MS-E2122 - Nonlinear Optimization
Lecture 1

Fabricio Oliveira

Systems Analysis Laboratory
Department of Mathematics and Systems Analysis

Aalto University
School of Science

Outline of this lecture

What is optimisation?

- Mathematical programming and optimisation
- Types of mathematical optimisation models

Applications

- Resource allocation
- The pooling problem: refinery operations planning
- Robust optimisation
- Classification: support vector machines

Outline of this lecture

What is optimisation?

Mathematical programming and optimisation

Types of mathematical optimisation models

Applications

Resource allocation

The pooling problem: refinery operations planning

Robust optimisation

Classification: support vector machines

What is optimisation?

Discipline of applied mathematics. The idea is to search values for variables in a given domain that maximise/minimise function values.

Can be achieved by

- Analysing properties of functions / extreme points or

What is optimisation?

Discipline of applied mathematics. The idea is to search values for variables in a given domain that maximise/minimise function values.

Can be achieved by

- Analysing properties of functions / extreme points or
- Applying numerical methods

What is optimisation?

Discipline of applied mathematics. The idea is to search values for variables in a given domain that maximise/minimise function values.

Can be achieved by

- Analysing properties of functions / extreme points or
- Applying numerical methods

Optimisation has important applications in fields such as

- **operations research (OR)**;
- economics;
- statistics;
- machine learning and artificial intelligence.

What is optimisation?

In this course, optimisation is viewed as the core element of mathematical programming.

Math. programming is a central OR modelling paradigm:

- **variables** $\rightarrow$ decisions: business decisions, parameter definitions, settings, geometries, ...;

What is optimisation?

In this course, optimisation is viewed as the core element of mathematical programming.

Math. programming is a central OR modelling paradigm:

- **variables** $\rightarrow$ decisions: business decisions, parameter definitions, settings, geometries, ...;
- **domain** $\rightarrow$ constraints: logic, design, engineering, ...;

What is optimisation?

In this course, optimisation is viewed as the core element of mathematical programming.

Math. programming is a central OR modelling paradigm:

- **variables** $\rightarrow$ decisions: business decisions, parameter definitions, settings, geometries, ...;
- **domain** $\rightarrow$ constraints: logic, design, engineering, ...;
- **function** $\rightarrow$ objective function: measurement of (decision) quality.

What is optimisation?

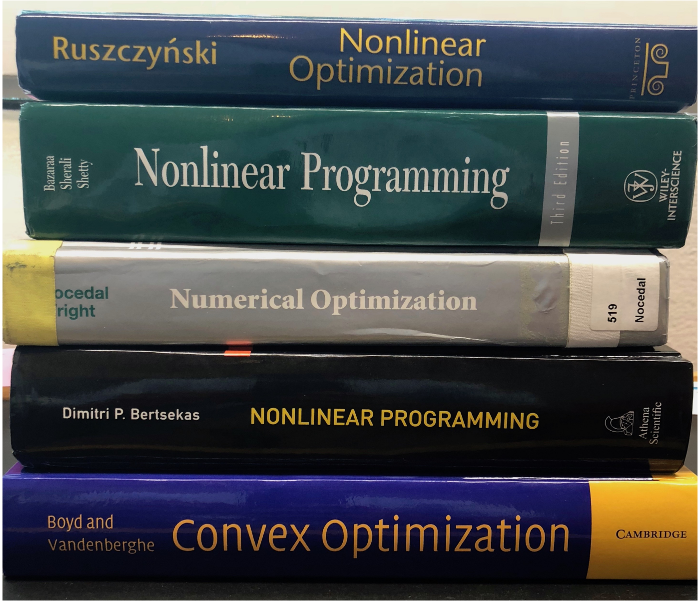
In this course, optimisation is viewed as the core element of mathematical programming.

Math. programming is a central OR modelling paradigm:

- **variables** $\rightarrow$ decisions: business decisions, parameter definitions, settings, geometries, ...;
- **domain** $\rightarrow$ constraints: logic, design, engineering, ...;
- **function** $\rightarrow$ objective function: measurement of (decision) quality.

