EE 127 Notes

Kanyes Thaker

Spring 2019

Optimization Models

Optimization refers to a branch of applied mathematics concerned with the minimization or maximization of a certain function, possibly under constraints. This course covers linear algebra, convex optimization problems, and duality. This guide is based on **Optimization Models and Applications** by Laurent Al Ghaoui, as well as lectures by Gireeja Ranade and Alexander Bayen. This course is extremeley rigorous and is not by any means an introduction to linear algebra. It is not a replacement of lectures, discussions, or homework, but it should be a good enough refresher to prepare you for exams as a high level overview of the course. Have fun :)

Contents

1 Linear Algebra

1.1 Vectors

A vector is a collection of numbers arranged in a column or a row, and can be thought of as point in space, or as providing a direction. The **scalar product** gives us information about the length of a vector, and we can also use it to find information about the angle between two vectors. In this way, we can think of vectors as **linear functions**.

1.1.1 Basics

Say we have a collection of n numbers in $\mathbb{R}$, labeled $x_1, ..., x_n$. We can represent these numbers as a single **point** in n-dimensional space. We usually write all the information about this point (the n numbers) in a column:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^\top$$

Independence

A set of vectors $\{\mathbf{x}_1, ..., \mathbf{x}_n\} \in \mathbb{R}^n$ is **independent** if and only if the following condition:

$$\left\{ \lambda \in \mathbb{R}^n : \sum_{i=1}^{n} \lambda_i \mathbf{x}_i = 0 \right\} \implies \lambda = \mathbf{0}$$

This means that no vector in that set can be expressed as a **linear combination** of the other vectors in that set.

A **subspace** of $\mathbb{R}^n$ is a subset that is closed under addition and scalar multiplication. All subspaces must pass through the origin (since 0 is a scalar). An important property in linear algebra (proved later on) is that a subspace $\mathcal{S}$ can be represented as the **span** of a set of vectors $\{\mathbf{x}_1, ..., \mathbf{x}_m\}, \mathbf{x}_i \in \mathbb{R}^n$. In other words, it is of the form

$$\mathcal{S} = \text{span}(\mathbf{x}_1, ..., \mathbf{x}_m) := \left\{ \sum_{i=1}^{m} \lambda_i \mathbf{x}_i : \lambda \in \mathbb{R}^m \right\}$$

An **affine set** is a **translated subspace**. It is a subspace that has been shifted so that it no longer necessarily passes through the origin. We can

define an affine set $\mathcal{A}$ based on our definition for a subspace:

$$\mathcal{A} = \left\{ \mathbf{x}_0 + \sum_{i=1}^{m} \lambda_i \mathbf{x}_i : \lambda \in \mathbb{R}^m \right\}$$

In shorthand notation, we can simply write this as $\mathcal{A} = \mathbf{x}_0 + \mathcal{S}$.

A **basis** in $\mathbb{R}^n$ is a set of n independent vectors. If the set of vectors $\mathbf{u}_1, ..., \mathbf{u}_n$ form a basis, we can describe any vector in $\mathbb{R}^n$ as a linear combination of the basis vectors, where $\mathbf{x} = \sum \lambda_i \mathbf{u}_i$. The **standard basis** in $\mathbb{R}^n$ consists of vectors $\mathbf{e}_1, ..., \mathbf{e}_n$ where $\mathbf{e}_i$ has 1 as its ith element and 0 everywhere else.

The basis of a **subspace** is any independent set of vectors whose span is $\mathcal{S}$. The number of vectors in the basis is independent of the choice of basis itself. For example, we only need 2 vectors to define an origin-containing plane in $\mathbb{R}^3$. The number of vectors we need to define a subspace is the **dimension** of $\mathcal{S}$. We can define the dimension of an affine space in much the same way, since it's just a translated subspace.

1.1.2 Scalar Product, Norms, and Angles

Scalar Product

The **scalar product**, **inner product**, or **dot product** between two vectors $\mathbf{x}$ and $\mathbf{y}$ is a scalar $\mathbf{x}^\top \mathbf{y}$ and is defined as $\mathbf{x}^\top \mathbf{y} = \sum x_i y_i$. We also denote the scalar product as $\langle \mathbf{x}, \mathbf{y} \rangle$. Two vectors are **orthogonal** if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$.

For the purposes of this course, we largely focus on 3 **vector norms** for vectors in $\mathbb{R}^n$.

Norms

1. ℓ_2 Norm: Also known as the **Euclidian Norm**, denoted $\|\mathbf{x}\|_2$, the ℓ_2 norm defined as

$$\|\mathbf{x}\|_2 = \sqrt{\sum x_i^2} = \sqrt{\mathbf{x}^\top \mathbf{x}},$$

   correponds to the traditional straight-line definition of distance. The set of points with equal ℓ_2 norms in 2 dimensions is a circle, in 3 dimensions is a sphere, and in higher dimensions is a hypersphere.

2. ℓ_1 Norm: Denoted as $\|\mathbf{x}\|_1$, and defined as

$$\|\mathbf{x}\|_1 = \sum |x_i|,$$

   the ℓ_2 norm corresponds to the distance traveled on a rectangular grid from one point to another.

3. ℓ_∞ Norm: Also called the **supremum norm** and defined as

$$\|\mathbf{x}\|_\infty = \max\{x_i\},$$

   the ℓ_∞ norm can be thought of as the maximum distance moved in a particular direction.

Cauchy-Schwarz Inequality binds the scalar products of two vectors in terms of their Euclidian norms. The proof for this inequality is widely covered in other courses such as EE16A, and Math54 and will not be reproduced here.

Cauchy-Schwarz

For any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

$$\mathbf{x}^\top \mathbf{y} \leq \|\mathbf{x}\|_2 \cdot \|\mathbf{y}\|_2.$$

The only time this inequality is an equality is if the two vectors are collinear.

Another definition of the scalar product is that $\mathbf{x}^\top \mathbf{y} = \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos\theta$. This has also been extensively covered in EE16A, Math54, and Math53, but we can easily use Cauchy-Schwarz to show that this is valid. The angle between two vectors is incredibly useful for determining the similarity or closeness of

two vectors.

1.1.3 Projection on a Line

Define a line $\{\mathbf{x}_0 + t\mathbf{u} : t \in \mathbb{R}\}$ where $\mathbf{u} \in \mathbb{R}^n$ is the direction of the line. Then define the **projection** of $\mathbf{x}$ onto the line is the vector $\mathbf{z}$ on the line such that the ℓ_2 norm between $\mathbf{x}$ and $\mathbf{z}$ is minimized (i.e. we use the previous definition of "closeness" to find the vector on the line that is "closest" to $\mathbf{x}$):

$$\min_t \|\mathbf{x} - \mathbf{x}_0 - t\mathbf{u}\|_2$$

This problem is part of the class of optimization problems known as **least squares**.

We now want to find a **closed-form expression** for this problem. We can expand the objective function into

$$(t - \mathbf{u}^\top(\mathbf{x} - \mathbf{x}_0))^2 + c.$$

The optimal t is then

$$\hat{t} = \mathbf{u}^\top(\mathbf{x} - \mathbf{x}_0).$$

We can then get $\mathbf{z} = \mathbf{x}_0 + \hat{t}\mathbf{u} = \mathbf{x}_0 + \mathbf{u}^\top(\mathbf{x} - \mathbf{x}_0)\mathbf{u}$. If $\|\mathbf{u}\|_2 \neq 1$, we can normalize this result to get the standard projection:

$$\mathbf{z} = \mathbf{x}_0 + \frac{\mathbf{u}^\top(\mathbf{x} - \mathbf{x}_0)}{\|\mathbf{u}\|_2}\mathbf{u}.$$

This allows us to generalize the scalar product and say that the scalar product $\mathbf{u}^\top\mathbf{x}$ is the component of $\mathbf{x}$ along the normalized direction $\mathbf{u}/\|\mathbf{u}\|_2$. Understand why this makes sense; if the algebra is troublesome, it's helpful to try and prove some of the key principles here yourself to convince yourself that the math is correct.

1.1.4 Orthogonalization and the Gram-Schmidt Procedure

Orthonormal

A basis $(\mathbf{u}_i)_{i=1}^n$ is **orthogonal** if $\mathbf{u}_i^\top\mathbf{u}_{j\neq i} = 0$. If $\|\mathbf{u}_i\|_2 = 1$ then it is (more strongly defined as) **orthonormal**.

Orthogonalization refers to the procedure of finding $\{\mathbf{q}_1, ..., \mathbf{q}_n\} \in \mathbb{R}^n$ from a given $\{\mathbf{a}_1, ..., \mathbf{a}_k\} \in \mathbb{R}^n$ such that $\mathcal{S} := \text{span}\{\mathbf{a}_1, ..., \mathbf{a}_k\}) = \{\mathbf{q}_1, ..., \mathbf{q}_r\}$ for

$r = \dim \mathcal{S}$, and $\{\mathbf{q}_1, ..., \mathbf{q}_r\}$ obeys the properties above, i.e. $\mathbf{q}$ forms an orthonormal basis for $\mathbf{a}$.

Consider a simple example: project $\mathbf{a}$ along the line $L(\mathbf{q}) = \{t\mathbf{q} : t \in \mathbb{R}\}$ where $\|\mathbf{q}\|_2 = 1$. The projection of $\mathbf{a}$ onto L is then simply $\hat{\mathbf{q}} = (\mathbf{q}^\top \mathbf{a})\mathbf{q}$. Note that we can write $\mathbf{a}$ as the sum of two vectors: $(\mathbf{a} - \hat{\mathbf{a}})$ and $\hat{\mathbf{a}}$. Note that these two vectors are orthogonal (take the scalar product to see why). In other words,

$$\mathbf{a} = (\mathbf{a} - \hat{\mathbf{a}}) + \hat{\mathbf{a}} = (\mathbf{a} - (\mathbf{q}^\top \mathbf{a})\mathbf{q}) + (\mathbf{q}^\top \mathbf{a})\mathbf{q}, (\mathbf{a} - (\mathbf{q}^\top \mathbf{a})\mathbf{q}) \perp (\mathbf{q}^\top \mathbf{a})\mathbf{q}.$$

Interpret this as removing the component of $\mathbf{a}$ along $\mathbf{q}$.

The **Gram-Schmidt Procedure** is a famous orthogonalization algorithm. We orthogonalize each vector with respect to the previous one and then normalize the result.

For Independent Vectors:
`procedure gram-schmidt():`
    `Set` $\tilde{\mathbf{q}}_1 = \mathbf{a}_1$
    `Normalize: Set` $\mathbf{q}_1 = \tilde{\mathbf{q}}_1 / \|\tilde{\mathbf{q}}\|_2$
    `Remove component of` $\mathbf{q}_1$ `in` $\mathbf{a}_2$`: Set` $\tilde{\mathbf{q}}_2 = \mathbf{a}_2 - (\mathbf{q}_1^\top \mathbf{a}_2)\mathbf{q}_1$
    `continue`

In the case when some vectors are dependent, we simply skip over the process for any $\tilde{\mathbf{q}}_i = 0$.

