ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ

A review of Convex Analysis and Optimization with applications to Stochastic Approximation and Statistical Learning

Dimitrios Tziotis

ΕΡΓΑΣΙΑ

Που υποβλήθηκε στο Τμήμα Στατιστικής

του Οικονομικού Πανεπιστημίου Αθηνών

ως μέρος των απαιτήσεων για την απόκτηση

Διπλώματος Μεταπτυχιακών Σπουδών

στη Στατιστική

ΑΘΗΝΑ

Ιούλιος, 2020

Contents

Introduction

Convex optimization is the subfield of mathematical optimization that makes use of convex analysis to formulate and solve optimization problems. There are many different classes of convex optimization problems, but they all share an important quality: They have a unique global optimum which can be determined deterministically and in efficient time. Once an optimization problem is formulated as a convex problem, it can be automatically deemed *solvable*. Convex optimization is a key concept in statistical estimation where the challenge lies in using a density function as the objective function to be maximized with computational efficiency. Through maximum a posteriori (MAP) and maximum likelihood estimation (MLE) and, the challenge arises in both Bayesian and frequentist settings.

A non-convex problem is considered *unsolvable* when there exists no efficient-time exact algorithm, i.e. one which is guaranteed to converge to the global optimum in (worse case) polynomial time. In a *Learning* application, where the objective function is not a probability density but a non-convex loss function, *stochastic optimization* can be used to approximate the problem's global optimum in efficient time. In that latter scenario, the combination of statistical simulation and convex analysis gives rise to the field of *stochastic approximation*, which allows any non-convex problem to be approached as a noisy convex problem. Stochastic approximation algorithms are mostly gradient-based and are directly derived from the deterministic methods used in convex optimization.

The goal of this work is to trace the path of convex optimization into theoretical statistics. This is achieved first by developing the theory needed to optimize convex density functions (estimation), then by building upon that theory and mixing it with statistical simulation in order to optimize noisy loss functions (learning). The first part of this work deals with the theoretical background of convex analysis. The second part focuses on convex optimization problems, their applications, and the deterministic algorithms that solve them. In the third part, we introduce stochastic approximation and its associated algorithms, along with a working application of stochastic gradient descent on a neural network model.

Part I

Convex Analysis

1. Convex analysis

In this chapter, I provide the theoretical foundations that lead up to the definition of a convex optimization problem. The chapter treats convex sets, convex functions, and separation theorems, paving the way for their application to extremum problems.

1.1. Affine sets

Let **R** be the real number system, with $\mathbf{R}^n$ being the vector space of real *n*-tuples $x = (\xi_1, \dots, \xi_n)$. By default, everything takes place in $\mathbf{R}^n$. We express the inner product of two vectors x and x^* in $\mathbf{R}^n$ by:

$$\langle x, x^* \rangle = \xi_1 \xi_1^* + \cdots + \xi_n \xi_n^*.$$

We use the symbol A to denote a real matrix $m \times n$ and the corresponding linear transformation $x \to Ax$ from $\mathbf{R}^n$ to $\mathbf{R}^m$. Consequently, we use A^* to denote the transpose matrix and the associated adjoint linear transformation from $\mathbf{R}^m$ to $\mathbf{R}^n$. This leads to the identity:

$$\langle Ax, y^* \rangle = \langle x, A^* y^* \rangle.$$

The notation * has no operational significance for vectors and, by default, all vectors are column vectors (for matrix multiplication).

We use symbol || to denote the end of a proof.

Suppose that x and y are different points in $\mathbf{R}^n$, then if we have a set of points of the form:

$$(1-\lambda)x+\lambda y = x+\lambda(y-x), \quad \lambda \in \mathbf{R},$$

we call it *the line through x and y*.

We call a subset M of $\mathbf{R}^n$ an *affine* set if:

$(1-\lambda)x+\lambda y \in M$ for every $x \in M$, $y \in M$ and $\lambda \in \mathbf{R}$.

Extreme examples of affine sets are the empty set $\emptyset$ and the space $\mathbf{R}^n$ itself, as well as the case where M is composed by a solitary point. An affine set must contain two different points and the entire line through those points, like an endless uncurved structure, such as a line or a plane in space. The geometry of affine sets is derived from the theorems of linear algebra about subspaces of $\mathbf{R}^n$.

(Sources: [1])

1.2. Convex sets

We say that a subset of C in $\mathbf{R}^n$ is *convex* if:

$$(1-\lambda)x+\lambda y \in C,$$

whenever $x \in C, y \in C$ and $0 < \lambda < 1$.

Under the above definition, all affine sets are convex (including $\emptyset$ and $\mathbf{R}^n$). Convex sets are more general than affine sets as they can only comprise any two distinct points x and y and a segment of the line passing through them, i.e.:

$$\{(1-\lambda)x+\lambda y \mid 0 \leq \lambda \leq 1\}.$$

This is the *closed line segment between x and y*.

If we have a non-zero $b \in \mathbf{R}^n$ and a $\beta \in \mathbf{R}$, then the sets:

$$\{x \mid \langle x, b\rangle \leq \beta\}, \qquad \{x \mid \langle x, b\rangle \geq \beta\},$$

are called *closed half-spaces*, whereas the sets:

$$\{x \mid \langle x, b\rangle < \beta\}, \qquad \{x \mid \langle x, b\rangle > \beta\},$$

are called *open half-spaces.* All of the above four sets are non-empty and convex, but we would be getting the exact same sets if b and β were to be replaced by $\lambda\beta$, for $\lambda \neq 0$. As it seems, these half-spaces depend solely on the hyperplane $H = \{x \mid \langle x, b\rangle = \beta\}$ (Theorem 1.3).

Theorem 2.1.
The intersection of an arbitrary collection of convex sets is convex.

Proof:
Obvious. ‖

Corollary 2.1.1.
Let $b_i \in \mathbf{R}^n$ and $\beta_i \in \mathbf{R}^n$ for $i \in I$, where I is an arbitrary index set. Then the set:

$$C = \{x \in \mathbf{R}^n \mid \langle x, b_i\rangle \leq \beta_i, \forall i \in I\}$$

is convex.

Proof:
Let $C_i = \{x \mid \langle x, b_i\rangle \leq \beta_i\}$, then C_i is a closed half-space or $\mathbf{R}^n$ or $\emptyset$ and $C = \cap_{i\in I} C_i$. ‖

It is to note that Corollary 2.1.1 is valid for any system of simultaneous linear inequalities and equations in n variables, such that the set C of solutions is a convex set in $\mathbf{R}^n$.

A *polyhedral* convex set is one that can be expressed as the *intersection of finitely many closed half spaces* of $\mathbf{R}^n$. Due to its lack of curvature, a polyhedral convex set is better behaved than a general convex set and its theory is applicable to the study of *finite* systems of simultaneous linear equations and weak linear inequalities.

Any vector sum:

$$\lambda_1 x_1 + \cdots + \lambda_m x_m$$

can be called a *convex combination* of $x_1, \ldots, x_m$ if the coefficients λ_i are non-negative and $\lambda_1 + \cdots + \lambda_m = 1$. Convex combinations in applied mathematics, $\lambda_1, \ldots, \lambda_m$, can be often interpreted as probabilities or proportions – e.g.:

If m particles with masses $\alpha_1, \ldots, \alpha_m$ are located at points $x_1, \ldots, x_m$ of $\mathbf{R}^3$, the center of gravity of the system is the point $\lambda_1 x_1 + \cdots + \lambda_m x_m$, where $\lambda_i = \alpha_i/(\alpha_1 + \cdots + \alpha_m)$. Here, we have a convex combination where λ_i is the proportion of the total weight at x_i.

Theorem 2.2.

A subset of $\mathbf{R}^n$ is said to be convex iff it contains all possible convex combinations of its elements.

Proof:

A set C is convex, by definition, iff $\lambda_1 x_1 + \lambda_2 x_2 \in C$ whenever $x_1 \in C$, $x_2 \in C$, $\lambda_1 \geq 0$, $\lambda_2 \geq 0$, and $\lambda_1 + \lambda_2 = 1$, i.e. the convexity of C implies that C is closed under taking convex combinations with $m = 2$. It can be shown that C is also closed under taking convex combinations with $m > 2$, by taking any $m > 2$ and making the induction hypothesis that C is closed under taking all convex combinations of any number of vectors fewer than m. For a given convex combination $x = \lambda_1 x_1 + \cdots + \lambda_m x_m$ of elements of C, we must obtain at least one scalar λ_i that is different than 1 (since $\lambda_1 + \cdots + \lambda_m = m \neq 1$). Let's call this scalar λ_1 and suppose that:

$$y = \lambda_2' x_2 + \cdots + \lambda_m' x_m, \ \lambda_i' = \lambda_i/(1 - \lambda_1).$$

Then $\lambda_i \geq 0$ for $i = 2, \dots, m$, and:

$$\lambda_2' + \cdots + \lambda_m' = (\lambda_2 + \cdots + \lambda_m)/(\lambda_2 + \cdots + \lambda_m) = 1.$$

Therefore, y is a convex combination of $m - 1$ elements of C and, by induction, $y \in C$. Since $x = (1 - \lambda_1)y + \lambda_1 x_1$, it follows that $x \in C$. ||

Let the intersection of all convex sets that contain a given subset S of $\mathbf{R}^n$ to be called the *convex hull* of S and denoted by $conv\ S$. By Theorem 2.1, $conv\ S$ is a convex set; the unique smallest one containing S.

Theorem 2.3.

For any $S \subset \mathbf{R}^n$, $conv\ S$ consists of all the convex combinations of the elements of S.

Proof:

All elements of S belong to $conv\ S$, so all convex combinations between them belong to $conv\ S$ (Theorem 2.2). If we have two convex combinations $x = \lambda_1 x_1 + \cdots + \lambda_m x_m$ and $y = \mu_1 y_1 + \cdots + \mu_r y_r$ where $x_i \in S$ and $y_i \in S$, then the vector:

$$(1 - \lambda)x + \lambda y =$$
$$(1 - \lambda)\lambda_1 x_1 + \cdots + (1 - \lambda)\lambda_m x_m + \lambda_1 \mu_1 y_1 + \cdots + \lambda_r \mu_r y_r,$$

where $0 \leq \lambda \leq 1$, will be another convex combination of elements of S. Therefore, the set of convex combinations of elements of S will be also a convex set and, by containing S, it must coincide with the smallest convex set $conv\ S$. ||

Theorem 2.3 should be enough to consider the convex combinations that involve $n + 1$ or fewer elements at a time; an important refinement that leads to *Caratheodory's Theorem*.

Corollary 2.3.1.

The convex hull of a finite subset $\{b_0, \dots, b_m\}$ of $\mathbf{R}^n$ consists of all vectors of the form:

$$\lambda_0 \beta_0 + \cdots + \lambda_m \beta_m, \text{ with } \lambda_0 \geq 0, \dots,\ \lambda_m \geq 0,\ \lambda_0 + \cdots + \lambda_m = 1.$$

Proof:

Every convex combination of elements in $\{b_0, \dots, b_m\}$ can be expressed as a convex combination of $b_0, \dots, b_m$ if we include the unnecessary vectors b_i with zero coefficients. ‖

(Sources: [1])

1.3. Convex functions

We say that a function f: $\mathbf{R}^n \to \mathbf{R}$ is convex if its domain is a convex set and for all x, y in its domain, and all $\lambda \in [0,1]$, we have*:*

$$f(\lambda x + (1-\lambda)y \leq \lambda f(x) + (1-\lambda)f(y).$$

If we take any two points x, y, then evaluate f at any convex combination of these two points, the evaluation of f should not be larger than the convex combination of $f(x)$ and $f(y)$. Geometrically, the line segment connecting $(x, f(x))$ to $(y, f(y))$ must sit above the graph of f. In order to assess convexity, it suffices to check the above definition by setting λ to any fixed value $\lambda \in (0,1)$. Consequently, we say that f is *concave* if $-f$ is convex.

(Sources: [3], [4])

1.3.1. Strict and strong convexity

A function f: $\mathbf{R^n} \to \mathbf{R}$ is *strictly convex* if:

$$\forall x, y, x \neq y\ , \forall \lambda \in (0,1),\ f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y).$$

It is *strongly convex* if $\exists a > 0$ such that $f(x) - a\|x\|^2$ is convex.

Lemma 1:

Strong convexity ⇒ Strict convexity ⇒ Convexity.

(The converse of these implication is not true.)

Proof:

It is obvious that strict convexity implies convexity, but we shall prove that strong convexity implies strict convexity. We can see that strong convexity of f implies:

$$f(\lambda x + (1-\lambda)y) - a\|\lambda x + (1-\lambda)y\|^2 \leq \lambda f(x) + (1-\lambda)f(y) - \lambda a\|x\|^2 - (1-\lambda)a\|y\|^2$$

However:

$$\lambda a\|x\|^2 + (1-\lambda)a\|y\|^2 - a\|\lambda x + (1-\lambda)y\|^2 > 0, \ \ \forall x, y, x \neq y, \ \ \forall \lambda \in (0,1)$$

since $\|x\|^2$ is strictly convex. The converse statements are not true because $f(x) = x$ is convex but not strictly convex while $f(x) = x^4$ is strictly convex but not strongly convex.

(Sources: [3], [4])

1.3.2. Multivariate convex functions

Affine functions:

$f(x) = a^T x + b$, for any $a \in \mathbf{R}^n, b \in \mathbf{R}$.

Affine functions are convex, though not strictly convex. They are also concave:

$\forall \lambda \in [0,1], \ f(\lambda x + (1-\lambda)y) =$
$= a^T(\lambda x + (1-\lambda)y) + b =$
$= \lambda a^T x + (1-\lambda)a^T y + \lambda b + (1-\lambda)b =$
$= \lambda f(x) + (1-\lambda)f(y).$

Affine functions are the only functions that are both convex and concave.

Quadratic functions:

$f(x) = x^T Q x + c^T x + d.$

Quadratic functions are convex iff $Q \succcurlyeq 0$, strictly convex iff $Q \succ 0$, concave iff $Q \preccurlyeq 0$, and strictly concave iff $Q \prec 0$. This can be proven via the second order characterization of convexity.

Any norm:

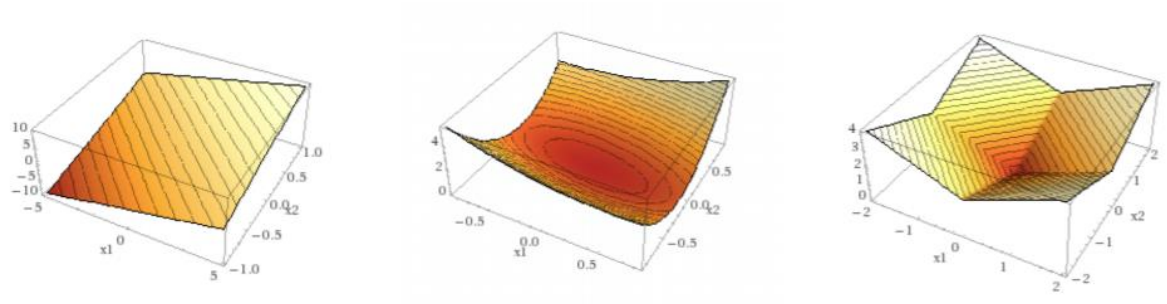
A norm is any function f that satisfies:

- $f(\alpha x) = |\alpha| f(x), \forall \alpha \in \mathbf{R}$
- $f(x + y) \leq f(x) + f(y)$
- $f(x) \geq 0,\ \forall x, f(x) = 0 \Rightarrow x = 0$

Proof:

$\forall \lambda \in [0,1], f(\lambda x + (1-\lambda)y) \leq f(\lambda x) + f\big((1-\lambda)y\big) = \lambda f(x) + (1-\lambda) f(y).$

The above inequality follows from triangle inequality and the equality follows from the homogeneity property.



An affine, a quadratic, and a 1-norm function. [3]

Convexity along all lines:

Theorem 1:

A function $f: \mathbf{R}^n \to \mathbf{R}$ is convex iff the function $g: \mathbf{R} \to \mathbf{R}$ given by $g(t) = f(x + ty)$ is convex (as a univariate function) for all x in domain of f and all $y \in \mathbf{R}^n$, where the domain of g is all t for which $x + ty$ is in the domain of f.

Proof:

Obvious from the definition.

The theorem may simplify various basic proofs in convex analysis but it will not make verifications of convexity much easier since the condition that it sets must hold for all lines among infinitely many. Many algorithms for convex optimization iteratively minimize the function over lines and the above statement ensures that every subproblem is also a convex optimization problem.

(Sources: [3], [4])

1.3.3. Properties of convex functions

Let f be a function whose domain is a subset S of $\mathbf{R}^n$ and whose values are real or $\pm\infty$. Then the set:

$$\{(x, \mu) | x \in S, \mu \in S, \mu \in \mathbf{R}, \mu \geq f(x)\},$$

will be called the *epigraph* of f and denoted by $epi\ f$. Function f is said to be a convex function on S if $epi\ f$ is convex as a subset of $\mathbf{R}^{n+1}$. Consequently, a *concave* function on S is a function whose negative is convex and an *affine* function on S is finite, convex, and concave.

We say that the *effective domain* of a convex function f on S, denoted by $dom\ f$, is the projection on $\mathbf{R}^n$ of the epigraph of f:

$$dom\ f = \{x | \exists \mu, (x, \mu) \in epi\ f\} = \{x | f(x) < +\infty\}.$$

The effective domain is the image of the convex set $epi\ f$ under a linear transformation and is, therefore, a convex set in $\mathbf{R}^n$ whose dimension is called the dimension of f. The convexity of f is equivalent to convexity of the restriction of f to $dom\ f$.

For a number of reasons, we cannot consider the class of all convex functions having a certain fixed C as their common effective domain and we prefer to focus on functions which are nowhere $+\infty$, i.e. where S would always coincide with $dom\ f$ (and would vary with f). Alternatively, we could focus on functions given on all of $\mathbf{R}^n$, since a convex function f on S can always be extended to a convex function on all of $\mathbf{R}^n$ by setting $f(x) = +\infty$ for $x \notin S$. The latter approach allows us to sidestep all technical details on effective domains.

We call a convex function *proper* if its epigraph is non-empty and contains no vertical lines, i.e. if $f(x) < +\infty$ for at least one x and $f(x) > -\infty$ for every x. Therefore, f is proper iff the convex set $C = dom\ f$ is non-empty and the restriction of f to C is finite. A proper convex function on $\mathbf{R}^n$ is a function that can be obtained by taking a finite convex function f on a non-empty convex set C and then extend it to all of $\mathbf{R}^n$ by setting $f(x) = +\infty$ for $x \notin C$.

A convex function that isn't proper, is called *improper*. Convex analysis focuses on proper convex functions, though improper functions may often arise from proper functions in practical situations.

Convex functions possess an important interpolation property. By the definition of a convex function, f is convex on S iff:

$$(1-\lambda)(x,\mu) + \lambda(y,v) = ((1-\lambda)x + \lambda y, (1-\lambda)\mu + \lambda v),$$

belongs to $epi\ f$ whenever (x,μ) and (y,v) belong to $epi\ f$ and $0 \le \lambda \le 1$.
We can, therefore, have $(1-\lambda)x + \lambda y \in S$ and:

$$f\big((1-\lambda)x + \lambda y\big) \le (1-\lambda)\mu + \lambda v,$$

whenever $s \in S$, $y \in S$, $f(x) \le \mu \in \mathbf{R}$, $f(y) \le v \in \mathbf{R}$ and $0 \le \lambda \le 1$. This is a condition that can be expressed in different ways.

Theorem 4.1:

Let f be a function from C to $(-\infty, +\infty)$, where C is a convex set (for example $C = R^n$). Then f is convex on C iff:

$$f\big((1-\lambda)x + \lambda y\big) \leq (1-\lambda)f(x) + \lambda f(y), \quad 0 < \lambda < 1,$$

For every x and y in C.

Theorem 4.2:

Let f be a function from $\mathbf{R}^n$ to $[-\infty, +\infty]$. Then f is convex iff:

$$f\big((1-\lambda)x + \lambda y\big) \leq (1-\lambda)\alpha + \lambda\beta, \quad 0 < \lambda < 1,$$

whenever $f(x) < \alpha$ and $f(y) < \beta$.

We can deduce this variant by applying Theorem 2.2 to epigraphs.

Theorem 4.3. (Jensen's Inequality):

Let f be a function from $\mathbf{R}^n$ to $(-\infty, +\infty]$. We assess that f is convex iff:

$$f(\lambda_1 x_1 + \cdots + \lambda_m x_m) \leq \lambda_1 f(x_1) + \cdots + \lambda_m f(x_m),$$

whenever $\lambda_1 \geq 0, \ldots, \lambda_m \geq 0, \lambda_1 + \cdots + \lambda_m = 1$.

Proof:

Concave functions satisfy the opposite inequalities under similar hypotheses, while affine functions satisfy the inequalities as equations. Therefore, affine functions on $\mathbf{R}^n$ are the affine transformations from $\mathbf{R}^n$ to $\mathbf{R}$.

The inequality in Theorem 4.1 can be perceived as the definition of the convexity of a function f from a convex set C to $(-\infty, +\infty]$, however, this may cause difficulties when f takes both $+\infty$ and $-\infty$ among its values and the expression $\infty - \infty$ can potentially arise. On the other

hand, the condition in Theorem 4.2 could be used as the definition of convexity in the general case. It should be noted that the initial definition is preferable because puts focus on geometry, which is fundamental to the theory of convex functions.

Theorem 4.4:

Let f be a twice continuously differentiable real-valued function on an open interval (α, β). Then f will be convex iff its second derivative f'' is non-negative throughout (α, β).

Proof:

Suppose that f'' is non-negative on (α, β) and f' is non-decreasing on (α, β). Then, for $a < x < y < \beta, 0 < \lambda < 1$ and $z = (1-\lambda)x + \lambda y$, we have:

$$f(z) - f(x) = \int_x^z f'(t)dt \leq f'(z)(z-x)$$

$$f(y) - f(z) = \int_z^y f'(t)dt \leq f'(z)(y-z)$$

Given that $z - x = \lambda(y-x)$ and $y - z = (1-\lambda)f'(z)(y-x)$, we have:

$f(z) \leq f(x) + \lambda f'(z)(y-x)$,
$f(z) \leq f(y) - (1-\lambda)f'(z)(y-x)$.

If we multiply the two inequalities by $(1-\lambda)$ and λ respectively and add them together, we get:

$(1-\lambda)f(z) + \lambda f(z) \leq (1-\lambda)f(x) + \lambda f(y)$.

The left side is just $f(z) = f((1-\lambda)x + \lambda y)$, which proves the convexity of f on (α, β) by Theorem 4.1. As for the converse assertion of the theorem: If f'' weren't non-negative on (α, β), then f^n would be negative on a certain subinterval (α', β') by continuity. Consequently, on (α', β') we would have:

$f(z) - f(x) > f'(z)(z-x)$,

$f(y) - f(z) > f'(z)(y - z)$.

and therefore:

$f\big((1-\lambda)x + \lambda y\big) > (1-\lambda)f(x) + \lambda f(y)$.

Thus f would not be convex on (α, β).

In the multidimensional scenario (by Theorem 4.1), every function of the form:

$f(x) = \langle x, a\rangle + \alpha, \;\; a \in \mathbf{R}^n, \;\; \alpha \in \mathbf{R}$

is convex on $\mathbf{R}^n$ and affine, thus, every affine function on $\mathbf{R}^n$ takes the above form. A quadratic function:

$f(x) = \frac{1}{2}\langle x, Qx\rangle + \langle x, a\rangle + \alpha,$

where Q is a symmetric $n \times m$ matrix, will be convex on $\mathbf{R}^n$ iff Q is positive semi-definite, i.e.:

$\langle z, Qz\rangle \geq 0$ for every $z \in \mathbf{R}^n$.

This is a direct conclusion via the multidimensional version of Theorem 4.4., described below.

Theorem 4.5.

Let f be a twice continuously differentiable real-valued function on an open convex set C in $\mathbf{R}^n$. Then f will be convex on C iff its Hessian matrix:

$Q_x = \big(q_{ij}(x)\big),$

$q_{ij}(x) = \frac{\partial^2 f}{\partial \xi_i \partial \xi_j}\,(\xi_1, \dots, \xi_n),$

is positive semi-definite for every $x \in C$.

