A Quick Tour of Linear Algebra and Optimization for Machine Learning

Masoud Farivar

January 8, 2015

Outline of Part I: Review of Basic Linear Algebra

- Matrices and Vectors
- Matrix Multiplication
- Operators and Properties
- Special Types of Matrices
- Vector Norms
- Linear Independence and Rank
- Matrix Inversion
- Range and Nullspace of a Matrix
- Determinant
- Quadratic Forms and Positive Semidefinite Matrices
- Eigenvalues and Eigenvectors
- Matrix Eigendecomposition

Outline of Part II: Review of Basic Optimization

- The Gradient
- The Hessian
- Least Squares Problem
- Gradient Descent
- Stochastic Gradient Descent
- Convex Optimization
- Special Classes of Convex Problems
- Example of Convex Problems in Machine learning
- Convex Optimization Tools

Matrices and Vectors

- Matrix: A rectangular array of numbers, e.g., $A \in \mathbb{R}^{m \times n}$:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

- Vector: A matrix with only one column (default) or one row, e.g., $x \in \mathbb{R}^n$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Matrix Multiplication

- If $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, $C = AB$, then $C \in \mathbb{R}^{m \times p}$:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

Properties of Matrix Multiplication:

- Associative: $(AB)C = A(BC)$
- Distributive: $A(B + C) = AB + AC$
- Non-commutative: $AB \neq BA$
- Block multiplication: If $A = [A_{ik}]$, $B = [B_{kj}]$, where A_{ik}'s and B_{kj}'s are matrix blocks, and the number of columns in A_{ik} is equal to the number of rows in B_{kj}, then $C = AB = [C_{ij}]$ where $C_{ij} = \sum_k A_{ik} B_{kj}$

Operators and properties

Transpose: $A \in \mathbb{R}^{m \times n}$, then $A^T \in \mathbb{R}^{n \times m}$: $(A^T)_{ij} = A_{ji}$

Properties:

- $(A^T)^T = A$
- $(AB)^T = B^T A^T$
- $(A + B)^T = A^T + B^T$

Trace: $A \in \mathbb{R}^{n \times n}$, then: $tr(A) = \sum_{i=1}^{n} A_{ii}$

Properties:

- $tr(A) = tr(A^T)$
- $tr(A + B) = tr(A) + tr(B)$
- $tr(\lambda A) = \lambda tr(A)$
- If AB is a square matrix, $tr(AB) = tr(BA)$

Special types of matrices

- Identity matrix: $I = I_n \in \mathbb{R}^{n \times n}$:

$$I_{ij} = \begin{cases} 1 & \text{i=j,} \\ 0 & \text{otherwise.} \end{cases}$$

- $\forall A \in \mathbb{R}^{m \times n}$: $AI_n = I_m A = A$
- Diagonal matrix: $D = diag(d_1, d_2, \ldots, d_n)$:

$$D_{ij} = \begin{cases} d_i & \text{j=i,} \\ 0 & \text{otherwise.} \end{cases}$$

- Symmetric matrices: $A \in \mathbb{R}^{n \times n}$ is symmetric if $A = A^T$.
- Orthogonal matrices: $U \in \mathbb{R}^{n \times n}$ is orthogonal if $UU^T = I = U^T U$

Vector Norms

A norm of a vector $||x||$ is a measure of it's "length" or "magnitude". The most common is the Euclidean or ℓ_2 norm.

- ℓ_p norm : $||x||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$
- ℓ_2 norm : $||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$

  used in ridge regression: $||y - X\beta||^2 + \lambda||\beta||_2^2$
- ℓ_1 norm : $||x||_1 = \sum_{i=1}^{n} |x_i|$

  used in ℓ_1 penalized regression: $||y - X\beta||^2 + \lambda||\beta||_1$
- ℓ_∞ norm : $||x||_\infty = \max_i |x_i|$

Linear Independence and Rank

- A set of vectors $\{x_1, \ldots, x_n\}$ is linearly independent if $\nexists\{\alpha_1, \ldots, \alpha_n\}$: $\sum_{i=1}^{n} \alpha_i x_i = 0$
- Rank: $A \in \mathbb{R}^{m \times n}$, then $rank(A)$ is the maximum number of linearly independent columns (or equivalently, rows)
- Properties:
  - $rank(A) \leq \min\{m, n\}$
  - $rank(A) = rank(A^T)$
  - $rank(AB) \leq \min\{rank(A), rank(B)\}$
  - $rank(A + B) \leq rank(A) + rank(B)$

Matrix Inversion

- If $A \in \mathbb{R}^{n \times n}$, $rank(A) = n$, then the inverse of A, denoted A^{-1} is the matrix that: $AA^{-1} = A^{-1}A = I$
- Properties:
  - $(A^{-1})^{-1} = A$
  - $(AB)^{-1} = B^{-1}A^{-1}$
  - $(A^{-1})^T = (A^T)^{-1}$
- The inverse of an orthogonal matrix is its transpose