Hyperplanes

A **hyperplane** is a set described by an equality, i.e. $\mathcal{H} = \{\mathbf{x} : \mathbf{a}^\top \mathbf{x} = \mathbf{b}\}$, with appropriate assumptions (i.e. $\mathbf{a} \neq 0, \mathbf{a} \in \mathbb{R}^n$). If $\mathbf{b} = 0$, then $\mathbf{x}$ is simply the set orthogonal to $\mathbf{a}$. If $\mathbf{b} \neq 0$, then we are referring to the same plane translated in the direction of $\mathbf{a}$. Note that by definition, if $\mathbf{x}_0 \in \mathcal{H}$ and $\mathbf{x} \in \mathcal{H}$ then $\mathbf{a}^\top \mathbf{x}_0 - \mathbf{a}^\top \mathbf{x} = \mathbf{b} - \mathbf{b} = 0$, i.e. $\mathcal{H} = \{\mathbf{x} : \mathbf{a}^\top (\mathbf{x}_0 - \mathbf{x}) = 0\}$.

Hyperplanes are useful because they have dimension $n - 1$ (proof omitted) and therefore **separate** the space into two regions (called **half-spaces**). We can then get information about the space based on what points lie on what side of the hyperplane. $\hat{\mathbf{x}} = \mathbf{ba}$ is the projection of the origin onto $\mathcal{H}$. In

other words, since $\mathcal{H}$ is orthogonal to $\mathbf{a}$, the vector from the origin to $\mathcal{H}$ that minimizes the Euclidian norm is $\mathbf{ba}$.

Half-Spaces

A **half-space** is defined by single equality that tells us which side of the hyperplane we are on. It is a space of the form $\mathcal{H} = \{\mathbf{x} : \mathbf{a}^\top \mathbf{x} \geq \mathbf{b}\}$. The angle between $\mathbf{x} - \mathbf{x}_0$ and $\mathbf{a}$ is acute, where $\mathbf{x}_0$ is the projection of the origin onto the boundary of the half-space.

1.1.5 Linear Functions and Maps

Linear Functions

Linear functions preserve the additive and scaling qualities of their arguments. Affine functions are "linear functions plus a constant." More formally, a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is linear if and only if

$$\forall \mathbf{x} \in \mathbb{R}^n, \alpha \in \mathbb{R} \implies$$

$$f(\alpha \mathbf{x}) = \alpha f(\mathbf{x}) \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n, f(\mathbf{x}_1 + \mathbf{x}_2) = f(\mathbf{x}_1) + f(\mathbf{x}_2).$$

A function is affine if and only if $\tilde{f} : \mathbb{R}^n \mapsto \mathbb{R} = f(\mathbf{x}) - f(0)$ is linear. A function f is also only affine if it can be expressed as $f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} + \mathbf{b}$.

The **gradient** of an affine function f at a point $\mathbf{x}$ ($\nabla f(\mathbf{x})$) is a vector of first derivatives with respect to $x_1, ..., x_n$.

$$\nabla f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} + \mathbf{b} = \mathbf{a}.$$

Say we want to approximate a non-linear function with a linear or affine one. Consider the one dimensional case, with $f : \mathbb{R} \mapsto \mathbb{R}$. Then we can evaluate a point $\mathbf{x}$ near $\mathbf{x}_0$ as

$$f(\mathbf{x}) \simeq l(\mathbf{x}) := f(\mathbf{x}_0) + f'(\mathbf{x} - \mathbf{x}_0)$$

For a multidimensional case, we consider a very similar construction. We must approximate f through an affine function l, meaning l must be of the form $\mathbf{a}^\top \mathbf{x} + \mathbf{b}$. We then get the following approximation:

$$f(\mathbf{x}) \approx l(\mathbf{x}) := f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

1.2 Matrices

Matrices are collections of vectors of same size, organized in a rectangular array. Via the matrix-vector product, we can interpret matrices as linear maps (vector-valued functions), which act from an input space to an output space, and preserve addition and scaling of the inputs.

1.2.1 Basics

Matrices are horizontally concatenated column vectors of equal dimension. If we have a set of column vectors $\mathbf{a}_1, ..., \mathbf{a}_n \in \mathbb{R}^m$, we can create the $m \times n$ matrix $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & ... & \mathbf{a}_n \end{bmatrix}$. We transpose a matrix by inverting the row and column of an element – i.e. $\mathbf{A}_{ij}$ becomes $\mathbf{A}_{ji}$ in $\mathbf{A}^\top$. We can also represent a matrix as a vertical stack of **transposed** column vectors.

1.2.2 Matrix Products

The Matrix-Vector Product

The **matrix-vector product** of an $m \times n$ matrix $\mathbf{A}$ and an n-vector $\mathbf{x}$, denoted $\mathbf{Ax}$ has definition

$$(\mathbf{Ax})_i = \sum_{j=1}^{n} A_{ij} x_j, i = 1, ..., m$$

$$\mathbf{Ax} = \sum_{i=1}^{n} x_i \mathbf{a}_i = \begin{bmatrix} \mathbf{a}_1^\top \mathbf{x} \\ \vdots \\ \mathbf{a}_m^\top \mathbf{x} \end{bmatrix}$$

Matrix-Matrix Products

We can extend the definition to matrix-matrix products. If $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $\mathbf{B} \in \mathbb{R}^{n \times m}$, then

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}.$$

$$\mathbf{AB} = \begin{bmatrix} \mathbf{Ab}_1 & ... & \mathbf{Ab}_n \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^\top \mathbf{B} \\ \vdots \\ \mathbf{a}_m^\top \mathbf{B} \end{bmatrix}$$

Matrix algebra generalizes to blocks (i.e. splitting up matrices into concatenated, smaller matrices). In this way $\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}\begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{bmatrix}^\top = \mathbf{A}_1\mathbf{B}_1+\mathbf{A}_2\mathbf{B}_2$. Note that if the transposes were in different orders we'd have a completely different product (an **outer product**).

$$\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}^\top\begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1\mathbf{B}_1 & \mathbf{A}_1\mathbf{B}_2\\ \mathbf{A}_2\mathbf{B}_1 & \mathbf{A}_2\mathbf{B}_2\end{bmatrix}$$

Trace

The **trace** of a square matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathrm{tr}(\mathbf{A})$, is the sum of the diagonal elements of the matrix.

$$\mathbf{tr}(\mathbf{A}) = \sum_{i=1}^{n} A_{ii}$$

$$\mathbf{tr}(\mathbf{A}) = \mathbf{tr}(\mathbf{A})^\top, \mathbf{tr}(\mathbf{AB}) = \mathbf{tr}(\mathbf{BA})$$

We can use the trace to define the scalar product between two matrices.

$$\langle \mathbf{A}, \mathbf{B}\rangle = \mathbf{tr}(\mathbf{A}^\top\mathbf{B}) = \mathbf{tr}(\mathbf{A}^\top\mathbf{B})^\top = \mathbf{tr}(\mathbf{B}^\top\mathbf{A}) = \langle \mathbf{B}, \mathbf{A}\rangle$$

.

1.2.3 Special Matrices

Identity

The $n \times n$ identity matrix (often denoted $\mathbf{I}_n$, or simply $\mathbf{I}$, if context allows), has ones on its diagonal and zeros elsewhere. It is square, diagonal and symmetric. This matrix satisfies $\mathbf{A} \cdot \mathbf{I}_n = \mathbf{A}$ for every matrix $\mathbf{A}$ with columns, and $\mathbf{I}_n \cdot \mathbf{B} = \mathbf{B}$ for every matrix $\mathbf{B}$ with n rows.

Diagonal

Diagonal matrices are square matrices with $\mathbf{A}_{ij} = 0$ when $i \neq j$. A diagonal $n \times n$ matrix $\mathbf{A}$ can be denoted as $\mathbf{A} = \mathrm{diag}(\mathbf{a})$, with $a \in \mathbb{R}^n$ being the vector containing the elements on the diagonal.

Symmetric

A matrix is symmetric if $A_{ij} = A_{ji}$. They will be covered in more depth later.

Triangular

A matrix $A \in \mathbb{R}^{m \times n}$ is **upper triangular** if $A_{ij} = 0$ for $i > j$. A matrix is **lower triangular** if its transpose is upper triangular.

Orthogonal

An **orthogonal** or **unitary** matrix is one whose columns form an orthonormal basis. Using the properties of orthonormal vectors, we see that this means that $\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$. Orthogonal matrices correspond to rotations or reflections across a specific direction. They preserve length and angles (can be easily proved from the definition of inner products).

Dyads

A matrix is a **dyad** if it is of the form $\mathbf{A} = \mathbf{u}\mathbf{v}^\top$. $\mathbf{A}\mathbf{x} = (\mathbf{u}\mathbf{v}^\top)\mathbf{x} = (\mathbf{v}^\top\mathbf{x})\mathbf{u}$. The output is always a scaling of $\mathbf{u}$ dependant on $\mathbf{v}$. Dyads can also be **normalized** (so that we can capture the scaling factor in a single coefficient).

1.2.4 The QR Decomposition of Matrices

The goal of **QR** decomposition is to factor a matrix as a product of two matrices. It is extremely similar to the Gram-Schmidt procedure applied to the columns of the matrix. Consider $\mathbf{A} \in \mathbb{R}^{m \times n}$, with n columns $\mathbf{a}_i \in \mathbb{R}^m$.

First assume that the columns of $\mathbf{A}$ are linearly independent. Then each step of the G-S procedure can be written as

$$\mathbf{a}_i = (\mathbf{a}_i^\top \mathbf{q}_1)\mathbf{q}_1 + ... + (\mathbf{a}_i^\top \mathbf{q}_{i-1})\mathbf{q}_{i-1} + \|\tilde{\mathbf{q}}_i\|_2 \mathbf{q}_i, i = 1, ..., n$$

Now let $r_{ji} = \mathbf{a}_i^\top \mathbf{q}_j$, $1 \leq j \leq i-1$, and $r_{ii} = \|\tilde{\mathbf{q}}_{ii}\|_2$. Since each $\mathbf{q}_i$ is unit length and normalized, the matrix $\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 & ... & \mathbf{q}_n \end{bmatrix}$ is unitary. Then the

QR decomposition of $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{QR}, \mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 & ... & \mathbf{q}_n \end{bmatrix}, \mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & & 0 & r_{nn} \end{bmatrix}$$

where $\mathbf{Q} \in \mathbb{R}^{m \times n}$, where $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}$, and $\mathbf{R} \in \mathbb{R}^{n \times n}$ is upper triangular.

1.2.5 Matrix Inverses

A square $n \times n$ is only invertible if it is **full rank**, i.e. all of its columns are linearly independent. Equivalently, a matrix $\mathbf{A}$ is only invertible if $|\mathbf{A}| \neq 0$. For any invertible $\mathbf{A}$, $\exists\ \mathbf{B} \in \mathbb{R}^{n \times n} : \mathbf{BA} = \mathbf{AB} = \mathbf{I}_n$. We denote $\mathbf{B}$ as $\mathbf{A}^{-1}$. Using the **QR** decomposition of $\mathbf{A}$, $\mathbf{A}^{-1} = \mathbf{R}^{-1}\mathbf{Q}^\top$. Additionally, $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$.