Proof:

The convexity of f on C is equivalent to the convexity of the restriction of f to each line segment in C, which is the similar to the convexity of the function $g(\lambda) = f(y + \lambda z)$ on the open real interval $\{\lambda | y + \lambda z \in C\}$ for each $y \in C$ and $z \in \mathbf{R}^n$. It can be trivially shown that:

$$g''(\lambda) = \langle z, Q_x z \rangle, \quad x = y + \lambda z.$$

Therefore, by Theorem 4.4, we have that g is convex for every $y \in C$ and $z \in \mathbf{R}^n$ iff $\langle z, Q_x z \rangle \geq 0$ for every $z \in \mathbf{R}^n$, whose convexity may be verified by Theorem 4.5. This gives rise to the negative of the geometric mean:

$$f(x) = f(\xi_1, \dots, \xi_n) = -(\xi_1 \xi_2 \dots \xi_n)^{\frac{1}{n}} \quad \text{if} \quad \xi_1 \geq 0, \dots, \xi_n \geq 0,$$
$$= +\infty \qquad \text{otherwise.}$$

By direct computation it can be shown that:

$$\langle z, Q_x z \rangle = n^{-2} f(x) [(\sum_{j=1}^{n} \left(\frac{\zeta_j}{\xi_j}\right))^2 - n \sum_{j=1}^{n} \left(\frac{\zeta_j}{\xi_j}\right)^2]$$

for $z = (\zeta_1, \dots, \zeta_n), \quad x = (\xi_1, \dots, \xi_n), \quad \xi_1 > 0, \dots, \xi_n > 0.$

This quantity is non-negative because $f(x) < 0$ and:

$$(\alpha_1 + \cdots + \alpha_n)^2 \leq n(\alpha_1^2 + \cdots + \alpha_n^2)$$

(also $2\alpha_j \alpha_k \leq \alpha_j^2 + \alpha_k^2$) for any real numbers α_j.

One of the most fundamental convex functions on $\mathbf{R}^n$ is the *Euclidean norm*:

$$|x| = \langle x, x \rangle^{\frac{1}{2}} = (\xi_1^2 + \cdots + \xi_n^2)^{\frac{1}{2}}$$

which is the absolute value function when $n = 1$. The convexity of the Euclidean norm follows from the standard laws:

$$|x + y| \leq |x| + |y|, \qquad |\lambda x| = \lambda |x| \qquad \text{for } \lambda \geq 0.$$

We can examine a number of correspondences between convex sets and convex functions. The simplest such correspondence associates each set C in $\mathbf{R}^n$ with the *indicator function* $\delta(\,.\,|C)$ of C, where:

$$\begin{aligned} \delta(x|C) &= 0 \qquad \text{if } x \in C, \\ &= +\infty \quad \text{if } x \notin C. \end{aligned}$$

Here, the epigraph of the indicator function is a "half-cylinder with cross-section C" and C is obviously a convex set iff $\delta(.\,|C)$ is a convex function on $\mathbf{R}^n$. Indicator functions play an important role in convex analysis, in a similar way that characteristic functions of sets do in other branches of analysis.

We define the *support function* $\delta^*(.\,|C)$ of a convex set C in $\mathbf{R}^n$ by:

$$\delta^*(x|C) = \sup\{\langle x, y\rangle | y \in C\}.$$

The *gauge* $\gamma(.\,|C)$ is defined by:

$$\gamma(x|C) = \inf\{\lambda \geq 0 | x \in \lambda C\}, \quad C \neq \emptyset.$$

We define the Euclidean *distance function* $d(.\,, C)$ by:

$$d(x, C) = \inf\{|x - y| \,\big|\, y \in C\}.$$

The convexity of these functions can be directly verified on $\mathbf{R}^n$ or shown to follow from general principles. A noteworthy assessment is that convex functions give rise to convex sets.

Theorem 4.6.

For any convex function f and any $\alpha \in [-\infty, +\infty]$, the level sets $\{x|f(x) < \alpha\}$ and $\{x|f(x) \leq \alpha\}$ are convex.

Proof:

In the scenario of strict inequality, the result is immediate from Theorem 4.2, by setting $\beta = \alpha$. The convexity of $\{x|f(x) \leq \alpha\}$ follows from the fact that it is the intersection of the convex sets $\{x|f(x) < \mu\}$ for $\mu > \alpha$. A geometric view of this convexity comes by observing that $\{x|f(x) \leq \alpha\}$ is the projection on $\mathbf{R}^n$ of the intersection of $epi\ f$ and the horizontal hyperplane $\{(x, \mu)|\mu = \alpha\}$ in $\mathbf{R}^{n+1}$, so that $\{x|f(x) \leq \alpha\}$ can be seen as a horizontal cross-section of $epi\ f$.

Corollary 4.6.1.

Let f_i be a convex function on $\mathbf{R}^n$ and α_i be a real number for each $i \in I$, where I is an arbitrary index set. Then, the set:

$$C = \{x|f_i(x) \leq \alpha_i, \forall i \in I$$

is a convex set.

Proof:

Like Corollary 2.1.1.

By taking f to be a quadratic convex function in Theorem 4.6, we conclude that the set of points satisfying a quadratic inequality:

$$\frac{1}{2}\langle x, Qx\rangle + \langle x, \alpha\rangle + \alpha \leq 0$$

is convex when Q is positive semi-definite (Theorem 4.5). Convex sets of this form include all "solid" ellipsoids and paraboloids, specifically spherical balls like $\{x|\langle x, x\rangle \leq 1\}$.

Theorem 4.6 and Corollary 4.6.1 play an important role to the theory of systems of nonlinear inequalities. Convexity takes part in the analysis of other aspects of the theory of inequalities due to the fact that various classical inequalities can be regarded as special cases of Theorem

4.3. For example, if f on **R** is negative of the logarithm, then for a convex combination of positive numbers $x_1, \dots, x_m$ we have (by Theorem 4.3):

$$-\log(\lambda_1 x_1 + \cdots + \lambda_m x_m) \leq -\lambda_1 log\ x_1 - \cdots - \lambda_m log\ x_m$$

Multiplying the above by -1 and taking the exponential of both sides, we get:

$$\lambda_1 x_1 + \cdots + \lambda_m x_m \geq x_1^{\lambda_1} \dots x_m^{\lambda_m}.$$

Specifically, for $\lambda_1 = \cdots = \lambda_m = \frac{1}{m}$,

$$(x_1 + \cdots + x_m) / m \ \geq \ (x_1 \dots x_m)^{\frac{1}{m}}.$$

This is known as the inequality between the arithmetic mean and geometric mean of a family of positive numbers.

We can use operations which preserve convexity to obtain new convex functions from already known convex functions. Some of these operations, such as pointwise addition of functions, come from ordinary analysis, while others, such as taking the convex hull of a collection of functions, are come from geometry. If we express the constructed function as a constrained infimum, then the function becomes is directly applicable to the theory of extremum problem – known as convex optimization. For this latter scenario, one has to use these operations to prove that a given function with a complicated formula is a convex function.

(Sources: [1])

1.4. Relative interior

Let X be a normed vector space and E be a convex subset of X. We denote by $affhull(E)$ the intersection of affine spaces containing E, and by $\overline{affhull}(E)$ its closure; the latter being the smallest closed affine space containing E. The *relative interior* of E, denoted by $rint(E)$, is the interior of E viewed as a subset of $affhull(E)$.

Proposition 1.19:

Let A and B be two nonempty subsets of X, with empty intersection. If $A - B$ is convex and has a nonempty relative interior, then there exists a hyperplane $H_{x^*,a}$ separating A and B, and such that:

$\langle x^*, a\rangle < \langle x^*, a\rangle$, *whenever* $(a, b) \in A \times B$ *and* $a - b \in rint(A - B)$.

Proof:

Set $E \coloneqq B - A$ and $Y \coloneqq \overline{affhull}(E)$. By theorem 1.12, there exists a y^* in Y^* separating 0 and E, with strict inequality for $rint(E)$. By theorem 1.7, there exists an $x^* \in X^*$ whose restriction to Y is y^*, and the conclusion holds with x^*.

Remark:

By the previous preposition, when $B = \{b\}$ is a singleton, noting that $rint(A - b) = rint(A) - b$, we obtain that when A is convex, if $b \notin rint(A)$ then there exists an $x^* \in X$ such that:

$\langle x^*, a\rangle < \langle x^*, b\rangle$, whenever $a \in rint(A)$.

Since any convex subset of a finite-dimensional subspace has a nonempty relative interior, we deduce the following:

Corollary 1.21:

Let A and B be two convex and nonempty subsets of a Euclidean space, with empty intersection. Then there exists a hyperplane $H_{x^*,a}$ separating A and B, such that the conclusion of Proposition 1.19 holds.

(Sources: [2])

1.4.1. Relative interiors of convex sets

Recall that the *Euclidean distance* between two points x and y in $\mathbf{R}^n$ is defined as:

$$d(x,y) = |x-y| = \langle x-y, x-y\rangle^{\frac{1}{2}}$$

The function d is called the *Euclidean metric* and is convex as a function on $\mathbf{R}^{2n}$. (since d is obtained by combining the Euclidean norm $f(z) = |z|$ with the linear transformation $(x,y) \to x-y$ from $\mathbf{R}^{2n}$ to $\mathbf{R}^n$.)
The topological concepts of *closed set*, *open set*, *closure*, and *interior* in $\mathbf{R}^n$ are usually introduced in terms of convergence of vectors with respect to the Euclidean metric, however, such convergence is equivalent to the convergence of a sequence of vectors in $\mathbf{R}^n$ component by component. It should be noted that the topological properties of convex sets in $\mathbf{R}^n$ are notably simpler than those of arbitrary sets.

Convex functions play an important role in the theory of open and closed convex sets. Any continuous real-valued function f on $\mathbf{R}^n$ may produce a family of open level sets $\{x \mid f(x) < \alpha\}$ and closed level sets $\{x \mid f(x) \leq \alpha\}$, and these sets will be convex if f is convex.

Let us denote by B the Euclidean unit ball in $\mathbf{R}^n$:

$$B = \{x \mid |x| \leq 1\} = \{x \mid d(x,0) \leq 1\}.$$

The Euclidean unit ball B is a closed convex set, i.e. a level set of the Euclidean norm, which is continuous and convex. For any $a \in \mathbf{R}^n$, the Euclidean ball with radius $\epsilon > 0$ and enter a is expressed by:

$$\{x|d(x,a) \leq \epsilon\} = \{a+y|\ |y| \leq \epsilon\} = a + \epsilon B.$$

For any set C in $\mathbf{R}^n$, the set of points x whose distance from C does not exceed ϵ is:

$$\{x|\exists y \in C, d(x,y) \leq \epsilon\} = \cup\{y + \epsilon B | y \in C\} = C + \epsilon B.$$

The closure $cl\ C$ and *interior* $int\ C$ of C are given by the formulas:

$$cl\ C = \cap \{C + \epsilon B \,|\, \epsilon > 0\},$$
$$int\ C = \{x \,|\, \exists \epsilon > 0, x + \epsilon B \subset C\}.$$

In convex sets, the concept of *interior* is being absorbed into the more convenient concept of *relative interior*. The *relative interior* of a convex set C in $\mathbf{R}^n$, denoted by $ri\ C$, is defined as the resulting interior when C is a subset of its affine hull $aff\ C$. Therefore, $ri\ C$ comprises all points $x \in aff\ C$ that have an $\epsilon > 0$, such that $y \in C$ whenever $y \in aff\ C$ and $d(x, y) \leq \epsilon$, i.e.:

$$ri\ C = \{x \in affC \mid \exists \epsilon > 0, (x + \epsilon B) \cap (aff\ C) \subset C\}.$$

and:

$$ri\ C \subset C \subset cl\ C.$$

The set difference $(cl\ C) \setminus (ri\ C)$ is called the *relative boundary* of C, where C is said to be *relatively open* if $ri\ C = C$. For a n-dimensional convex set, we have by definition:

$$aff\ C = \mathbf{R}^n,$$

therefore:

$$ri\ C = int\ C.$$

By definition an affine set is open, but it is also closed at the same time. This is reflected by the fact that an affine set is an intersection of hyperplanes, where every hyperplane H can be expressed as a level set of a continuous function:

$$H = \{x = (\xi_1, \dots, \xi_n) \mid \beta_1\xi_1 + \cdots + \beta_n\xi_n = \beta\}.$$

where see that:

$cl\ C \subset (aff\ C) = aff\ C$

for any C. Therefore, any line passing through two different points of $cl\ C$ lies entirely in $aff\ C$.

We see that closures and relative interiors are preserved under translations as well as under any one-to-one affine transformation of $\mathbf{R}^n$ onto itself. Transformations of this type will preserve affine hulls and will be continuous in both directions. This results from the fact that the components of the image of a vector x, under an affine transformation, are either linear or affine functions of the components ξ_i of x.

A fundamental property of closures and relative interiors of convex sets follows.

Theorem 6.1.

Let C be a convex set in $\mathbf{R}^n$. Let $x \in ri\ C$ and $y \in cl\ C$. Then, $(1-\lambda)x + \lambda y$ belongs to $ri\ C$ (and to C) for $0 \leq \lambda < 1$.

Proof:

We can focus on the case where C is n-dimensional, so that $ri\ C = int\ C$. Let $\lambda \in [0,1)$; we must prove that $(1-\lambda)x + \lambda y + \epsilon B$ is contained in C for some $\epsilon > 0$. Since $y \in cl\ C$, we have $y \in C + \epsilon B$, for every $\epsilon > 0$. Therefore, for every $\epsilon > 0$:

$$(1-\lambda)x + \lambda y + \epsilon B \subset (1-\lambda)x + \lambda(C + \epsilon B) + \epsilon B$$
$$= (1-\lambda)[x + \epsilon(1+\lambda)(1-\lambda)^{-1}B] + \lambda C.$$

If ϵ is sufficiently small, the above set is contained within $(1-\lambda)C + \lambda C = C$, as $x \in int\ C$ by hypothesis.

(Sources: [1])

1.5. Separation of convex sets

From the Hahn-Banach theorem, valid in a vector space setting, we can derive results of separation of convex sets in normed vector spaces.

The Hahn-Banach Theorem:

Let X be a vector space. We say that $p: X \to \mathbf{R}$ is *positively homogeneous* and *subadditive* if it satisfies the following conditions:

(i) $p(ax) = ap(x)$, for all $x \in X$ and $a > 0$

(ii) $p(x + y) \leq p(x) + p(y)$, for all x and y in X.

We see that, for $x = 0$ in (i), we obtain that $p(0) = 0$, and so we could as well take $a = 0$ in (i). If $\beta \in (0,1)$, combining the above relations, we obtain:

$$p(\beta x + (1 - \beta)y) \leq \beta p(x) + (1 - \beta)p(y)$$

i.e., p is convex. It derives conversely that a positively homogeneous (finite-valued) convex function is subadditive.

The analytical form of the *Hahn-Banach theorem*, a nontrivial consequence of Zorn's lemma, is as follows.

Theorem 1.7:

Let p satisfy (1.13), X_1 be a vector subspace of X, and λ be a linear form defined on X_1 that is dominated by p in the sense that:

$\lambda(x) \leq p(x)$, *for all* $x \in X_1$.

Then there exists a linear form μ on X, dominated by p, whose restriction to X_1 coincides with λ.

We say that a real vector space X is a *normed space* when endowed with a mapping $X \to \mathbf{R}$, $x \mapsto \|x\|$, satisfying the three axioms:

$\|x\| \geq 0$, with equality iff $x = 0$,
$\|ax\| = |a|\|x\|$, for all $a \in \mathbf{R}, x \in X$,
$\|x + x'\| \leq \|x\| + \|x'\|$, (triangle inequality).

Then $(x, y) \mapsto \|x - y\|$ is a metric over X. We denote the norm of Euclidean spaces, i.e., finite-dimensional spaces endowed with the norm $(\sum_i x_i^2)^{1/2}$, by $|x|$.
A sequence x_k in a normed vector space X is said to be a *Cauchy sequence* if $\|x_p - x_q\| \mapsto 0$ when $p, q \uparrow \infty$. We say that X is a Banach space if every Cauchy sequence has a (necessarily unique) limit.
The *topological dual* X^* of the normed vector space X is the set of *continuous* linear forms (maps $X \mapsto \mathbf{R}$) on X. In the sequel, by dual space we will mean the topological dual. We denote the duality product between $x^* \in X$ and $x \in X$ by $\langle x^*, x \rangle_X$ or simply $\langle x^*, x \rangle$. Note that a linear form, say ℓ over X, is continuous iff it is continuous at 0, which holds iff $\sup\{\ell(x);\ \|x\| \leq 1\} < \infty$. So we may endow X^* with the norm:

$$\|x^*\|_* := \sup\{\langle x^*, x \rangle;\ \|x\| \leq 1\}.$$

It is easily check that X^* is a Banach space. The dual of $\mathbf{R}^n$ (space of vertical vectors) is denoted by $\mathbf{R}^{n*}$ (space of horizontal vectors).
In the sequel we may denote the dual norm by $\|x^*\|$. If X and Y are Banach spaces, we denote by $L(X, Y)$ the Banach space of linear continuous mappings $X \mapsto Y$, endowed with the norm $\|A\| := \sup\{\|Ax\|; \|x\| \leq 1\}$. We denote by B_X (resp. $\bar{B}_X$) the open (resp. closed) unit ball of X. If x_1^* is a continuous linear form on a linear subspace X_1 of X, its norm is defined accordingly:

$$\|x_1^*\|_{1,*} = \sup\{\langle x^*, x \rangle;\ \ x \in X_1, \|x\| \leq 1\}.$$

Here are some other corollaries of the Hahn-Banach theorem.

***Corollary 1.8**:*

Let x_1^* be a continuous linear form on a linear subspace X_1 of the normed space X. Then there exists an $x^* \in X^*$ whose restriction to X_1 coincides with x_1^*, and such that:

$\|x^*\|_* = \|x_1^*\|_{1,*}$.

***Proof**:*

Apply theorem 1.7 with $p(x) \coloneqq \|x_1^*\|_{1,*}\|x\|$. Since $\langle x^*, \pm x\rangle \leq p(x)$, we have that $\|x\| \leq 1$ implies $\langle x^*, \pm x\rangle \leq \|x_1^*\|_{1,*}$. The result follows.

***Corollary 1.9**:*

Let x_0 belong to the normed vector space X. Then there exists an $x^* \in X^*$ such that $\|x^*\| = 1$ and $\langle x^*, x_0\rangle = \|x_0\|$.

***Proof**:*

Apply Corollary 1.8 with $X_1 = \mathbf{R}x_0$ and $x_1^*(tx_0) = t\|x_0\|$, for $t \in \mathbf{R}$. The orthogonal of $E \subset X$ is the closest subspace of X^* defined by:

$E^{\perp} \coloneqq \{x^* \in X^*;\ \langle x^*, x\rangle = 0, for\ all\ x \in E\}$.

***Lemma 1.10**:*

Let E be a subspace of X. Then $E^{\perp} = \{0\}$ iff E is dense.

Proof:

(a) If E is dense, given $x \in X$, there exists a sequence x_k in E, $x_k \to x$ and hence, for all $x^* \in E^{\perp}$, $\langle x^*, x\rangle = \lim_k \langle x^*, x_k\rangle = 0$, proving that $x^* = 0$.

(b) If E is not dense, let $x_0 \notin \bar{E}$ (closure of E). We may assume that $\|x_0\| = 1$ and that $B(x_0, \epsilon) \cap E = \emptyset$ for some $\epsilon > 0$. Let $E_0 \coloneqq E \oplus (\mathbf{R}x_0)$ denote the space spanned by E_0 and x_0. Consider the linear form λ on E_0 defined by:

$\lambda(\epsilon + \alpha x_0) = \alpha$, foa all $\epsilon \in E$ and $\alpha \in \mathbf{R}$.

Since any $x \in E_0$ has a unique decomposition as $x = \epsilon + \alpha x_0$ with $e \in E$ and $\alpha \in \mathbf{R}$, the linear form is well-defined. Let such an x satisfy $\alpha \neq 0$. Since $e' \coloneqq -e/\alpha$ does not belong to $B(x_0, \epsilon)$, we have that $\|x\| = |\alpha|\|x_0 - e'\| \geq \epsilon|\alpha|$, and hence, $\lambda(x) = a \leq \frac{\|x\|}{\epsilon}$. If $a = 0$ we still have $\lambda(x) \leq \frac{\|x\|}{\epsilon}$. By Corollary 1.8, λ has an extension to a continuous form on X, which is a nonzero element of $E^{\perp}$.

Bidual space, Reflexivity.

Given $x \in X$, the mapping $\ell_x : X^* \to \mathbf{R}$, $x^* \mapsto \langle x^*, x \rangle$ is by [1.17] linear continuous. Since $|\langle x^*, x \rangle| \leq \|x^*\|\|x\|$, its norm $\|\ell_x\|$ (in the bidual space X^{**}) is not greater than $\|x\|$, and as a consequence of Corollary 1.9, is equal to $\|x\|$: the mapping $x \mapsto \ell_x$ is *isometric*. This allows us to identify X with a closed subspace of X^{**}. We say that X is *reflexive* if $X = X^{**}$. The Hilbert spaces are reflexive.

(Sources: [2])

1.5.1. Separating hyperplanes

Let X be a normed vector space. A topological hyperplane of X is a set of the form:

$H_{x^*,\alpha} \coloneqq \{x \in X;\ \ \langle x^*, x \rangle = \alpha,\ \ \text{for some } (x^*, \alpha) \in X^* \times \mathbf{R},\ x^* \neq 0\}.$

Then, a closed *half-space* of X will be a set of the form:

$\{x \in X;\ \ \langle x^*, x \rangle \leq \alpha\},$ where $x^* \neq 0.$

Definition 1.11

Let A and B be two subsets of X. We say that the hyperplane $H_{x^*,\alpha}$ *separates* A and B if:

$\langle x^*, a \rangle \leq \alpha \leq \langle x^*, b \rangle,\ \ \forall (a, b) \in A \times B.$ [1.24]

We speak of a *strict separation* if:

$\langle x^*, a \rangle < \langle x^*, b \rangle,\ \ \forall (a, b) \in A \times B.$ [1.25]

We speak of a *strong separation* if, for some $\epsilon > 0$:

$\langle x^*, a\rangle + \epsilon \leq \alpha \leq \langle x^*, b\rangle - \epsilon, \ \forall (a,b) \in A \times B.$ [1.26]

We say that $x^* \in X$ separates A and B if [1.24] holds for some α, strictly separates A and B if [1.25] holds, and strongly separates A and B if [1.26] holds for some $\epsilon > 0$ and α. If A is the singleton $\{a\}$, then we say that x^* separates a and B, etc.

Given two subsets A and B of a vector space X, we define their *Minkowski sum* and *difference* as:

$A + B = \{a + b; \quad a \in A, b \in B\},$
$A - B = \{a - b; \quad a \in A, b \in B\}.$

The first geometric form of the *Hahn-Banach theorem* is as follows:

Theorem 1.12

Let A and B be two nonempty subsets of the normed vector space X, with empty intersection. If $A - B$ is convex and has a nonempty interior, then there exists a hyperplane $H_{x^*,a}$ separating A and B, such that:

$\langle x^*, a\rangle + \langle x^*, b\rangle, \ whenever \ (a,b) \in A \times B \ and \ a - b \in int(A - B).$

Note that $A - B$ has a nonempty interior whenever either A or B has a nonempty interior. The proof needs the following concept.

Definition 1.13

Let C be a convex subset of X whose interior contains 0. The *gauge function* of C is:
$g_C(x) \coloneqq \inf\{\beta > 0; \ \beta^{-1}x \in C\}.$

Example 1.14: If C is the closed unit ball of X, then $g_c(x) = \|x\|$ for all $x \in X$.

A gauge function is obviously positively homogeneous and finite. If $B(0,\epsilon) \subset C$ for some $\epsilon > 0$, then:

$g_C(x) \leq \|x\|/\epsilon$, for all $x \in X$,

So it is bounded over bounded sets. In addition, for any $\beta > g_C(x)$ and $\gamma > 0$, since $x \in \beta C$ and $B(0,\gamma\epsilon) \subset \gamma C$, we get $x + B(0,\gamma\epsilon) \subset (\beta+\gamma)C$, so that $g_C(y) \leq g_C(x) + \gamma$, for all $y \in B(x,\gamma\epsilon)$. We have proved that:

If $B(0,\epsilon) \subset C$, then g_C is Lipschitz with constant $1/\epsilon$.

It easily follows that:

$\{x \in X; g_C(x) < 1\} = int(C) \subset \bar{C} = \{x \in X; g_C(x) \leq 1\}$.

Lemma 1.15:
A gauge is subadditive and convex.

Proof:
Let x and y belong to X. For all $b_x > g_C(x)$ and $b_y > g_C(y)$, we have that $(\beta_x)^{-1}x \in C$ and $(\beta_y)^{-1}y \in C$, so that:

$$\frac{x+y}{\beta_x+\beta_y} = \frac{\beta_x}{\beta_x+\beta_y}(\beta_x)^{-1}x + \frac{\beta_y}{\beta_x+\beta_y}(\beta_y)^{-1}y \in C$$

Therefore $g_C(x+y) \leq \beta_x + \beta_y$. Since this holds for any $\beta_x > g_C(x)$ and $\beta_y > g_C(y)$, we obtain that g_C is sugadditive. Since g_C is positively homogeneous, it easily follows that g_C is convex.

