Optimization & Linear Algebra Review

MInDS @ Mines

Linear algebra provides the notation of data representation in machine learning and makes it simple to represent distances in mathematical notation. As we often want to minimize distances, we use matrix norms for a concise representation of them. Optimization methods are then used to solve the minimization problem whether constrained or unconstrained.

Linear Algebra

In machine learning, we represent a model as its objective to minimize or maximize[1] by learning some model parameters. Machine learning is therefore essentially *model parameter estimation*. The objective function of a model is a distance measurement between some properties of the data or some function of the data. We therefore need a concise notation or representation of measurements which in this case is linear algebra. Measurements are with respect to the space they are in:

[1] Maximizing a function, $f(x)$ is equivalent to minimizing $-f(x)$. For consistency we often use the minimization version of objective functions.

- **Scalars** are real numbers. $x \in \mathbb{R}$.
- **Vectors** generalize scalars in d dimensions. $\mathbf{x} \in \mathbb{R}^d$.
- **Matrices** generalize vectors in $m \times n$ dimensions. $\mathbf{X} \in \mathbb{R}^{m\times n}$.
- **Tensors** generalize matrices in any dimensions. $\mathcal{X} \in \mathbb{R}^{a\times b\times c\times \cdots}$.

Throughout this course, we use a commonly found notation of lowercase letters for scalars, bold lowercase letters for vectors, bold uppercase letters for matrices and calligraphic uppercase letters for tensors.

When taking the product of various measurements, we must ensure that the inner dimensions align. A product of a measurement in $a \times b$ with a measurement in $b \times c$ results in a measurement in $a \times c$.

$$\begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i \tag{1}$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix} = \begin{bmatrix} x_1y_1 & \dots & x_1y_n \\ \vdots & \ddots & \vdots \\ x_my_1 & \dots & x_my_n \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} \begin{bmatrix} y_1 & y_2 & y_3 \\ y_4 & y_5 & y_6 \end{bmatrix} = \begin{bmatrix} x_1 \times y_1 + x_2 \times y_4 & x_1 \times y_2 + x_2 \times y_5 & x_1 \times y_3 + x_2 \times y_6 \\ x_3 \times y_1 + x_4 \times y_4 & x_3 \times y_2 + x_4 \times y_5 & x_3 \times y_3 + x_4 \times y_6 \end{bmatrix} \tag{3}$$

This shows how different measurements interact but now let's get back to the original goal of representing a distance. A **norm** is a function $f(\mathbf{x})$ that assigns a strictly positive length or size to each vector in a vector space. A norm is non-negative ($f(\mathbf{x}) \geq 0$), definite ($f(\mathbf{x}) = 0$ if and only if $\mathbf{x} = 0$), homogenous ($f(t\mathsf{x}) = |t|f(\mathsf{x})$), and follows the triangle inequality ($f(\mathsf{x}+\mathsf{y}) \leq f(\mathsf{x}) + f(\mathsf{y})$).

The ℓ_p norm of a vector, x, where $p \geq 1$, is denoted as,

$$||\mathsf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}} \tag{4}$$

$$||\mathsf{x}||_1 = \sum_{i}^{n} |x_i|$$

$$||\mathsf{x}||_2 = \sqrt{\sum_{i}^{n} x_i^2}$$

$$||\mathsf{X}||_{2,2} = ||\mathsf{X}||_F = \sqrt{\sum_{ij} x_{ij}^2} = \sqrt{\mathsf{tr}(\mathsf{X}\mathsf{X}^T)}$$

The $\ell_{p,q}$ norm of a matrix, $\mathsf{X} \in \mathbb{R}^{n\times m}$, where $p, q \geq 1$, is denoted as,

$$||\mathsf{X}||_{p,q} = \left(\sum_{i=1}^{n}\left(\sum_{j=1}^{m} |x_{ij}|^p\right)^{\frac{q}{p}}\right)^{\frac{1}{q}} = \left(\sum_{i=1}^{n} ||\mathsf{x}^i||_p^q\right)^{\frac{1}{q}} \tag{5}$$

The $\ell_{p,q}$ norm of a matrix can be thought of as calculating the ℓ_p norm of the row vectors followed by the ℓ_q norm of the resulting vector.

Matrix Properties and Operations

In this section, we list a variety of important properties and operations about matrices. This section is dense with definitions that you should be aware of.

A symmetric matrix S is one where $s_{ij} = s_{ji}$. Symmetric matrices have a special operation called **trace** which is the sum of the diagonal of that matrix represented as, $\mathsf{tr}(\mathsf{S}) = \sum_{i=1}^{n} s_{ii}$.