A matrix is **full column rank** if its columns are independent ($m \geq n$). A matrix has full column rank iff $\exists \mathbf{B} : \mathbf{BA} = \mathbf{I}$ where $\mathbf{A} = \mathbf{Q} \begin{bmatrix} \mathbf{R}_1 & 0 \end{bmatrix}^\top$. Then $\mathbf{B} = \begin{bmatrix} \mathbf{R}_1^{-1} & 0 \end{bmatrix} \mathbf{Q}^\top$. We can also write

$$\mathbf{B} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top.$$

A matrix is **full row rank** if its rows are independent ($m \leq n$). A matrix has full row rank iff $\exists \mathbf{B} : \mathbf{AB} = \mathbf{I}$. If $\mathbf{A} = \begin{bmatrix} \mathbf{R}^\top & 0 \end{bmatrix} \mathbf{Q}^\top$ then $\mathbf{B} = \mathbf{Q} \begin{bmatrix} \mathbf{R}^{-1} & 0 \end{bmatrix}^\top$. We can also write

$$\mathbf{B} = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1}.$$

1.2.6 Matrix Norms

A matrix $\mathbf{A}$ can **induce** a linear map via the matrix-vector product, i.e. $f : \mathbf{x} \mapsto \mathbf{Ax}$. Say, however that we have a noisy input vector $(\mathbf{x} + \mathbf{v})$. Then our map will amplfy the error and leave us with an error in our output of $\mathbf{Av}$. We can **quantify** this error by capturing how $\|\mathbf{Av}\|$ varies in $\mathbf{v}$. Since scaling $\mathbf{v}$ scalse $\|\mathbf{Av}\|$, we can restrict $\|\mathbf{v}\| = 1$.

Frobenius Norm

Let's examine that the error vector can be any one of the standard basis vectors. Then the average squared error is

$$\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{A}\mathbf{e}_i\|_2^2 = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{a}_i\|_2^2$$

Here, $\mathbf{a}_i$ is the ith column of $\mathbf{A}$. This quantity is equal to $\frac{1}{n}\|\mathbf{A}\|_F^2$, where

$$\|\mathbf{A}\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{A}_{ij}^2} = \sqrt{\mathbf{tr}(\mathbf{A}^\top\mathbf{A})}$$

Intuitively, the Frobenius norm $\|\cdot\|_F$ is a matrix-equivalent to the 2-norm, as it is the Euclidian norm of the vector $\mathbf{x} \in \mathbb{R}^{nm}$ formed with the coefficients of $\mathbf{A}$. Computation of $\|\cdot\|_F$ is in $\mathrm{O}(nm)$.

Norms such as the LSV norm and peak norm are covered in the book, but aren't essential to know so I'm omitting them from these notes. Looking over their definitions is still highly encouraged, however.

1.3 The Fundamental Theorem of Linear Algebra

The solution set to $\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{y}, \mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{y} \in \mathbb{R}^m$ is (if it exists) an affine subspace, i.e. of the form $\mathbf{x} = \mathbf{x}_0 + \mathbf{L}$. We want to know if $\mathbf{x}$ exists at all; and if it does, find $\mathbf{x}_0$ and an orthonormal basis for $\mathbf{L}$.

Range

The **range** or **image** of $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a subset of $\mathbb{R}^m$. It describes all the vectors that can be attained by multiplying $\mathbf{A}$ by any arbitrary $\mathbf{x}$. The **range** is equivalent to the column span of $\mathbf{A}$.

$$\mathcal{R}(\mathbf{A}) := \{\mathbf{A}\mathbf{x} : \mathbf{x} \in \mathbb{R}^n\}$$

The **rank** is the dimension of the range. The rank is capped by the smallest dimension of $\mathbf{A}$, i.e. $\textbf{rank}(\mathbf{A}) \leq \min\{m, n\}$. $\mathbf{A}$ is **full rank** if $\textbf{rank}(\mathbf{A}) = \min\{m, n\}$.

Nullspace

Nullspace: The **nullspace** also known as the **kernel** of $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a subspace of $\mathbb{R}^n$. The nullspace is a measure of **ambiguity** in $\mathbf{A}$. If $\mathbf{z} \in \mathcal{N}(\mathbf{A})$, then $\mathbf{A}(\mathbf{x} + \mathbf{z}) = \mathbf{A}\mathbf{x} = \mathbf{y}$, so we cannot extract $\mathbf{x}$ from $\mathbf{y}$. In order for us to be able to completely determine $\mathbf{x}$ we require $\mathcal{N}(\mathbf{A}) = \{0\}$, known as the **singleton**.

$$\mathcal{N}(\mathbf{A}) := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = 0\}$$

The **nullity** of $\mathbf{A}$ is the dimension of of the nullspace.

$\mathbf{A}$ is **full row rank** or **onto** if $\mathcal{R}(\mathbf{A}) = \mathbb{R}^m$. An equivalent condition is that $\mathbf{A}\mathbf{A}^\top$ is invertible. $\mathbf{A}$ is **full column rank** or **one-to-one** if $\mathcal{N}(\mathbf{A}) = \{0\}$. An equivalent condition is that $\mathbf{A}^\top\mathbf{A}$ is invertible.

The Rank-Nullity Theorem

The rank and nullity of $\mathbf{A} \in \mathbb{R}^{m \times n}$ sum to its column dimension n, i.e. the nullity of $\mathbf{A} \in \mathbb{R}^{m \times n}$ is $n - r$ where $r = \textbf{rank}(\mathbf{A})$. Equivalently, $\textbf{rank}(\mathbf{A}) + \dim(\mathcal{N}(\mathbf{A})) = n$.

The Fundamental Theorem of Linear Algebra

The range of a matrix is the orthogonal complement of the nullspace of its transpose. That is, for $\mathbf{A} \in \mathbb{R}^{m \times n}$, the following holds:

$$\mathcal{N}(\mathbf{A}) \oplus \mathcal{R}(\mathbf{A}^\top) = \mathbb{R}^n$$

$$\mathcal{N}(\mathbf{A}^\top) \oplus \mathcal{R}(\mathbf{A}) = \mathbb{R}^n$$

1.4 Least Squares

The **ordinary least squares** (OLS) problem is defined as

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{y} \in \mathbb{R}^m$. Intuitively, the least squares problem describes how to fit a function to a set of data so that we minimize the error between our predicted output $\mathbf{A}\mathbf{x}$ and the observed output $\mathbf{y}$. We usually use least squares where $\mathbf{A}\mathbf{x} = \mathbf{y}$ has no direct **feasible** solution, and is instead used to find the smallest δ such that $\mathbf{A}\mathbf{x} = \mathbf{y} + \delta\mathbf{y}$ becomes feasible.

We can reduce the problem to a minimization problem on the rows of $\mathbf{A}$, i.e. rewrite it as

$$\min_{\mathbf{x}} \sum_{i=1}^{m} (\mathbf{y}_i - \mathbf{a}_i^\top \mathbf{x})^2.$$

This expression makes it expressly clear that we are determining how to fit each component of $\mathbf{y}$ with its corresponding input $\mathbf{a}_i$ using $\mathbf{x}$ as the coefficients of the linear map.

When $\mathbf{A}$ is full rank, the solution to the problem is unique. The global minimizer of $f = \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$ is where $\nabla_{\mathbf{x}} f = 0$, i.e. where

$$\nabla_{\mathbf{x}} (\mathbf{A}\mathbf{x} - \mathbf{y})^\top (\mathbf{A}\mathbf{x} - \mathbf{y}) = 0$$

Solving for $\mathbf{x}$ gives us the standard closed form OLS solution,

$$\hat{\mathbf{x}} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{y}.$$

In the case where $\mathbf{A}$ is not full rank, we can express any single solution $\mathbf{x}_0$ as $\mathbf{x}_0 + \mathcal{N}(\mathbf{A})$ for reasons mentioned in the previous section. Anything more in depth than that is beyond the scope of this course, and is omitted here. It is sometimes acceptable to leave the solution as $(\mathbf{A}^\top \mathbf{A})\mathbf{x} = \mathbf{A}^\top \mathbf{y}$.

1.5 Eigenvalue Decomposition for Symmetric Matrices

A matrix $\mathbf{A}$ is symmetric if $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{A}^\top = \mathbf{A}$. The set of symmetric matrices is denoted as

$$\mathbb{S}^n \subseteq \mathbb{R}^{n \times n}.$$

Quadratic Functions

A function $q : \mathbb{R}^n \mapsto \mathbb{R}$ is quadratic if

$$q(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{A}_{ij} \mathbf{x}_i \mathbf{x}_j + 2 \sum_{i=1}^{n} \mathbf{b}_i \mathbf{x}_i + c$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$, and $c \in \mathbb{R}$. When $\mathbf{b} = 0, c = 0$ we say that q is in **quadratic form**. The Hessian $\nabla^2_{\mathbf{x}} q(\mathbf{x})$ is constant.

There is a link between symmetric matrices and quadratic functions. In fact, we can see that we can decompose our above definition of q into the product of vectors and matrices:

$$q(\mathbf{x}) = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^\top & c \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \mathbf{x}^\top \mathbf{A} \mathbf{x} + 2\mathbf{b}^\top \mathbf{x} + c$$

where $\mathbf{A} \in \mathbb{S}^n$, $\mathbf{b} \in \mathbb{R}^n$, and $c \in \mathbb{R}$.

The scalar $\lambda \in \mathbb{R}$ is an **eigenvalue** of $\mathbf{A}$ if:

$$\exists \mathbf{u} \in \mathbb{R}^n \neq \mathbf{0} : \mathbf{A}\mathbf{u} = \lambda \mathbf{u}.$$

Here $\mathbf{u}$ is an **eigenvector** of $\mathbf{A}$. If the eigenvector is **normalized**, i.e. $\|\mathbf{u}\| = 1$, then $\mathbf{u}^\top \mathbf{A} \mathbf{u} = \lambda \mathbf{u}^\top \mathbf{u} = \lambda$. The eigenvalues of $\mathbf{A}$ are **characterized** by the equation

$$\det(\lambda \mathbf{I} - \mathbf{A}) = 0$$

The function $t \mapsto p(t) := \det(\lambda \mathbf{I} - \mathbf{A})$ is an n-degree polynomial called the **characteristic polynomial**. This is a concept that's covered pretty heavily in EE16A, EE16B, and Math54.

Spectral Theorem

Any symmetric matrix can be decomposed into the following symmetric eigenvalue decomposition:

$$\forall \mathbf{A} \in \mathbb{S}^n, \mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top, \mathbf{\Lambda} = \text{diag}(\lambda_1, ..., \lambda_n), \mathbf{U}^\top = \mathbf{U}^{-1}$$

More simply, any symmetric matrix can be decomposed into the product of a unitary matrix, its eigenvalue matrix, and the transpose of the first unitary matrix. Calculating $\mathbf{\Lambda}$ is in $\text{O}(n^3)$.

Rayleigh Quotients

We can find $\lambda_{\min}$ and $\lambda_{\max}$ of a symmetric matrix:

$$\lambda_{\min}(\mathbf{A}) = \min_{\mathbf{x}}\{\mathbf{x}^\top \mathbf{A}\mathbf{x} : \mathbf{x}^\top \mathbf{x} = 1\}, \lambda_{\max}(\mathbf{A}) = \max_{\mathbf{x}}\{\mathbf{x}^\top \mathbf{A}\mathbf{x} : \mathbf{x}^\top \mathbf{x} = 1\}$$

We call this form the **variational form**, because this is a way to describe the eigenvalues as the solutions to optimization problems (historically referred to as **variational** problems) of quadratic functions over the Euclidian $\mathbb{R}$-ball. In other words, we can rewrite these values as the minimum and maximum values of the **Rayleigh Quotient**,

$$\frac{\mathbf{x}^\top \mathbf{A}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}}$$

Positive Semi-Definite Matrices

The definition of a **positive semi-definite** (PSD, $\mathbf{A} \succeq 0$) matrix is as follows:

$$(\mathbf{A} \in \mathbb{R}^{n \times n} \subseteq \mathbb{S}^n) \succeq 0 \iff \mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0 \forall \mathbf{x} \in \mathbb{R}^n$$

A matrix is **positive definite** (PD, $\mathbf{A} \succ 0$) if $\mathbf{x}^\top \mathbf{A}\mathbf{x} = 0$ iff $\mathbf{x} = 0$.

