{x}^T\mathbf{y} = \sum_{i=1}^{n} x_i y_i \in \mathbb{R}$$

- matrix-vector product, for $A \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$

$$Ax \in \mathbb{R}^m, (Ax)_i = \sum_{k=1}^{n} A_{ik} x_k$$

Properties

- associative $A(BC) = (AB)C$
- distributive $A(B + C) = AB + AC$
- not commutative $AB \neq BA$
- transpose of product $(AB)^T = B^T A^T$

Special matrices

Identity

- $I \in \mathbb{R}^{n \times n} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$. For any $A \in \mathbb{R}^{m \times n}$, $AI = A = IA$

Constant

- zero matrix : $0 \in \mathbb{R}^{m \times n} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$, used for block matrices, e.g.

  $\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$

- all-ones vector : $\mathbf{1} \in \mathbb{R}^{n} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$, for compact sums : $\mathbf{1}^T\mathbf{x} = \sum_{i=1}^{n} x_i$

Symmetric matrix

- A square matrix A is symmetric if $A = A^T$.
- For $A \in \mathbb{R}^{m \times n}$, $A^T A \in \mathbb{R}^{m \times m}$ is symmetric

Inverse matrix

The inverse of a square matrix $A \in \mathbb{R}^{n \times n}$, denoted A^{-1},

$$AA^{-1} = I = A^{-1}A.$$

This inverse may not exist (non-singular matrix has inverse, singular does not)

$$A^{-1} \text{ exists iff } Ax \neq 0 \ \forall x \neq 0$$

Exercise 1: Matrices

Let $A = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}$ and $B = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$.

Give, if it is possible

- the transpose of A, and the sum $A + B$
- the two possible product AB (algebraic and element-wise)
- the inverse of A and B

Definition

- for $A \in \mathbb{R}^{n \times n}$, $\lambda \in \mathbb{C}$ is an **eigenvalue**, and $\mathbf{v} \in \mathbb{C}^n$ is an **eigenvector** if

$$A\mathbf{v} = \lambda \mathbf{v}$$

- for $A \in \mathbb{R}^{n \times n}$, there exists n solutions to this equation (implying n eigenvalues λ_i and n eigenvectors $\mathbf{v}_i$, $i = 1, \cdots, n$)
- let $V = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n]$, then A can be factorized as $A = V \Lambda V^{-1}$, where Λ is the diagonal matrix such that $\Lambda_{ii} = \lambda_i$
- question : what can we say about A^k ?

Notation and rules

- the gradient $\nabla_{\mathbf{x}} f(\mathbf{x}) \in \mathbb{R}^n = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}$
- $\nabla_{\mathbf{x}}(\mathbf{x}^T A \mathbf{x}) = (A + A^T)\mathbf{x}$
- $\nabla_{\mathbf{x}}(\mathbf{b}^T \mathbf{x}) = \mathbf{b}$

What is optimization?

- find the minimizer of a function subject to constraints

$$\text{minimize}_{\mathbf{x}}\ f_0(\mathbf{x})$$
$$\text{subject to } f_i(\mathbf{x}) \leq b_i, i = \{1, \cdots, m\}$$

- the vector $\mathbf{x}$ is the optmization variable, f_0 the objective function, and the f_i are the constraints fucntions
- a vector $\mathbf{x}^\star$ is said to be optimal (or solution) if it has the smallest objective value, given the constraints holds

Example 1: Portofolio optimization

- In portfolio optimization, we seek the best way to invest some capital in a set of n assets. The variable x_i represents the investment in the ith asset, so $\mathbf{x} \in \mathbb{R}^n$ describes the overall portfolio allocation across the set of assets.
- The constraints might represent a limit on the budget (i.e., a limit on the total amount to be invested), the requirement that investments are nonnegative (assuming short positions are not allowed).
- The objective or cost function might be a measure of the overall risk. The optimization problem corresponds to choosing a portfolio allocation that minimizes risk, among all possible allocations that meet the firm requirements.

Optimization

Why do we care about optimization?

Optimization is at the heart of many (most practical?) machine learning algorithms.

- linear regression

$$\min_w \|Xw - y\|^2$$

- classification (eg SVM)

$$\min_w \|w\|^2 + C\sum_{i=1}^{n} \xi_i \text{ s.t. } \xi_i \geq 1 - y_i x_i^T w, \xi_i \geq 0$$

- collaborative filtering

$$\min_w \sum_{i \prec j} \log\left(1 + \exp(w^T x_i - w^T w_j)\right)$$

- clustering

$$\min_\mu \sum_{j=1}^{k} \sum_{i \in C_j} \|x_i - \mu_j\|^2$$

Solving optimization problems

General optimization problem

- very difficult to solve
- methods involve some compromise, e.g., very long computation time, or not always finding the solution (which may not matter in practice)

Exceptions: certain problem classes can be solved efficiently and reliably

- least-squares problems
- linear programming problems
- convex optimization problems

$$\text{minimize } \|Ax - b\|_2^2$$

Solving least-squares problems

- analytical solution: $x^\star = (A^T A)^{-1} A^T b$
- reliable and efficient algorithms and software
- computation time proportional to n^{2k} ($A \in \mathbb{R}^{k \times n}$); less if structured
- a mature technology