Proof of Theorem 1.12:
Let $x_0 \in int(B-A)$; since $A \cap B = \emptyset$, $x_0 \neq 0$. Set:

$C \coloneqq \{a - b + x_0, b \in B, a \in A\}$.

We easily check that $0 \in int(C)$. Obviously, x^* separates A *and* B iff it separates C and $\{x_0\}$. Let λ be the linear form defined on $X_1 \coloneqq \mathbf{R}x_0$ by $\lambda(tx_0) = t$, for $t \in \mathbf{R}$. Since $A \cap B = \emptyset, x_0 \notin C$, and hence, $g_C(x_0) \geq 1$. It easily follows that λ is dominated by g_C on X_1. By theorem 1.7, there exists a (possibly not continuous) linear form x^* on X, dominated by g_C, whose restriction to X_1 coincides with λ. Since $0 \in int(C)$ we have that (1.30) holds for some $\epsilon > 0$, and hence, being dominated by g_C, x^* is continuous. It follows that $\langle x^*, x\rangle \leq 1$, for all $x \in C$, or equivalently:

$\langle x^*, a\rangle - \langle x^*, b\rangle + \langle x^*, x_0\rangle \leq 1$, for all $(a, b) \in A \times B$,

Whereas $\langle x^*, x_0\rangle = 1$. Therefore x^* separates A and B. In addition, if $a - b \in int(A - B)$, say $B(a - b, \epsilon) \subset A - B$ for some $\epsilon > 0$, then $\langle x^*, a - b + \epsilon\rangle \leq 0$ whenever $\|e\| \leq \epsilon$; maximizing over $e \in B(0, \epsilon)$ we obtain that $\langle x^*, a - b\rangle \leq -\epsilon\|x^*\|$. Relation (1.28) follows.

Corollary 1.16

Let E be a closed convex subset of the normed space X. Then there exists a hyperplane that strongly separates any $x_0 \notin E$ and E.

Proof:

For $\epsilon > 0$ small enough, the open convex set $A \coloneqq B(x_0, \epsilon)$ has empty intersection with E. By Theorem 1.12, there exists an $x^* \neq 0$ separating A and E, that is:

$\langle x^*, x_0\rangle + \epsilon\|x^*\|_* = \sup\{\langle x^*, a\rangle; a \in A\} \leq \inf\{\langle x^*, b\rangle; b \in E\}$.

The conclusion follows.

Remark 1.17: Corollary 1.16 can be reformulated as follows: any closed convex subset of a normed space is the intersection of half spaces in which it is contained. The following example shows that, even in a Hilbert space, one cannot in general separate two convex sets with empty intersection.

Example 1.18: Let $X = \ell^2$ be the space of the real sequences whose sum of squares of coefficients is summable. Let C be the subset of X of sequences with finitely many nonzero coefficients, the last one being positive. The C is a convex cone that does not contain 0. Let x^* separate 0 and C. We can identify the Hilbert space X with its dual, and therefore x^* with an element of X. Since each element e_i of the natural basis belongs to the cone C, we must have $x_i^* \geq 0$ for all I, and $x_j^* > 0$ for some j. For any $\epsilon > 0$ small enough, $x := -\epsilon_j + \epsilon e_{j-1}$ belongs to C, but $\langle x^*, x \rangle = -x_j^* + \epsilon x_{j+1}^* < 0$. This shows that one cannot separate the convex sets. So, 0 and C cannot be separated.

(Sources: [2])

1.5.2. Separation theorems

The concept of *separation* has a special place in convexity theory and its applications. This has to do with the fact that a hyperplane in $\mathbf{R}^n$ divides $\mathbf{R}^n$ evenly in two, such that the complement of the hyperplane is the union of two disjoint open convex sets, which are the open half-spaces associated with the hyperplane.

Suppose that C_1 and C_2 are two non-empty sets in $\mathbf{R}^n$. A hyperplane H *separates* C_1 and C_2 if C_1 is contained in one of the closed half-spaces associated with H and C_2 lies in the opposite closed half-space. To separate C_1 and C_2 *properly*, it must hold that C_1 and C_2 are not *both* contained in H. To separate C_1 and C_2 *strongly*, there must exists some $\epsilon > 0$ such that $C_1 + \epsilon B$ is contained in one of the open half-spaces associated with H and $C_2 + \epsilon B$ is contained in the opposite open half-space, in which B is the unit Euclidean ball $\{x \mid |x| \leq 1\}$.

A different type of separation is *strict* separation, in which C_1 and C_2 must belong to opposing open half-spaces. However, *proper* separation and strong separation are likely the most useful as they correspond directly to extrema of linear functions.

Theorem 11.1.

Let C_1 and C_2 be non-empty sets in $\mathbf{R}^n$. There exists a hyperplane separating C_1 and C_2 *properly*, iff there exists a vector b such that:

(a) $\inf \{\langle x, b\rangle \mid x \in C_1\} \geq \sup \{\langle x, b\rangle \mid x \in C_2\}$,

(b) $\sup\{\langle x, b\rangle \mid x \in C_1\} \geq \inf \{\langle x, b\rangle \mid x \in C_2\}$.

There exists a hyperplane separating C_1 and C_2 *strongly*, iff there exists a vector such that:

$(c) \inf \{\langle x, b\rangle \mid x \in C_1\} > \sup \{\langle x, b\rangle \mid x \in C_2\}$.

Proof:

Suppose that b satisfies condition (a) and (b), and choose any β between the infimum over C_1 and the supremum over C_2. We have $b \neq 0$ and $\beta \in R$, so that $H = \{x \mid \langle x, b\rangle = \beta\}$ is a hyperplane.

The half-space $\{x \mid \langle x, b\rangle \geq \beta\}$ contains C_1, while $\{x \mid \langle x, b\rangle \leq \beta\}$ contains C_2, while condition (b) implies that not both C_1 and C_2 are contained in H. Therefore, H separates C_1 and C_2 *properly*.

In addition, when C_1 and C_2 are separated properly, then the *separating hyperplane* and associated closed half-spaces containing C_1 and C_2 can be expressed in the above manner for some b and β. Then, we have $\langle x, b\rangle \geq \beta$ for every $x \in C_1$ and $\langle x, b\rangle \leq \beta$ for every $x \in C_2$, with strict inequality for at least one $x \in C_1$ or $x \in C_2$ and, therefore, b satisfies both conditions (a) and (b).

If b satisfies the stronger condition (c), we can choose $\beta \in \mathbf{R}$ and $\delta > 0$ such that $\langle x, b\rangle \geq \beta + \delta$ for every $x \in C_1$, and $\langle x, b\rangle \leq \beta - \delta$ for every $x \in C_1$, and $\langle x, b\rangle \leq \beta - \delta$ for every $x \in C_2$. As the unit ball B is bounded, $\epsilon > 0$ can be chosen so small that $\left|\langle y, b\rangle\right| < \delta$ for every y in ϵB. Then:

$C_1 + \epsilon B \subset \{x \mid \langle x, b\rangle > \beta\}$,

$C_2 + \epsilon B \subset \{x \mid \langle x, b\rangle < \beta\}$,

So that condition (c) holds. ‖

Whether two sets can be separated is an existence question and many applications of separation theory are found in the proofs of various existence theorems. Having vector b with certain properties, we can construct a pair of convex sets C_1 and C_2 so that those vectors b correspond to the hyperplanes separating C_1 and C_2. We can subsequently invoke a theorem saying that C_1 and C_2 can be in fact separated.

The existence of separating hyperplanes in $\mathbf{R}^n$ does not involve the axiom of choice and the fundamental construction is given in the proof of the theorem below.

Theorem 11.2.

Suppose that C is a non-empty open convex set in $\mathbf{R}^n$, and let M be a non-empty affine set in $\mathbf{R}^n$ that does not meet C. Then, there exists a hyperplane H that contains M, such that one of the open half-spaces associated with H contains C.

Proof:

If M is a hyperplane, then one of the associated open half-spaces must contain C, otherwise M should meet C, contradicting the initial hypothesis. Assuming that M is found inside a hyperplane, we can show how to construct an affine set M' of one higher dimension than M that does not meet C. This construction yields a hyperplane H with the desired properties after n steps (or less), proving thus the theorem. The main separation theorem is presented below.

Theorem 11.3.

Let C_1 and C_2 be two non-empty convex sets in $\mathbf{R}^n$. For a hyperplane separating C_1 and C_2 properly to exist, it should hold that $ri\ C_1$ and $ri\ C_2$ have no point in common.

Proof:

Let C be a convex set and $C = C_1 - C_2$. The relative interior of C is $ri\ C_1 - ri\ C_2$ by corollary 6.6.2, therefore, $0 \in ri\ C$ iff $ri\ C_1$ and $ri\ C_2$ have no point in common. If $0 \in ri\ C$, there exists a hyperplane containing $M = \{0\}$ such $ri\ C$ is contained in one of the associated open half-spaces. Then, the closure of that half-space will contain C, given that $C \subset cl(ri\ C)$. Therefore, if $0 \notin ri\ C$, there exists a vector b such that:

$$0 \leq \inf_{\alpha \in C} \langle x, b \rangle = \inf_{\alpha_1 \in C_1} \langle x_1, b \rangle - \sup_{\alpha_2 \in C_2} \langle x_2, b \rangle,$$

and

$$0 \leq \sup_{\alpha \in C} \langle x, b \rangle = \sup_{\alpha_1 \in C_1} \langle x_1, b \rangle - \inf_{\alpha_2 \in C_2} \langle x_2, b \rangle.$$

In this manner, C_1 and C_2 can be separated properly (according to Theorem 11.1). The above conditions also imply that $0 \notin ri\, C$, as they assert the existence of a half-space $D = \{x \mid \langle x, b \rangle \geq 0\}$ that contains C, whose interior $ri\, D = \{x \mid \langle x, b \rangle > 0\}$ meets C. Hence, $ri\, C \subset ri\, D$ (Corollary 6.5.2). ‖

Proper separation allows that (exclusively) one of the sets can be contained in the separating hyperplane. This is a required provision, as shown by the sets:

$$C_1 = \{(\xi_1, \xi_2) \mid \xi_1 > 0, \xi_2 \geq \xi_1^{--1}\}$$
$$C_2 = \{(\xi_1, 0) \mid \xi_1 \geq 0\}$$

These convex sets are disjoint in $\mathbf{R}^2$ and the only separating hyperplane is the ξ_1-axis, which contains C_2. It follows that not every pair of disjoint closed convex sets can be separated strongly.

Theorem 11.4.

Suppose that C_1 and C_2 are non-empty convex sets in $\mathbf{R}^n$. For a hyperplane that separates C_1 and C_2 strongly to exist, it is required that:

$$\inf\{|x_1 - x_1| \mid x_1 \in C_1, x_2 \in C_2\} > 0,$$

i.e. that $0 \in cl(C_1 - C_2)$.

Proof:

If we assume that C_1 and C_2 can be separated strongly, then for some $\epsilon > 0$, $C_1 + \epsilon B$ will not meet $C_2 + \epsilon B$, but if the latter holds, then the convex sets $C_1 + \epsilon B$ and $C_2 + \epsilon B$ can be separated properly, according to the preceding theorem. Given that $\epsilon B = \epsilon' B + \epsilon' B$ for $\epsilon' = \epsilon/2$, then the sets $(C_1 + \epsilon' B) + \epsilon' B$ and $(C_2 + \epsilon' B) + \epsilon' B$ belong to opposite closed half-

spaces, so that $C_1 + \epsilon' B$ and $C_2 + \epsilon' B$ are in opposite open half-spaces. Therefore, C_1 and C_2 can be separated strongly iff, for some $\epsilon > 0$, the origin does not belong to the set:

$$(C_1 + \epsilon B) - (C_2 + \epsilon B) = C_1 - C_2 - 2\epsilon B$$

which implies that:

$$2\epsilon B \cap (C_1 - C_2) = \emptyset$$

for some $\epsilon > 0$, i.e. $0 \notin cl(C_1 - C_2)$. ‖

Corollary 11.4.1.

Let C_1 and C_2 be non-empty disjoint closed convex sets in $\mathbf{R}^n$ having no common directions of recession. Then there exists a hyperplane separating C_1 and C_2 strongly.

Proof:

We have $0 \notin (C_1 - C_2)$ since C_1 and C_2 are disjoint. But $cl\,(C_1 - C_2) = C_1 - C_2$ under the recession condition by Corollary 9.1.2. ‖

Corallary 11.4.2.

Suppose that C_1 and C_2 are two non-empty convex sets in $\mathbf{R}^n$, whose closures are disjoint. If either of those sets is bounded, then there exists a hyperplane separating C_1 and C_2 strongly.

Proof:

By applying the first corollary to $cl\ C_1$ and $cl\ C_2$, we see that one of them has no directions of recession at all. ‖

The set of solutions x to a system of weak linear inequalities $\langle x, b_i \rangle \leq \beta_i, i \in I$ is a closed convex set (because it is an intersection of closed half-spaces); albeit it can be shown that every closed convex set in $\mathbf{R}^n$ can be represented as a solution set of this sort.

Theorem 11.5.

A closed convex set C is the intersection of the closed half-spaces that contain it.

Proof:

Suppose that $\emptyset \neq C \neq \mathbf{R}^n$ and, given any $a \notin C$, the sets $C_1 = \{a\}$ and $C_2 = C$ satisfy the condition in Theorem 11.4. There exists, therefore, a hyperplane separating $\{a\}$ and C strongly. One of the closed half-spaces associated with this hyperplane will contain C but will not contain a. Therefore, the intersection of the closed half-spaces containing C will contain no points other than those in C. ‖

Corollary 11.5.1.

Let S be any subset of $\mathbf{R}^n$. Then $cl\ (conv\ S)$ is the intersection of all the close half-spaces containing S.

Proof:

A closed half-space contains $C = cl\ (conv\ S)$ iff it contains S. ‖

Corollary 11.5.2.

Suppose that C is a convex subset of $\mathbf{R}^n$ (other than $\mathbf{R}^n$ itself), then, there exists a closed half-space containing C; i.e. there exists some $b \in \mathbf{R}^n$ such that the linear function $\langle\ .\ , b\rangle$ is bounded above on C.

Proof:

It is implied by the hypothesis that $cl\ C \notin \mathbf{R}^n$ (for otherwise $\mathbf{R}^n = ri\ (cl\ C) \subset C$). According to the theorem, a point belongs to $cl\ C$, thus the collection of closed half-spaces containing $cl\ C$ cannot be empty. ‖

The geometric concept of tangency is an important tool in analysis as tangent lines to curves and tangent planes to surfaces are defined (classically) in terms of differentiation. In convex analysis, the opposite concept is exploited, i.e. how can a generalized tangency be defined geometrically in terms of separation. This is a notion used to develop a generalized theory of differentiation. The concept of *generalized tangency* is expressed by *supporting* hyperplanes and half-spaces.

Let C be a convex set in $\mathbf{R}^n$. We say that a *supporting half-space* to C is a closed half-space which contains C and has a point of C in its boundary. In addition, a *supporting hyperplane* to C, is a hyperplane which is the boundary of a supporting half-space to C. We can say that the supporting hyperplanes to C are the hyperplanes which can be represented in the form:

$H = \{x \mid \langle x, b\rangle = \beta\}, b \neq 0,$

where $\langle x, b\rangle \leq \beta$ for every $x \in C$ and $\langle x, b\rangle = \beta$ for at least one $x \in C$.

Therefore, we see that a supporting hyperplane to C is associated with a linear function which achieves its maximum on C. The supporting hyperplanes that pass through a given point $a \in C$, correspond to vectors b *normal* to C at a (as defined earlier). If C is not *n*-dimensional, so that $aff\ C \neq \mathbf{R}^n$, we can extend $aff\ C$ to a hyperplane that contains all of C.

Theorem 11.6.

Let C be a convex set and D be a non-empty convex subset of C. For the existence of a non-trivial supporting hyperplane to C containing D, it is required that D be disjoint from $ri\ C$.

Proof:

Given that $D \subset C$, the non-trivial supporting hyperplanes to C which contain D are the same as the hyperplanes which separate D and C properly. By a preceded theorem (11.3), such a hyperplane exists iff $ri\ D$ is disjoint from $ri\ C$, and this condition is equivalent to D being disjoint from $ri\ C$.

Corollary 11.6.1.

A convex set has a non-zero normal at each one of its boundary points.

Corallary 11.6.2.

Let C be a convex set; then any $x \in C$ is a relative boundary of C iff there exists a linear function h, which is not constant on C, such that h achieves its maximum over C at x.

1.6. Characterization of convex functions

Theorem 2:

Let $f: \mathbf{R}^n \to \mathbf{R}$ be a twice differentiable function over an open domain., where the following conditions are equivalent:

(i) f is convex.

(ii) $f(y) \geq f(x) + \nabla f(x)^T(y - x)$, for all $x, y \in dom(f)$, i.e. the first order Taylor expansion at any point is a *global* under estimator of the function.

(iii) $\nabla^2 f(x) \succcurlyeq 0$, for all $x \in dom(f)$, i.e. f has a nonnegative curvature everywhere and in a single dimension it holds: $f''(x) \geq 0, \forall x \in dom(f)$.

Proof:

We can prove that (i) $\Leftrightarrow$ (ii) and (ii) $\Leftrightarrow$ (iii).

First, we prove that (i) $\Rightarrow$ (ii). If f is convex, by definition, we have:

$$f(\lambda y + (1 - \lambda)x) \leq \lambda f(y) + (1 - \lambda) f(x), \forall \lambda \in [0,1], x, y \in dom(f).$$

This can also be expressed as:

$$f\big(x + \lambda(y - x)\big) \leq f(x) + \lambda\big(f(y) - f(x)\big)$$

$$\Rightarrow f(y) - f(x) \geq \frac{f\big(x+\lambda(y-x)\big) - f(x)}{\lambda}, \forall \lambda \in (0,1]$$

As $\lambda \downarrow 0$, we have:

$$f(y) - f(x) \geq \nabla f^T(x)(y - x). \qquad [1]$$

Then we prove that (ii) $\Rightarrow$ (i). Suppose that [1] holds for all $x, y \in dom(f)$, take any $x, y \in dom(f)$, and let:

$$z = \lambda x + (1 - \lambda)y.$$

Then, we have:

$f(x) \geq f(z) + \nabla f^T(x)(x - z)$ [2]

$f(y) \geq f(z) + \nabla f^T(z)(y - z)$ [3]

If we multiply [2] by λ and [3] by $(1 - \lambda)$, then add them together, we get:

$\lambda f(x) + (1 - \lambda) f(y) \geq f(z) + \nabla f^T(z)(\lambda x + (1 - \lambda) y - z)$
$= f(z)$
$= f(\lambda x + (1 - \lambda) y).$

Proving (ii) $\Rightarrow$ (iii):
We prove both premises in dimension 1 and generalize.

Proving (ii) $\Rightarrow$ (iii) for $n = 1$:
Let $x, y \in dom(f), y > x$. We have:

$f(y) \geq f(x) + f'(x)(y - x)$ [4]

and:

$f(x) \geq f(y) + f'(y)(x - y)$ [5]
$\Rightarrow f'(x)(y - x) \leq f(y) - f(x) \leq f'(y)(y - x)$

Using [4] and [5], dividing LHS and RHS by $(y - x)^2$, we get:

$\frac{f'(y) - f'(x)}{y - x} \geq 0, \forall x, y, x \neq y.$

By letting $y \to x$, we get:

$f''(x) \geq 0, \forall x \in dom(f).$

Finally, we prove that (iii) $\Rightarrow$ (ii) for $n = 1$:
Let $f''(x) \geq 0, \forall x \in dom(f)$. Using the mean value version of Taylor's theorem, we have:

$f(y) = f(x) + f'(x)(y-x) + \frac{1}{2}f''(z)(y-x)^2$, for some $z \in [x, y]$.

$\Rightarrow f(y) \geq f(x) + f'(x)(y-x)$.

In order to prove (ii)⇔(iii) in general dimension, we use the principle that convexity is equivalent to convexity along all lines, meaning that:

$f: \mathbf{R}^n \to \mathbf{R}$ is convex if $g(\alpha) = f(x_0 + \alpha v)$ is convex $\forall x_0 \in dom(f)$ and $\forall v \in \mathbf{R}^n$.

We have proved that this holds iff:

$g''(\alpha) = v^T \nabla^2 f(x_0 + \alpha v) v \geq 0$,
$\forall x_0 \in dom(f), \forall v \in \mathbf{R}^n$ and $\forall \alpha$ s.t. $x_0 + \alpha v \in dom(f)$.

Therefore, f is convex iff $\nabla^2 f(x) \succcurlyeq 0$ for all $x \in dom(f)$.

Corollary 1:
Suppose we have an unconstrained optimization problem:

$\min f(x)$
such that $x \in \mathbf{R}^n$,

where f is convex and differentiable. Then, any point $\bar{x}$ which satisfies $\nabla f(\bar{x}) = 0$ is a global minimum.

Proof:
Using the first order characterization of convexity, we have:

$f(y) \geq f(x) + \nabla f^T(x)(y-x), \forall x, y.$

which becomes:

$f(y) \geq f(\bar{x}) + \nabla f^T(\bar{x})(y-x), \forall y.$

since $\nabla f(\bar{x}) = 0$.

Hence, we have:

$f(y) \geq f(\bar{x}), \forall y$.

It should be noted that $\nabla f(x) = 0$ is always a necessary condition for local optimality in an unconstrained problem. For convex problems, the theory states that $\nabla f(x) = 0$ is necessary and sufficient in order to have *local and global optimality*. On the other hand, in the absence of convexity, condition $\nabla f(x) = 0$ is *not sufficient* for local optimality (for example: $f(x) = x^3$ and $\bar{x} = 0$). A necessary condition for unconstrained local optimality for a point x in the non-convex scenario is $\nabla^2 f(x) \succcurlyeq 0$, but in the convex scenario this condition automatically holds.

(Sources: [1])

1.6.1. Characterization of strict convexity

We have previously stated that a function $f: \mathbf{R}^n \to \mathbf{R}$ is strictly convex if:

$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y)$.

$\forall x, y, x \neq y$, for all $\lambda \in (0,1)$.

If f is strictly convex, then f is also convex; albeit the converse does not true (i.e. $f(x) = x, x \in \mathbf{R}$).

Second order sufficient condition:
If $\nabla^2 f(x) \succ 0$, for all $x \in \Omega$, then f is strictly convex on Ω. Again, the converse is not true.

First order characterization:
A function f is strictly convex on $\Omega \subseteq \mathbf{R}^n$ iff:

$f(y) > f(x) + \nabla f^T(x)(y - x), \forall x, y \in \Omega, x \neq y.$

There are similar characterizations for strongly convex functions, e.g. f is strongly convex iff $\exists m > 0$, such that:

$f(y) \geq f(x) + \nabla^T f(x)(y - x) + m\|y - x\|^2, \forall x, y \in dom(f),$

or iff there exists $m > 0$ such that:

$\nabla f(x) \succcurlyeq mI$, for all $x \in dom(f)$.

It should be noted that the main use of strict convexity is to ensure *uniqueness of the optimal solution*, in other words, that a local optimum corresponds to a global optimum.

(Sources: [3], [4])

1.6.2. Strict convexity and uniqueness of optimal solutions

Theorem 3:
Consider the following optimization problem:

$\min f(x)$
such that $x \in \Omega$,

where function $f: \mathbf{R}^n \to \mathbf{R}$ is strictly convex on Ω (Ω being a convex set). Then, the optimal solution will be a unique solution.

Proof:
If there were two optimal solutions $x, y \in \mathbf{R}^n$, then it would hold that $x, y \in \Omega$ and:

$f(x) = f(y) \leq f(z)$, for all $z \in \Omega$.

Considering that $z = \frac{x+y}{2}$ and, by convexity of Ω, we have $z \in \Omega$.

Then, by strict convexity, we have:

$$f(z) = f\left(\frac{x+y}{2}\right)$$
$$< \frac{1}{2}f(x) + \frac{1}{2}f(y)$$
$$= \frac{1}{2}f(x) + \frac{1}{2}f(x) = f(x).$$

However, the above contradicts (6).

(Sources: [3], [4])

1.6.3. Optimality conditions for convex optimization

Theorem 4:

Consider the following optimization problem:

$\min f(x)$
such that $x \in \Omega$,

where function $f: \mathbf{R}^n \to \mathbf{R}$ is convex and differentiable, and Ω is a convex set. We say that a point x is optimal iff $x \in \Omega$ and:

$$\nabla f(x)^T (y - x) \geq 0, \forall y \in \Omega.$$

The above condition implies that, if we move from x towards any feasible y, there will be a local increase of f. In addition, vector $-\nabla f(x)$ is the supporting hyperplane of the feasible set Ω at x. The necessity of this condition holds independent of the convexity of f, though convexity is used in order to establish sufficiency. If it holds that $\Omega = \mathbf{R}^n$, then the condition is reduced to our first order unconstrained optimality condition $\nabla f(x) = 0$. Finally, if x is in

the interior of Ω and is optimal, then it must hold that $\nabla f(x) = 0$ (e.g. by taking $y = x - \alpha\nabla f(x)$, for small α).