However, math. programming has many applications in fields other than OR, which causes some confusion;

We will study math. programming in its most general form: both constraints and objectives are nonlinear functions.


Ruszczyński
Nonlinear Optimization
PRINCETON
Bazaraa
Sherali
Shetty
Nonlinear Programming
Third Edition
WILEY-INTERSCIENCE
ocedal
right
Numerical Optimization
519
Nocedal
Dimitri P. Bertsekas
NONLINEAR PROGRAMMING
Athena Scientific
Boyd and
Vandenberghe
Convex Optimization
CAMBRIDGE

Types of programming

The simpler are the assumptions which define a type of problems,
the better are the methods to solve such problems.

Types of programming

The simpler are the assumptions which define a type of problems, the better are the methods to solve such problems.

Some useful notation:

- $x \in \mathbb{R}^n$ - vector of (decision) variables x_j, $j = 1, \dots, n$;

Types of programming

The simpler are the assumptions which define a type of problems, the better are the methods to solve such problems.

Some useful notation:

- $x \in \mathbb{R}^n$ - vector of (decision) variables x_j, $j = 1, \dots, n$;
- $f : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$ - objective function;

Types of programming

The simpler are the assumptions which define a type of problems, the better are the methods to solve such problems.

Some useful notation:

- $x \in \mathbb{R}^n$ - vector of (decision) variables x_j, $j = 1, \dots, n$;
- $f : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$ - objective function;
- $X \subseteq \mathbb{R}^n$ - ground set (physical constraints);

Types of programming

The simpler are the assumptions which define a type of problems, the better are the methods to solve such problems.

Some useful notation:

- $x \in \mathbb{R}^n$ - vector of (decision) variables x_j, $j = 1, \dots, n$;
- $f : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$ - objective function;
- $X \subseteq \mathbb{R}^n$ - ground set (physical constraints);
- $g_i, h_i : \mathbb{R}^n \to \mathbb{R}$ - constraint functions;

Types of programming

The simpler are the assumptions which define a type of problems, the better are the methods to solve such problems.

Some useful notation:

- $x \in \mathbb{R}^n$ - vector of (decision) variables x_j, $j = 1, \dots, n$;
- $f : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$ - objective function;
- $X \subseteq \mathbb{R}^n$ - ground set (physical constraints);
- $g_i, h_i : \mathbb{R}^n \to \mathbb{R}$ - constraint functions;
- $g_i(x) \leq 0$ for $i = 1, \dots, m$ - inequality constraints;

Types of programming

The simpler are the assumptions which define a type of problems, the better are the methods to solve such problems.

Some useful notation:

- $x \in \mathbb{R}^n$ - vector of (decision) variables x_j, $j = 1, \dots, n$;
- $f : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$ - objective function;
- $X \subseteq \mathbb{R}^n$ - ground set (physical constraints);
- $g_i, h_i : \mathbb{R}^n \to \mathbb{R}$ - constraint functions;
- $g_i(x) \leq 0$ for $i = 1, \dots, m$ - inequality constraints;
- $h_i(x) = 0$ for $i = 1, \dots, l$ - equality constraints.

Types of programming

Our goal will be to solve variations of the general problem P:

$$
\begin{aligned}
(P): \quad \text{min. } & f(x) \\
\text{subject to: } & g_i(x) \leq 0, i = 1, \dots, m \\
& h_i(x) = 0, i = 1, \dots, l \\
& x \in X.
\end{aligned}
$$

Types of programming

Our goal will be to solve variations of the general problem P:

$$\begin{aligned}(P): \quad \text{min.} \quad & f(x) \\ \text{subject to:} \quad & g_i(x) \leq 0, i = 1, \dots, m \\ & h_i(x) = 0, i = 1, \dots, l \\ & x \in X.\end{aligned}$$

- **Linear programming (LP):** linear $f(x) = c^\top x$ with $c \in \mathbb{R}^n$; constraint functions $g_i(x)$ and $h_i(x)$ are affine ($a_i^\top x - b_i$, with $a_i \in \mathbb{R}^n$, $b \in \mathbb{R}$); $X = \{x \in \mathbb{R}^n : x_j \geq 0, j = 1, \dots, n\}$.