$\mathbf{A}$ is only positive semidefinite if all of the eigenvalues of $\mathbf{A}$ are nonnegative. By definition, whether or not $\mathbf{A}$ is PSD depends only on the eigenvalues of $\mathbf{A}$, not on the eigenvectors. In fact, for all $\mathbf{B}$ with n rows, $\mathbf{B}^\top \mathbf{A}\mathbf{B}$ is PSD so long as $\mathbf{A}$ is.

Ellipsoids

An **ellipsoid** is any affine transform of the unit ball for the Euclidian norm:

$$\varepsilon = \{\hat{\mathbf{x}} + \mathbf{L}\mathbf{z} : \|\mathbf{z}\| = 1\}$$

Here, $\mathbf{L}$ is an invertible square matrix. Rearranging terms, we get

$$\varepsilon = \{\mathbf{x} : \|\mathbf{L}^{-1}(\mathbf{x}-\hat{\mathbf{x}})\|_2 \leq 1\} = \{\mathbf{x} : (\mathbf{x}-\hat{\mathbf{x}}^\top \mathbf{A}^{-1}(\mathbf{x}-\hat{\mathbf{x}}) \leq 1\}, \mathbf{A} = \mathbf{L}\mathbf{L}^\top \succ 0$$

1.5.1 Sample Covariance Matrices

For $\mathbf{z} \in \mathbb{R}^m$, the variance σ^2 shows how much the data varies around the mean $\hat{\mathbf{z}}$, i.e.

$$\hat{\mathbf{z}} := \frac{1}{m}\sum_{i=1}^{m} \mathbf{z}_i, \mathrm{Var}(\mathbf{z}) := \frac{1}{m}\sum_{i=1}^{m}(\mathbf{z}_i - \hat{\mathbf{z}})^2$$

Consider a data matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, with m data points represented by column vectors. To describe how much variance exists in the data set, we project along $\mathbf{u} \in \mathbb{R}^n$ to get the row vector $\mathbf{z} = \begin{bmatrix}\mathbf{u}^\top\mathbf{x}_1 & \dots & \mathbf{u}^\top\mathbf{x}_m\end{bmatrix} = \mathbf{u}^\top\mathbf{X}$. Then using our definitions above,

$$\hat{\mathbf{z}} = \mathbf{u}^\top\hat{\mathbf{x}}, \mathrm{Var}(\mathbf{z}) = \sigma^2(\mathbf{u}) := \frac{1}{m}\sum_{k=1}^{m}[\mathbf{u}^\top(\mathbf{x}_k - \hat{\mathbf{x}})^2] = \mathbf{u}^\top\boldsymbol{\Sigma}\mathbf{u}$$

Here, $\boldsymbol{\Sigma}$ is the PSD **sample covariance matrix**, defined as

$$\boldsymbol{\Sigma} = \frac{1}{m}\sum_{k=1}^{m}(\mathbf{x}_k - \hat{\mathbf{x}})(\mathbf{x}_k - \hat{\mathbf{x}})^\top$$

1.5.2 Principal Component Analysis

When we have a set of data that is represented in an extremeley high number of dimensions, it sometimes becomes beneficial to try and **reduce** the number of dimensions our data is represented in. In order to preserve the differences in our data points, we try to project along the directions of **maximum variance**. In that way, the **variance maximization problem** is to find the solution to

$$\max_{\mathbf{u}:\|\mathbf{u}\|=1} \mathbf{u}^\top\boldsymbol{\Sigma}\mathbf{u}$$

Here $\mathbf{\Sigma}$ is the sample covariance matrix described in the previous section. Note the similarity between this problem and the previously mentioned Rayleigh quotient – solving this problem is the same as solving for the largest eigenvalue of $\mathbf{\Sigma}$.

The goal of **principal component analaysis**, or PCA, is to project our data onto the hyperplane orthogonal to the directions that correspond to the maximum variance between our data points. These directions, or **principal components**, are simply the eigenvectors of our covariance matrix. For example, the projection onto the first two principal components is $\mathbf{x} \mapsto \mathbf{Px}$, where $\mathbf{P} = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2\end{bmatrix}^\top$ is composed of the first two eigenvectors of $\mathbf{\Sigma}$ corresponding to the two largest eigenvalues of $\mathbf{\Sigma}$.

The total variance of our data is $\mathbf{tr}(\mathbf{\Sigma}) = \mathbf{tr}(\mathbf{U\Lambda U}^\top) = \mathbf{tr}(\mathbf{U}^\top\mathbf{U\Lambda}) = \mathbf{tr}(\mathbf{\Lambda}) = \sum_{i=1}^{n} \lambda_n$. The variance of our projected data is then just $\sum_{i=1}^{k} \lambda_k$, where k is the number of principal components we use for our projection. The ratio of projected variance to total variance tells us how much of the variance we're capturing in our projection.

1.6 Singular Value Decomposition

Any rank-one matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ can be written as $\mathbf{A} = \sigma\mathbf{uv}^\top$ for $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^n$, $\sigma > 0$. We can turn this into a general statement for a rank r matrix by treating it as the sum of r rank-one matrices, i.e.

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$$

Here, each u_i is mutually orthogonal, and each v_i is mutually orthogonal. The values of σ are the **singular values** of $\mathbf{A}$.

SVD Theorem

The following result applies to **any** matrix $\mathbf{A}$ and helps define the map $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$:
An arbitrary matrix A in $\mathbf{R}^{m\times n}$ admits a decomposition of the form

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top = \mathbf{U}\tilde{\mathbf{\Sigma}}\mathbf{V}^\top, \tilde{\mathbf{\Sigma}} := \begin{pmatrix} \mathbf{\Sigma} & 0 \\ 0 & 0 \end{pmatrix}$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$, $\mathbf{V} \in \mathbb{R}^{n\times n}$ are both orthogonal matrices, and the matrix $\mathbf{\Sigma}$ is diagonal:

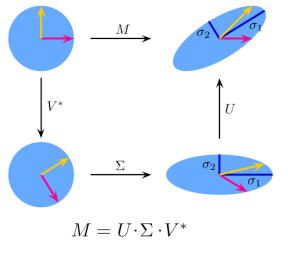
$$\mathbf{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_r)$$

where the positive numbers $\sigma_1 \geq \ldots \geq \sigma_r > 0$ are unique, and are called the singular values of $\mathbf{A}$.

The number $r \leq \min\{m, n\}$ is equal to the rank of $\mathbf{A}$, and the triplet $(\mathbf{U}, \tilde{\mathbf{\Sigma}}, \mathbf{V})$ is called a singular value decomposition (SVD) of $\mathbf{A}$. The first r columns of $\mathbf{U}$: $\mathbf{u}_i, i = 1, \ldots, r$ (resp. $\mathbf{V} : \mathbf{v}_i, i = 1, \ldots, r$) are called left (resp. right) singular vectors of $\mathbf{A}$, and satisfy

$$\mathbf{A}\mathbf{v}_i = \sigma_i \mathbf{u}_i, \quad \mathbf{A}^\top \mathbf{u}_i = \sigma_i v_i, \quad i = 1, ..., r$$

Computing the SVD of a matrix is on the order of $\text{O}(mn \min\{m, n\})$



2 Convex Models

Recall that we were able to use linear algebra to easily solve ordinary least-squares problems. We can apply this approach in an **iterative** fashion in order to minimize any function that is **bowl-shaped**, or **convex**.

2.1 Convex Sets

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is **convex** if and only if it contains the entire line segment between any two points in $\mathcal{C}$,

$$\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C},\ \forall \lambda \in [0,1]\ :\ \lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in \mathcal{C}$$

By this definition, subspaces and affine sets (i.e."flat" planes) are convex sets, since they contain the entire **line** between two points, not just the line segment. A set is a convex **cone** if $\forall \mathbf{x} \in \mathcal{C},\ \forall \alpha \in \mathbb{R},\ \alpha\mathbf{x} \in \mathcal{C}$. The intersection of family of convex sets is convex, and if a map $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ is affine and $\mathcal{C}$ is convex, then

$$f(\mathcal{C}) := \{f(\mathbf{x}) : \mathbf{x} \in \mathcal{C}\}.$$

One of the most important tools in convex optimization is our ability to draw a line between two convex sets.

Separating Hyperplane Theorem

If $\mathcal{C} \subseteq \mathbb{R}^n$ is convex and non-empty, then for any $\mathbf{x}_0$ at the boundary of $\mathcal{C}$, there exists a supporting hyperplane to $\mathcal{C}$ at $\mathbf{x}_0$, meaning that there exists $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{a} \neq 0$ such that $\mathbf{a}^\top(\mathbf{x} - \mathbf{x}_0) \leq 0\ \forall \mathbf{x} \in \mathcal{C}$.

Additionally, this means that if $\mathcal{C} \subseteq \mathbb{R}^n$ and $\mathcal{D} \subseteq \mathbb{R}^n$ are non-intersecting convex sets, there exists a **separating hyperplane**; that is, $\mathbf{a} \in \mathbb{R}^n,\ \mathbf{a} \neq 0$ and $\mathbf{b} \in \mathbb{R},\ \mathbf{b} \neq 0$ such that $\mathbf{a}^\top\mathbf{x} \leq \mathbf{b}\ \forall \mathbf{x} \in \mathcal{C}$ and $\mathbf{a}^\top\mathbf{x} \geq \mathbf{b}\ \forall \mathbf{x} \in \mathcal{D}$.

2.2 Convex Functions

The **domain** of a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a set $\mathbf{dom}f \subseteq \mathbb{R}^n$ where f is well-defined.

$$\mathbf{dom}f := \{\mathbf{x} \in \mathbb{R}^n : -\infty < f(x) < \infty\}$$

A function f is a **convex function** if

1. $\mathbf{dom} f$ is convex.
2. $\forall \mathbf{x}, \mathbf{y} \in \mathbf{dom} f,\ \forall \theta \in [0,1] : f(\theta \mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y})$

A function where $-f$ is convex is called a **concave** function.

Note that the fact that $\mathbf{dom} f$ is convex is essential to this definition, and therefore the domain on which a function is defined is important. For example, say we have the funciont $f : \mathbb{R} \mapsto \mathbb{R}$, $f(x) = 1/x$. If we define $f(x) = 1/x$ for $x > 0$ and $f(x) = +\infty$ for $x \leq 0$, then $f(x)$ is convex since its domain is $\mathbb{R}$, a convex set. However, if we define $f(x)$ on the domain $\mathbb{R}\backslash\{0\}$, f would not be a convex function since the set $\mathbb{R}\backslash\{0\}$ is not convex.

Alternate Characterizations of Convexity

1. **Epigraph:** A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex if and only if its **epigraph** is convex. The epigraph of a function can intuitively be thought of as the space above the boundary of a function; for the function $f(x) = x^2$, $\mathbf{epi} f = \{(x, y) : y \geq x^2\}$. More formally,
$$\mathbf{epi} f := \{(x, t) : t \geq f(x)\}$$

2. **First Order Condition:** If f is differentiable ($\mathbf{dom} f$ is open and the gradient is well-defined on the domain) then f is convex if and only if
$$\forall \mathbf{x}, \mathbf{y} : f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})$$
This can be interpreted intuitively as saying that f is convex if f is always bounded below by one of its tangent lines.