Proof:

To prove sufficiency, suppose that $x \in \Omega$ satisfies:

$$\nabla f(x)^T(y - x) \geq 0, \forall y \in \Omega. \qquad [8]$$

By the first order characterization of convexity defined previously, we have:

$$f(y) \geq f(x) + \nabla f^T(x)(y - x), \forall y \in \Omega. \qquad [9]$$

Consequently, adding together [8] and [9] gives:

$$f(y) \geq f(x), \forall y \in \Omega$$

which implies that x is optimal.

To prove necessity, suppose that x is optimal but for some $y \in \Omega$ we have:

$$\nabla f^T(x)(y - x) < 0.$$

Given that $g(\alpha) \coloneqq f(x + (\alpha(y - x))$ and, since Ω is convex, we have:

$$\forall \alpha \in [0,1], x + a(y - x) \in \Omega.$$

Consequently, we have:

$$g'(\alpha) = (y - x)^T\nabla f(x + \alpha(y - x))$$
$$\Rightarrow g'(0) = (y - x)^T\nabla f(x) < 0,$$

from which we derive:

$\exists \delta > 0, s.t.\ g(\alpha) < g(0), \forall \alpha \in (0, \delta)$
$\Rightarrow f\big(x + \alpha(y - x)\big) < f(x), \forall \alpha \in (0, \delta).$

However, this contradicts the optimality of x. Below we have a special case of this theorem.

Theorem 5:
Consider the optimization problem:

$\min f(x)$ [10]
such that $\forall x = b$,

where f is a convex function and $A \in \mathbf{R}^{m \times n}$.
Then, a point $x \in \mathbf{R}^n$ is optimal to [10] iff it is feasible and $\exists \mu \in \mathbf{R}^m$ such that:

$\nabla f(x) = A^T \mu.$

Proof:
By the optimality condition of convexity, a feasible x is optimal iff:

$\nabla f^T(x)(y - x) \geq 0, \forall y\ s.t. Ay = b.$

So any y with $Ay = b$ can be expressed as $y = x + v$, where v is a point in the nullspace of A (i.e. $Av = 0$). Therefore, a feasible x will be optimal iff:

$\nabla f^T(x)v \geq 0, \forall v\ s.t.\ Av = 0$

for any v. Since $Av = 0$ implies $A(-v) = 0$, we see that $\nabla f^T(x)v \leq 0$. Thus, the optimality condition reads:

$\nabla f^T(x)v = 0, \forall v\ s.t.\ Av = 0.$

The above condition implies that $\nabla f(x)$ is in the orthogonal complement of the nullspace of A, which we know from linear algebra that it is equal to the row space of A (or the column space of A^T). Hence, we have:

$\exists \mu \in \mathbf{R}^m \; s.t. \nabla f(x) = A^T \mu.$

(Sources: [3], [4])

1.6.4. Strong convexity and implications

We say that a function is strongly convex on S when there exists an $m > 0$, such that:

$\nabla^2 f(x) \succcurlyeq mI, \quad \forall x \in S.$

Strong convexity has several interesting consequences. For $x, y \in S$ we have:

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(z)(y - x)$$

For some z on the line segment $[x, y]$. By the strong convexity assumption, the last term on the righthand side is at least $\left(\frac{m}{2}\right)\|y - x\|_2^2$, so we can have the inequality:

$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}\|y - x\|_2^2, \quad \forall x, y \in S.$ [9.8]

When $m = 0$, we recover the basic inequality characterizing convexity; for $m > 0$ we obtain a better lower bound on $f(y)$ than follows from convexity alone.
We will first show that the inequality [9.8] can be used to bound $f(x) - p^*$, which is the suboptimality of the point x, in terms of $\|\nabla \mathrm{f(x)}\|_2$. The righthand side of [9.8] is a convex quadratic function of y (for fixed x). Setting the gradient with respect to y equal to zero, we find that $\tilde{y} = x - (1/m)\nabla f(x)$ minimizes the righthand side. Therefore we have:

$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}\|y - x\|_2^2$

$$\geq f(x) + \nabla f(x)^T(\tilde{y} - x) + \frac{m}{2}\|\tilde{y} - x\|_2^2$$
$$= f(x) - \frac{1}{2m}\|\nabla f(x)\|_2^2.$$

Since this holds for any $y \in S$, we have:

$$p^* \geq f(x) - \frac{1}{2m}\|\nabla f(x)\|_2^2. \qquad [9.9]$$

This inequality shows that if the gradient is small at a point, then the point is nearly optimal. The inequality [9.9] can also be interpreted as a condition for suboptimality which generalizes the optimality condition [9.2]:

$$\|\nabla f(x)\|_2 \leq (2m\epsilon)^{\frac{1}{2}} \Rightarrow f(x) - p^* \leq \epsilon.$$

We can also derive a bound on $\|x - x^*\|_2$, the distance between x and any optimal point x^*, in terms of $\|\nabla f(x)\|_2$:

$$\|x - x^*\|_2 \leq \frac{2}{m}\|\nabla f(x)\|_2.$$

To see this, we apply [9.8] with $y = x^*$ to obtain:

$$p^* = f(x^*) \geq f(x) + \nabla f(x)^T(x^* - x) + \frac{m}{2}\|x^* - x\|_2^2$$
$$\geq f(x) - \|\nabla f(x)\|_2\|x^* - x\|_2 + \frac{m}{2}\|x^* - x\|_2^2,$$

Where we use the Cauchy-Schwartz inequality in the second inequality. Since $p^* \leq f(x)$, we must have:

$$-\|\nabla f(x)\|_2\,\|x^* - x\|_2 + \frac{m}{2}\|x^* - x\|_2^2 \leq 0,$$

From which [9.11] follows. One consequence of [9.22] is that the optimal point x^* is unique.

(Sources: [5], [6])

Part II

Convex Optimization

2. Convex optimization

In this chapter, I lay down the formal definition of convex optimization and describe different classes of convex problems, as well as some of the deterministic algorithms used widely to solve these problems. I additionally describe some popular applications of convex optimization in statistical estimation and learning theory.

2.1. Generic optimization problem

A mathematical optimization problem has the form:

$$\begin{aligned} &\textit{minimize} \quad f_0(x) \\ &\textit{subject to} \quad f_i(x) \leq b_i, \quad i = 1, \dots, m \end{aligned}$$

where vector $x = (x_1, \dots, x_n)$ is the *optimization variable* (or decision variable),
function $f_0: \mathbf{R}^n \to \mathbf{R}$ is the *objective function*,
functions $f_i: \mathbf{R}^n \to \mathbf{R}$, $i = 1, \dots, m$ are the inequality *constraints functions*,
and constants $b_i, \dots, b_m$ are the *bounds* (or limits) for the constraints.

A vector denoted x^* is the *optimal solution* to an optimization problem, if it has the smallest *objective value* (output of the objective function) between all vectors that satisfy the problem constraints, i.e. for any z, where $f_1(z) \leq b_1, \dots, f_m(z) \leq b_m$, we have:

$$f_0(z) \geq f_0(x^*), \forall z$$

Optimization problems can be divided into different families or classes, depending on the structure of the objective function and the associated problem constraints. An important classification of optimization problems has to do with linearity:

- *Linear problem* (or linear program): An optimization problem where the objective function and the constraint functions are linear, i.e. satisfy:

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

  for all $x, y \in \mathbf{R}^n$ and $\alpha, \beta \in \mathbf{R}$.

- *Nonlinear problem* (or nonlinear program): An optimization problem where the objective function and the constraint functions are nonlinear.

A more general classification of optimization problems has to do with convexity. A convex optimization problem is one in which the objective and constraint functions are convex, i.e. the satisfy the inequality:

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

for all $x, y \in \mathbf{R}^n$ and $\alpha, \beta \in \mathbf{R}$, with $\alpha + b = 1,\ \alpha \geq 0,\ \beta \geq 0$. Since inequality replaces the more restrictive equality, we can consider linear programming to be a subclass of convex optimization.

In learning theory and data fitting, we select a model from a family of models that best fits the observed data and potentially some prior beliefs. The optimization variables are the parameters of the model while the optimization constraints are the equivalent of prior information or limits on the parameters. The objective function represents the error between the actual/observed values and the values predicted by the model.

(Sources: [5], [6])

2.2. Convex optimization problem

A mathematical optimization problem has the form:

$$\begin{aligned} &\textit{minimize} \quad f_0(x) \\ &\textit{subject to} \quad f_i(x) \leq b_i, \quad i = 1, \dots, m \end{aligned}$$

where functions $f_0, \dots, f_m$: $\mathbf{R}^n \rightarrow \mathbf{R}$ are convex, meaning that they satisfy the condition:

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

for all $x, y \in \mathbf{R}^n$ and all $\alpha, \beta \in \mathbf{R}$ with $\alpha + \beta = 1,\ \alpha \geq 0,\ \beta \geq 0$.

The least-squares and linear programming problems are two sub-classes of convex optimization problems which come with mature solution methods, whereas methods for solving general convex optimization problems are mostly iterative and not analytical. Mature algorithms for solving general convex optimization problems in very large datasets is a growing technology, however, many of the existing iterative methods are very fast and efficient, e.g. interior-point methods can solve a problem with hundreds of variables and thousands of constraints within a few seconds.

If an optimization problem can be formulated as a convex problem, it can be solved efficiently. On the other hand, the process of recognizing convex optimization problems, or non-convex problems that can be formulated as convex problems, can be a difficult task. Still, once this task is done, any solution method applied to a convex optimization problem guarantees finding the unique optimal solution in computationally efficient (worst-case polynomial) time. Different classes of optimization problems, such as non-linear optimization (non-linear objective and constraints), local optimization (searching for local optima), and global optimization (exact solution exponential-time search) all fail in producing solutions which are both efficient and accurate. This is what makes convex optimization extremely useful in industrial applications.

We can therefore reformulate a convex optimization problem as:

$$
\begin{aligned}
&\textit{minimize} && f_0(x) \\
&\textit{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\
& && h_i(x) \leq 0, \quad i = 1, \dots, p
\end{aligned}
$$

The problem can be described as "finding the value of x that minimizes $f_0(x)$ among all possible values of x, which satisfy the given conditions (constraints). We call $x \in \mathbf{R}^n$ the *optimization variable* and function f_0 the objective function (or cost function). Our constraints are divided into the inequality constraints $f_i(x) \leq 0$ and equality constraints $h_i(x) \leq 0$, with f_i and h_i being the inequality and equality constraint functions, respectively. In the case of the absence of all constraints, we are in the scenario of an *unconstrained* optimization problem. The set of points for which the objective function and the constraints functions are defined, is the *domain* of the optimization problem. A *feasible* point x or a *feasible solution*, is a value that satisfies all problem constraints. A problem is called feasible when it has at least one feasible solution, infeasible otherwise. All feasible solutions to a problem are called the *feasible set*. The optimal value p^* to the problem can be defined has:

$$p^* = \inf \{f_0(x) | f_i(x) \leq 0,\ i = 1, \dots, m,\ h_i(x) = 0,\ i = 1, \dots, p\}.$$

Now consider the following problem formulation:

$$
\begin{aligned}
&\textit{minimize} && f_0(x) \\
&\textit{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\
& && a_i^T(x) = b_i, \quad i = 1, \dots, p
\end{aligned}
$$

We refer to the precedent problem formulation as an optimization problem in *standard form*, in which we adapt the convention that the righthand side of the inequality and equality constraints are zero. When in standard form, it requires that the objective function and the inequality constraint functions are convex, and that the equality constraint functions are affine. The feasible set of a convex optimization problem is also convex, therefore we optimize a convex objective function over a convex set.

With a slight change in notation, we can also treat a *concave maximization problem* as a convex minimization problem:

$$\begin{aligned} &\textit{maximize} && f_0(x) \\ &\textit{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && a_i^T x = b_i, \quad i = 1, \dots, p \end{aligned}$$

where the objective function f_0 is concave and the inequality constraints $f_1, \dots, f_m$ are convex.

Equality constraints and slack variables:

Equality constraints can be introduced into a convex optimization problem on the condition that they are linear, so that the resulting problem will remain convex. The introduction of slack variables gives us the new constraints $f_i(x) + s_i = 0$ and, since equality constraints must be affine in a convex problem, f_i must be affine as well.

(Sources: [5], [6])

2.2.1. Local and global optima

As already seen, the critical advantage of convex optimization is that a local optimum is also the global optimum. Suppose that x is a local optimum for a convex optimization problem. We should, therefore, have:

$$f_0(x) = \inf\{f_0(z) | \, z \text{ feasible}, \|z - x\|_2 \leq R\},$$

for some $R > 0$. Suppose also that x is not a global optimum, such that there is a feasible y such that $f_0(y) < f_0(x)$, which implies that $\|y - x\|_2 > R$. Now consider the point z such that:

$$z = (1 - \theta)x + \theta y, \qquad \theta = \frac{R}{2\|y - x\|_2}.$$

Then, we shall have $\|z - x\|_2 = \frac{R}{2} < R$, which implies that z is feasible. By convexity of f_0 we have:

$f_0(z) \leq (1-\theta) f_0(x) + \theta f_0(y) < f_0(x)$,

which contradicts the initial premise on $f_0(x)$. Therefore, there exists no feasible y, for which $f_0(y) < f_0(x)$, i.e. x is a global optimum.

2.2.2. Optimality criterion for a differentiable objective function

Let f_0 be the differentiable objective function of a convex optimization problem. Then, for all $x, y \in dom\ f_0$, we have:

$f_0(y) \geq f_0(x) + \nabla f_0(x)^T (y - x)$.

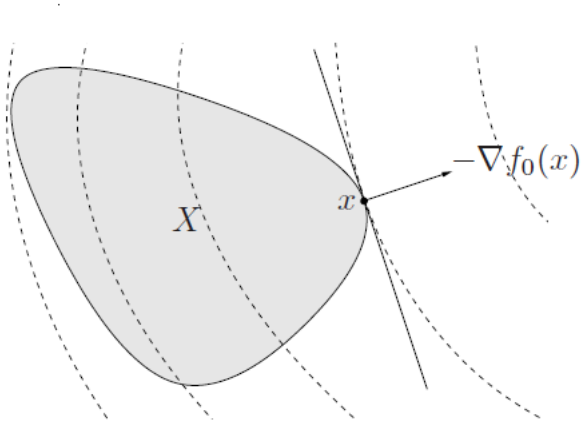
Now, suppose that X is the feasible set:

$X = \{x | f_i(x) \leq 0, i = 1, \dots, m,\ \ h_i(x) = 0, i = 1, \dots, p\}$.

Then, for all $x \in X$, x is optimal iff:

$\nabla f_0(x)^T (y - x) \geq 0$ for all $y \in X$.

Geometrically, this optimality criterion implies that $-\nabla f_0(x)$ is a supporting hyperplane to the feasible set X at point x (shown in figure blow), given that $\nabla f_0(x) \neq 0$.



Geometric representation of optimality condition, where the feasible set X is shown as shaded lines. Level curves of f_0 are shown as dashed lines and point x is optimal. [5]

Proof:

Suppose that $x \in X$ satisfies the precedent optimality condition. If $y \in X$, then $f_0(y) \geq f_0(x)$, and x is an optimal point for the convex optimization problem in standard form. Conversely, if x is an optimal point but the optimality condition does not hold, i.e. there exist $y \in X$ for which:

$$\nabla f_0(x)^T(y - x) < 0$$

Now, suppose that the point $z(t) = ty + (1 - t)x$, with parameter $t \in [0,1]$. Then $z(t)$ is feasible, since the feasible set is convex and $z(t)$ is on the line segment between x and y. Then, for a small and positive t, we have:

$$f_0\big(z(t)\big) < f_0(x),$$

which proves that x is not optimal. This can be shown by:

$$\frac{d}{dt} f_0\big(z(t)\big)\,\big|_{t=0} = \nabla f_0(x)^T(y - x) < 0,$$

which implies that, for a small positive t, is holds that $f_0\big(z(t)\big) < f_0(x)$.

(Sources: [5], [6])

2.2.3. Unconstrained problems

In an unconstrained problem, it holds that $m = p = 0$, and the optimality condition becomes the necessary and sufficient condition:

$$\nabla f_0(x) = 0$$

which must hold in order for x to be optimal.

Proof:

Suppose that x is optimal, i.e. $x \in dom\ f_0$ and for all feasible y it holds that:

$\nabla f_0(x)^T (y - x) \geq 0.$

Given that f_0 is differentiable, its domain is open, therefore all y sufficiently close to x will be feasible. Let $y = x - t\nabla f_0(x)$, $t \in R$, with t small and positive such that y is feasible and:

$\nabla f_0(x)^T (y - x) = -t\|\nabla f_0(x)\|_2^2 \geq 0,$

from which we derive that $\nabla f_0(x) = 0$.

Depending on whether or not there are optimal points for $\nabla f_0(x) = 0$, we say that the problem is *unbound below* or that the optimal value is *finite but not attained*.

2.2.4. Epigraph problem form

A convex problem can be formulated in *epigraph form* as such:

$$\begin{aligned} &\textit{minimize} \quad t \\ &\textit{subject to} \quad f_0(x) - t \leq 0, \\ &\qquad\qquad f_i(x) \leq 0, \quad i = 1, \dots, m \\ &\qquad\qquad a_i^T(x) = b_i, \quad i = 1, \dots, p \end{aligned}$$

The objective function is linear (therefore convex), and the new constraint function $f_0(x) - t$ is convex in (x, t), which implies that the epigraph form problem is also convex.

The linear objective function is said to be "universal" for convex problems due to the fact that any convex problem can be transformed into a problem with linear objective. The epigraph form can simplify a convex problem and its algorithmic solution by assuming that the objective function is linear.

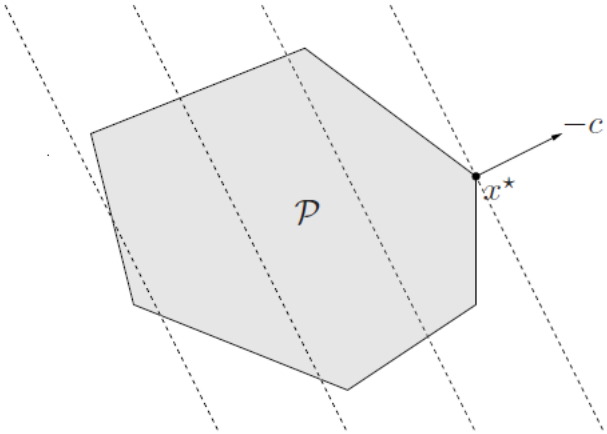
(Sources: [5], [6])

2.2.5. Linear optimization problems

Linear optimization (or linear programming) is another important class of optimization problems where the objective function and its constraints are all affine. The general linear program (LP) has the form:

$$\begin{aligned} &\textit{minimize} && c^T x + d \\ &\textit{subject to} && Gx \leq h \\ & && Ax = b \end{aligned}$$

where $G \in \mathbf{R}^{m \times n}$ and $A \in \mathbf{R}^{p \times n}$. The constant d is sometimes omitted as it does not affect the feasible set. We can also refer to maximization problem with affine objective and constraints as a LP, since by minimizing $-c^T x - d$ we also maximize an affine objective $c^T x + d$. In a geometric interpretation, the feasible set of a LP is polyhedron $\mathcal{P}$ and the objective to be minimized is the affine function $c^T x + d$ over $\mathcal{P}$.



Geometric interpretation of a LP. The shaded polyhedron is the feasible set $\mathcal{P}$, the linear objective $c^T x$ and its level curves are hyperplanes (dashed lines) orthogonal to c. The optimal point x^* is the point in $\mathcal{P}$ that goes as far as possible in the direction $-c$. [5]

(Sources: [5])

Example in Linear optimization: Chebysev inequalities

Let $p \in \mathbf{R}^n$ be a vector describing the probability distribution of a discrete random variable x on a set $\{u_1, \ldots, u_n\} \subseteq \mathbf{R}$ with n elements. So for vector p, we have:

$p_i = prob(x = u_i)$,

hence, it holds that p satisfies $p \succcurlyeq 0$ and $1^T p = 1$. Conversely, the two latter premises imply that p defines a probability distribution for x. We can assume that u_i are known and fixed, while the distribution p is unknown. If f is a function of x, its expected value:

$E[f] = \sum_i^n p_i f(u_i)$

will be a linear function of p. Even though p is unknown, we have the prior knowledge that upper and lower bounds on expected values of some functions of x and probabilities of some subsets of $\mathbf{R}$, which can be expressed as linear inequality constraints on p:

$$\alpha_i \leq \alpha_i^T p \leq \beta_i, \qquad i = 1, \dots, m.$$

The problem consists of Chebyshef inequalities consists of assigning lower and upper bounds on $E[f_0(x)] = a_0^T p$, where f_0 is some function of x. Finding the lower bound comes down to solving the following LP:

$$\begin{aligned} &\textit{minimize} && a_0^T p \\ &\textit{subject to} && p \succcurlyeq 0, \quad 1^T p = 1 \\ &&& a_i \leq a_i^T p \leq \beta_i, \quad i = 1, \dots, m \end{aligned}$$

The optimal value of this linear program yields the lowest possible value of $E[f_0(X)]$, for any probability distribution that satisfies the conditions of prior knowledge. Equivalently, finding the optimal upper bound consists of maximizing $a_0^T p$, subject to the same constraints.

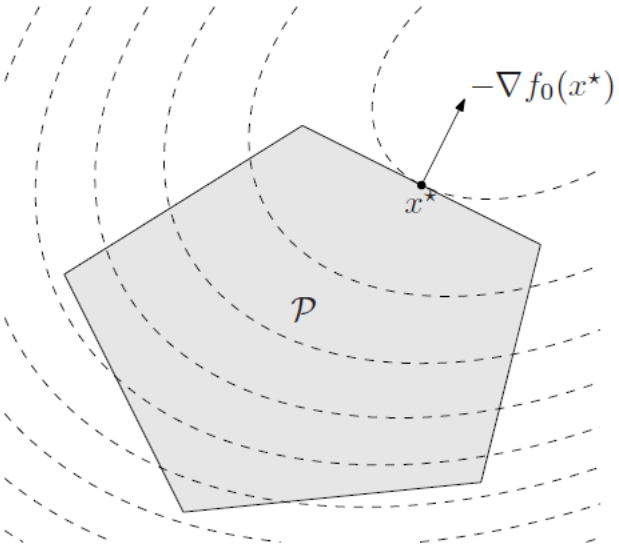
(Sources: [5])

2.2.6. Quadratic optimization

A convex optimization problem is called a *quadratic problem* (QP) if its objective function is quadratic and its constraints are affine. A quadratic problem is expressed in the following form:

$$\begin{aligned} &\textit{minimize} && \left(\frac{1}{2}\right) x^T P x + q^T x + r \\ &\textit{subject to} && Gx \preccurlyeq h \\ &&& Ax = b \end{aligned}$$

where $P \in \mathbf{S}_i^n$, $G \in \mathbf{R}^{m \times n}$, and $A \in \mathbf{R}^{p \times n}$. In a quadratic program, we minimize a convex quadratic function over a polyhedron (as illustrated).



Quadratic program: A Geometric illustration where the feasible set $\mathcal{P}$ is shown as a shaded polyhedron and the quadratic objective function as dashed contour lines. [5]

When the constraint functions are all also convex, we have the following problem formulation:

$$\begin{aligned} &\textit{minimize} && \left(\frac{1}{2}\right) x^T P x + q^T x + r_0 \\ &\textit{subject to} && \left(\frac{1}{2}\right) x^T P_i x + q_i^T x + r_i \leq 0, \quad i = 1, \dots m \\ & && Ax = b \end{aligned}$$

where $P_i \in \mathbf{S}_+^n, \quad i = 0, 1, \dots, m$. Here, the problem is called a *quadratically constrained* quadratic program (QCQP), and the minimization of a convex quadratic function takes place over a feasible region that is geometrically the intersection of ellipsoids.

Linear programs can be seen as a special case of quadratic programs when, in the quadratic objective function, it holds that $P = 0$.

(Sources: [5], [6])

Example in Quadratic optimization: Least-squares and regression

The least-squares problem is an unconstrainted optimization problem where the objective function is a sum of squares of terms of the form $a_i^T x - b_i$.
More specifically, it consists of minimizing the convex quadratic function:

$minimize\ \|Ax - b\|_2^2 = x^T A^T Ax - 2b^T Ax + b^T b$.

It is an unconstrained quadratic program with a long history of applications in parameter learning, where it is commonly known as *regression analysis* or *least-squares approximation*. This problem is simple enough to have the analytical solution $x = A^\tau b$, where A^τ is the pseudo-inverse of A.