The Identity matrix, I is a special version of a diagonal matrix that returns the same matrix when multiplied by another.

$$\mathsf{I}_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{6}$$

$$\mathsf{D}_{ij} = \begin{cases} d_i & i = j \\ 0 & i \neq j \end{cases} \tag{7}$$

The rank of a matrix is the maximum number of linearly independent column vectors or linearly independent row vectors. The maximum rank of a matrix is therefore less than the minimum of its dimensions, $\mathsf{rank}(\mathsf{X}) \leq \min(n, m)$ where $\mathsf{X} \in \mathbb{R}^{n\times m}$.

Another useful operation on matrices is **transpose**. The transpose of matrix is represented as X^T. If $\mathsf{Y} = \mathsf{X}^T$, then $y_{ij} = x_{ji}$. Note that if a matrix is in $a \times b$ dimensions, its transpose will be in $b \times a$ dimensions. Symmetric matrices have the special property that $\mathsf{S}^T = \mathsf{S}$.

The inverse X^{-1} of a matrix $\mathsf{X} \in \mathbb{R}^{n\times n}$ is defined such that $\mathsf{A}^{-1}\mathsf{A} = \mathsf{A}\mathsf{A}^{-1} = \mathsf{I}$. A matrix is **invertible** if its inverse exists and **singular** otherwise.

Eigenvalues λ and eigenvectors v of a matrix X satisfy, $\mathsf{X}\mathsf{v}_i = \lambda_i \mathsf{v}_i$. For a symmetric positive matrix S, $\mathsf{eig}(\mathsf{S}^T\mathsf{S}) = \mathsf{eig}(\mathsf{S}\mathsf{S}^T) = \mathsf{eig}(\mathsf{S}) \circ \mathsf{eig}(\mathsf{S})$.

A good explainer on eigenvalues and eigenvectors is available at `http://setosa.io/ev/eigenvectors-and-eigenvalues/`.

Two vectors, $\mathsf{x} \in \mathbb{R}^n, \mathsf{y} \in \mathbb{R}^n$ are **orthogonal** if $\mathsf{x}^T\mathsf{y} = 0$. A matrix, U, is orthogonal if every pair of its column vectors are orthogonal, $\mathsf{U}^T\mathsf{U} = \mathsf{I}$.

Any matrix can be decomposed into the product of several matrices. If the matrix $\mathsf{X} \in \mathbb{R}^{n\times m}$ is not symmetric we have,

$$\mathsf{X} = \mathsf{U}\boldsymbol{\Sigma}\mathsf{V}^T, \tag{8}$$

where $\mathsf{U} \in \mathbb{R}^{n\times n}$ is the eigenvectors of $\mathsf{X}\mathsf{X}^T$, $\mathsf{V} \in \mathbb{R}^{m\times m}$ is the eigenvectors of $\mathsf{X}^T\mathsf{X}$, and $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times m}$ is a diagonal of the $\sqrt{\mathsf{eig}(\mathsf{X}\mathsf{X}^T)}$. This is called **singular value decomposition** (SVD) and $\boldsymbol{\Sigma}$ is a diagonal of X's singular values. A special case of SVD exists when the matrix is symmetric called **eigenvalue decomposition** where,

$$\mathsf{X} = \mathsf{V}\boldsymbol{\Lambda}\mathsf{V}^T, \tag{9}$$

where $\mathsf{V} \in \mathbb{R}^{n\times n}$ is the eigenvectors of X, and $\boldsymbol{\Lambda}$ is a diagonal of the eigenvalues of X.

Matrix Calculus

When solving convex optimization problems, we often set the derivative of a function to 0 and find the values that make it so. This approach determines the minimization or maximization of the problem. The same applies to functions of matrices and so it is important to be aware of some common derivatives of matrices.

Chapter 2 of the matrix cookbook (available at `http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=3274`) offers a large collection of useful derivatives commonly found in most function definitions. We mainly focus on derivatives of traces and norms when considering most formulations in machine learning.

The one important definition to be aware of aside from the derivatives that you will want to simply lookup is the Hessian matrix. The Hessian matrix is a square matrix of the second order partial derivatives of a function.

$$H(f) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}. \tag{10}$$

Convex Optimization

Now that we've covered the notation to express various measurements, operations, and distances, we can begin to formulate objective functions that we care about. First, let's formalize the presentation of an optimization problem. An optimization problem contains the following; whether it's a minimization or maximization problem, the variable(s) to change, the objective function to evaluate, and its subject to constraints - equalities and inequalities. An example objective function is,

$$\begin{aligned} \min_{\mathsf{w}}\; & f(\mathsf{w}) = \frac{1}{2}\mathsf{w}^T\mathsf{X}\mathsf{w} + \mathsf{c}^T\mathsf{w} \\ s.t.\quad & \mathsf{A}\mathsf{w} \leq \mathsf{b} \qquad \leftarrow \text{ inequality constraint} \\ & \mathsf{E}\mathsf{w} = \mathsf{d} \qquad \leftarrow \text{ equality constraint} \end{aligned} \tag{11}$$

After presenting a problem or formulation of a model, we use optimization methods to find the best parameters of the model. Generally, we like the problems we present to be convex so that it is easier to solve. A set, $\mathcal{S}$, is convex, if for any $\theta, \theta' \in \mathcal{S}$, $\lambda\theta + (1-\lambda)\theta' \in \mathcal{S}, \forall\lambda \in [0,1]$. A function, $f(\theta)$ is convex if it is defined on a convex set, and if any $\theta, \theta' \in \mathcal{S}$ and $\forall\lambda \in [0,1]$, $f(\lambda\theta + (1-\lambda)\theta') \leq f(\lambda\theta) + f((1-\lambda)\theta')$.