Range and Nullspace of a Matrix

- Span: $span(\{x_1, \ldots, x_n\}) = \{\sum_{i=1}^{n} \alpha_i x_i | \alpha_i \in \mathbb{R}\}$
- Projection: $Proj(y; \{x_i\}_{1 \leq i \leq n}) = argmin_{v \in span(\{x_i\}_{1 \leq i \leq n})} \{||y - v||_2\}$
- Range: $A \in \mathbb{R}^{m \times n}$, then $\mathcal{R}(A) = \{Ax | \, x \in R^n\}$ is the span of the columns of A
- $Proj(y, A) = A(A^T A)^{-1} A^T y$
- Nullspace: $null(A) = \{x \in \mathbb{R}^n | \, Ax = 0\}$

Determinant

- $A \in \mathbb{R}^{n \times n}$, $a_1, \ldots, a_n$ the rows of A, then $det(A)$ is the volume of the $S = \{\sum_{i=1}^{n} \alpha_i a_i | \, 0 \leq \alpha_i \leq 1\}$.
- Properties:
  - $det(I) = 1$
  - $det(\lambda A) = \lambda det(A)$
  - $det(A^T) = det(A)$
  - $det(AB) = det(A)det(B)$
  - $det(A) \neq 0$ if and only if A is invertible.
  - If A invertible, then $det(A^{-1}) = det(A)^{-1}$

Quadratic Forms and Positive Semidefinite Matrices

- $A \in \mathbb{R}^{n \times n}$, $x \in \mathbb{R}^n$, $x^T A x$ is called a quadratic form:

$$x^T A x = \sum_{1 \leq i,j \leq n} A_{ij} x_i x_j$$

- A is positive definite if $\forall\, x \in \mathbb{R}^n : x^T A x > 0$
- A is positive semidefinite if $\forall\, x \in \mathbb{R}^n : x^T A x \geq 0$
- A is negative definite if $\forall\, x \in \mathbb{R}^n : x^T A x < 0$
- A is negative semidefinite if $\forall\, x \in \mathbb{R}^n : x^T A x \leq 0$

Eigenvalues and Eigenvectors

- $A \in \mathbb{R}^{n\times n}$, $\lambda \in \mathbb{C}$ is an eigenvalue of A with the corresponding eigenvector $x \in \mathbb{C}^n$ $(x \neq 0)$ if:

$$Ax = \lambda x$$

- eigenvalues: the n possibly complex roots of the polynomial equation $det(A - \lambda I) = 0$, and denoted as $\lambda_1, \dots, \lambda_n$
- Properties:
  - $tr(A) = \sum_{i=1}^{n} \lambda_i$
  - $det(A) = \prod_{i=1}^{n} \lambda_i$
  - $rank(A) = |\{1 \leq i \leq n | \lambda_i \neq 0\}|$

Matrix Eigendecomposition

- $A \in \mathbb{R}^{n \times n}$, $\lambda_1, \ldots, \lambda_n$ the eigenvalues, and $x_1, \ldots, x_n$ the eigenvectors. $X = [x_1|x_2|\ldots|x_n]$, $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$, then $AX = X\Lambda$.
- A called diagonalizable if X invertible: $A = X\Lambda X^{-1}$
- If A symmetric, then all eigenvalues real, and X orthogonal (hence denoted by $U = [u_1|u_2|\ldots|u_n]$):

$$A = U\Lambda U^T = \sum_{i=1}^{n} \lambda_i u_i u_i^T$$

- A special case of Singular Value Decomposition

The Gradient

Suppose $f : \mathbb{R}^{m\times n} \rightarrow \mathbb{R}$ is a function that takes as input a matrix A and returns a real value. Then the gradient of f is the matrix

$$\nabla_A f(A) \in \mathbb{R}^{m\times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdots & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}$$

Note that the size of this matrix is always the same as the size of A. In particular, if A is the vector $x \in \mathbb{R}^n$,

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

The Hessian

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a function that takes a vector in $\mathbb{R}^n$ and returns a real number. Then Hessian matrix with respect to x, the $n \times n$ matrix:

$$\nabla_x^2 f(x) \in \mathbb{R}^{n\times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

Gradient and Hessian of Quadratic and Linear Functions:

- $\nabla_x b^T x = b$
- $\nabla_x x^T A x = 2Ax$ (if A symmetric)
- $\nabla_x^2 x^T A x = 2A$ (if A symmetric)

Least Squares Problem

Solve the following minimization problem:

$$\text{minimize} \quad \tfrac{1}{2}||Ax - b||_2^2$$

Note that

$$\begin{aligned} ||Ax - b||_2^2 &= (Ax - b)^T(Ax - b) \\ &= x^T A^T A x - 2b^T A x + b^T b \end{aligned}$$

Taking the gradient with respect to x we have (and using the properties above):

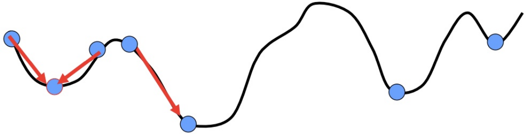
$$\begin{aligned} \nabla_x(x^T A^T A x - 2b^T A x + b^T b) &= \nabla_x x^T A^T A x - \nabla_x 2b^T A x + \nabla_x b^T b \\ &= 2A^T A x - 2A^T b \end{aligned}$$