When linear inequality constraints are added to the least squares problem, the problem is transformed into *constrained least-squares* or *constrained regression*, and there is no longer an analytical solution. Let us consider regression with upper and lower bounds on the variables:

$minimize\ \|Ax - b\|_2^2$
s.t: $l_i \leq x_i \leq u_i, \quad i = 1, \dots, n,$

which is a quadratic programming problem.

The least-squares problem is fundamental in regression analysis and related methods involving statistical learning, parameter estimation, and data fitting. Its statistical interpretation is the Maximum Likelihood Estimation of the observed data (vector x), given linear measurements that have been corrupted by Gaussian noise. The interpretation of least-squares as an optimization problem requires that the objective function is quadratic and positive semidefinite. Two well-known variations of this problem are described below.

Weighted least-squares with the optimization criterion:

$$\sum_i^k w_i(a_i^T x - b_i^2)$$

where $w_1, \dots, w_k$ are positive and set according to the relative importance of individual predictors. Its statistical interpretation is that the Gaussian errors have unequal variances.

Another common variation is least squares with *regularization*, a version of the optimization problem where extra cost terms are added to the objective function.

Simplest case of *regularized least-squares* with the a positive multiple of the sum of squares of all independent variables added to the objective function:

$$\sum_{i}^{k} w_i(a_i^T x - b_i^2) + \rho \sum_{i=1}^{n} x_i^2$$

where $\rho > 0$, which is manually set in order to control the balance between the original optimization term and the regularization term. The purpose of the extra terms is to penalize large values of x, which may derive a sensible solution when the minimization of the first term fails to do so. Regularization is used in statistical estimation when the observed data is given a prior distribution or when the size of data causes maximum likelihood estimation to overfit.

The exact solution of the least-squares problem comes down to solving the set of linear equations:

$$(A^T A)x = A^T b,$$

from which we obtain the following analytical solution:

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

The time complexity of the least-squares solution is $O(n^2 k)$, where k is a known constant.

(Sources: [5], [6])

2.2.7. Vector optimization

The standard form of a convex problem can be extended to include vector valued objective and constraint functions. We call this scenario a *vector optimization* problem and denote it as:

$$\begin{array}{ll} \textit{minimize (with respect to K)} & f_0(x_0) \\ \textit{subject to} & f_i(x) \leq 0, \;\; i = 1, \dots, m \\ & h_i(x) = 0, \;\; i = 1, \dots, p \end{array}$$

where $x \in \mathbf{R}^n$ is the optimization variable, $K \subseteq \mathbf{R}^Q$ is a proper cone, $f_0 : \mathbf{R}^n \to \mathbf{R}^q$ is the objective function, $f_i : \mathbf{R}^n \to \mathbf{R}$ are the inequality constraint functions, and $h_i : \mathbf{R}^n \to \mathbf{R}$ are the equality constraint functions. The difference from the standard (scalar) convex optimization problem is that, in the case of vector optimization, the objective function takes values in $\mathbf{R}^q$, therefore multiple objective values are being compared via a proper cone K. The vector optimization problem is a convex problem if the objective f_0 and its associated equality and inequality constraints are all K-convex.

Optimal points and values in vector optimization

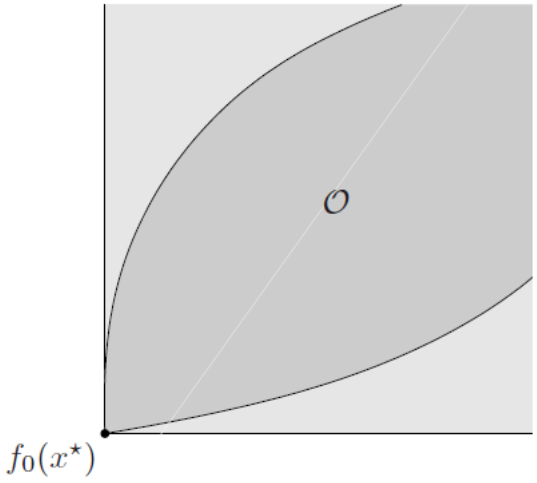
Suppose that we have the following objective values of feasible points:

$$\mathcal{O} = \{f_0(x) | \exists x \in \mathcal{D}, f_i(x), i = 1, \dots, m, h_i(x) = 0, i = 1, \dots, p\} \subseteq \mathbf{R}^q,$$

which we call the set of *achievable* objective values. If there exists a feasible set x such that $f_0(x) \preccurlyeq_K f_0(y)$, for all feasible y, then the set $\mathcal{O}$ has a minimum value and we say that x is optimal, with $f_0(x)$ being the problem's unique optimal value. Therefore, if x^* is an optimal point, then the objective $f_0(x^*)$ can be compared to the objective of any other feasible point. In this case, a point x^* is optimal iff it is feasible and it holds that:

$$\mathcal{O} \subseteq f_0(x^*) + K$$

The set K can be interpreted as the set of values that are worse than or equal to $f_0(x^*)$ and, as illustrated below, the precedent condition implies that every achievable value falls in the set $\mathcal{O}$.



The set of achievable values $\mathcal{O}$ is depicted shaded while the point $f_0(x^*)$ is the optimal value as yielded by the optimal point $x^*$and the objective function f_0. [5]

It should be noted that the majority of vector optimization problems do not possess an optimal point and an optimal value.

(Sources: [5])

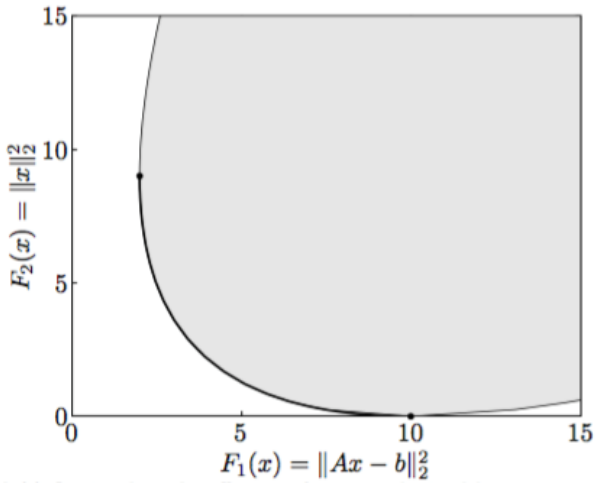
Example in Vector optimization: Regularized least-squares

Suppose that we are given $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$, and that we want to find $x \in \mathbf{R}^{\mathrm{n}}$ while taking into account two quadratic objectives:

- $F_1(x) = \|Ax - b\|_2^2 = x^T A^T Ax - 2b^T Ax + b^T b$ is the measure of misfit between Ax and b,
- $F_2(x) = \|x\|_2^2 = x^T x$ is a measure of the size of x.

The goal is to find x that gives a good fit (small F_1) and is, at the time, not large (small F_2). We formulate this problem as a vector optimization problem with respect to the cone $\mathbf{R}_+^2$, i.e. an unconstrained bi-criterion problem:

$\min f_0(x) = (F_1(x), F_2(x))$.



The optimal trade-off curve of regularized least-squares is shown darker at the lower left part of the boundary. The shaded set represents the set of achievable values.

We scalarize the problem by taking $\lambda_1 > 0$ and $\lambda_2 > 0$ and then by minimizing the scalar weighted sum objective:

$\lambda^T f_0(x) = \lambda_1 F_1(x) + \lambda_2 F_2(x)$
$= x^T(\lambda_1 A^T A + \lambda_2 I)x - 2\lambda_1 b^T Ax + \lambda_1 b^T b,$

which yields:

$x(\mu) = (\lambda_1 A^T A + \lambda_2 I)^1 \lambda_1 A^T b = (A^T A + \mu I)^{-1} A^T b,$

where $\mu = \lambda_2/\lambda_1$. For any $\mu > 0$, this point will be the Pareto optimal for the bi-criterion problem. The value of $\mu = \lambda_1/\lambda_2$ can then be interpreted as the relative weight of F_1 in comparison to F_2.

This is a method to produce all Pareto optimal points those two associated to the extremes $\mu \to \infty$ and $\mu \to 0$. The former extreme will yield the Pareto optimal solutions $x = 0$, via scalarization with $\lambda = (0,1)$, and the latter extreme will yield the Pareto optimal solution $A^{\mathcal{T}}b$, with $A^{\mathcal{T}}$ being the pseudo-inverse of A. The Pareto optimal solution is the limit of the optimal solution of the scalarized problem as $\mu \to 0$, i.e. $\lambda \to (1,0)$.

(Sources: [5])

2.3. Parametric statistical estimation

2.3.1. Maximum Likelihood Estimation (MLE)

Suppose that we have a family of probability distributions on $\mathbf{R}^m$, which are indexed by a vector $x \in \mathbf{R}^m$ with densities given by function $p_x(.)$. If p_x is a function of x, for any fixed $y \in \mathbf{R}$, then the function p_x becomes a *likelihood function*. Due to properties of the logarithmic function, it is usually easier to work with the logarithm of p_x, which is denoted by l and called the *log-likelihood function*:

$$l(x) = \log(p_x(y)).$$

Any constraints on the values of the parameter x represent the *prior knowledge* on x, which is the domain of the likelihood function. To account for these constraints, we can simply assign $p_x(y) = 0$ whenever the constraints on x are violated.

Suppose we were to estimate the value of parameter x, based on an observed sample y draw from the distribution. The *maximum likelihood estimation* will estimate x by solving the following problem:

$$\hat{x}_{ML} argmax_x p_x(y) = argmax_x l(x),$$

i.e. the optimal value of x is the one that maximizes the likelihood function for the observed values of y. The prior information on x, represented as $x \in C \subseteq \mathbf{R}^n$, is the constraint $x \in C$ that can be added explicitly or implicitly by setting $p_x(y) = 0$ if $x \notin C$.

We can formally express the optimization problem of finding the maximum likelihood estimate for the parameter vector x as:

$$maximize\ l(x) = \log\big(p_x(y)\big)$$
$$subject\ to\ x \in C$$

where $x \in C$ is expresses the prior information on vector x. In this formulation, vector x (the parameter of the probability density) is the optimization variable and vector y (observed data) an optimization parameter.
The maximum likelihood estimation problem is a convex problem *when the log-likelihood function is concave* for every observed value in y and the set C is described by linear equality and convex inequality constraints. These convexity conditions are met in many estimation problems, for which we can use convex optimization via MLE to obtain "optimal" point estimates.

(Sources: [5], [6])

MLE estimation of linear measurements with IID noise

Suppose we have the following linear model:

$$y_i = a_i^T x + v_i, \qquad i = 1, \dots, m$$

where $x \in \mathbf{R}^n$ is the parameter vector to be estimated, $y_i \in \mathbf{R}$ is the observed data, and v_i are the IID measurements errors (noise) with density p on $\mathbf{R}$. The likelihood function can be expressed as:

$$p_x(y) = \prod_{i=1}^{m} p(y_i - a_i^T x)$$

and the log-likelihood as:

$$l(x) = \log\big(p_x(y)\big) = \sum_{i=1}^{m} \log\left(p(y_i - a_i^T x\right).$$

Then any optimal point of the following problem:

$$maximize \sum_{i=1}^{m} \log\Big(p(y_i - a_i^T x)\Big)$$

will be the ML estimate. The problem is convex if the density function p is log-concave and is in the form of a penalty approximation problem with penalty $-\log(p)$.

2.3.2. Maximum a Posteriori estimation (MAP)

In a Bayesian setting, the posterior distribution of a parameter x can be derived analytically via Bayes rule or approximated via stochastic simulation (Markov Chain Monte Carlo):

$$p(x|y) = \frac{p(x)p(y;x)}{p(y)}$$

where random variable x is the vector to be estimated and random variable y is the observed data. The main philosophical difference with classical estimation (MLE) is that x is seen as a random variable rather than a fixed parameter and will therefore be estimated as a distribution rather than a single value. This distribution, denoted $p(x|y)$, is called the posterior distribution. However, sometimes all we need is a single value for x, while the posterior $p(x|y)$ does not automatically give us a point estimate $\hat{x}$. In the scenario of a non-symmetric posterior where the mode cannot be computed analytically, an emerging optimization problem has to do with finding the point $\hat{x}$ where the posterior becomes maximum:

$$\hat{x}|y = argmax_x\, p(x|y)$$

In that respect, maximum a posteriori can be viewed as the Bayesian equivalent of maximum likelihood estimation with a prior probability density on the parameter x. The prior density of x is expressed as:

$$p_x(x) = \int p(x,y)dy$$

The above density is the prior distribution of the values of x before observing y. The prior density of y is given by:

$$p_y(y) = \int p(x,y)dx,$$

which is the prior information about the observed data.
The conditional (predictive) density of y given x is expressed as:

$$p_{y|x}(y;x) = \frac{p(x,y)}{p_x(y)}$$

In MAP estimation, the conditional density $p_{y|x}$ is the parameter-dependent equivalent of MLE's marginal density p_x.

The conditional (posterior) density of x given y is expressed as:

$$p_{x|y}(x;y) = \frac{p(x,y)}{p_y(y)} = p_{y|x}(x;y)\frac{p_x(x)}{p_y(y)}$$

The substitution of the observed data y into $p_{x|y}$, gives us the posterior distribution of x.

Therefore, the MAP estimate of x given y is given by:

$$\hat{x}_{MAP} = argmax_x\, p_{x|y}(x;y)$$
$$= argmax_x\, p_{y|x}(x;y)p_x(x)$$

$$= argmax_x \, p(x, y)$$

which leads to the optimization problem described at the beginning of this section. By taking as the estimate of x the value that maximizes the conditional distribution of $x|y$. The only difference between the calculation of MAP and MLE is the prior density $p_x(x)$. In MLE, the optimization is applied directly on the likelihood function, whereas in MAP it is applied on the product of the likelihood function and the prior density. When the prior density if uniform then the $p_x(x)$ becomes a constant and the MAP estimation overlaps with MLE. Taking the logarithm of $p(x, y)$ we can express the MAP estimate as:

$$\hat{x}_{MAP} = argmax_x \left(\log\left(p_{y|x}(x, y)\right) + \log\left(p_x(x)\right)\right)$$

The first term in the above expression is the log-likelihood function and the second is the logarithm of the prior density that penalizes less probable values of x (when $p_x(x)$ small).
Without considering the philosophical debate between frequentist and Bayesian statistics, the only mathematical difference between MAP and MLE estimates is the prior density term in the optimization problem. Consequently, for any MLE problem with a concave log-likelihood, the addition of a convex prior density will preserve convexity in the resulting MAP problem.

(Sources: [5], [6])

MAP estimation of linear measurements with IID noise in a Bayesian setting

Let $x \in \mathbf{R}^n$ and $y \in \mathbf{R}^n$ are linked by the following relation:

$$y_i = a_i^T x + v_i, \quad i = 1, \dots, m$$

where v_i is a vector of independent and identically distributed random variables (IID) with distribution p_v on $\mathbf{R}$, with x having a prior distribution p_x on $\mathbf{R}^n$. The joint distribution of x and y is:

$$p(x,y) = p_x(x)\prod_{\text{i}}^{\text{m}} p_v(y_i - a_i^T x)$$

The MAP estimate is the solution of the following optimization problem:

$$maximize\ \log\big(p_x(x)\big) + \sum_i^m \log p_v(y_i - a_i^T x)$$

The above problem is convex if the density functions for p_x and p_v are log-concave. By removing the term $\log\big(p_x(x)\big)$, we have the objective function of the MLE scenario.
To demonstrate this application, suppose that:

$x \sim Normal(\bar{x}, \Sigma)$
$v_i \sim Uniform(-a, a)$

Then the MAP estimate is reduced to the following quadratic problem:

$$minimize\ \ (x-\bar{x})^T\Sigma^{-1}(x-\bar{x})$$
$$subject\ to\ \ \|Ax-y\|_\infty \le a$$

with optimization variable x.

(Sources: [5])

2.4. Statistical learning

2.4.1. Classification

In statistical classification, we have two sets of points in $\mathbf{R}^n$, $\{x_1, \dots, x_N\}$ and $\{y_1, \dots, y_M\}$. We therefore wish to find a function $f: \mathbf{R}^n \to \mathbf{R}$ that is positive on one set and negative on the other:

$$f(x_i) > 0, i = 1, \dots, N$$

$$f(y_i) < 0, i = 1, \dots, M$$

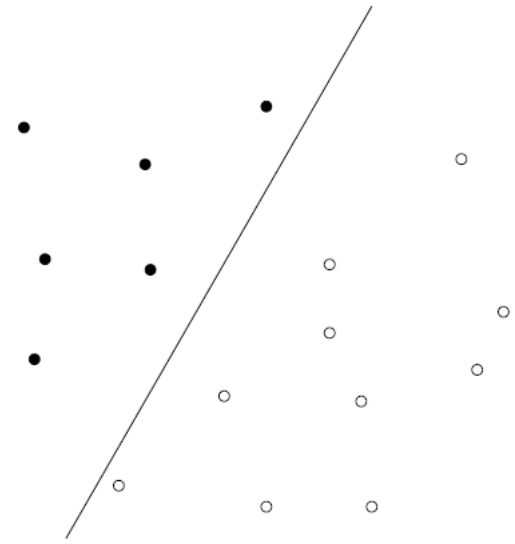
When these inequalities are satisfied, we say that f classifies, separates, or discriminates the two sets. Accordingly, we speak of a weak separation holds when the weak versions of the inequalities are satisfied.

We speak of linear discrimination, when the two sets of points can be separated by a straight line. Thus, we seek an affine function $f(x) = a^T x - b$ that classifies the points:

$$a^T x_i - b > 0, \quad i = 1, \dots, N$$
$$a^T y_i - b < 0, \quad i = 1, \dots, M$$

In the geometrical interpretation, we seek a hyperplane that separates the points from two sets. We already know that the strict inequalities are homogenous in a and b, they are feasible iff the set of nonstrict linear inequalities is also feasible:

$$a^T x_i - b \geq +1, \quad i = 1, \dots, N$$
$$a^T y_i - b \leq -1, \quad i = 1, \dots, M$$



The open and filled circles represent points $x_1, \dots, x_N$ and $y_1, \dots, y_M$, respectively, separated by the affine function f. [5]

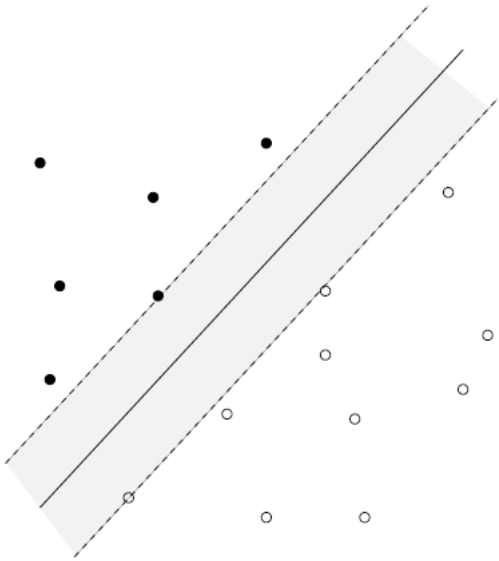
(Sources: [5], [7])

2.4.2. Robust linear discrimination

The *robust linear discrimination problem* is a linear discrimination where we seek to optimize some measure of robustness. For example, this measure could be the output of a function that gives the maximum gap between the sets of points of two (or more) classes. Generally, an affine classifying function $f(x) = a^T x - b$ is equivalent to a set of linear inequalities in variables a and b that define f. The polyhedron of affine functions that linearly discriminates the two sets will be optimizing the measure of robustness. To find the maximum gap between values at the points x_i and y_i, we should first normalize or scale a and b by a positive constant and solve the emerging convex optimization problem:

$$
\begin{aligned}
&maximize\ t \\
&subject\ to\ a^T x_i - b \geq t, \quad i = 1, \dots, N \\
&a^T y_i - b \leq -t, \quad i = 1, \dots, M \\
&\|a\|_2 \leq 1
\end{aligned}
$$

The optimal solution t^* of the above problem is positive if and only if the two sets of points (classes) are linearly separable. When this holds, the inequality $\|a\|_2 \leq 1$ is always tight at the optimum, such that $\|a^*\|_2 = 1$.



The linear discrimination problem with the affine function that must yield the largest separation gap between the two sets of points (classes) [5]

(Sources: [5], [7])

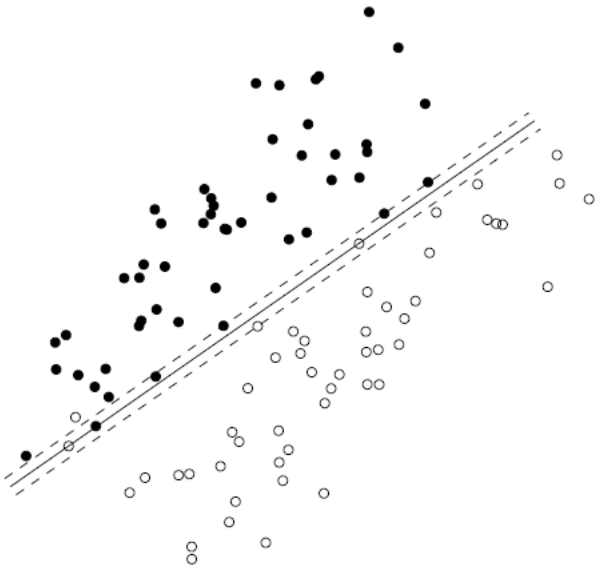
2.4.3. Approximate linear discrimination via support vector classification

In the scenario where it is not possible to linearly separate the two classes (sets of points), one can seek an affine function that approximates separation. For example, one such objective would be to minimize the number of misclassified points. The exact solution to this problem would be through hard combinatorial optimization, though a heuristic for approximate linear discrimination can be obtained by *support vector classifiers*. We can relax the constraints of a feasibility problem by introducing nonnegative variables $u_1, \dots, u_N$ and $v_1, \dots, v_M$ thus forming the inequalities:

$$a^T x_i - b \geq 1 - u_i, \quad i = 1, \dots, N$$
$$a^T y_i - b \leq -(1 - v_i), \quad i = 1, \dots, M$$

The values of u_i and v_i can be interpreted as the degree that the constraints $a^T x_i - b \geq 1$ and $a^T y_i - b \leq -1$ are being violated. Therefore, we are interested in finding a, b and sparse nonnegative u and v which satisfy those inequalities. For this purpose, we may use a heuristic approach which minimizes the sum of the variables u_i and v_i by solving the following linear problem:

$$\begin{aligned} &minimize \;\; 1^T u + 1^T v \\ &subject\ to \;\; a^T x_i - b \geq 1 - u_i, \;\; i = 1, \dots, N \\ &a^T y_i - b \leq -(1 - v_i), \;\; i = 1, \dots, M \\ &u \geqslant 0, v \leqslant 0 \end{aligned}$$



Approximate solution to the linear discrimination problem via linear programming. The two sets of points are not linearly separable and one of the points gets inevitably misclassified. [5]

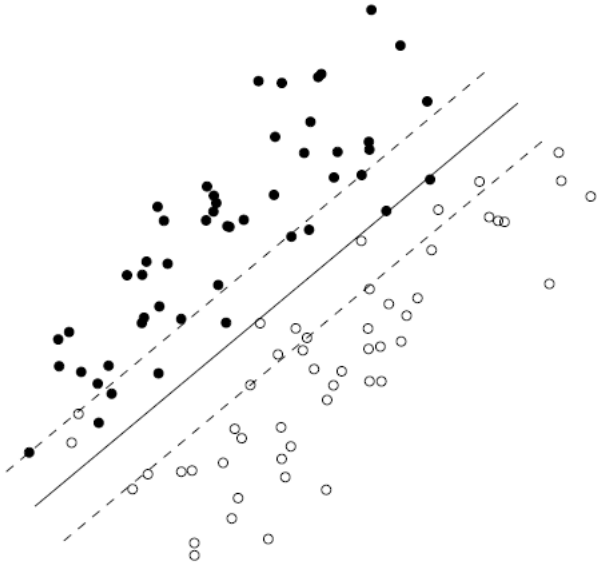
In the above example, the affine function $a^T z - b$ misclassifies 1 out of 100 points and when $0 < u_i < 1$ it holds that points x_i and y_i might be correctly classified but violate their corresponding inequality constraints. Therefore, the objective function of that linear problem can be seen as a relaxation of the number of points that violate their constraints or a relaxation of the number of misclassified points plus the number of correctly classified points which lie in the slab defined by:

$$-1 < a^T z - b < 1$$

Thus, there is a tradeoff between misclassified points and the width of the slab $\{z \mid -1 < a^T z - b < 1\}$, which is obtained by $\frac{2}{\|a\|_2}$. Consequently, the support vector classifier for the sets $\{x_1, \dots, x_N\}$ and $\{y_1, \dots, y_N\}$ is defined by the solution of the following linear problem:

$$\begin{aligned} &minimize \quad \|a\|_2 + \gamma(1^T u + 1^T v) \\ &subject\ to \quad a^T x_i - b \geq 1 - u_i, \quad i = 1, \dots, N \\ &\qquad\qquad a^T y_i - b \leq -(1 - v_i), \quad i = 1, \dots, M \\ &\qquad\qquad u \succcurlyeq 0, \quad v \succcurlyeq 0 \end{aligned}$$

where γ is a positive value expressing the relative weight of misclassified points. The problem in question is illustrated below.