Types of programming

Our goal will be to solve variations of the general problem P:

$$\begin{aligned}(P):\ \ \text{min. }\ & f(x)\\ \text{subject to: }\ & g_i(x) \leq 0, i = 1, \dots, m\\ & h_i(x) = 0, i = 1, \dots, l\\ & x \in X.\end{aligned}$$

- **Linear programming (LP):** linear $f(x) = c^\top x$ with $c \in \mathbb{R}^n$; constraint functions $g_i(x)$ and $h_i(x)$ are affine ($a_i^\top x - b_i$, with $a_i \in \mathbb{R}^n$, $b \in \mathbb{R}$); $X = \{x \in \mathbb{R}^n : x_j \geq 0, j = 1, \dots, n\}$.
- **Nonlinear programming (NLP):** some (or all) of the functions f, g_i or h_i are nonlinear;

Types of programming

Our goal will be to solve variations of the general problem P:

$$\begin{aligned}(P):\ \ \text{min.}\ \ & f(x)\\ \text{subject to:}\ \ & g_i(x) \leq 0, i = 1, \dots, m\\ & h_i(x) = 0, i = 1, \dots, l\\ & x \in X.\end{aligned}$$

- **Linear programming (LP):** linear $f(x) = c^\top x$ with $c \in \mathbb{R}^n$; constraint functions $g_i(x)$ and $h_i(x)$ are affine $(a_i^\top x - b_i$, with $a_i \in \mathbb{R}^n$, $b \in \mathbb{R})$; $X = \{x \in \mathbb{R}^n : x_j \geq 0, j = 1, \dots, n\}$.
- **Nonlinear programming (NLP):** some (or all) of the functions f, g_i or h_i are nonlinear;
- **(Mixed-)integer programming ((M)IP):** LP where (some of the) variables are binary (or integer). $X \subseteq \mathbb{R}^k \times \{0,1\}^{n-k}$

Types of programming

Our goal will be to solve variations of the general problem P:

$$
\begin{aligned}
(P): \quad \text{min.} \quad & f(x) \\
\text{subject to:} \quad & g_i(x) \leq 0, i = 1, \dots, m \\
& h_i(x) = 0, i = 1, \dots, l \\
& x \in X.
\end{aligned}
$$

- **Linear programming (LP):** linear $f(x) = c^\top x$ with $c \in \mathbb{R}^n$; constraint functions $g_i(x)$ and $h_i(x)$ are affine ($a_i^\top x - b_i$, with $a_i \in \mathbb{R}^n$, $b \in \mathbb{R}$); $X = \{x \in \mathbb{R}^n : x_j \geq 0, j = 1, \dots, n\}$.
- **Nonlinear programming (NLP):** some (or all) of the functions f, g_i or h_i are nonlinear;
- **(Mixed-)integer programming ((M)IP):** LP where (some of the) variables are binary (or integer). $X \subseteq \mathbb{R}^k \times \{0, 1\}^{n-k}$
- **Mixed-integer nonlinear programming (MINLP):** MIP+NLP.

Outline of this lecture

What is optimisation?

- Mathematical programming and optimisation
- Types of mathematical optimisation models

Applications

- Resource allocation
- The pooling problem: refinery operations planning
- Robust optimisation
- Classification: support vector machines

Resource allocation and portfolio optimisation

Problem statement. Plan production that maximises return. Let

- $I = \{1, \dots, i, \dots, M\}$ resources;
- $J = \{1, \dots, j, \dots, N\}$ products;
- c_j - return per unit of product $j \in J$;
- a_{ij} - resource $i \in I$ requirement for making product $j \in J$;
- b_i - availability of resource $i \in I$;
- x_j - production of $j \in J$.