Using least-squares

- least-squares problems are easy to recognize
- a few standard techniques increase flexibility (e.g., including weights, adding regularization terms)

Linear programming

$$\begin{array}{c} \text{minimize } c^T x \\ \text{subject to } a_i^T x \leq b_i,\ i = 1, \cdots, m \end{array}$$

Solving linear programs

- no analytical formula for solution
- reliable and efficient algorithms and software
- computation time proportional to $n^2 m$ if $m \geq n$; less with structure
- a mature technology

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)

$$\text{minimize } f_0(x)$$
$$\text{subject to } f_i(x) \leq b_i,\ i = 1, \cdots, m$$

- objective and constraint functions are convex:

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y)$$

  if $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$
- includes least-squares problems and linear programs as special cases

Solving convex optimization problems

- no analytical solution
- reliable and efficient algorithms
- computation time (roughly) proportional to $\max\{n^3, n^2m, F\}$, where F is cost of evaluating f_i's and their first and second derivatives
- almost a technology

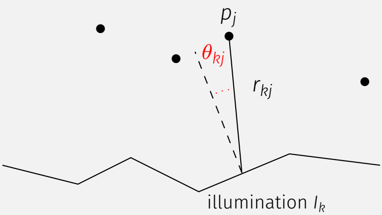
Using convex optimization

- often difficult to recognize
- many tricks for transforming problems into convex form
- surprisingly many problems can be solved via convex optimization

Practical example

Example 2: The lamps

m lamps illuminating n (small, flat) patches



Intensity I_k at patch k depends linearly on lamp powers p_j:

$$I_k = \sum_{j=1}^{m} a_{kj} p_j, \quad a_{kj} = r_{kj}^{-2} \max(\cos\theta_{kj}, 0)$$

Problem: achieve desired illumination I_{des} with bounded lamp powers

$$\min_{p} \max_{k=1,\cdots,n} |\log I_k - \log I_{des}|, \quad 0 \leq p_j \leq p_{max}, j = 1, \cdots, m$$

How to solve?

- use uniform power: $p_j = p$, vary p
- use least-squares:

$$\min_p \sum_{k=1}^{n} (I_k - I_{des})^2$$

  round p_j if $p_j > p_{max}$ or $p_j < 0$
- use weighted least-squares:

$$\min_p \sum_{k=1}^{n} (I_k - I_{des})^2 + \sum_{j=1}^{m} w_j (p_j - p_{max}/2)^2$$

  iteratively adjust weights w_j until $0 \le p_j \le p_{max}$
- use linear programming:

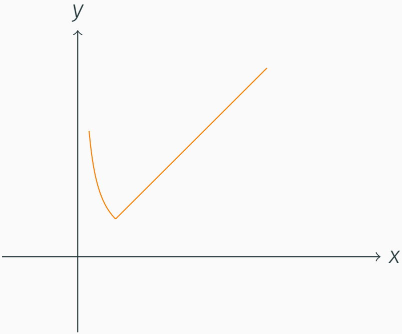
$$\min_p \max_{k=1,\cdots,n} |I_k - I_{des}|$$

  subject to $0 \le p_j \le p_{max}, j = 1, \cdots, m$ which can be solved via linear programming of course these are approximate (suboptimal) 'solutions'

- use convex optimization, equivalent to

$$\min f_0(p) = \max_{k=1,\cdots,n} h(I_k/I_{des}), \text{ s.t. } 0 \leq p_j \leq p_{max}, j = 1, \cdots, m$$

with $h(u) = \max\{u, 1/u\}$



f_0 is convex because a maximum of convex functions is convex
exact solution obtained with effort $\approx$ modest factor $\times$ least-square effort

additional constraints

Does adding 1. or 2. below complicate the problem ?

1. no more than half of total power is in any 10 lamps
2. no more than half of the lamps are on ($p_j > 0$)

additional constraints

Does adding 1. or 2. below complicate the problem ?

1. no more than half of total power is in any 10 lamps
2. no more than half of the lamps are on ($p_j > 0$)

- answer: with 1., still easy to solve; with 2., extremely difficult
- moral: (untrained) intuition doesn't always work; without the proper background very easy problems can appear quite similar to very difficult problems

Exercise 2: Gradient

Let $f(x, y) = 2x^2 + 3y^3 - 8x - 3y + 8$
Give, the following optimum values (and precise if they are maximum or minimum)

- $\min_x f(x, y)$
- $\min_y f(x, y)$
- $\min_x f(x, y) + \|x\|^2$

Linear algebra

- The matrix cookbook https://www.ics.uci.edu/~welling/teaching/KernelsICS273B/MatrixCookBook.pdf
- Linear algebra (Hefferon) http://joshua.smcvt.edu/linearalgebra/book.pdf

Optimization

- Convex optimization (Boyd) http://web.stanford.edu/~boyd/cvxbook/
- An introduction to optimization (Chong and Zak) (google it)