Approximate solution to the linear discrimination problem via the support vector classifier (shown as the solid line) with three points being misclassified. [5]

The first term of the objective function is proportional to the width of the slab while the second term is a convex relaxation for the number of misclassified points.

(Sources: [5], [7])

2.5. Deterministic algorithms for unconstrained minimization

Suppose we are in the scenario of an arbitrary unconstrained optimization problem:

$$minimize\ f(x)$$

Let the objective function $f: \mathbf{R}^n \to \mathbf{R}$ be convex and twice differentiable. The problem is solvable, i.e. it has a unique optimal solution, represented by the optimal point x^*. For convenience, let $p^* = f(x^*)$. Given that f is differentiable and convex, x^* will exist under the necessary and sufficient condition:

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

The solution to the unconstrained problem is therefore reduced to finding the point where the above optimality condition is satisfied. In some problem instances the solution may be derived analytically but most of the time the problem is solved by an iterative numerical method that computes a *minimizing sequence* of points $x^{(0)}, x^{(1)}, \ldots \in dom(f)$ with $f\left(x^{(k)}\right) \to p^*$ as $k \to \infty$. The algorithm converges when $f\left(x^{(k)}\right) - p^* \leq \epsilon$, with $\epsilon > 0$ being a tolerance parameter. A starting point for this search must lie in $dom\, f$, while the sublevel set $S = \{x \in dom\, f \mid f(x) \leq f\left(x^{(0)}\right)\}$ must be closed (which is satisfied for all $x^{(0)} \in dom\, f$, if f is closed).

(Sources: [5], [6])

2.5.1. Gradient descent method

Basic descent

A descent-type of algorithm is one that produces a minimizing sequence $x^{(k)}$, $k = 1, .,$ where:

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

with $t^{(k)} > 0$. Here, Δx is a vector in $\mathbf{R}^n$ which is called the *step* or *search direction*, $k = 0, 1, \ldots$ is the number of iterations, $t^{(k)} \geq 0$ is the step size at iteration k. It is often convenient to omit superscripts and focus on a single iteration of the algorithm, by replacing:

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(x)}$$

with the lighter notation:

$$x^+ = x + t\Delta x$$

Generally, for any descent method it holds that:

$$f\left(x^{(k+1)}\right) < f(x^{(k)})$$

except in the case of x^*. Therefore, for all k we have $x^{(k)} \in S$ (the initial subset level) and $x^{(k)} \in dom\, f$. By convexity, we have that:

$$\nabla f\left(x^{(k)}\right)^T \left(y - x^{(k)}\right) \geq 0$$

implies:

$$f(y) \geq f(x^{(k)})$$

Therefore, the search direction in a descent algorithm much satisfy:

$$\nabla f\left(x^{(k)}\right)^T \Delta x^{(k)} < 0.$$

In other words, the search direction must make an acute angle with the negative gradient in order to be called a *descent direction*. The algorithm of the general descent method alternates between two steps:

- Determine descent direction Δx
- Select step size t

The pseudocode for the general descent algorithm is provided below:

Algorithm: General descent method

given a starting point $x \in \mathbf{dom}\, f$.
repeat
1. Determine a descent direction Δx.
2. *Line search.* Choose a step size $t > 0$.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

By selecting step size t during *line search*, we determine where the next iteration will be along the line $\{x + t\Delta x \mid t \in R_+\}$. Different variants of the basic descent may have the stopping criterion checked after the descent direction Δx has been computed. A typical form of the stopping criterion is:

$$\|\nabla f(x)\|_2 \leq \eta$$

for small positive values of η.

During *exact line search*, we use the line search method and choose t to minimize f along the ray $\{x + t\Delta x \mid t \geq 0\}$:

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

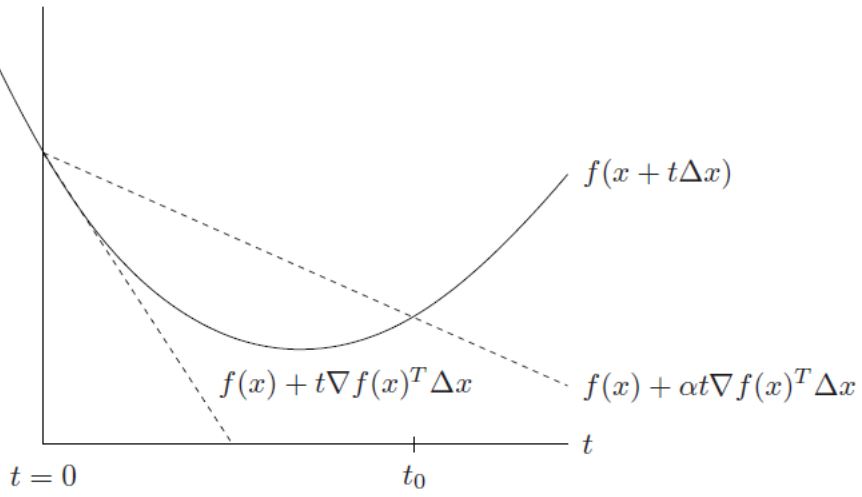
Exact line search is useful when the cost of minimization is lower than the cost of computing the search direction.

An *inexact* alternative to exact line search is *backtracking line search*, where the step length is chosen to approximately minimize f along the ray $\{x + t\Delta x \mid \geq 0\}$. Exact line search depends on constants α and β, with $0 < \alpha < 0.5$ and $0 < \beta < 1$.

Algorithm: Backtracking line search

given a descent direction Δx for f at $x \in \mathbf{dom}\, f$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
$t := 1$.
while $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x$, $\quad t := \beta t$.

Therefore, the search starts with a unit step size and reduces it by a factor of β until the stopping condition $f(x + t\nabla x) \leq f(x) + at\nabla f(x)^T \Delta x$ is met, hence the name 'backtracking'.



Backtracking line search where the linear extrapolation of f is depicted by the lower dashed line. [5]

(Sources: [5])

Gradient descent

The variant of descent method that uses the negative gradient $\Delta x = -\nabla f(x)$ as its search direction, is called *gradient descent*.

Algorithm: Gradient descent method

given a starting point $x \in \mathbf{dom}\, f$.
repeat
1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

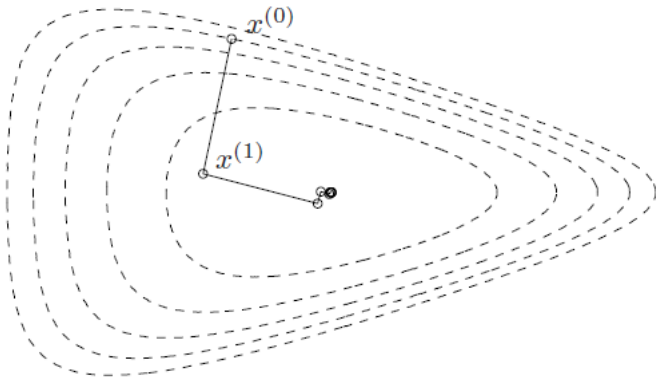
until stopping criterion is satisfied.

We normally use the standard stopping criterion of the form $\|\nabla f(x)\|_2 \leq \eta$, with small and positive η.

A simple convergence analysis for the gradient method following. Let f be strongly convex on S and $\Delta x = -\nabla f(x)$. There are positive constants m and M such that $mI \preccurlyeq \nabla^2 f(x) \preccurlyeq MI, \forall x \in S$. The function $f: \mathbf{R} \to \mathbf{R}$ is defined by $\tilde{f}(t) = f\big(x - t\nabla f(x)\big)$, so f is a function of

step length t in the direction of the negative gradient. By considering only t for which $x - t\nabla f(x) \in S$ and $y = x - t\nabla f(x)$, we have a quadratic upper bound on $\tilde{f}$:

$$\tilde{f}(t) \leq f(x) - t\|\nabla f(x)\|_2^2 + \frac{Mt^2}{2}\|\nabla f(x)\|_2^2.$$



Iterations of the gradient descent algorithm using exact line search. [5]

(Sources: [5])

2.5.2. Newton's method

The Newton's method is essentially a basic descent method that uses the *Newton step* as search direction, however, the stopping criterion is checked right after the search direction is computed (before the update).

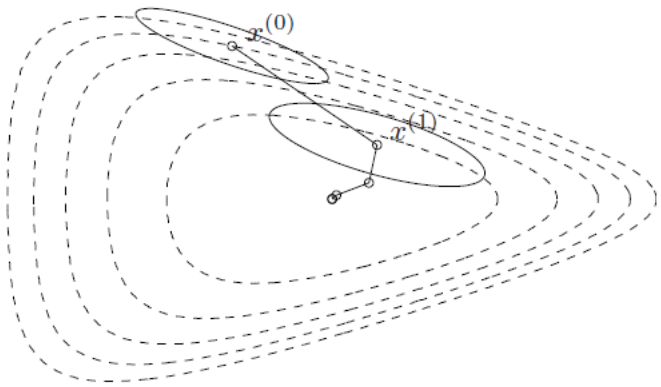
Algorithm: Newton's method

given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.
repeat
1. *Compute the Newton step and decrement.*
   $\Delta x_{\mathrm{nt}} := -\nabla^2 f(x)^{-1}\nabla f(x)$; $\lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1}\nabla f(x)$.
2. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \epsilon$.
3. *Line search.* Choose step size t by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\mathrm{nt}}$.

As before, the objective function f is assumed to be twice continuously differentiable and strongly convex with constant m, such that $\nabla^2 f(x) \leq MI, \forall x \in S$. Suppose that the Hessian of f is Lipschitz continuous on S with constant L:

$$\|\nabla^2 f(x) - \nabla^2 f(y)\|_2 \leq L\|x - y\|_2, \qquad \forall x, y \in S.$$

The Lipschitz constant L is important to the performance of Newton's method. It serves as a bound on the third derivative of f and measures how well f can be approximated by a quadratic model. The coefficient L can be taken as zero for a quadratic function, though any small value of L should make a quadratic function vary slowly.



Iterations of Newton's method using backtracking line search.

(Sources: [5])

Part III

Stochastic Optimization

3. Stochastic optimization

Stochastic optimization (also known as randomized optimization) involves optimization methods that have randomness as a core ingredient [12]. This element of randomness may exist as part of the objective function in the form of random variables (noisy, convex problem), or as part of a stochastic *Neighborhood function* (such as the learning rate function) in the search algorithm (deterministic non-convex problem). Stochastic convex optimization, specifically, refers to the scenario of the noisy objective function.

3.1. Generic stochastic optimization

Stochastic optimization is a major branch of computational statistics. It plays an important role to the design and operation of modern systems as it provides methods of dealing with inherent noise, high nonlinearity, and high dimensionality. Stochastic optimization algorithms are generally applicable to optimization scenarios of the following categories:

- Global optimization: Problems where multiple local optima and complicated constraints are present, often involving a mix of continuous and discrete variables [12]. In this scenario, randomization can be used to escape local optima and approximate convexity in nonconvex functions.

- Noisy optimization: Problems where the gradient is *noisy*, i.e. the underlying objective function contains random variables and only estimates of the function's value at a given point are available. In this noisy scenario, the objective function in considered unknown and can be obtained by approximating the expected value of its random variables via

Monte Carlo simulation. The gradient of the noisy objective function can be estimated via numerical differentiation. [12]

- Sequential learning: Scenarios of statistical machine learning problems where an objective function (Likelihood or Loss) has to be optimized in a sequential manner. Examples of such learning problems would be the sequential estimation of the parameters of a Gaussian distribution via maximum likelihood and the on-line training of the weights of an artificial neural network using backpropagation. Sequential methods allow data points to be processed one at a time before being discarded and are particularly useful in on-line applications where data inputs are naturally sampled sequentially, or applications with very large data-sets where batch processing of all data at once is infeasible. [7]

Comparison between deterministic and stochastic optimization

Algorithms for stochastic optimization apply when: [13]

- There is random noise in the measurements of loss function $L(\boldsymbol{\theta})$, i.e. $L(\boldsymbol{\theta})$ is a noisy objective function
- A random choice (via Monte Carlo) determines the direction of the search at every step of the algorithm

Algorithms for deterministic optimization apply when:

- There is perfect information available in the measurements of loss function $L(\boldsymbol{\theta})$ and its derivatives (if it is differentiable).
- This perfect information adjusts the direction of the search deterministically at every step of the algorithm.

Types of stochastic optimization algorithms

We can consider stochastic optimization methods for both noisy and deterministic objective problems. In the former case, randomness is found in the regression function, and in the latter

case, in the neighborhood function. The scenario of optimizing a regression function is usually that of a convex problem while the deterministic objective function is that of non-convex problems. The latter assumption is made because in the convex deterministic objective problem, we are guaranteed to reach the global optimum by means of a deterministic search (there is no use for a stochastic component).

In that sense, we can divide stochastic optimization algorithms into two groups:

- *Noisy objective optimization*, which can be gradient-based or gradient-free, solving a convex optimization problem with a noisy objective function.
- *Deterministic objective optimization*, which are essentially gradient-free, solving a non-convex optimization problem with a deterministic objective function and a randomized neighborhood function.

We refer to gradient-based algorithms as *stochastic gradient methods* when they integrate the element of randomness to introduce the strengths of convex optimization into stochastic optimization. This work focuses mainly on Convex Gradient-based methods, where convex optimization is a central topic, though some Simulation-based non-convex methods are also presented in chapter "Monte Carlo optimization".

3.2. Stochastic Approximation

Stochastic approximation is the area of numerical analysis that makes up the cornerstone of stochastic gradient-based methods and stochastic optimization as a whole. It was introduced by Robbins and Monro (1951) as a root-finding method for the scenario of a *noisy optimization*, where only noisy measurements of the gradient are available. [13] If we treat the function of the root-finding problem as a gradient of some objective function, the Robbins-Monro algorithm can be viewed as a minimization procedure. Motivated by this idea, Kiefer and Wolfowitz (1952) rephrased the algorithm as an optimization procedure which uses central-difference approximations of the gradient to update the optimum point estimator. [10]

Stochastic approximation is a widely researched field with numerous applications in optimization and root-finding problems (often in the guise of stochastic gradient descent). [10]. Since the original paper of Robbins and Monro in 1951, there has been a steady increase on the applicability of stochastic approximation, which peaked in recent years with the advent of artificial intelligence and deep learning neural networks. Nowadays applications include fields such as queueing networks, wireless communications, manufacturing systems, repeated games, and computational learning. [14]

The basic stochastic approximation algorithm of Robbins-Monro can be simply described as a stochastic difference equation with a small step size, where the main performance concern is the rate of convergence. [13]

A common problem in the applicability of the basic algorithm has to do with the amount of noise in the observations, which may require the use of *variance reduction methods* that will make the algorithm both more effective and complex. For a variation of the basic algorithm that is robust and resistant to large noise, one may have to incorporate constraints for vector-valued iterates to be confined to a given bounded set, though this gives rise to the question of whether the averaging of the iterate sequence will yield improved estimates. A wide range of techniques has been developed for dealing with diverse random processes, a common one being a Lagrangian method for the constrained minimization of a convex function, where only noise-corrupted observations on the function and its constraints are available. [13]

3.2.1. Generic Stochastic Approximation procedure

The method can be formulated as a standard optimization problem. Suppose we have a minimization on $\mathcal{X} \subseteq \mathbf{R}^n$ of the form:

$$\min_{x \in \mathcal{X}} S(x),$$

Where S is an unknown function of the form $E[\hat{S}(x, \xi)]$, with ξ is a random vector and $\hat{S}$ a known function. In a typical scenario, $S(x)$ is the expected performance measure from a Monte

Carlo simulation. This is a problem of *noisy optimization* because, even though $S(x)$ cannot be directly observed, its noisy realization $\hat{S}(x, \xi)$ is observed.

We can't apply classical (deterministic) optimization methods because the gradient ∇S is unknown. The method of stochastic approximation mimics the simple gradient descent by replacing a deterministic gradient with a random subgradient approximation. We assume that $\widehat{\nabla \mathrm{S}}(x)$ is an estimate of the gradient of S which is available at any point $x \in \mathcal{X}$. There are several ways of obtaining $\widehat{\nabla \mathrm{S}}(x)$:

- Finite difference method,
- Infinitesimal perturbation analysis,
- Score function method,
- Method of weak derivatives,

all of which involve replacing S by ℓ and x by θ.

Similar to the gradient descent methods, stochastic approximation starts with some initial value $x_1 \in \mathcal{X}$ and produces a sequence of iterates by applying the following update rule:

$$x_{t+1} = \Pi_{\mathcal{X}}\left[x_t - \beta_t \widehat{\nabla \mathrm{S}}(x_t)\right],$$

where $\beta_1,\ \beta_2$, ... is a sequence of strictly positive step sizes and $\Pi_{\mathcal{X}}$ is a projection operator that takes a point in $\mathbf{R}^n$ and returns a closest (Euclidean distance) point in $\mathcal{X}$. The role of operator $\Pi_{\mathcal{X}}$ is to ensure that further iterates are feasible, that is:

$$y \in \mathbf{R}^n, \Pi_{\mathcal{X}}(y) \in argmin_{z \in \mathcal{X}} \|z - y\|.$$

It follows that, if $\mathcal{X} = \mathbf{R}^n$, then $\Pi_{\mathcal{X}}(y) = y$. The generic stochastic approximation algorithm known as the *Robbins-Monro procedure* is as follows:

Algorithm: Stochastic Approximation

1. Initialize $x_1 \in \mathcal{X}$
   Set $t = 1$.

2. Obtain an estimated gradient $\widehat{\nabla S}(x_t)$ of S at x_t.

3. Determine a step size β_t.

4. Set $x_{t+1} = \Pi_{\mathcal{X}}[x_t - \beta_t \widehat{\nabla S}(x_t)]$.

5. If a convergence criterion is met, stop;
   Else, set $t = t + 1$ and go to step 2.

There are numerous theorems on the convergence of stochastic approximation algorithms [see citation 14]. For an arbitrary deterministic positive sequence $\beta_1, \beta_2, \ldots$ such that:

$$\sum_{t=1}^{\infty} \beta_t = \infty, \quad \sum_{t=1}^{\infty} \beta_t^2 < \infty,$$

The random sequence $x_1, x_2, \ldots$ converges in the mean square sense to the optimal value (minimizer) x^* of $S(x)$ under specific regularity conditions [see citation [15], section 5.9).

One of the simplest convergence theorems is as follows [citation [25]]:

Theorem 12.1.1
(Convergence of Robbins-Monro Stochastic Approximation)

Let us assume that the following conditions are satisfied:

1. The feasible set $\mathcal{X} \subset \mathbf{R}^n$ is convex, nonempty, closed, and bounded.

2. $\Pi_{\mathcal{X}}$ is the Euclidean projection operator.

3. The objective function S is well defined, finite valued, continuous, differentiable, and strictly convex in $\mathcal{X}$ with parameter $\beta > 0$. That is, there exists a $\beta > 0$ such that:

$$(y - x)^T\big(\nabla S(y) - \nabla S(x)\big) \geq \beta\|y - x\|^2, \quad \forall x, y \in \mathcal{X}.$$

4. The error in the stochastic gradient vector $\widehat{\nabla \mathrm{S}}(x)$ possesses a bounded second moment. That is, for some $\mathrm{K} > 0$,

$$E\left[\left\|\widehat{\nabla \mathrm{S}}(x_t)\right\|^2\right] \leq K^2 < \infty, \ \ \forall x \in \mathcal{X}.$$

Then, if $\beta_t = c/t$ for $c > 1/(2\beta)$,

$$E[\|x_t - x^*\|^2] \leq \frac{Q(c)}{t}, \quad t = 1, 2, \ldots,$$

where:

$$Q(c) = \max\{c^2K^2(2c\beta - 1)^{-1}, \ \ \|x_1 - x^*\|^2\},$$

with minimal Q attained by choosing $\mathrm{c} = 1/\beta$. In other words, the expected error in terms of Euclidean distance of the iterates is of order $\mathrm{O}(\mathrm{t}^{-1/2})$.
Furthermore, if x^* is an interior point of $\mathcal{X}$ and if there is some constant $\mathrm{L} > 0$ such that $\nabla \mathrm{S}(\mathrm{x})$ is uniformly Lipschitz continuous in $\mathcal{X}$, i.e:

$$\|\nabla S(y) - \nabla S(x)\| \leq L\|y - x\|, \ \ \forall x, y \in \mathcal{X},$$

then we have:

$$E[\,\big|\,S(x_t) - S(x^*)\,\big|\,] \leq \frac{LQ(c)}{2t}, \ \ t = 1,2, \ldots$$

We can conclude that the expected error (in terms of Euclidean distance) of the objective function values is of order $\mathcal{O}(\mathrm{t}^{-1})$.

An advantage of the Stochastic Approximation procedure is that it becomes particularly easy to implement when the projection operator $\Pi_{\mathcal{X}}$ is easily computed. For example, using *box-constraints*, where $\mathcal{X} = [a_1, b_1] \times ... \times [a_n, b_n]$, any component x_k of $\mathbf{x}$ is projected to a_k if $x_k < a_k$ and to b_k if $x_k < b_k$, otherwise it remains unchanged.

A disadvantage of Stochastic Approximation is the difficulty in choosing step size $\beta_1, \beta_2, \ldots, \beta_n$, since small step sizes may lead to slow convergence while large step sizes can cause a "zigzagging" behavior of the iterates (failure to converge). As theorem 12.1.1 suggests, the rule $\beta_t = c/t$, for some constant c, can be a common choice. Step size hyperparameter c can then be adaptively tuned with every problem at hand.

If $\beta_t \,/\, \beta_{t+1} = 1 + o(\beta_t)$, as in e.g. $\beta_t = 1/t^{\gamma}$ with $\gamma \in (0, 1)$, then the averaged iterate sequence defined by $\bar{x}_t = \frac{1}{t}\sum_{k=1}^{t} x_k$ has been observed to yield better results than $\{x_t\}$. This rule is called *Polyak averaging* or *iterate averaging* and will cause the algorithm to take larger step sizes than the $1/t$ case, reducing convergence time.

When $\widehat{\nabla \mathrm{S}}(x_t)$ is an unbiased estimator of $\nabla \mathrm{S}(x_t)$, then the generic stochastic approximation method of algorithm 12.1 is referred to as the *Robbins-Monro algorithm*. When we use finite differences to estimate $\widehat{\nabla \mathrm{S}}(x_t)$, then the generic method is referred to as the *Kiefer-Wolfowitz algorithm*. In high dimensions, the *random directions* procedure can also be used instead of the finite differences in order to reduce the number of function evaluations per gradient estimate to two. The Kiefer-Wolfowitz algorithm is also known as *Finite Difference Stochastic Approximation* (FDSA).

(Sources: [12], [14])

3.2.2. Robbins-Monro procedure

The original paper in Stochastic Approximation was by Robbins and Monro in 1951. The motivation behind the work of Robbins and Monro was the sequential estimation of the location of the root of a function when that function is unknown and only *noise-corrupted observations* can be made and corrected at very small steps. Starting with an observation at the initial estimator of the root, then use that observation to make small correction in the last estimate, then draw a new observation using the last estimator, and repeat this process until convergence. Keeping a small step size is crucial to convergence, as it guarantees that the noise is averaged. [14]

Due to the small step size, the behavior of the algorithm can be approximated by a *mean flow*. This is equivalent to the solution of an ordinary differential equation (ODE) referred to as *mean ODE*, where the right-hand side of the equation is the mean value of the driving term and its limit points are the same as those of the stochastic approximation process. [14] The Robbins-Monro procedure is the starting point to an enormous literature on general recursive stochastic algorithms with a large number of actual applications.

Algorithm: Robbins-Monro

Let $g(.)$ be a real-valued function of a real variable θ, whose root has to be found. If g were known and continuously differentiable, then Newton's method could be used to generate a sequence of estimators θ_n of the root $\bar{\theta}$, defined recursively by:

$$\theta_{n+1} = \theta_n - [g_\theta(\theta_n)]^{-1} g(\theta_n) \qquad [1.1]$$

where g_θ is the derivative of g with respect to θ. Suppose that $g(\theta) < 0,\ \forall \theta > \bar{\theta}$ and $g(\theta) > 0,\ \forall \theta < \bar{\theta}$, and that $g_\theta(\theta)$ is strictly negative and bounded in a neighborhood of θ. Then θ_n converges to θ if θ_0 is in a very small neighborhood of θ. If there exists step size ϵ which is sufficiently small and greater than zero, the learning rule of [1.1] becomes:

$$\theta_{n+1} = \theta_n + \epsilon g(\theta_n)\ . \quad [1.2]$$

The updated learning rule of [1.2] does not require differentiability and is guaranteed to converge if $\theta_0 - \theta$ is sufficiently small.