Resource allocation and portfolio optimisation

Problem statement. Plan production that maximises return. Let

- $I = \{1, \dots, i, \dots, M\}$ resources;
- $J = \{1, \dots, j, \dots, N\}$ products;
- c_j - return per unit of product $j \in J$;
- a_{ij} - resource $i \in I$ requirement for making product $j \in J$;
- b_i - availability of resource $i \in I$;
- x_j - production of $j \in J$.

$$\begin{aligned} \text{max.} \quad & \sum_{j \in J} c_j x_j \\ \text{subject to:} \quad & \sum_{j \in J} a_{ij} x_j \le b_i, \forall i \in I \\ & x_j \ge 0, \forall j \in J \end{aligned}$$

Resource allocation and portfolio optimisation

Problem statement. Plan production that maximises return. Let

- $I = \{1, \dots, i, \dots, M\}$ resources;
- $J = \{1, \dots, j, \dots, N\}$ products;
- c_j - return per unit of product $j \in J$;
- a_{ij} - resource $i \in I$ requirement for making product $j \in J$;
- b_i - availability of resource $i \in I$;
- x_j - production of $j \in J$.

$$
\begin{aligned}
\text{max. } & \sum_{j \in J} c_j x_j \\
\text{subject to: } & \sum_{j \in J} a_{ij} x_j \leq b_i, \forall i \in I \\
& x_j \geq 0, \forall j \in J
\end{aligned}
$$

Remark:

- notice that max. $f(x) =$ min. $-f(x)$;

Resource allocation and portfolio optimisation

Problem statement. Plan production that maximises return. Let

- $I = \{1, \dots, i, \dots, M\}$ resources;
- $J = \{1, \dots, j, \dots, N\}$ products;
- c_j - return per unit of product $j \in J$;
- a_{ij} - resource $i \in I$ requirement for making product $j \in J$;
- b_i - availability of resource $i \in I$;
- x_j - production of $j \in J$.

$$\begin{aligned} \text{max. } & \sum_{j \in J} c_j x_j \\ \text{subject to: } & \sum_{j \in J} a_{ij} x_j \leq b_i, \forall i \in I \\ & x_j \geq 0, \forall j \in J \end{aligned}$$

Remark:

- notice that max. $f(x) =$ min. $-f(x)$;
- the base of most practical optimisation problems; exploits mature LP technology.

Portfolio optimization

Problem statement. Plan portfolio of assets to minimise exposition to risk. Let

- $J = \{1, \dots, j, \dots, N\}$ assets;
- μ_j - expected relative return of asset $j \in J$;
- Σ - covariance matrix;
- ϵ - minimum expected return;
- x_j - position of asset $j \in J$

Portfolio optimization

Problem statement. Plan portfolio of assets to minimise exposition to risk. Let

- $J = \{1, \dots, j, \dots, N\}$ assets;
- μ_j - expected relative return of asset $j \in J$;
- Σ - covariance matrix;
- ϵ - minimum expected return;
- x_j - position of asset $j \in J$

$$
\begin{aligned}
\text{min. } \quad & x^\top \Sigma x \\
\text{subject to: } \quad & \mu^\top x \geq \epsilon \\
& 0 \leq x_j \leq 1, \forall j \in J
\end{aligned}
$$

Portfolio optimization

Problem statement. Plan portfolio of assets to minimise exposition to risk. Let

- $J = \{1, \dots, j, \dots, N\}$ assets;
- μ_j - expected relative return of asset $j \in J$;
- Σ - covariance matrix;
- ϵ - minimum expected return;
- x_j - position of asset $j \in J$

$$
\begin{aligned}
\text{min. } \; & x^\top \Sigma x \\
\text{subject to: } \; & \mu^\top x \geq \epsilon \\
& 0 \leq x_j \leq 1, \forall j \in J
\end{aligned}
$$

Remarks:

- The term $x^\top \Sigma x$ measures exposition to risk. It is credited to Harry Markowitz (1952).
- Another important class: quadratic programming (nonlinear).

Refinery Operations Planning Problem

Oil refinery operational planning

- Goal is to maximize profit;
- Several possible configurations;
- Product property specifications must be met;

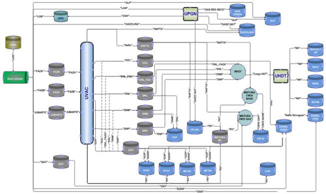
Refinery Operations Planning Problem

Oil refinery operational planning

- Goal is to maximize profit;
- Several possible configurations;
- Product property specifications must be met;

Model characteristics:

- Bilinear (nonconvex) and mixed-integer;
- Large number of flows;
- Several nonlinear constraints.