3. **Restriction to a Line:** The function f is convex if and only if its restriction to **any** line is convex. This means that for any $\mathbf{x}_0 \in \mathbb{R}^n$ and any $v \in \mathbb{R}$, $f(x_0 + tv)$ is convex.

4. **Second Order Condition:** If f is twice differentiable, then it is convex if and only if its Hessian $\nabla^2 f \succeq 0$ everywhere on $\mathbf{dom} f$. This is the common "calculus" definition of convexity.

Operations that Preserve Convexity

1. **Composition with an Affine Function:** If $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^n$ and $f : \mathbb{R}^m \mapsto \mathbb{R}$ is convex, then the function $g : \mathbb{R}^n \mapsto \mathbb{R}$, $g(x) = f(\mathbf{A}\mathbf{x}+\mathbf{b})$ is also convex over $\mathbf{dom}g := \{\mathbf{x} : \mathbf{A}\mathbf{x}+\mathbf{b} \in \mathbf{dom}f\}$.

2. **Pointwise Maximum:** Let $(f_\alpha)_{\alpha \in \mathcal{A}}$ is a family of convex functions, then
$$f(x) := \max_{\alpha \in \mathcal{A}} f_\alpha(x)$$
is convex. This is one of the most important ways of proving convexity. This follows from the assertion that the intersection of the epigraphs of convex functions is convex. Intuitively, this means that the set of all points that form a "least upper bound" for all the functions in (f_α) is convex.

3. **Nonnegative Weighted Sum:** The nonnegative weighted sum of convex functions is convex.
$$\forall \mathbf{w} \in \mathbb{R}_+^{\mathbf{card}\mathcal{A}}, \sum_\alpha w_\alpha f_\alpha$$
is convex if each element of $\{f_\alpha : \alpha \in \mathcal{A}\}$ is convex.

4. **Composition with Monotone Convex Function:** Compositions with functions generally do not preserve convexity. However, if $f = h \circ g$ with g convex and h monotonically increasing, then f is convex.

The **dual norm** is of the form
$$\mathbf{x} \mapsto \sup_{\mathbf{y}:\|\mathbf{y}\| \leq 1} \mathbf{y}^\top \mathbf{x}$$

This function is convex because linear functions are by definition convex (and concave) on their entire domain. Intuitively, think of the dual norm as taking a linear map $f_\mathbf{y} = \mathbf{y}^\top \mathbf{x}$ and seeing how "big" it is compared to $\mathbf{x}$.

2.3 Convex Problems

An optimization problem in standard form

$$\min_{\mathbf{x}} f_0(\mathbf{x}): \quad \begin{aligned} f_i(\mathbf{x}) &\leq 0 \quad i = 1, \ldots, m \\ h_i(\mathbf{x}) &= 0 \quad i = 1, \ldots, p \end{aligned}$$

is **convex** if

1. The objective function f_0 is convex.
2. The functions defining the inequality constraints $f_i,\ i = 1, \ldots, m$ are convex.
3. The functions defining the equality constraints $h_i,\ i = 1, \ldots, p$ are affine.

Optimality Theorem

For convex problems, any locally optimal point is globally optimal. The optimal set is convex.

We can rewrite our optimization problem as $\min_{\mathbf{x}\in\mathcal{X}} f_0(\mathbf{x})$ where $\mathcal{X}$ is the feasible set. From our first order condition of convex functions, we know that

$$f_0(\mathbf{y}) \geq f_0(\mathbf{x}) + \nabla f_0(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}).$$

Then $\mathbf{x}$ is optimal if and only if

$$\mathbf{x} \in \mathcal{X},\ \forall \mathbf{y} \in \mathcal{X} : \nabla f_0(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \geq 0$$

If $\nabla f_0(\mathbf{x})$ is nonzero, it defines a supporting hyperplane to $\mathcal{X}$ at $\mathbf{x}$. When the problem is unconstrained, we find our optimum at $\nabla f_0(\mathbf{x}) = 0$.

We say that a constraint is **active** at the optimum $\mathbf{x}^* \in \mathcal{X}$ if $f_i(\mathbf{x}^*) = 0$ and that a constraint is **inactive** if $f_i(\mathbf{x}^*) < 0$.

There are also cases when we want to maximize a convex function over a set $\mathcal{S}$. This is a difficult problem! We can say that this is equivalent to maximizing over the **convex hull** of the set. For any $\mathcal{S} \subseteq \mathbb{R}^n$ and any convex function $f : \mathbb{R}^n \mapsto \mathbb{R}$, we have

$$\max_{\mathbf{x}\in\mathcal{S}} f(x) = \max_{\mathbf{x}\in\mathbf{Co}\mathcal{S}} f(x)$$

In other words, instead of having to consider every point in the set, we can simply find the maximum at each vertex of the convex hull. However,

finding the convex hull remains a computationally intensive problem. For more study on this problem and this class of problems, make sure to check out CS170.

2.4 Linear Optimization

A **linear program** is an optimization problem where all functions involved are affine. The feasible set is a **polyhedron**, the intersection of a set of half spaces. A **polyhedral function** is one whose epigraph is a polyhedron. These functions include sums of maxima of affine functions, and can be solved through linear programming.

Recall that a **half-space** is defined by a single affine inequality, i.e.

$$\mathcal{H} = \{\mathbf{x} : \mathbf{a}^\top \mathbf{x} \leq \mathbf{b}\}$$

Now we can also say that a half space is a convex set bounded by a hyperplane. A hyperplane splits the entire space in half.

Hyperplanes correspond to functional **level sets**. In this way, the **half-space** defined by a hyperplane is is a **sub-level set** of a linear function. If $b \geq 0$, 0 is in the half-space (i.e. $\mathbf{a}^\top(0) \leq b$).

A **polyhedron** is a set described by a finite number of affine functions, i.e.

$$\mathcal{P} = \{\mathbf{x} : \mathbf{a}_i^\top \mathbf{x} \leq \mathbf{b}_i, i = 1, \ldots, m\}$$

where $\mathbf{a}_i \in \mathbb{R}^n$, $\mathbf{b}_i \in \mathbb{R}$. We can then express the polyhedron as the intersection of finitely many half spaces, i.e.

$$\mathcal{P} = \bigcap_{i=1}^{m} \{x : \mathbf{a}_i^\top \mathbf{x} \leq \mathbf{b}_i\}$$

Geometrically, a polyhedron is a convex set with flat sides where each side is a hyperplane. The vectors $\mathbf{a}$ point away from the interior of the set.

Equality constraints are also allowed in the definition of a polyhedron; the set then becomes

$$\mathcal{P} = \{\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b},\ \mathbf{C}\mathbf{x} = \mathbf{d}\}$$

Standard Form

The standard form of a linear program is

$$\min_{\mathbf{x}} f_0(\mathbf{x}) : \quad f_i(\mathbf{x}) \leq 0 \quad i = 1, \ldots, m$$

Here we can assume that f_0 is linear and f_i is affine. Each constraint in an LP is affine in $\mathbf{x}$. Each constraint tell us that $\mathbf{x}$ lies in a half-space; all the constraints together tell us that $\mathbf{x}$ exists in the interior of a polyhedron. If a function is affine, we know it is of the form $f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} - \mathbf{b}$. We can then rewrite the linear program as

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} : \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \quad i = 1, \ldots, m$$

Conic Form

A function that can be expressed in the form

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} : \quad \mathbf{A}\mathbf{x} = \mathbf{b} \quad i = 1, \ldots, m \quad \mathbf{x} \geq 0$$

is in **conic form**. Every LP can be expressed in conic form (as can every QP). Here, instead of having a sign constraint $\mathbf{x} \geq 0$ we represent $\mathbf{x}$ as $\mathbf{x} \in \mathcal{K}$, where $\mathcal{K}$ is a convex cone.

2.4.1 Minimizing Polyhedra

A function is polyhedral if its epigraph is a polyhedron. For a more formal definition, $f : \mathbb{R}^n \mapsto \mathbb{R}$ is polyhedral iff

$$\exists \mathbf{C} \in \mathbb{R}^{m \times n+1},\ \mathbf{d} \in \mathbb{R}^m : \mathbf{epi}(f) = \{(\mathbf{x}, t) \in \mathbb{R}^{n+1} : t \geq f(\mathbf{x})\}$$

$$\implies \mathbf{epi}(f) = \left\{ (\mathbf{x}, t) \in \mathbb{R}^{n+1} : \mathbf{C} \begin{bmatrix} \mathbf{x} \\ t \end{bmatrix} \leq \mathbf{d} \right\}$$

Example: Conversion of ℓ_∞ Norm to LP:
Objective: Write the following minimization problem as a linear program:

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_\infty.$$

We can use the above epigraph definition for this LP, substitute the objective with the variable t and then constrain the objective. The new program

becomes

$$\min_t t \ : \ \max_{x_i} |\mathbf{a}_i^\top \mathbf{x} - b_i| \leq t \ \ \forall i \in [1, m]$$

which is then equivalent to

$$\min_t t \ : \ \mathbf{a}_i^\top \mathbf{x} - b_i \leq t, \ -(\mathbf{a}_i^\top \mathbf{x} - b_i) \leq t \ \ \forall i \in [1, m],$$

which is a linear program.

Example: Conversion of ℓ_1 Norm to LP
Objective: Write the following minimization problem as a linear program:

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1.$$

Again, we can use the epigraph definition:

$$\min_{\mathbf{u}} \sum u_i \ : \ \mathbf{a}_i^\top \mathbf{x} - \mathbf{b}_i \leq u_i, \ -(\mathbf{a}_i^\top \mathbf{x} - \mathbf{b}_i) \leq u_i \ \ \forall i \in [1, m]$$

2.5 Convex Quadratic Programs

A **quadratic program** is one which whose objective can be written generically as

$$f_0(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} + \mathbf{c}^\top \mathbf{x} + d,$$

where $d \in \mathbb{R}$, $\mathbf{c} \in \mathbb{R}^n$, and $\mathbf{H}$ is symmetric and in $\mathbb{R}^{n \times n}$. Note that at **linear function** is a special case of a quadratic function, where $\mathbf{H} = \mathbf{0}$. If $\mathbf{H}$ is not symmetric, we can replace it with its symmetric part $\frac{1}{2}(\mathbf{H} + \mathbf{H}^\top)$.

A quadratic function is **convex** if its Hessian is positive semidefinite ($\nabla^2 f_0(\mathbf{x}) \succeq 0$) – this means that every eigenvalue of $\mathbf{H}$ must be nonnegative.

The model defined as

$$p* = \min_{\mathbf{x}} f_0(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} + \mathbf{c}^\top \mathbf{x} \ : \ \mathbf{A}\mathbf{x} \leq \mathbf{b}, \ \mathbf{C}\mathbf{x} = \mathbf{d}$$

is a **quadratic program**, or QP. Such a program is **convex** if $\mathbf{H}$ is symmetric and PSD and non-convex otherwise.

2.5.1 Unconstrained Minimization of Linear Functions

We begin by considering a simple linear case, i.e. one where

$$p^* = \min_{\mathbf{x}} f_0(x) = \mathbf{c}^\top \mathbf{x} + d$$

Note that if $\mathbf{c} \neq \mathbf{0}$ then the program will have an **unbounded minimum**, i.e. if $\mathbf{c} \neq \mathbf{0}$ then $p^* = -\infty$. This is because we can let $\mathbf{x} = -\alpha\mathbf{c}$ for any large $|\alpha|$. We then get $f_0(\mathbf{x}) = -\alpha\|\mathbf{c}\|_2^2$ which we can arbitrarily drive towards $-\infty$. If $\mathbf{c} = \mathbf{0}$ then we get the minimum as $p* = d$.