Let us now assume that g is unknown and only noise-corrupted observations of $g(\theta)$ can be collected for certain values of θ. Newton's method cannot be used due to the observational noise, but we can use learning rule [1.2] if we replace $g(\theta_n)$ with an estimate of its true value that we obtain by averaging a multitude of observations. The procedure of taking and averaging an excessive number of observations to estimate $g(\theta_n)$ was deemed inefficient since we only need a direction (slope) and not an estimated value for $g(\theta_n)$. Therefore, Robbins and Monro [203], proposed the following algorithm:

$$\theta_{n+1} = \theta_n + \epsilon Y_n \ , \quad [1.3]$$

where ϵ_n is an appropriate sequence which satisfies:

$$\epsilon_n > 0, \quad \epsilon_n \to 0, \quad \sum_n \epsilon_n = \infty, \quad [1.4]$$

and Y_n is a "noisy" estimate of $g(\theta_n)$. It is noteworthy that:

(a) the condition $\sum_n \epsilon_n^2 < \infty$ is used but can be weakened.

(b) Decreasing the step size would imply that the rate of change in $\theta_n$slows down as $n \to \infty$.

(c) The value of sequence$\{\epsilon_n\}$ is crucial to the efficiency of the algorithm as the decreasing step sizes provide implicit averaging of the observations.

The form of training rule [1.3] is motivated by the form of the recursive linear least squares estimator of the mean value of a random variable, which hints how a decreasing step size leads to an averaging of the observations. Let $\{\xi_n\}$ be a sequence of real-valued, mutually independent, i.i.d. random variables finite variance and unknown mean $\bar{\theta}$. Given the observations ξ_i, $1 \le i \le n$, the linear least squares estimate of $\bar{\theta}$ is $\theta_n = \sum_{i=1}^{n} \frac{\xi_i}{n}$, which can be written in the recursive form:

$$\theta_{n+1} = \theta_n + \epsilon_n[\xi_{n+1} - \theta_n], \quad [1.5]$$

where $\theta_0 = 0$ and $\epsilon_n = \frac{1}{n+1}$.

Therefore, from the use of decreasing step sizes ϵ_n derives an estimator equivalent to averaging the observations, with [1.5] being a special case of [1.3]. In the "recursive filter" form of [1.5], shows that the estimator changes only by $\epsilon_n[\xi_{n+1} - \theta_n]$, where ϵ_n is the "reliability" and $[\xi_{n+1} - \theta_n]$ is the "estimation error". (The use of recursive stochastic algorithms in communications and control theory predates the work of Robbins and Monro).

In the generic Robbins-Monro procedure, Y_n is a "noise-corrupted" observation of a vector-valued function $\bar{g}(.)$, whose root we are seeking. The error $Y_n - \bar{g}(\theta_n)$ can be a complicated function of θ_n or of past values of θ_i. In most applications, observed values take the form $Y_n = g(\theta_n, \xi_n) + \delta M_n$, where ξ_n is a correlated stochastic process, for δM_i holds that $E[\delta M_n | Y_i, \delta M_i, i < n] = 0$, and Y_n is an estimator of $\bar{g}(\theta)$ such that $\bar{g}(\theta) = Eg(\theta, \xi_n)$. Despite the numerous forms the observed values can take, the main mathematical interest lies in the asymptotic properties of the sequence θ_n and its dependence on algorithm complexity and randomness.

(Sources: [14], [15])

Example 1 in Robbins-Monro: Finding the zeros of an unknown

Let θ be a parameter that is a point in $\mathbf{R}^r$ and $G(., \theta)$ be an unknown distribution function of a real-valued random variable, where:

$$m(\theta) = \int yG(dy, \theta)$$

is the mean value under θ. When the desired level $\bar{m}$ is given, the problem consists of finding a value $\bar{\theta}$ such that $m(\bar{\theta}) = \bar{m}$. Distribution function $G(., \theta)$ is unknown and we, therefore, require a nonparametric sampling method to approximate it via the Robbins-Monro procedure. Suppose that $m(.)$ is nondecreasing and has a unique root $m(\theta) = \bar{m}$. Let θ_n be the n^{th} estimator of $\bar{\theta}$ and Y_n be the observation at time n for parameter value θ_n. Then θ_n can be defined recursively as:

$$\theta_{n+1} = \theta_n + \epsilon_n[\bar{m} - Y_n] \Leftrightarrow$$
$$\theta_{n+1} = \theta_n + \epsilon_n[\bar{m} - m(\theta_n)] + \epsilon_n[m(\theta_n) - Y_n]. \qquad [1.7]$$

Suppose that:

$$E[Y_n - m(\theta_n)|Y_i, \theta_i, i < n, \theta_n] = 0 \qquad [1.8]$$

If the noise terms $Y_n - m(\theta_n) \equiv \delta M_n$ have bounded variances $\sigma^2(\theta_n)$, then the above expression (1.8) implies that δM_n terms are *martingale differences*, i.e. $E[\delta M_n|\delta M_i, i < n] = 0$, with probability equal to one. The martingale difference noise sequence can be thought of as the simplest type of noise and comes with probability one convergence results.

Martingale difference property:

The martingale difference property often arises under the following conditions: Suppose that the distribution of Y_n is conditioned on $\{\theta_0, Y_i, i < n\}$, depends only on θ_n, i.e. the value of used to observe Y_n (it is equivalent to say that the successive observations are independent). The noise terms δM_n, however, are not mutually independent since θ_n depends on the observations $\{Y_i, i < n\}$, but do not have the martingale difference property, which would guarantee good convergence results.

It is possible to show that the noise terms of [1.7] "average to zero" and do not affect the algorithm's asymptotic behavior. For small $\Delta > 0$, we define m_n^{Δ} by:

$$\sum_{i=n}^{n+m_n^{\Delta}-1} \epsilon_i = \Delta.$$

We then have:

$$\theta_{n+m_n^{\Delta}} - \theta_n = \Delta[\bar{m} - m(\theta_n)] + \text{error},$$

Where:

$$\text{error} = \sum_{i=n}^{n+m_n^{\Delta}-1} \epsilon_i \, \delta M_i. \qquad [1.9]$$

From equation [1.8] we conclude that $\{\delta M_n\}$ is a sequence of zero mean orthogonal random variables, i.e:

$$E\delta M_i \delta M_j = 0, \forall i \neq j$$

with variance of the error being:

$$E\left[\sum_{i=n}^{n+m_n^{\Delta}-1} \epsilon_i \, \delta M_i\right]^2 = \sum_{i=n}^{n+m_n^{\Delta}-1} E\epsilon_i^2 \, \delta M_i^2 = \sum_{i=n}^{n+m_n^{\Delta}-1} O(\epsilon_i^2) = O(\Delta)\epsilon_n.$$

Along with [1.9], these bounds tell us that the mean change in the value of the parameter is more important than the noise (for small Δ and large n, over iterate intervals $[n, n + m_n^{\Delta}]$). Consequently, the difference equation [1.9] implies that asymptotic behavior of the algorithm can be approximated by the asymptotic behavior of the solution to the ODE:

$$\theta = \bar{g}(\theta) = \bar{m} - m(\theta). \qquad [1.9b]$$

Such ODEs have been shown to play a crucial role in convergence theory. If $\bar{\theta}$ is an asymptotically stable point of [1.9b], then $\theta_n \to \bar{\theta}$ with $P[\{\theta_n \to \bar{\theta}\}] = 1$ (probability one). Since $\epsilon_n \to 0$, we have that $m_n^{\Delta} \to \infty$ as $n \to \infty$. The statement that "the noise locally averages to zero" implies that the noise effects go towards zero as n goes towards infinity and Δ goes towards zero. The above description can be thought of as a heuristic summary of the associated convergence theorems. The most important element is the intuition of "time scale separation" between sequences $\{\theta_n \to \bar{\theta}\}$ and $\{Y_i - m(\theta_i), i \leq n\}$, for a large n, and the role of the ODE becomes magnified when the noise sequence is strongly correlated.

(Sources: [14])

Example 2 in Robbins-Monro: Minimization by Recursive Monte Carlo

This section describes a function minimization problem, similar to the ones encountered in learning applications. In this example of parametric optimization of dynamical systems, the Robbins-Monro procedure uses the *noise-corrupted observations of the derivatives* for sequential Monte Carlo minimization. The θ-derivatives of the mean are unknown, but one could observe values of the θ-derivative by sampling $[y_n - \theta'\phi_n]^2/2$ with the desired values of θ, then using these estimated derivatives in the iterative algorithm in the place of the exact derivatives.

Let θ be a $\mathbf{R}^r$-valued parameter of a dynamical system in $\mathbf{R}^k$. The evolution of the system can be described by the equation:

$$\bar{X}^{m+1}(\theta) = b(\bar{X}^m(\theta), \theta, \chi^m), \qquad m = 0, 1, \dots, \qquad \bar{X}^0(\theta) = X_0, \qquad [1.20]$$

where χ^m are random variables, function $b(.)$ is known and continuously differentiable in (x, θ), and $\bar{X}_i^m(\theta), i \leq k$ are the components of $\bar{X}^m(\theta)$. Given a real-valued function $F(.)$, we want to minimize $f(\theta) = E[F(\bar{X}^N(\theta), \theta)]$ over θ, for a given value of N, therefore the system is studied over the finite horizon $[0, N]$. The combined dynamics of a tracking and intercept problem can be represented by equation [1.20], in which θ parametrizes the tracker controller, and the objective function yields the probability of getting within striking distance before terminal time N (the probability has to be maximized).

Let $\bar{\chi} = \{\chi^m, m = 0, \dots, N-1\}$ and suppose that the distribution of $\bar{\chi}$ is known. We assume that function $F(.)$ is known and continuously differentiable in (x, θ), so that noisy/sampled values of the system state (cost and pathwise θ-derivatives) can be simulated via Monte Carlo. A deterministic algorithm requires good estimates of $E[F(\bar{X}^N(\theta), \theta)]$ at selected values of θ, but this is not feasible when the problem becomes complicated. In this scenario, a standard Monte Carlo simulation would yield values of θ far from the optimal point. However, a recursive Monte Carlo method is a viable alternative.

Recursive Monte Carlo requires simulations of the system on time interval $[0, N]$, under varying parameter values. Let $\bar{U}_j^m(\theta) = \frac{\partial}{\partial \theta^j}\bar{X}^m$, with components $\bar{U}_{j,i}^m(\theta) = \frac{\partial}{\partial \theta^j}\bar{X}_i^m, j \leq r$, where θ^j is the jth component of vector θ. Then $\bar{U}_i^0(\theta) = 0, \forall i$, and for $m \geq 0$ we have:

$$\bar{U}_i^{m+1}(\theta) = b'_x(\bar{X}^m(\theta), \theta, \chi^m)\bar{U}_i^m(\theta) + b_{\theta^i}(\bar{X}^m(\theta), \theta, \chi^m).$$

If the operations of differentiation and expectation are assumed to be interchangeable, let $\bar{g}(\theta) = (\bar{g}_i(\theta), i \leq r)$ be defined by:

$$\frac{\partial E[F(\bar{X}^N(\theta), \theta)]}{\partial \theta} = E\left[\frac{\partial F(\bar{X}^N(\theta), \theta)}{\partial \theta}\right] =$$

$$= E\left[F'_x(\bar{X}^N(\theta), \theta)\bar{U}_i^N(\theta) + F_{\theta^i}(\bar{X}^N(\theta), \theta)\right] =$$

$$= -\bar{g}_i(\theta).$$

Now, assuming that θ_0 is given, let $\theta_n = (\theta_{n,1}, \dots, \theta_{n,r})$ be the nth estimator of the minimizing value of θ.

We may define the system in the nth simulation by:

$$X_n^{m+1} = b(X_n^m, \theta_n, \chi_n^m), \quad X_n^0 = X_0, \quad m = 0,1, \dots N-1,$$

where $\chi_n = \{\chi_n^m, m < N\}$ is a random sequence with the same distribution as $\bar{\chi}$, et every n. Let $U_{n,i}^m = \frac{\partial}{\partial \theta_{n,i}} X_n^m$, where $\theta_{n,i}$ is the ith component of θ_n. Let us additionally define:

$$Y_{n,i} = -F'_x(X_n^N, \theta_n)U_{n,i}^N - F_{\theta^i}(X_n^N, \theta_n)$$

with $Y_n = \left(Y_{n,i}, i = 1, \dots, r\right)$. Then, for this problem, a Robbins-Monro recursive Mote Carlo procedure can be expressed as:

$$\theta_{n+1} = \theta_n + \epsilon_n Y_n = \theta_n + \epsilon_n \bar{g}(\theta_n) + \epsilon_n[Y_n - \bar{g}(\theta_n)].$$

In general, the noise terms $[Y_n - \bar{g}(\theta_n)]$ are martingale differences, if $\{x_n\}$ are assumed to be mutually independent. However, when variance reduction is applied, correlated values of $\{x_n\}$

can be used, as long as the noise terms "locally average to zero". The mean ODE, $\dot{\theta} = \bar{g}(\theta)$, characterizes the asymptotic behavior.

In the scenario where actual data is observed on a physical system rather than simulated from a hypothesized data generating process, the exact form of the dynamical equations governing the system will be unknown. In the case of actual having observations, the calculated pathwise derivatives will not be accurate but, even in that scenario, the optimization procedure will likely work well since the approximation theorems will still hold true.

(Sources: [14])

3.2.3. Kiefer-Wolfowitz procedure

In all previous examples of stochastic approximation with Robbins-Monro, we were concerned with the minimization of a function of unknown form (noisy), a problem typically met in "Learning". As seen in previous chapters on deterministic convex optimization, numerical methods such as the Newton-Raphson algorithm are commonly used for the recursive computation of the minimum of a smooth known function. However, when dealing with an unknown or noisy objective function, i.e. one with noise-corrupted observations at parameter values obtained via Monte Carlo simulation, then one could use a stochastic recursive method, equivalent to Newton-Raphson, where the gradient is estimated via numerical differentiation (such as pathwise of finite derivatives) using the noisy measurements at small step sizes.
With Robbins-Monro, we would explicitly differentiate the sample error functions at the current parameter value and use these derivates as the noisy estimates of the derivates of mean performance of interest at those parameter values. However, when pathwise differentiation is not possible, the gradient estimate can be obtained via a *finite difference* or *random directions* method. The variant of stochastic approximation for the minimization of a noisy function via numerical differentiation is called the *Kiefer-Wolfowitz procedure*. Depending on whether random directions or finite differences are used, the variations of the algorithm are abbreviated as *Random Direction Stochastic Approximation* (RDSA) or *Finite Difference Stochastic Approximation* (FDSA). The most popular approach for random direction is random selection from a Bernoulli distribution, in which case RDSA goes also by the name *Simultaneous Perturbation Stochastic Approximation* (SPSA). [14]

Finite Difference Stochastic Approximation (FDSA)

Let $E[F(\theta,\chi)] = f(\theta)$ be a function that we wish to minimize over the $\mathbf{R}^r$-valued parameter θ, where χ is a random vector, $f(.)$ is continuously differentiable, and the forms of F and f are not completely known. Suppose we are in the scenario of stochastic approximation with a finite difference method. Let $c_n \to 0$ be a finite difference interval, e_i be unit vector in the ith coordinate direction. Let θ_n be the nth estimate of the minimum and suppose that, for i, n and random vectors $\chi_{n,i}^{+}, \chi_{n,i}^{-}$, we observe the finite difference estimate:

$$Y_{n,i} = -\frac{\left[F\left(\theta_n + c_n e_i, \chi_{n,i}^{+}\right) - F\left(\theta_n - c_n e_i, \bar{\chi}_{n,i}\right)\right]}{2c_n}. \qquad [2.1]$$

We now define $Y_n = \left(Y_{n,1}, \dots, Y_{n,r}\right)$, and we update θ_n by:

$$\theta_{n+1} = \theta_n + \epsilon_n Y_n. \qquad [2.2]$$

When equation [2.2] uses an estimator of finite differences Y_n as defined by [2.1], then the stochastic approximation variant is called the *Kiefer-Wolfowitz* algorithm.

Let:

$$\psi_{n,i} = \left[f(\theta_n + c_n e_i) - F\left(\theta_n + c_n e_i, \chi_{n,i}^{+}\right)\right] - \left[f(\theta_n - c_n e_i) - F\left(\theta_n - c_n e_i, \chi_{n,i}^{+}\right)\right]$$

And express:

$$\frac{\left[F\left(\theta_n + c_n e_i, \chi_{n,i}^{+}\right) - F\left(\theta_n - c_n e_i, \bar{\chi}_{n,i}\right)\right]}{2c_n} \equiv \gamma_{n,i} = f_{\theta^i}(\theta_n) - \beta_{n,i}, \qquad [2.3]$$

Where $-\beta_{n,i}$ is the bias of the finite difference estimate of $f_\theta(\theta_n)$, using central differences. By setting $\psi_n = (\psi_{n,1}, \dots, \psi_{n,r})$ and $\beta_n = (\beta_{n,1}, \dots, \beta_{n,r})$, we can rewrite equation [2.2] as:

$$\theta_{n+1} = \theta_n - \epsilon_n f_\theta(\theta_n) + \epsilon_n \frac{\psi_n}{2c_n} + \epsilon_n \beta_n. \qquad [2.4]$$

In order to converge to a local minimum, we would need $\beta_n \to 0$, with the bias being proportional to the finite difference interval $c_n \to 0$. For convergence, we would also need the noise terms $\epsilon_n \psi_n / 2c_n$ to average locally to zero. Then, the ODE which characterizes the asymptotic behavior of the algorithm becomes:

$$\dot{\theta} = -f_\theta(\theta).$$

With the effective noise being of the order of $1/2c_n$, a disadvantage of Kiefer-Wolfowitz in comparison to the Robbins-Monro procedure is the need of using a variance reduction method in order to improve the estimates of the derivatives. When c_n is restricted from going to zero, a small bias can make up for smaller noise effects.

Variance reduction:

Driving noise is an essential part of a system where the objective is the minimization of an average value and Monte Carlo simulation is involved. Suppose that $F(x, \chi)$ is continuously differentiable for each value of χ and it holds that:

$$\frac{F(\theta_n - e_i c_n, \chi_{n,i}^-) - F(\theta_n + e_i c_n, \chi_{n,i}^+)}{2c_n} + f_{\theta^i}(\theta_n)$$
$$= \frac{1}{2c_n}\left[F(\theta_n, \chi_{n,i}^-) - F(\theta_n, \chi_{n,i}^+)\right]$$
$$- \left[\frac{1}{2}\left(F_{\theta^i}(\theta_n, \chi_{n,i}^+) + F_{\theta^i}(\theta_n, \chi_{n,i}^-)\right) - f_{\theta^i}(\theta_n)\right] + \tilde{\beta}_{n,i} \equiv \tilde{\psi}_{n,i} + \tilde{\beta}_{n,i},$$

Where $\tilde{\beta}_{n,i}$ is the bias of the finite difference estimate. The algorithm then becomes:

$$\theta_{n+1} = \theta_n - \epsilon_n f_\theta(\theta_n) + \epsilon_n \tilde{\psi}_n + \epsilon_n \tilde{\beta}_n.$$

It is preferable to use $\chi_{n,i}^{+} = \chi_{n,i}^{-}$ as it removes the $1/c_n$ factor in effective noise, i.e. $\tilde{\psi}_{n,i}$ is not inversely proportional to c_n. This can be advantageous even without differentiability. Let $\theta_n = \theta$, $E[F\left(\theta \pm c_n e_i, \chi_{n,i}^{\pm}\right) = f(\theta \pm c_n e_i)$, and $\tilde{F}(\theta, \chi) = F(\theta, \chi) - f(\theta)$. Then, the variance of the effective noise is:

$$E[\tilde{F}\left(\theta + c_n e_i \chi_{n,i}^{+}\right)]^2 + E[\tilde{F}\left(\theta - c_n e_i \chi_{n,i}^{-}\right)]^2 - 2E[\tilde{F}\left(\theta + c_n e_i \chi_{n,i}^{+}\right)]^2 [\tilde{F}\left(\theta - c_n e_i \chi_{n,i}^{-}\right)]^2,$$

all divided by $4c_n^2$, suggesting that the largest correlation becomes between $\chi_{n,i}^{\pm}$, the smaller the noise variance we get for small c_n. If we suppose that $\{(\chi_n^{+}, \chi_n^{-}), n = 0,1, \dots\}$ is a sequence of independent random variables, the ψ_n and $\bar{\psi}_n$ are martingale differences for each n. If the noise terms ψ_n and $\bar{\psi}_n$ are complicated functions of θ_n, the dependence on θ_n is omitted in proofs of convergence unless $\chi_n^{\pm}$ are correlated in n.

Simultaneous Perturbation Stochastic Approximation (SPSA)

SPSA refers to the classical Kiefer-Wolfowitz procedure with Random directions. The classical algorithm uses either $2r$ or $r + 1$ observations for two-sided and one-sided differences, respectively. The symmetric two-sided difference is usually preferred due to its better convergence rate and finite difference bias. When the sequential form of the algorithm is used with one component of θ being updated at a time, then $2r$ steps are required for a full derivative estimate. Sometimes the derivative can be estimated directly (without finite differences), but in the opposite case and when dimension r is large, the classical Kiefer-Wolfowitz procedure becomes applicable. An alternative method would be to use a finite difference estimate to update only one out of two directions that is selected randomly at every iteration (so that every step gets only two observations).

Let $\{d_n\}$ be a sequence of random direction vectors, where values $d_n d_n'$ average locally to the identity matrix in $\mathbf{R}^n$, with difference intervals $0 < c_n \to 0$. Then the algorithm becomes:

$$\theta_{n+1} = \theta_n - \epsilon_n d_n \frac{[Y_n^{+} - Y_n^{-}]}{2c_n}, \qquad [2.9]$$

where $Y_n^{\pm}$ are observed at parameter values $\theta_n \pm c_n d_n$. If the difference interval is a constant, the method is still applicable with the advantage of reducing noise effects and yielding a more robust algorithm, with a small bias. Given some suitable function $F(.)$ on random variables $\chi_n^{\pm}$, the observations $Y_n^{\pm}$ can can be written in the form:

$$Y_{n,i}^{\pm} = F(\theta_n \pm e_i c_n, \chi_n^{\pm}) = f(\theta_n \pm c_n d_n) + \psi_n^{\pm}, \qquad [2.10]$$

where $\psi_n^{\pm}$ is the effective observation noise. If $f(.)$ is continuously differentiable in $\mathbf{R}^n$, then [2.9] becomes:

$$\theta_{n+1} = \theta_n - \epsilon_n d_n d_n' f_\theta(\theta_n) + \epsilon_n \beta_n + \epsilon_n \frac{d_n[\psi_n^- - \psi_n^+]}{2c_n},$$

where β_n is the bias in the symmetric finite difference estimator of the derivative of f at θ_n, in the direction d_n, and with difference interval $c_n d_n$. Centering $d_n d_n'$ about the identity matrix gives:

$$\theta_{n+1} = \theta_n - \epsilon_n[f_\theta(\theta_n) - \beta_n] + \epsilon_n \frac{d_n[\psi_n^- - \psi_n^+]}{2c_n} + \epsilon_n \psi_n^d, \qquad [2.12]$$

where $\psi_n^d = [I - d_n d_n']f_\theta(\theta_n)$ is the *random direction noise*. Then the mean ODE which characterizes the asymptotic behavior of the algorithm is the same as that for the Kiefer-Wolfowitz procedure, i.e. the *gradient descent* form:

$$\dot{\theta} = -f_\theta(\theta).$$

If $\chi_n^+ = \chi_n^-$ then the term in [2.12] that is proportional to $1/c_n$ can be replaced by $\epsilon_n d_n \psi_n$ in order to go back to a form of the Robbins-Monro procedure.

(Sources: [14], [16])

3.2.4. Stochastic Approximation in Learning problems

Training an Animal Learning model

The purported learning behavior of an animal trying to maximize its reward per unit time can be modeled as a single-agent learning problem that can be solved by stochastic approximation. This specific problem description is taken from Kushner and Lin (2003).

Problem*:*
A lizard has a fixed home location and can hunt in an arbitrary finite region. For the sake of simplicity, the lizard's hunting region can be thought of as a circle. Insects and other food sources of varying value in calories (weight) may appear randomly inside the hunting region. If a food source appears when the lizard is at its home location, the lizard has to decide whether to hunt or not, given the implicit objective of maximizing long-term return per unit time. The purpose of the application is to observe whether the lizard's learning behavior is consistent with the optimization of a "return for effort" principle.

The sequence of intervals between actions (decision times) for the lizard have to be defined. At time zero t_0, The lizard is at its home base and the process commences; at time t_1, the first insect appears. If the lizard decides to hunt, r_1 is the time to pursue the insect and return to home base, while r_2 is the time between the return and the arrival of the next insect. If the lizard decides to stay home and not hunt, then r_2 is the time until the arrival of the next insect. If we were to generalize this notation, r_n can indicate any of the following time intervals:

- time between the successive arrivals of two insects, if the first insect was not pursued,
- time between the return of the lizard to home base from the last pursuit and the arrival of the next insect,
- time of duration of a pursuit, right after the lizard decides to "pursue".