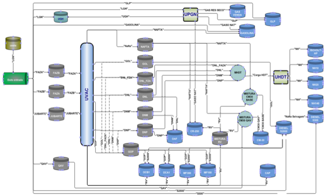


Refinery Operations Planning Problem

Objective: maximize profit

Variables:

- Stream Flows (crude, intermediate and final products);
- Storage;
- Stream properties.



Refinery Operations Planning Problem

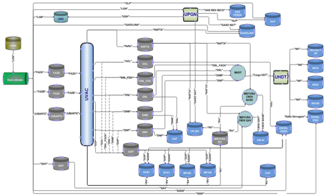
Objective: maximize profit

Variables:

- Stream Flows (crude, intermediate and final products);
- Storage;
- Stream properties.

Constraints

- Mass balance;
- Market features (supply and demand);
- Unit capacities;
- Stream property limits;
- Calculation of mix properties (nonlinear).

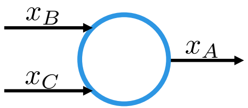


Refinery Operations Planning Problem

The challenging aspect is how to model the calculation of product properties in a mix. Let:

- x_p be the volume of product $p \in P$ and
- q_p the value of a given chemical property (sulphur content, octane content, viscosity...).

In a given mix, mass and property balances are calculated as:



$$x_A = x_B + x_C$$

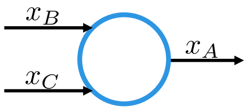
$$q_A = \frac{q_B x_B + q_C x_C}{x_A}$$

Refinery Operations Planning Problem

The challenging aspect is how to model the calculation of product properties in a mix. Let:

- x_p be the volume of product $p \in P$ and
- q_p the value of a given chemical property (sulphur content, octane content, viscosity...).

In a given mix, mass and property balances are calculated as:



$$x_A = x_B + x_C$$
$$q_A = \frac{q_B x_B + q_C x_C}{x_A}$$

Remarks:

- More complex mixes (such as nonlinear balances) might need to be considered.
- These are bilinear programming problems (nonlinear).

Robust optimisation

Is a subarea of mathematical programming concerned with uncertainty in the input data.

It's a risk-averse perspective that seeks protection against variability.

Robust optimisation

Is a subarea of mathematical programming concerned with uncertainty in the input data.

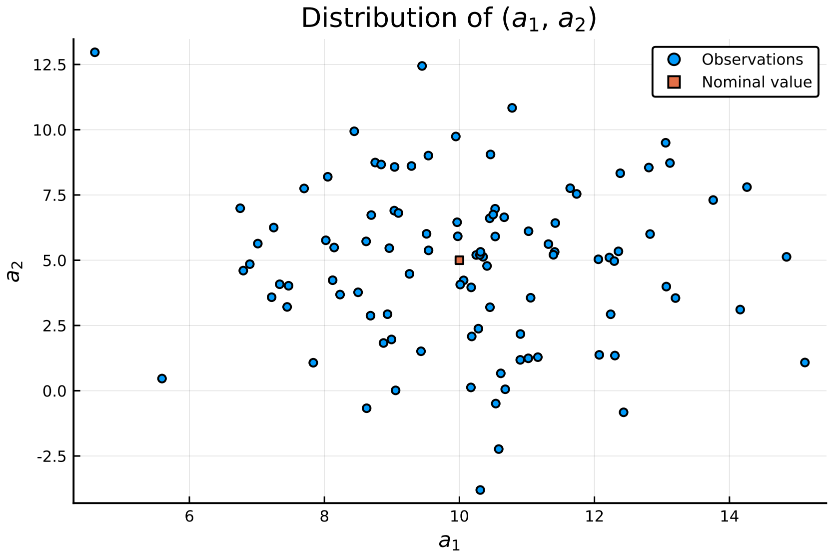
It's a risk-averse perspective that seeks protection against variability.

Consider the resource allocation problem under uncertainty:

$$\begin{aligned} \text{max. } & c^\top x \\ \text{subject to: } & \tilde{a}_i^\top x \leq b_i, \forall i \in I \\ & x_j \geq 0, \forall j \in J, \end{aligned}$$

where $\tilde{a}_i$ is a random variable.