Setting this to zero and solving for x gives the following closed form solution (psuedo-inverse):

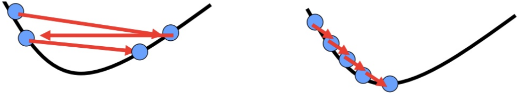
$$x = (A^T A)^{-1} A^T b$$

Gradient Descent

- Gradient Descent (GD): takes steps proportional to the negative of the gradient (first order method)



- Advantage: very general (we'll see it many times)
- Disadvantage: Local minima (sensitive to starting point)
- Step size
  - not too large, not too small
  - Common choices:
    - Fixed
    - Linear with iteration (May want step size to decrease with iteration)
    - More advanced methods (e.g., Newton's method)



Gradient Descent

A typical machine learning problem aims to minimize Error(loss) + Regularizer (penalty):

$$\min_{w} F(w) = f(w; y, x) + g(w)$$

Gradient Descent (GD):

- choose initial $w^{(0)}$
- repeat

$$w^{(t+1)} = w^{(t)} - \eta_t \nabla F(w^{(t)})$$

- until

$$||w^{(t+1)} - w^{(t)}|| \leq \epsilon \quad \text{or} \quad ||\nabla F(w^{(t)})|| \leq \epsilon$$

Stochastic (Online) Gradient Descent

- Use updates based on individual data points chosen at random
- Applicable when minimizing an objective function is sum of differentiable functions:

$$f(w; y, x) = \frac{1}{n} \sum_{i=1}^{n} f(w; y_i, x_i)$$

- Suppose we receive an stream of samples (y_t, x_t) from the distribution, the idea of SGD is:

$$w^{(t+1)} = w^{(t)} - \eta_t \nabla_w f(w^{(t)}; y_t, x_t)$$

- In practice, we typically shuffle data points in the training set randomly and use them one by one for the updates.

Stochastic (Online) Gradient Descent

- The objective does not always decrease for each step
- comparing to GD, SGD needs more steps, but each step is cheaper
- mini-batch (say pick up 100 samples and average) can potentially accelerate the convergence

Convex Optimization

- A set of points S is convex if, for any $x, y \in S$ and for any $0 \leq \theta \leq 1$,

$$\theta x + (1 - \theta) y \in S$$

- A function $f : S \to \mathbb{R}$ is convex if its domain S is a convex set and

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

  for all $x, y \in S$, $0 \leq \theta \leq 1$.
- Convex functions can be efficiently minimized.

Convex Optimization

A convex optimization problem in an optimization problem of the form

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

where f is a convex function, C is a convex set, and x is the optimization variable. Or equivalently:

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, p \end{array}$$

where g_i are convex functions, and h_i are affine functions.

Theorem

All locally optimal points of a convex optimization problem are globally optimal.

Note that the optimal solution is not necessarily unique.

Special Classes of Convex Problems

- Linear Programming:

$$\begin{array}{ll} \text{minimize} & c^T x + d \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

- Quadratic Programming:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} x^T P x + c^T x + d \\ \text{subject to} & Gx \preceq h \\ & Ax = b \end{array}$$

- Quadratically Constrained Quadratic Programming:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} x^T P x + c^T x + d \\ \text{subject to} & \frac{1}{2} x^T Q_i x + r_i^T x + s_i \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

- Semidefinite Programming:

$$\begin{array}{ll} \text{minimize} & \mathrm{tr}(CX) \\ \text{subject to} & \mathrm{tr}(A_i X) = b_i, \quad i = 1, \ldots, p \\ & X \succeq 0 \end{array}$$

Example of Convex Problems in Machine learning

- Support Vector Machine (SVM) Classifier:

$$\begin{array}{lll} \text{minimize} & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{m}\xi_i & \\ \text{subject to} & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, & i = 1, \ldots, m \\ & \xi_i \geq 0, & i = 1, \ldots, m \end{array}$$

This is a quadratic program with optimization variables $\omega \in \mathbb{R}^n$, $\xi \in \mathbb{R}^m$, $b \in \mathbb{R}$, and the input data $x(i), y(i), i = 1, \ldots m$, and the parameter $C \in \mathbb{R}$

Convex Optimization Tools

In many applications, we can write an optimization problem in a convex form. Then we can use several software packages for convex optimization to efficiently solve these problems. These convex optimization engines include:

- MATLAB-based: CVX, SeDuMi, Matlab Optimization Toolbox (linprog, quadprog)
- Machine Learning: Weka (Java)
- libraries: CVXOPT (Python), GLPK (C), COIN-OR (C)
- SVMs: LIBSVM, SVM-light
- commerical packages: CPLEX, MOSEK

References

The source of this review are the following:

- Boyd, S. and Vandenberghe, L. (2004). Convex Optimization. Cambridge University Press.
- Course notes from CMU's 10-701.
- Course notes from Stanford's CS229, and CS224w
- Course notes from UCI's CS273a