2.5.2 Unconstrained Minimization of Convex Quadratic Functions

Consider the convex QP:

$$p^* = \min_{\mathbf{x}} f_0(\mathbf{x}) = \mathbf{x}^\top \mathbf{H}\mathbf{x} + \mathbf{c}^\top \mathbf{x} + d.$$

If $\mathbf{H} \succ 0$, i.e. $\mathbf{H}$ is positive definite, then we can rewrite this

$$f(x) = (\mathbf{x} - \mathbf{x}_0)^\top \mathbf{H}(\mathbf{x} - \mathbf{x}_0) + \alpha \geq \alpha,$$

where $\mathbf{x}_0 = -\mathbf{H}^{-1}\mathbf{c}$ and $\alpha = d - \mathbf{x}_0^\top \mathbf{H}\mathbf{x}_0 = d - \mathbf{c}^\top \mathbf{H}^{-1}\mathbf{c}$. The unique minimizer is $\mathbf{x}^* = \mathbf{x}_0$.

If $\mathbf{H}$ is only positive semi-definite, and $\mathbf{c} \in \mathcal{R}(\mathbf{H})$, then any $\mathbf{x}_0 : \mathbf{H}\mathbf{x}_0 + \mathbf{c} = 0$ is optimal. The set of solutions is then

$$\{-\mathbf{H}^\dagger + \zeta,\ \zeta \in \mathcal{N}(\mathbf{H})\}.$$

If $\mathbf{H}$ is only positive semi-definite and $\mathbf{c} \notin \mathcal{R}(\mathbf{H})$, then the function is unbounded below. By the FLTA, we can express $\mathbf{c}$ as any $\mathbf{c} = -\mathbf{H}\mathbf{x}_0 + \mathbf{r}$ for some $\mathbf{x}_0, \mathbf{r} \neq 0 \in \mathbb{R}^n$, $\mathbf{H}\mathbf{r} = 0$. If we let $\mathbf{x}(t) = \mathbf{x}_0 - t\mathbf{r}$, then evaluating $f(\mathbf{x}(t)) = \beta - t(\mathbf{r}^\top \mathbf{r})$ for a constant β. $\lim_{t\to\infty} f(\mathbf{x}(t)) = -\infty$, so the function is unbounded below.

2.5.3 Quadratic Minimization under Linear Equality Constraints

We can convert a quadratic program constrained by linear equalities into an unconstrained version. Take a quadratic program of the form

$$p^* = \min_{\mathbf{x}} f_0(\mathbf{x}) \ :\ \mathbf{A}\mathbf{x} = \mathbf{b}.$$

Parameterize all $\mathbf{x}$, with $\mathbf{Ax} = \mathbf{b}$ as $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Nz}$, where $\bar{\mathbf{x}}$ is one solution of $\mathbf{Ax} = \mathbf{b}$, $\mathbf{N}$ is a columnwise basis for $\mathcal{N}(\mathbf{A})$, and $\mathbf{z}$ is a vector of free variables. We then form the unconstrained minimization problem as

$$p^* = \min_{\mathbf{z}} \varphi_0(\mathbf{z}) = \frac{1}{2}\mathbf{z}^\top \bar{\mathbf{H}}\mathbf{z} + \bar{\mathbf{c}}^\top \mathbf{z} + \bar{d},$$

$$\bar{H} = \mathbf{N}^\top \mathbf{H}\mathbf{N},\ \mathbf{c} = \mathbf{N}^\top(\mathbf{c} + \mathbf{H}\bar{\mathbf{x}}),\ d = d + \mathbf{c}^\top \bar{\mathbf{x}} + \frac{1}{2}\bar{\mathbf{x}}^\top \mathbf{H}\bar{\mathbf{x}}$$

2.5.4 Linear-Quadratic Control

We want to figure out what sequence of inputs we should be feeding a system so that we can reach a target state $\mathbf{x}_d$ at some time $T > t_0$. We can pose this problem as a quadratic program:

$$\min_{(\mathbf{x}(t))_{t=t_0}^T, (\mathbf{u}(t))_{t=t_0^T}} \|\mathbf{x}(T) - x^d\|_2^2 + \sum_{t=t_0}^{T} \|\mathbf{u}(t)\|_2^2$$
$$\text{s.t. } \mathbf{x}(t) = \mathbf{A}^{t-t_0}\mathbf{x}(t_0) + \sum_{i=t_0}^{t-1} \mathbf{A}^{t-i-1}\mathbf{B}\mathbf{u}(i),\ t \in [t_0, T].$$

The optimal $\mathbf{u}^*$ can be written as a linear function of the state $\mathbf{x}$, hence the term **linear-quadratic**.

2.6 Second Order Cones

Second-order cone programming (SOCP) generalizes linear and quadratic programs to allow for affine combinations of variables constrained within a special convex set, the **second-order cone**. Linear programs and convex quadratic programs are special cases of second-order cone programs. SOCPs are especially useful in problems concerning geometry, as well as approaches to linear optimization that involve random uncertainty in the data.

In $\mathbb{R}^3$, the second-order cone is the set of vectors $(\mathbf{x}_1, \mathbf{x}_2, t)$ where $\sqrt{\mathbf{x}_1^2 + \mathbf{x}_2^2} \leq t$. Horizontal slices of the cone at level $\alpha \geq 0$ are disks of radius α. In an arbitrary dimension, an $n+1$ dimension SOC is

$$\mathcal{K}_n = \{(\mathbf{x}, t), \mathbf{x} \in \mathbb{R}^n, t \in \mathbb{R} : \|\mathbf{x}\|_2 \leq t\}.$$

The **rotated** second-order cone in $\mathbb{R}^{n+2}$ is the set

$$\mathcal{K}_n^r = \{(\mathbf{x}, y, z), \mathbf{x} \in \mathbb{R}^n, y \in \mathbb{R}, z \in \mathbb{R} : \mathbf{x}^\top \mathbf{x} \leq 2yz, y \geq 0, z \geq 0\}.$$

The rotated second-order cone in $\mathbb{R}^{n+2}$ is actually just a linear transform (a rotation in this case) of the standard SOC in $\mathbb{R}^{n+2}$, since

$$\|\mathbf{x}\|_2^2 \leq 2yz, y \geq 0, z \geq 0 \implies \left\| \begin{bmatrix} \mathbf{x} \\ \frac{1}{\sqrt{2}}(y-z) \end{bmatrix} \right\| \leq \frac{1}{\sqrt{2}}(y+z).$$

The above result can be translated to mean that $(\mathbf{x}, y, z) \in \mathcal{K}_n^r$ if and only if $(\mathbf{w}, t) \in \mathcal{K}_n$, where $\mathbf{w} = (\mathbf{x}, (y-z)/\sqrt{2})$, $t = (y+z)/\sqrt{2}$. Constraints of the form $\mathbf{x}^\top \mathbf{x} \leq 2yz$ are referred to as **hyperbolic** constraints.

Constraints for second-order cones take the form of $\|\mathbf{y}\|_2 \leq t$, meaning that for some variable $\mathbf{x}$, the constraint $(\mathbf{y}, t) \in \mathcal{K}_m \implies \|\mathbf{y}\|_2 \leq t$ for $\mathbf{y}, t$ affine transforms of $\mathbf{x}$. Simply put, the constraints must be in the form

$$\|\mathbf{A}\mathbf{x} + \mathbf{b}\| \leq \mathbf{c}^\top \mathbf{x} + d.$$

We can write the quadratic constraint $\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{c}^\top \mathbf{x} \leq t$ as the conic

$$\left\| \begin{bmatrix} \sqrt{2}\mathbf{Q}^{1/2}\mathbf{x} \\ t - \mathbf{c}^\top \mathbf{x} - 1/2 \end{bmatrix} \right\|_2 \leq t - \mathbf{c}^\top \mathbf{x} + 1/2.$$

Second Order Cone Programs

A second-order cone program is a convex optimization problem with a linear objective and SOC constraints. They are in the general form of

$$p^* = \min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \ : \ \|\mathbf{A}_i \mathbf{x} + \mathbf{b}_i\|_2 \leq \mathbf{c}_i^\top \mathbf{x} + d_i \ \ \forall i \in [1, m].$$

We can cast the standard form LP

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \ : \ \mathbf{a}_i^\top \mathbf{x} \leq b_i \ \ \forall i \in [1, m]$$

as the SOCP:

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \ : \ \|\mathbf{C}_i \mathbf{x} + \mathbf{d}_i\|_2 \leq \mathbf{b}_i - \mathbf{a}_i^\top \mathbf{x} \ \ \forall i \in [1, m]$$

where $\mathbf{C}_i = 0$ and $\mathbf{d}_i = 0$.

We can cast the QP

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{Q} \mathbf{x} + \mathbf{c}^\top \mathbf{x} \ : \ \mathbf{a}_i^\top \mathbf{x} \leq \mathbf{b}_i \ \ \forall i \in [1, m]$$

with $\mathbf{Q} = \mathbf{Q}^\top \succeq 0$ as the SOCP

$$\min_{\mathbf{x}, y} \mathbf{c}^\top \mathbf{x} + y \ : \ \left\| \begin{bmatrix} 2\mathbf{Q}^{1/2} \mathbf{x} \\ y - 1 \end{bmatrix} \right\|_2 \leq y + 1, \mathbf{a}_i^\top \mathbf{x} \leq b \ \ \forall i \in [1, m].$$

The convex **quadratically constrained quadratic program** (QCQP) with standard form

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{Q}_0 \mathbf{x} + \mathbf{a}_0^\top \mathbf{x} \ : \ \mathbf{x}^\top \mathbf{Q}_i \mathbf{x} + \mathbf{a}_i^\top \mathbf{x} \leq b_i \ \ \forall i \in [1, m]$$

with $\mathbf{Q}_i = \mathbf{Q}_i^\top \succeq 0$ can be cast as the SOCP

$$\min_{\mathbf{x}, y} \mathbf{c}^\top \mathbf{x} + y \ : \ \left\| \begin{bmatrix} 2\mathbf{Q}_0^{1/2} \mathbf{x} \\ y - 1 \end{bmatrix} \right\|_2 \leq y + 1,$$

$$\left\| \begin{bmatrix} 2\mathbf{Q}_i^{1/2} \mathbf{x} \\ b_i - \mathbf{a}_i^\top \mathbf{x} - 1 \end{bmatrix} \right\|_2 \leq b_i - \mathbf{a}_i^\top \mathbf{x} + 1 \ \ \forall i \in [1, m].$$

2.7 Robust Optimization

A lot of times, we work in an **ideal** setting, where we can have confidence in our data. However, in the real world this is rarely the case. Data usually has a degree of **uncertainty**. **Estimation** errors can affect the problem parameters, while **implementation** errors impact the final decision. Uncertainty can lead to highly unstable solutions or degraded performance. We can take our **nominal problem**

$$\min_{\mathbf{x}} f_0(\mathbf{x}) \ : \ f_i(\mathbf{x}) \leq 0 \ \ \forall i \in [1, m]$$

and transform it into its **robust** counterpart,

$$\min_{\mathbf{x}} \max_{u \in \mathcal{U}} f_0(\mathbf{x}, \mathbf{u}) \ : \ \forall \mathbf{u} \in \mathcal{U}, \ f_i(\mathbf{x}, \mathbf{u}) \leq 0 \ \ \forall i \in [1, m].$$

Essentially, we are now making our objective and constraints dependent on a second vector $\mathbf{u}$ representing our **uncertainty**, where $\mathbf{u}$ is constrained to a set $\mathcal{U}$. This problem also convex, and gets this convexity from the nominal. This problem is generally pretty complex, but under certain conditions relaxations exist.