Robust optimisation



Robust optimisation

Assume that, for any $i \in I$, $\tilde{a}_i \in \epsilon_i = \{\overline{a}_i + P_i u : ||u||_2 \leq \Gamma_i\}$, where

- $\overline{a}_i$ is the nominal (average) value;
- P_i is the characteristic matrix of the ellipsoid ϵ;
- Γ_i is risk-aversion control parameter.

Robust optimisation

Assume that, for any $i \in I$, $\tilde{a}_i \in \epsilon_i = \{\overline{a}_i + P_i u : ||u||_2 \leq \Gamma_i\}$, where

- $\overline{a}_i$ is the nominal (average) value;
- P_i is the characteristic matrix of the ellipsoid ϵ;
- Γ_i is risk-aversion control parameter.

Then, the robust counterpart can be stated as

$$\begin{aligned} \text{max. } & c^\top x \\ \text{subject to: } & \max_{a_i \in \epsilon_i} \left\{ a_i^\top x \right\} \leq b_i, \forall i \in I \\ & x_j \geq 0, \forall j \in J. \end{aligned}$$

Robust optimisation

Assume that, for any $i \in I$, $\tilde{a}_i \in \epsilon_i = \{\overline{a}_i + P_i u : ||u||_2 \leq \Gamma_i\}$, where

- $\overline{a}_i$ is the nominal (average) value;
- P_i is the characteristic matrix of the ellipsoid ϵ;
- Γ_i is risk-aversion control parameter.

Then, the robust counterpart can be stated as

$$\begin{aligned} \text{max. } & c^\top x \\ \text{subject to: } & \max_{a_i \in \epsilon_i} \left\{ a_i^\top x \right\} \leq b_i, \forall i \in I \\ & x_j \geq 0, \forall j \in J. \end{aligned}$$

Notice that

$$\max_{a_i \in \epsilon_i} \left\{ a_i^\top x \right\} = \overline{a}_i^\top x + \max_{u} \left\{ u^\top P_i x : ||u||_2 \leq \Gamma_i \right\} = \overline{a}_i^\top x + \Gamma_i ||P_i x||_2$$

Robust optimisation

The robust counterpart can be equivalently stated as:

$$
\begin{aligned}
\text{max.} \ & c^\top x \\
\text{subject to:} \ & \overline{a}_i^\top x + \Gamma_i ||P_i x||_2 \leq b_i, \forall i \in I \\
& x_j \geq 0, \forall j \in J.
\end{aligned}
$$

Remarks:

- In case data is available, P_i can be obtained from the empirical covariance matrix;

Robust optimisation

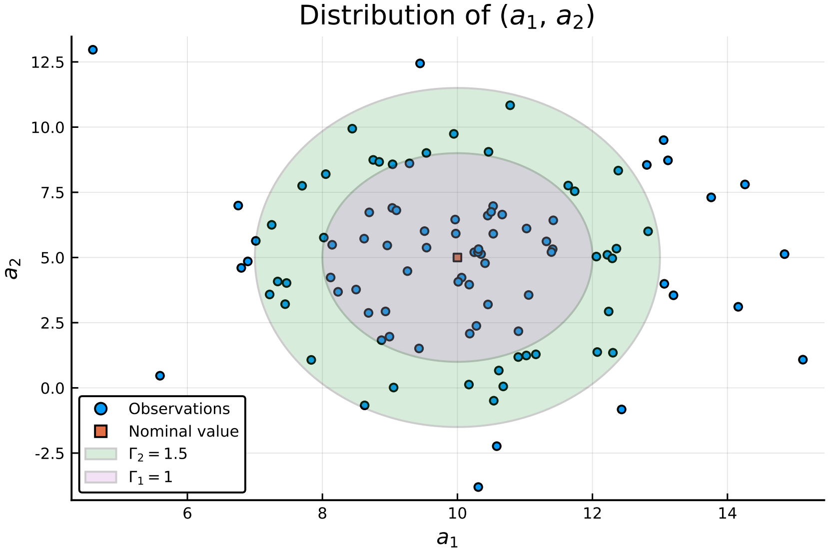
The robust counterpart can be equivalently stated as:

$$\begin{aligned} \text{max.} \quad & c^\top x \\ \text{subject to:} \quad & \overline{a}_i^\top x + \Gamma_i ||P_i x||_2 \leq b_i, \forall i \in I \\ & x_j \geq 0, \forall j \in J. \end{aligned}$$

Remarks:

- In case data is available, P_i can be obtained from the empirical covariance matrix;
- Values of Γ_i can be drawn, for example, from a Chi-squared distribution. Γ_i is sometimes called the budget of uncertainty.