Let w_n be the "weight" (value in calories) of the nth insect and J_n be the indicator function of the insect being caught, I_n be the indicator function of the event of pursuit at the nth insect arrival, and T_n be the time taken to complete the action of the $(n-1)$st decision. The pair (w_n, r_n) is the information received by the lizard when an insect arrives.

Define weight function:

$$W_n = \sum_{i=1}^{n-1} w_i I_i J_i$$

and let $\theta_n = T_n/W_n$ be the inverse of the sample consumption rate per unit time. Suppose that random variables (w_n, r_n, r_n, J_n) for $n \geq 1$ are i.i.d. with bounded second moments and $w_n > 0$, $r_n > 0$. Let $p(.)$ be a continuous function such that $E[J_n|r_n, w_n] = p(r_n, w_n) > 0$, that the lizard has knowledge of.

Learning algorithm:

Let $\bar{\theta} = \liminf_n ET_n/W_n$, where the infinimum comprises all (n) realizable strategies. The learning algorithm produces a sequence of estimates θ_n, such that $\theta_n \to \bar{\theta}$, where threshold $\bar{\theta}$ yields the optimal long-term decision rule:

Pursue only if $r_n \leq \bar{\theta} w_n p(r_n, w_n)$.

Therefore, the optimal strategy consists of pursuing an instinct only if the ratio of the required pursuit time over the expected gain is not greater than threshold $\bar{\theta}$. The learning algorithm approximates the optimal strategy asymptotically. We can extend the animal model by supposing that the estimates of $p(r_n, w_n)$ and r_n are prone to error/noise, in which case we can introduce additional random variables to account for noise.

Let $n \geq 1$ and θ_1 be an arbitrary real number. If the lizard does not decide to pursue at the nth opportunity, we have:

$$\theta_{n+1} = \frac{T_n + t_n}{W_{n+1}} = \frac{T_n + t_n}{W_n}. \quad [1.1a]$$

If the lizard pursues and successfully catches the insect, then:

$$\theta_{n+1} = \frac{T_n + t_n + r_n}{W_{n+1}} = \frac{T_n + t_n + r_n}{W_n + w_n}. \quad [1.1b]$$

If the lizard pursues the insect but fails to capture it, we have:

$$\theta_{n+1} = \frac{T_n + t_n + r_n}{W_n}. \qquad [1.1c]$$

Therefore, whenever the lizard pursues, we have:

$$\theta_{n+1} = \frac{T_n + t_n + r_n}{W_n + w_n} J_n + \frac{T_n + t_n + r_n}{W_n}(1 - J_n). \qquad [1.1d]$$

The lizard chooses the action that yields the minimal conditional expectations of the right-hand sides of (1.1a) and (1.1d) given (r_n, w_n, θ_n).

Suppose that $\epsilon_n = 1/W_n$, and that ϵ_n decreases every time the lizard pursues and catches an insect. Then, depending on the decision "pursue" or "not pursue", either (1.2a) or (1.2b) will hold:

$$\theta_{n+1} = \theta_n + \epsilon_n t_n,$$
$$\theta_{n+1} = \theta_n + \epsilon_n(t_n + r_n) - \epsilon_n \theta_n w_n J_n + O(\epsilon_n^2) J_n.$$

We have that $\epsilon_n \to 0$ with probability one, while we can prove that $O(\epsilon_n^2)$ in (1.2b) is asymptotically insignificant in relation to the other terms and will therefore be ignored. So if the insect is pursued, we now have:

$$\theta_{n+1} = \theta_n + \epsilon_n(t_n + r_n) - \epsilon_n \theta_n w_n p(r_n, w_n) + \epsilon_n \theta_n w_n (p(r_n, w_n) - J_n).$$

With $K_n = I_{\{r_n - \theta_n w_n p(r_n, w_n) < 0\}}$ we have:

$$\theta_{n+1} = \theta_n + \epsilon_n t_n + \epsilon_n \min\{r_n - \theta_n w_n p(r_n, w_n), 0\} + \epsilon_n \theta_n w_n (p(r_n, w_n) - J_n) K_n.$$

We can see that step sizes decrease randomly. For a fixed nonrandom θ, we define the mean value:

$$\bar{g}(\theta) = E[t_n] + E[\min\{r_n - \theta w_n p(r_n, w_n), 0\}],$$

where the noise term ξ_n is defined as:

$$\xi_n = t_n + \min\{r_n - \theta_n w_n p(r_n, w_n), 0\} - \bar{g}(\theta_n) + \theta_n w_n (p(r_n, w_n) - J_n) K_n.$$

Function $\bar{g}(.)$ is *Lipschitz continuous*, positive for $\theta = 0$, proportional to $-\theta$ for as θ grows larger, and there exists a unique root $\theta = \bar{\theta}$, for which $\bar{g}(\theta) = 0$. We can rewrite the learning rule as:

$$\theta_{n+1} = \theta_n + \epsilon_n \bar{g}(\theta_n) + \epsilon_n \xi_n.$$

The asymptotic behavior of the learning algorithm is determined by the mean ODE $\dot{\theta} = \bar{g}(\theta)$. The final form of the algorithm and the value of ϵ_n are consequences of the representation of the definition: $\epsilon_n = 1/W_n$. This representation brings the learning algorithm into the form of stochastic approximation.

(Sources: [14])

Training a Neural network model

A neural network is a type of *generalized linear model* in which the output y_n is a nonlinear function of all inputs. In its simplest form, the linear perceptron is a binomial GLM with a logit link function that becomes a binary logistic regression model when solved for y_n.

The GLM form of the binomial logit model is the following:

<u>*Stochastic component*</u>:

$y_i \sim Binomial(n = 1, \pi_i), \quad i = 1 \dots N$

where N is the number of binomial observations, and $E[y_i] = N\pi_i = \pi_i$.

Deterministic component:

$\pi_i = \sum_j^M \beta_{ij} x_{ij}$,

where M is the number of independent variables.

Link function:

$g(.) = logit(.)$

$g(\pi_i) = \sum_j^M \beta_{ij} x_{ij}$

In the binomial logit GLM we would have:

$logit(\pi_i) = \sum_j^M \beta_{ij} x_{ij}$

$log\left(\frac{\pi_i}{1-\pi_i}\right) = \sum_j^M \beta_{ij} x_{ij}$

$log\left(Odds_{y_i|x_{i.}}\right) = \sum_j^M \beta_{ij} x_{ij}$

In the linear perceptron case we have:

$\pi_i = g^{-1}\left(\sum_j^M \beta_{ij} x_{ij}\right)$

$\pi_i = logistic\left(\sum_j^M \beta_{ij} x_{ij}\right)$

By developing the logistic function and substituting $<\pi_i, \beta_{ij}, x_{ij}>$, by $<y_n, \alpha_n, \phi_n>$, we get the linear perceptron model:

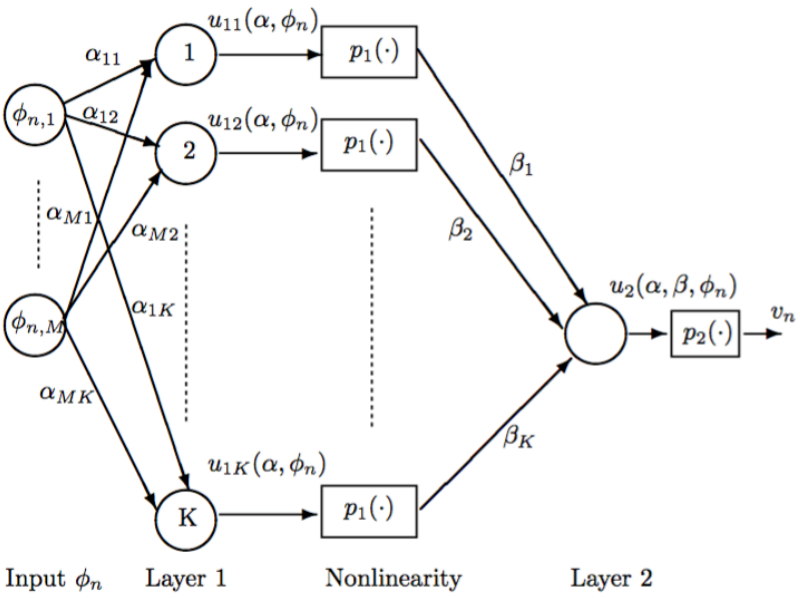
$$y_n = \frac{1}{1 + e^{-\sum_n \alpha_n \phi_n}}$$

If we set link function $g^{-1}(.)$ to the *activation function* $p(.)$, to which we can add non-linearity structure "on demand" by setting $p_1(p_2(p \ldots (.)))$, then we can easily a highly flexible GLM that is traditionally called a multi-layer perceptron.

In previous decisions problems, we came across a sequence of patterns where the nth pattern was denoted by y_n. In this setting, patterns are unobservable but we can observe a random ϕ_n that is correlated with y_n, at time n. We are searching for an affine function of the observables (affine decision rule) that minimizes the mean squared error between observed and actual values. The statistics of (y_n, ϕ_n) are unknown but, during a search procedure that is called *training phase*, many samples of those pairs become available while a recursive linear least squares procedure is used to converge sequentially to the optimal weights (parameters) of the affine decision function. During the training phase, a sequence of inputs $\{\phi_n\}$ is sequentially fed into the training rule until the algorithm converges to the optimal value of θ, i.e. the one that best matches the sequence of correct decisions y_n. A neural network works in the same way, only that output v_n can be a general nonlinear function of the inputs. Even at very high dimensions, this process can be optimized efficiently via the random directions Kiefer-Wolfowitz algorithm.

The simple neural network of this example has three layers of neurons: the input layer, the hidden layer, and the output layer (sometimes the inputs are not counted as a separate layer). The output layer consists of only one neuron, with or without an activation (link) function. The hidden layer (which can be more than just one) is what adds non-linearity to the regression function and makes the neural network a universal function approximator. Suppose that the hidden layer has K neurons and the output layer has a single neuron. A hidden neuron's input at time n is a linear combination of the observable random variables at that point in time, while a hidden neuron's output is a nonlinear function of its input. Similarly, in the output layer, the

neuron's input is a linear combination of the outputs of the hidden layer, and the network output is a nonlinear function of that neuron's input. The sigmoid (inverse logit) function is a common choice for the network's activation function, though any suitable nonlinear transformation can be used insofar as any continuous vector-valued map between input and output can be approximated by appropriate choices of the number of neurons and hidden layers.



Graphical representation of neural network model with one hidden layer (round nodes) and the logistic activation function (rectangular nodes).

The estimation of weights or "training" of the network can be described by its input, output, and the relationship between them. Assuming fixed weights at time n, the network is associated to a sequence of M-dimensional vectors: input vector $\phi_n = (\phi_{n,1}, \dots, \phi_{n,M})$, observable network output vector v_n, actual/desired output vector y_n. Thus, for a well-trained network with weights a_{ij}, v_n becomes a good approximation to y_n. Then, at time n, the input to neuron j in the hidden layer becomes:

$$u_{1j}(\alpha, \phi_n) = \sum_{i=1}^{M} \alpha_{ij} \phi_{n,i}\,, \qquad j = 1, \dots, K.$$

Let $p_1(u_{1j}(\alpha, \phi_n))$ be the output of this hidden layer neuron, where $p_1(.)$ is a real-valued antisymmetric nondecreasing and continuously differentiable function of a real variable (that we call the *activation function*). Let $\dot{p_1}(.)$ be the derivative of $p_1(.)$, and β_i be the weights of the output layer neuron. The input to neuron of the output layer is the linear combination of the output of all neurons in the hidden layer:

$$u_2(\alpha, \beta, \phi_n) = \sum_{i=1}^{M} \beta_j p_1(u_{1j}(\alpha, \phi_n)).$$

The neuron in the output layer is:

$$u(\alpha, \beta, \phi_n) = p_2(u_2(\alpha, \beta, \phi_n))$$

Where activation function p_2 has the same properties as p_1 and the derivative of p_2 is denoted by $\dot{p_2}$. The training phase of the network (estimation of weights) consists of an optimization problem where the errors between desired and actual output are minimized. Suppose that the set of pairs for desired output $\{(\phi_n, y_n), n = 1, \dots\}$ and the actual outputs $\{v_n\}$ are given, and that the input-output pairs are mutually independent. Let $\theta = (\alpha, \beta)$ be the vector of weights to be estimated and let θ_n be the value at the nth training session, with $v_n = u(\theta_n, \phi_n)$. The weighs will be update sequentially such that the mean squared error $E[y - u(\theta, \phi)]^2$ is minimized. This error function is derived by defining sample mean square error $e(\theta, \phi, y) = \frac{1}{2}[y - u(\theta, \phi)]^2$ and sample error $e_n = [y_n - v_n]$.

The form of the adaptive algorithm is the following:

$$\theta_{n+1,i} = \theta_{n,i} - e_n \frac{\partial e(\theta_n, \phi_n, y_n)}{\partial \theta^i} = \theta_{n,i} + \epsilon_n e_n \frac{\partial u(\theta_n, \phi_n)}{\partial \theta^i}, \quad i = 1, \dots, r.$$

Using the formula:

$$v = u(\alpha, \beta, \phi) = p_2\big(u_2(\alpha, \beta, \phi)\big) = p_2\big(\sum_{j=1}^{K} \beta_j p_1\left(u_{1j}(\alpha, \phi)\right)\big),$$

with the above derivatives computed via repeated differentiation:

$$\frac{\partial u(\theta,\phi)}{\partial \beta_i} = p_2\big(u_2(\alpha,\beta,\phi)\big)p_1\left(u_{1j}(\alpha,\phi)\right),$$
$$\frac{\partial u(\theta,\phi)}{\partial \alpha_{ij}} = p_2\big(u_2(\alpha,\beta,\phi)\big)\beta_j p_1\left(u_{1j}(\alpha,\phi)\right)\phi^i,$$

with ϕ^i being the ith element of input vector ϕ. The asymptotic behavior of the algorithm is characterized by the following ODE:

$$\dot{\theta}^i = g^{-i}(\theta) = -\frac{E\partial e(\theta,\phi,y)}{\partial \theta^i}, \;\; i = 1,\dots,r.$$

With above gradient method, θ_n converges to a stationary point of this ODE. Though approximated as a convex problem, the underlying objective function is in fact a non-convex function with multiple local minima and different training sessions will yield different weight estimates. In a well-tuned model, the variance of those estimates should be relatively small. Training time is directly proportional to the number of weights in the network, while neural network research has led to the development of specialized stochastic approximation algorithms.

(Sources: [14])

3.3. Application: Estimating neural network weights with stochastic gradient descent

The estimation of the weights of a single-layer neural network is a special case of the well-studied *backpropagation algorithm* and the core of all neural network training procedures. A linear neural network is one which contains no hidden layers and is also known as a *linear perceptron*. When the activation function is the logistic function, the optimization of the linear perceptron is the equivalent of a non-parametric logistic regression, i.e. the estimation of the parameters of a logistic model without using the binomial likelihood nor making any probabilistic assumption whatsoever. This task can be achieved by minimizing the mean

squared error of the logistic functions via stochastic gradient descent. An analytical training rule can be obtained by applying gradient descent on the error loss of the network, which leads to the estimation network weights via online learning rather than batch learning. The derivation is provided below.

3.3.1. Perceptron training rule

Let g be the differentiable link function associated to the deterministic component of a generalized linear model with coefficients w_{ji}. In neural network terminology, let the activation function $g(x)$ be associated to neuron j and weight i, with ith input x_i, predicted output $\hat{y}_j$, actual output y_j, learning rate α, and weighted sum of inputs h_j.
The output of the network with a single neuron (linear perceptron) can be expressed as:

$$\epsilon = \sum_j \frac{1}{2}\left(y_j - \hat{y}_j\right)^2$$

To move through the weight space of the neuron in proportion to the gradient of the error function we compute the partial derivative of the error with respect to each weight:

$$\frac{\partial \epsilon}{\partial w_{ij}}$$

We can develop the above by plugging in the error function and simplify it (for the case of a single neuron) by omitting the summation:

$$\frac{\partial \epsilon}{\partial w_{ij}} = \frac{\partial\left(\frac{1}{2}\left(y_j - \hat{y}_j\right)\right)}{\partial w_{ij}}$$

We can further develop using the chain rule:

$$= \frac{\partial\left(\frac{1}{2}(y_j - \hat{y}_j)\right)}{\partial y_j}\frac{\partial y_j}{\partial w_{ji}}$$

And we again use chain rule to solve the left-hand side:

$$= -(y_j - \hat{y}_j)\frac{\partial y_j}{\partial w_{ji}}$$

We solve similarly the right-hand side but by differentiating w.r.t. the total input:

$$= -(y_j - \hat{y}_j)\frac{\partial y_j}{\partial h_{ji}}\frac{\partial h_j}{\partial w_{ji}}$$

Then we substitute the output of the jth neuron with the activation function applied on the neuron's input h_j and rewrite the derivative of y_j w.r.t. h_j as the first derivative of g:

$$= -(y_j - \hat{y}_j)g'(h_j)\frac{\partial h_j}{\partial w_{ji}}$$

We express h_j as the sum of all k weights times the corresponding inputs x_k:

$$= -(y_j - \hat{y}_j)g'(h_j)\frac{\partial(\sum_k x_k w_{jk})}{\partial w_{ji}}$$

Since we only care about the ith weight, we can simplify the above expression by omitting the summation:

$$\frac{\partial x_i w_{ji}}{\partial w_{ji}} = x_i$$

which leads to the final equation of the gradient:

$$\frac{\partial \epsilon}{\partial w_{ji}} = -(y_j - \hat{y}_j) g'(h_j) x_i$$

Therefore, the change of each weight should be proportional to the gradient. A proportionality constant α can be applied to let us move the weight to the negative direction of the gradient (removing minus sign) and minimize the error, driving thus the perceptron training rule:

$$\Delta w_{ji} = \alpha(y_j - \hat{y}_j) g'(h_j) x_i$$

The implementation of the linear perceptron and the multi-layer perceptron in MATLAB code are provided in the next section.

3.3.2. Source code

Linear perceptron

```
% Function that learns a linear model over a training dataset.
% x: training set (to build model from)
% z: training set's output vector (known classes)
% b: bias (usually 0)
% r: learning rate (varies but usually 0.5)
% T: Threshod for classification decision (where applicable)
% MAXIT:  % maximum number of iterations before forced convergence

function w = LinearPerceptronTrain(x, z, b, r, T, MAXIT)

% Samples to classify on the rows, variables on the columns (4x2)
n = size(x,1);      % number of inputs
m = size(x,2);     % number of features (variables)
w = zeros(1,m);  % initial weight vector

% Parameters:
error_count = -1;
```

```
c = 1;       % iteration counter

% Start training:
while error_count~=0 && c<=MAXIT
% while error value is not zero and max interations haven't been reached
   error_count = 0;
   for i=1:n
      s = x(I,:) * w' + b;   % (i) calculate output y
      if s > T
         y = 1;
      else
         y = 0;
      end
      error = z(i)-y;              % (ii) calculate error
      d = x(I,:)*r*error;          % error correction value
      w = w + d;                   % (iii) update weight
      if error ~= 0
      error_count = error_count + 1;   % error counter
      end
   end
   if c==1 || mod(c,200)==0     % print out algorithm's progress
      fprintf('Iteration count: %d/%d   Error value: %f\n', c, MAXIT, error);
   end
   c = c + 1;
end

if c>=MAXIT
   fprintf('Perceptron terminated because the maximal number of %d iterations was reached.\n',
MAXIT);
else
   fprintf('Perceptron converged.\n');
 end
end
```

Multi-layer perceptron

```
%% Network training phase:

function [w_ih, w_ih_bias, w_ho, w_ho_bias, out_train, errors_vec] = MLPRegressionTrain(x_train,
y_train, bias, r, MAXIT, length_il, length_hl, length_ol)

%% Network initialisation:

% Input layer:
x_in = zeros(length_il,1); %outgoing x

% Hidden layer:
rand('state',sum(100*clock));
w_ih = -1 +2.*rand(length_il, length_hl);   % weights between input and hidden layers
w_ih_bias = -1 +2.*rand(1, length_hl);      % weights between input and hidden layer bias nodes
x_hidd = zeros(length_hl, 1);               % outgoing x
%x_hidd = [-1; x_hidd];
delta_hidd = zeros(length_hl, 1);           % error per hidden node

%Output layer:
rand('state',sum(100*clock));
w_ho = -1 +2.*rand(length_hl, length_ol);  % weights between hidden and output layers
w_ho_bias = -1 +2.*rand(1, length_ol);    % weights between hidden and output layers bias nodes
x_out = zeros(length_ol, 1);               % ougoing x
delta_out = zeros(length_ol, 1);           % error per output node
out_train = zeros(size(y_train));          % predicted value
errors_vec = zeros(MAXIT, length_ol);   % keep record of error evolution
allout = zeros(MAXIT,1);                % keep record of all output values of iteration c

%% Training phase:

c = 1;                  % iteration counter
while(c <= MAXIT)  % do until convergence
```

```
% Iterate through all inputs:
for i=1:size(x_train,1) % for all training inputs

   % (i) Calculate/update net/x values of all neurons/nodes:
   % Input layer x: Load training inputs
   x_in = x_train(i,:)';
   % Hidden layer x:
   for j=1:length_hl   % for all nodes
      x_hidd(j) = sum(w_ih(:,j).*x_in) + w_ih_bias(j)*bias; % H = sum(w*x) + bias*w_bias
      x_hidd(j) = 1/(1+exp(-x_hidd(j)));   % sigmoid(x) (activation function)
   end
   % Output layer x:
   for j=1:length_ol
      x_out(j) = sum(w_ho(:,j).*x_hidd) + w_ho_bias(j)*bias;
      x_out(j) = 1/(1+exp(-x_out(j)));
   end

   % (ii) Back-propagate the network to calculate error/delta for all hidden nodes:
   % Output layer:
   for j=1:length_ol
      errors_vec(c,j) = y_train(i)-x_out(j);
      delta_out(j) = x_out(j)*(1-x_out(j))*errors_vec(c,j);
   end
   % Hidden layer:
   for j=1:length_hl
      delta_hidd(j) = x_hidd(j)*(1-x_hidd(j))*sum(w_ho(j,:).*delta_out');
   end

   %(iii) Weight update: wi = wi + xi * (r*delta_i)
   % Hidden layer x:
   for j=1:length_il
      for k=1:length_hl
         w_ih(j,k) = w_ih(j,k) + x_in(j)*(r*delta_hidd(k));
```

```
        end
    end
    for j=1:length_hl
        w_ih_bias(j) = w_ih_bias(j) + bias*r*delta_hidd(j); % bias node on hidden layer
    end
    % Output layer x:
    for j=1:length_hl
        w_ho(j,:) = w_ho(j,:) + x_hidd(j).*(r*delta_out');
    end
    for j=1:length_ol
        w_ho_bias(j) = w_ho_bias(j) + bias*r*delta_out';  % bias node on output layer
    end

    % Print progress:
    fprintf('Iteration: %d, Predicted value: %.4f, Real value: %.4f\n', c, x_out, y_train(i));

    allout(c) = x_out;
    out_train(i) = x_out;
  end
  c = c+1;

 end
end
```

References

1. R. T. Rockafellar, Convex Analysis (1970)
2. J. F. Bonnans, Convex and Stochastic Optimization (2019)
3. A. A. Ahmadi, Lecture Notes on Convex and Conic Optimization (2020)
4. A. Ben-Tal and A. Nemirovski, Lecture Notes on Modern Convex Optimization
5. S. P. Boyd and L. Vandenberghe, Convex Optimization (2004)
6. D. Bertsekas, Convex Analysis and Optimization (2003)
7. C. Bishop, Pattern Recognition and Machine Learning (2006)
8. S. Theodoridis, Machine Learning: A Bayesian and Optimization Perspective (2015)
9. R. T. Rockafellar, The Convex Analysis of Random Variables (2013)
10. V. Kanade, Lectures Notes on advanced Machine Learning
11. A. N. Giannakopoulos, Lectures Notes on Stochastic Processes II
12. D. P. Kroese, T. Taimre, Z. I. Botev, Handbook of Monte Carlo Methods (2011)
13. J. E. Gentle, W. K. Härdle, Y. Mori, Handbook of Computational Statistics: Concepts and Methods (2012)
14. H. Kushner and G. C. Lin, Stochastic Approximation and Recursive Algorithms and Applications (2003)
15. H. Robbins and S. Monro, A Stochastic Approximation Algorithm (1951)
16. J. Kiefer and J. Wolfowitz, Stochastic Estimation of the Maximisation of a Regression Function (1952)
17. J. C. Spall, Introduction to Stochastic Search and Optimization (2003)


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