Robust Linear Programming

Take the nominal problem

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \ : \ \mathbf{a}_i^\top \mathbf{x} \leq b_i \ \ \forall i \in [1, m].$$

Assume that $\mathbf{c}$ is drawn from a set $\mathcal{U} \subseteq \mathbb{R}^n$. The robust program is then

$$\min_{\mathbf{x}} \max_{\mathbf{c} \in \mathcal{U}} \mathbf{c}^\top \mathbf{x} \ : \ \mathbf{a}_i^\top \mathbf{x} \leq b_i \ \ i \in [1, m].$$

This method of robustness can be extended for uncertainty affecting the coefficient matrix $\mathbf{A}$ or the right-hand side vector $\mathbf{b}$.

2.7.1 Robust Single Inequality

Here we examine the robust variant of a single inequality constraint, namely that

$$\forall \mathbf{a} \in \mathcal{U}, \ \mathbf{a}^\top \mathbf{x} \leq \mathbf{b}$$

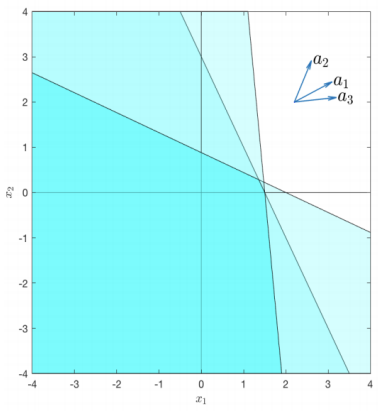
Here we let $\mathcal{U}$ take one of three forms – the "scenario" case, where $\mathcal{U}$ is a finite set of scenarios or cases; the "box" case; or the case where $\mathcal{U}$ is a

sphere or ellipsoid. Note that the above inequality can be written as

$$b \geq \max_{\mathbf{a} \in \mathcal{U}} \mathbf{a}^\top \mathbf{x}.$$

We begin with the "scenario" case. This assumes that we only know that $\mathbf{a}$ lies in a finite subset of $\mathbb{R}^n, \mathcal{U} = \{\mathbf{a}^{(1)}, ..., \mathbf{a}^{(K)}\}$ where each $\mathbf{a}^{(k)}$ represents a "scenario." We then have

$$\max_{\mathbf{a} \in \mathcal{U}} \mathbf{a}^\top \mathbf{x} = \max_{1 \leq k \leq K} \mathbf{a}^{(k)\top} \mathbf{x}$$

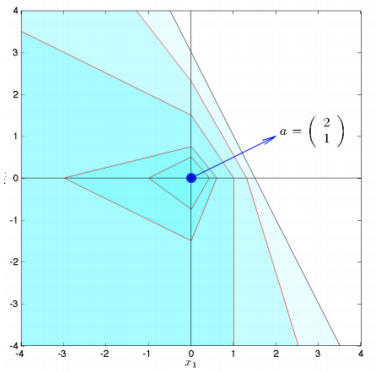


In the box case, we say that we only know that the coefficient vector $\mathbf{a}_i$ lies within a "box", a hyperrectangle in $\mathbb{R}^n$. In the simplest case, this means that

$$\mathcal{U} = \{\mathbf{a} : \|\mathbf{a} - \hat{\mathbf{a}}\|_\infty \leq \rho\} = \{\mathbf{a} + \rho\mathbf{u} : \|\mathbf{u}\|_\infty \leq 1\}.$$

Here ρ represents the size of the uncertainty and $\hat{\mathbf{a}}$ is the nominal value of the coefficient vector. We then have

$$\max_{\mathbf{a} \in \mathcal{U}} \mathbf{a}^\top \mathbf{x} = \hat{\mathbf{a}}^\top \mathbf{x} + \rho \cdot \left(\max_{\mathbf{u}: \|\mathbf{u}\|_\infty \leq 1} \mathbf{u}^\top \mathbf{x} \right) = \hat{\mathbf{a}}^\top \mathbf{x} + \rho \|\mathbf{x}\|_1.$$

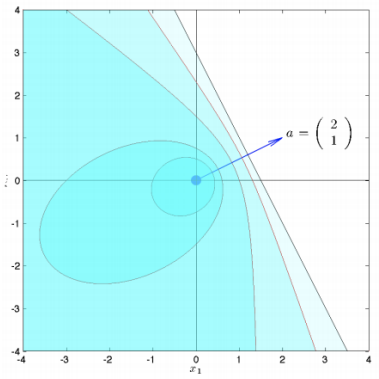


In the case of spherical uncertainty, we assume that $\mathbf{a}_i$ is only known to lie in a sphere. Then we have

$$U = \{\mathbf{a} : \|\mathbf{a} - \hat{\mathbf{a}}\|_2 \leq \rho\} = \{\mathbf{a} + \rho\mathbf{u} \, \|\mathbf{u}\|_2 \leq 1\}.$$

Here $\rho \geq 0$ measures uncertainty and $\hat{\mathbf{a}}$ is the nominal coefficient vector. Then we have

$$\max_{\mathbf{a}\in\mathcal{U}} \mathbf{a}^\top \mathbf{x} = \hat{\mathbf{a}}^\top \mathbf{x} + \rho \cdot \left(\max_{\mathbf{u}:\|\mathbf{u}\|_2 \leq 1} \mathbf{u}^\top \mathbf{x} \right) = \hat{\mathbf{a}}^\top \mathbf{x} + \rho \|\mathbf{x}\|_2.$$



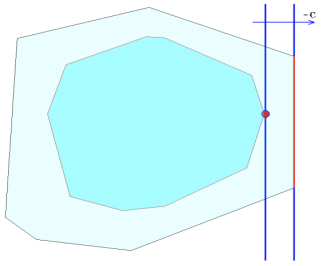
We also have **ellipsoidal** uncertainty, where we know $\mathbf{a}_i$ to be in an ellipse, i.e.

$$\mathcal{U} = \{\mathbf{a} : (\mathbf{a} - \hat{\mathbf{a}})^\top \mathbf{P}^{-1} (\mathbf{a} - \hat{\mathbf{a}}) \leq 1\} = \{\mathbf{a} = \hat{\mathbf{a}} + \mathbf{R}\mathbf{u} : \|\mathbf{u}\|_2 \leq 1\}$$

since $\mathbf{P} \succ 0$ and therefore $\mathbf{P} = \mathbf{R}^\top \mathbf{R}$. We then have

$$\max_{\mathbf{a}\in\mathcal{U}} \mathbf{a}^\top \mathbf{x} = \hat{\mathbf{a}}^\top \mathbf{x} + \max_{\mathbf{u}:\|\mathbf{u}\|_2 \leq 1} (\mathbf{R}\mathbf{u})^\top \mathbf{x} = \hat{\mathbf{a}}^\top \mathbf{x} + \|\mathbf{R}\mathbf{u}\|_2.$$

It can be difficult to visualize what exactly robust LP is trying to correct for, so the following visualization is given:



In the above diagram, the red line represents the nominal solution. Note that this implies that our data is inherently unstable; changing the direction of our objective (changing $\mathbf{c}$) could drastically change our optimum. However, the red point (which represents the solution to the robust problem) will stay the same even if we perturb the direction of our objective slightly. This is the advantage of robust programming.

2.7.2 Robust Least Squares

We now observe a robust version of the classic least squares optimization problem. Beginning with the standard LS formulation

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2, \ \mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{y} \in \mathbb{R}^m.$$

Now say that we only know that $\mathbf{A}$ exists within a certain **distance** of the nominal matrix $\hat{\mathbf{A}}$, i.e.

$$\|\mathbf{A} - \hat{\mathbf{A}}\| \leq \rho$$

(the norm above is the largest singular value norm, $\|\cdot\| = \sigma_{\max}(\cdot)$. An equivalent case is to say that $\mathbf{A} = \hat{\mathbf{A}} + \Delta$, where $\|\Delta\| \leq \rho$ where both Δ and ρ represent uncertainty. We are then left with the **robust least squares** problem,

$$\min_{\mathbf{x}} \max_{\|\Delta\|\leq\rho} \|(\hat{\mathbf{A}} + \Delta)\mathbf{x} - \mathbf{y}\|_2.$$

Think of this as minimizing $\mathbf{x}$ with the worst-case error Δ.Since the Euclidian norm is convex, $\|\mathbf{a} + \mathbf{b}\|_2 \leq \|\mathbf{a}\|_2 + \|\mathbf{b}\|_2$ so therefore

$$\|(\hat{\mathbf{A}} + \Delta)\mathbf{x} - \mathbf{y}\|_2 \leq \|\hat{\mathbf{A}}\mathbf{x} - \mathbf{y}\|_2 + \|\Delta\mathbf{x}\|_2$$

Additionally, $\|\Delta\mathbf{x}\| \leq \|\Delta\| \cdot \|\mathbf{x}\|_2 \leq \rho\|\mathbf{x}\|_2$, meaning we now have a bound on our objective of

$$\max_{\|\Delta\|\leq\rho} \|(\hat{\mathbf{A}} + \Delta)\mathbf{x} - \mathbf{y}\|_2 \leq \|\hat{\mathbf{A}}\mathbf{x} - \mathbf{y}\|_2 + \rho\|\mathbf{x}\|_2$$

The problem then becomes

$$\min_{\mathbf{x}} \|\hat{\mathbf{A}}\mathbf{x} - \mathbf{y}\|_2 + \rho\|\mathbf{x}\|_2,$$

which is just a regularized least squares problem that can be cast as the SOCP

$$\min_{\mathbf{x},\mathbf{u},\mathbf{v}} \mathbf{u} + \rho\mathbf{v} \ : \ \|\hat{\mathbf{A}}\mathbf{x} - \mathbf{y}\|_2 \leq \mathbf{u}, \ \|\mathbf{x}\|_2 \leq \mathbf{v}.$$

3 Duality

3.1 Weak Duality

Starting with any minimization problem (the **primal**), we can formulate a maximization problem (the **dual**) that provides a lower bound for it. This is known as **weak duality**. If the optimum value of the dual is the same as the primal, we have **strong duality**.

Consider a standard form minimization problem, that is one of the form

$$p^* = \min_{\mathbf{x}} f_0(\mathbf{x}) \quad f_i(\mathbf{x}) \leq 0 \quad i = 1, \ldots, m$$

Here p^* is the primal problem. Here $\mathbf{x} \in \mathbb{R}^n$ are the **primal variables**, and the optimum solution is the **primal value**.