Based on the definitions of convex problems, you might notice that they have some properties that make it easier to solve. First, there is one global optimum. Second, we can reach the global optimum by continuing along the gradient of the function. This leads us to our first solution algorithm.

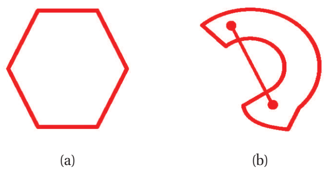

Figure 1: Two example sets, with a) showing a convex set and b) showing a non-convex set.

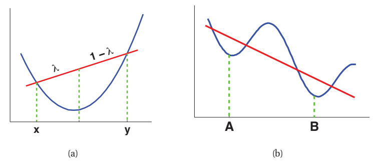

Figure 2: Two example functions, with a) showing a convex function and b) showing a non-convex function with a local minimum and global minimum.

Gradient Descent

Gradient descent is a commonly used and referenced solution algorithm to optimization problems due to its simplicity. The idea behind gradient descent is to 1) select a valid point in the solution space, 2) calculate the gradient at that point, and then 3) update the point by some amount by going down the gradient. In mathematical terms we represent the gradient as ∇, the learning rate[2], or the portion of the gradient to determine a step size, as γ, and the gradient descent updates as,

[2] The selected learning rate can heavily influence the speed of convergence and whether it happens. A large learning rate can overshoot past the optimal value, a small learning rate can take a very long time to reach optimal.

$$\mathsf{x}_{n+1} = \mathsf{x}_n - \gamma_n \nabla f(\mathsf{x}_n). \tag{12}$$

Newton's Method

While gradient descent focuses on the first order derivative of the objective, Newton's method adds some additional information from the second order derivative. It is, $\mathsf{x}_{n+1} = \mathsf{x}_n - \frac{f'(\mathsf{x}_n)}{f''(\mathsf{x}_n)}$ which in matrix form is,

$$\mathsf{x}_{n+1} = \mathsf{x}_n - \left[H(f(\mathsf{x}_n))\right]^{-1} \nabla f(\mathsf{x}_n), \tag{13}$$

where $H(f(\mathsf{x}_n))$ is the Hessian matrix of f. We may also add a learning rate to the step size for more granular control similar to gradient descent.

Constrained Problems

So far, we presented solutions to unconstrained objective functions. Objective constraints can add complexity to the problem and so we would much prefer an unconstrained problem. The way we deal with constrained problems is to convert them to unconstrained problems.

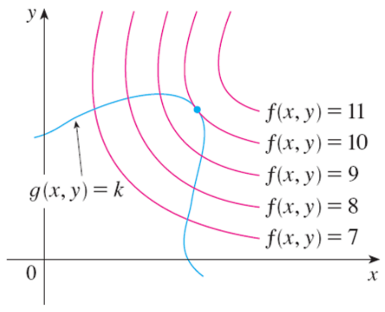

Figure 3: An illustration of a constrained problem and finding the optimal objective value.

One such way is to use a Lagrange multiplier, often denoted as λ and a Lagrangian function. The way a Lagrangian function works is by adding a variable to change in the optimization problem, the Lagrange multiplier, and moving the constraint to the objective function itself. We can convert,

$$\begin{aligned} &\min_{\mathsf{x}} f(\mathsf{x}), \\ &s.t. \quad g(\mathsf{x}) = 0, \end{aligned} \tag{14}$$

to

$$\min_{\mathsf{x},\lambda} \; f(\mathsf{x}) + \lambda g(\mathsf{x}). \tag{15}$$

The solution to the objective function would exist where the derivative with respect to the changing variables is equal to zero. With respect to λ, we get $g(\mathsf{x}) = 0$ thereby maintaining the constraint. The function $\mathcal{J}(\mathsf{x}, \lambda) = f(\mathsf{x}) + \lambda g(\mathsf{x})$ is called the Lagrangian function.

To handle inequality constraints we follow the same approach but we have to ensure what are called the Karush-Kuhn-Tucker (KKT) conditions.[3] The KKT conditions state that $g(\mathsf{x}) \leq 0$, $\lambda \leq 0$, and $\lambda g(\mathsf{x}) \leq 0$.

[3] This applies to ≤ 0 constraints. ≥ 0 constraints can be converted to ≤ 0 by multiplying both sides by -1.