Lagrange Function

Let $f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) & \cdots & f_m(\mathbf{x}) \end{bmatrix}$. The **Lagrangian** or **Lagrange function** is $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$ defined as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} y_i f_i(\mathbf{x}) = f_0(\mathbf{x}) + \mathbf{y}^\top f(\mathbf{x})$$

3.1.1 Dual Function

Based on the Lagrangian, we can construct a new function composed of **dual variables** that provides a lower bound to our original function. Fix some $\mathbf{y} \geq 0$ and note that $\mathbf{x} \mapsto \mathcal{L}(\mathbf{x}, \mathbf{y})$ is a **penalized** objective, meaning that violating any of the primal constraints incurs a **penalty**. The further we stray from the constraint, the bigger the penalty. If no constraints are violated, then $\mathbf{y}^\top f(\mathbf{x}) \leq 0$. Then

$$\forall \mathbf{x} \text{ feasible } f_0(\mathbf{x}) \geq \mathcal{L}(\mathbf{x}, \mathbf{y}).$$

We can now define the dual function

$$g(\mathbf{y}) := \min_{\mathbf{z}} \mathcal{L}(\mathbf{z}, \mathbf{y}) = \min f_0(\mathbf{z}) + \mathbf{y}^\top f(\mathbf{z})$$

The definition of $g(\mathbf{y})$ is a pointwise maximum, therefore g is concave. Combining this with our original inequality tells us

$$\forall \mathbf{x} \text{ feasible } f_0(\mathbf{x}) \geq g(\mathbf{y})$$

and minimizing over $\mathbf{x}$ gives us

$$\forall \mathbf{y} \geq 0 : p^* \geq g(\mathbf{y})$$

3.1.2 Dual Problem

We aren't just trying to find an arbitrary lower bound, however – we're looking for the **largest** lower bound; i.e. we're trying to find

$$p^* \geq d^* := \max_{\mathbf{y} \geq 0} g(\mathbf{y})$$

The problem is then to find the **best** lower bound. This is the **dual problem**, where the optimal value is the **dual value**. Therefore the dual problem, a maximization problem with g concave, is a convex optimization problem itself.

Duality is a powerful tool, since we can bound **any** minimization problem (convex or otherwise) into a convex optimization problem. However, the dual is not always necessarily easy to solve.

Example: The minimum distance to a polyhedron
The minimum distance to a polyhedron can be expressed by the function $p^* = \min \frac{1}{2}\mathbf{x}^\top \mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}$. Its Lagrangian is $\mathcal{L}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{x}^\top \mathbf{x} + \mathbf{y}^\top(\mathbf{A}\mathbf{x} - \mathbf{b})$. Then its dual is $g(\mathbf{y}) = \min_{\mathbf{x}} = \mathcal{L}(\mathbf{x}, \mathbf{y}) = \min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \mathbf{x} + \mathbf{y}^\top(\mathbf{A}\mathbf{x} - \mathbf{b})$. Using calculus, we can solve and see that $\mathbf{x}^* = -\mathbf{A}^\top \mathbf{y}$. Substituting this back gives us $g(\mathbf{y}) = \mathcal{L}(\mathbf{x}^*, \mathbf{y}) = -\frac{1}{2}\|\mathbf{A}^\top \mathbf{y}\|_2^2 - \mathbf{b}^\top \mathbf{y}$. The dual problem then becomes

$$d^* = \max_{\mathbf{y} \geq 0} -\mathbf{b}^\top \mathbf{y} - \frac{1}{2}\|\mathbf{A}^\top \mathbf{y}\|_2^2$$

Minimax and the Right Angle Property

The right angle property says that for $f \leq 0$ and $\mathbf{y} \geq 0$, the angle between $-f, \mathbf{y} \in \mathbb{R}_+^n$ is at most 90 degrees. This gives us a bound for the feasible primal set, i.e. $\max_{\mathbf{y} \geq 0} \mathbf{y}^\top f(\mathbf{x})$ is 0 for any feasible $\mathbf{x}$ but $+\infty$ for a non-feasible $\mathbf{x}$. Therefore $p^* = \min_{\mathbf{x}} \max_{\mathbf{y} \geq 0} \mathcal{L}(\mathbf{x}, \mathbf{y})$. The **minimax inequality** states that for any function of two vector variables $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ and $\mathcal{X} \subseteq \mathbb{R}^n, \mathcal{Y} \subseteq \mathbb{R}^m$, we have

$$\min_{\mathbf{x} \in \mathcal{X}} \max_{\mathbf{y} \in \mathcal{Y}} \mathcal{L}(\mathbf{x}, \mathbf{y}) \geq \max_{\mathbf{y} \in \mathcal{Y}} \min_{\mathbf{x} \in \mathcal{X}} \mathcal{L}(\mathbf{x}, \mathbf{y})$$

3.2 Strong Duality

When the primal and dual values coincide, we say that **strong duality** holds. In general, strong duality does not hold, but under a certain set of conditions it does. Here we explore those conditions.

3.2.1 Slater Condition for Strong Duality

Consider a convex constrained optimization problem in standard form:

$$p^* := \min_{\mathbf{x}} f_0(\mathbf{x}) : \mathbf{A}\mathbf{x} = \mathbf{b},\ f_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, m$$

for affine inequality constraints (i.e. h_i) $\mathbf{A} \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$ and $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ are convex functions. Here we have the Lagrangian as

$$\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \mapsto \mathbb{R},\ \mathcal{L}(\mathbf{x}, \lambda, \nu) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \nu^\top (\mathbf{A}\mathbf{x} - \mathbf{b})$$

for the dual variables λ, ν. The dual function is then

$$g(\lambda, \nu) = \min_{\mathbf{x}} f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \nu^\top (\mathbf{A}\mathbf{x} - \mathbf{b})$$

$$= \min_{\mathbf{x}} f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x})$$

The dual problem becomes

$$d^* = \max_{\lambda \geq 0, \nu} g(\lambda, \nu)$$

We saw previously that weak duality implies that $p^* \geq d^*$. Strong duality holds if $p^* = d^*$.

Slater's Condition

Slater's theorem provides a sufficient condition for strong duality, i.e. strong duality holds if

1. The primal problem is convex.
2. It is strictly feasible, i.e. $\exists x_0 \in \mathbb{R}^n : \mathbf{A}\mathbf{x}_0 = \mathbf{b},\ f_i(\mathbf{x}) < 0,\ i = 1, \ldots, m$.

Intuitively, this means that the primal is convex and there is a point inside the feasible set that is not on the border of the feasible set (i.e. there is a point that is fully contained within the boundaries of the feasible set).

Sion's Minimax Theorem

Let $\mathcal{X}$ be a compact convex subset of $\mathbb{R}^n$ and $\mathcal{Y}$ a convex subset of $\mathcal{R}^m$. If f is a real valued continuous function on $\mathcal{X} \times \mathcal{Y}$, with $f(\mathbf{x}, \cdot)$ continuous and concave on $\mathcal{Y}$, $\mathbf{x} \in \mathcal{X}$ and $f(\cdot, \mathbf{y})$ continuous and convex on $\mathcal{X}$, $\mathbf{y} \in \mathcal{Y}$, then

$$\min_{\mathbf{y} \in \mathcal{Y}} \max_{\mathbf{x} \in \mathcal{X}} f(x, y) = \max_{\mathbf{x} \in \mathcal{X}} \min_{yx \in \mathcal{Y}} f(x, y)$$

If a problem satisfies strong duality, then for an optimal primal variable $\mathbf{x}^*$ and optimal dual variables λ^*, ν^* we have

$$f(\mathbf{x}^*) = g(\lambda^*, \nu^*) \leq f_0(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* f_i(\mathbf{x}^*) + \sum_{i=1}^{p} \nu_i^* h_i(\mathbf{x}^*) \leq f_0(\mathbf{x}^*)$$

Here the first inequality is an expression of the dual problem as a minimization over $\mathbf{x}$ of the Langrangian, and the second inequality from the fact that $\mathbf{x}^*$ is feasible. Every term in the sum is nonpositive, therefore each individual term in the sum must be 0 to satisfy the above. The implication is that if the ith constraint is strictly satisfied, the corresponding dual variable λ_i must be 0. This is known as **complementary slackness**.

Karush-Kuhn-Tucker Conditions (KKT)

1. Primal Feasibility
2. Dual Feasibility
3. Complementary Slackness
4. Lagrangian Stationarity (every function involved is differentiable, i.e.)

$$\nabla f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \nabla f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i \nabla h_i(\mathbf{x}) = 0$$

These set of conditions are the **Karush-Kuhn-Tucker conditions (KKT)** of optimality. If a convex problem satisfies Slater's condition, then the primal point is optimal if and only if (λ, ν) s.t. the KKT conditions hold. Conversely, the conditions show strong duality and an optimal (x, λ, ν).

3.3 Descent Methods

Here we discuss algorithms for finding an optimal point when traditional methods are not easy/feasible. To be more precise, we aim to find a **minimizing sequence** $\mathbf{x}^{(k)}$ where $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$. For convex problems, we focus on **descent** methods, meaning that we need $f(x^{(k+1)}) \leq f(\mathbf{x}^{(k)})$ except for optimal $\mathbf{x}^{(k)}$. From our convexity definitions, we know that $\nabla f(\mathbf{x}^{(k)})^\top(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$ gives us the aforementioned inequality, $f(x^{(k+1)}) \leq f(\mathbf{x}^{(k)})$. Therefore $\nabla f(\mathbf{x}^{(k)})^\top \Delta\mathbf{x}^{(k)} < 0$.

General descent methods follow the following set of steps:
`Begin with` $\mathbf{x} \in$ `dom`(f).

1. `Begin with a descent direction` $\Delta\mathbf{x}$.
2. `Line Search determine a step size` $t > 0$.
3. `Update` $\mathbf{x} := \mathbf{x} + \Delta\mathbf{x}$.

`Repeat until the descent direction is` $\leq \epsilon$.

There are several methods for determining the step size t. In **exact line search**, we determine

$$t =_{s \geq 0} f(\mathbf{x} + s\Delta\mathbf{x}),$$

i.e. we determine t based on minimizing f on the ray $\{\mathbf{x} + t\Delta\mathbf{x} : t \geq 0\}$.

Frequently we don't try to find the exact minimizer of f – instead, we just try to find a value that minimizes f **enough**. One popular method is known as **backtracking line search**, depending on two constants $\alpha \in (0, 0.5)$ and $\beta \in (0, 1)$.

`Begin with a given direction` $\Delta\mathbf{x}$.
`Let` $t := 1$.
`while` $f(\mathbf{x} + \Delta\mathbf{x}) > f(\mathbf{x}) + \alpha t \nabla f(\mathbf{x})^\top \Delta\mathbf{x},\ t = \beta t$.

3.3.1 Gradient Descent

One of the most natural methods of selecting $\Delta\mathbf{x}$ is to use the **gradient** of f. `Begin with` $\mathbf{x} \in$ `dom`(f).

1. `Begin with a descent direction` $\Delta\mathbf{x} = -\nabla f(\mathbf{x})$.
2. `Check if this is` $\leq \epsilon$`; if so, break.`
3. `Line Search determine a step size` $t > 0$.
4. `Update` $\mathbf{x} := \mathbf{x} + t\Delta\mathbf{x}$.

Now say that we have the update equation $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta\nabla f(\mathbf{x}_k)$ (note that the two notations expressed thusfar are equivalent). We can write $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{b}$. Then the gradient descent algorithm will converge if $\|\lambda_{max}(\mathbf{A})\| < 1$.

A function f is L-Lipschitz if

$$\|f(\mathbf{x}) - f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2$$

. This additionally implies that $\|\nabla f(\mathbf{x})\|_2 \leq L$.
Now let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a convex, differentiable, L-Lipschitz function, with $\{\mathbf{x}_i, ..., \mathbf{x}_k\}$ being the iterates of gradient descent. Then let $\mathbf{x}^*$ be the optimum point and $\|\mathbf{x}_1 - \mathbf{x}^*\|_2 \leq R$. Then summing across all $k = 1, ..., t$ gives us

$$f\left(\frac{1}{t}\sum_{i=1}^{t}\mathbf{x}_i\right) - f(\mathbf{x}^*) \leq \frac{RL}{\sqrt{t}}.$$

Then we choose the optimum step size of

$$\eta = \frac{R}{L\sqrt{t}}$$

in order to minimize the convergence gap and maximize the convergence rate.