Robust optimisation



Classification

Suppose we are given some data $D \subset \mathbb{R}^n$ that can be separated into two sets in $\mathbb{R}^n$: $I^- = \{x_1, \dots, x_N\}$ and $I^+ = \{x_1, \dots, x_M\}$.

Each element in D is an observation of a given set of features; belonging to either I^- or I^+ defines a classification.

Classification

Suppose we are given some data $D \subset \mathbb{R}^n$ that can be separated into two sets in $\mathbb{R}^n$: $I^- = \{x_1, \dots, x_N\}$ and $I^+ = \{x_1, \dots, x_M\}$.

Each element in D is an observation of a given set of features; belonging to either I^- or I^+ defines a classification.

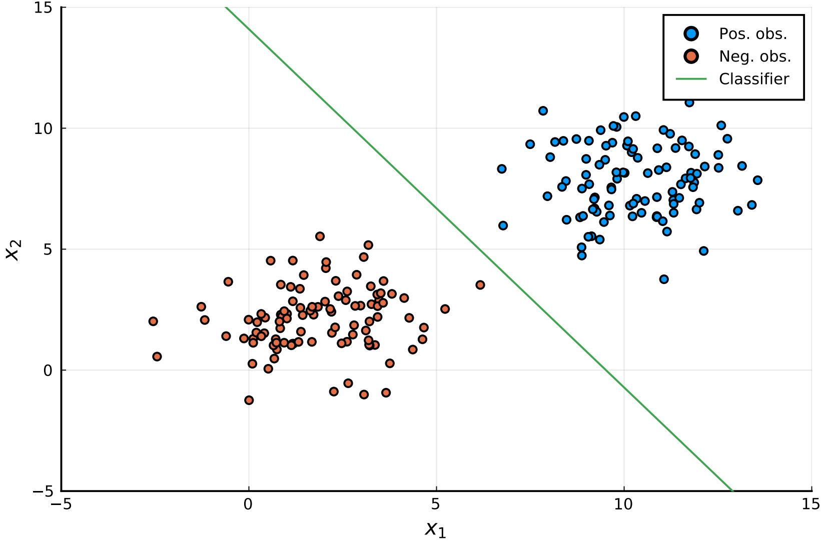
Our task is to select a function $f : \mathbb{R}^n \to \mathbb{R}$ from a given family of functions such that

$$f(x_i) < 0, \ \forall x_i \in I^- \text{ and } f(x_i) > 0, \ \forall x_i \in I^+.$$

Typically, f is selected as a linear classifier, i.e., $f(x_i) = a^\top x_i - b$.

Of course, there is always the possibility of misclassification and, therefore, we want to determine the best possible classifier.

Classification



Classification

Let us define the following error measures:

$$e^-(x_i \in I^-; a, b) := \begin{cases} 0, \text{ if } a^\top x_i - b \leq 0, \\ a^\top x_i - b, \text{ if } a^\top x_i - b > 0. \end{cases}$$

$$e^+(x_i \in I^+; a, b) := \begin{cases} 0, \text{ if } a^\top x_i - b \geq 0, \\ b - a^\top x_i, \text{ if } a^\top x_i - b < 0. \end{cases}$$

Classification

Let us define the following error measures:

$$e^{-}(x_i \in I^{-}; a, b) := \begin{cases} 0, & \text{if } a^\top x_i - b \leq 0, \\ a^\top x_i - b, & \text{if } a^\top x_i - b > 0. \end{cases}$$

$$e^{+}(x_i \in I^{+}; a, b) := \begin{cases} 0, & \text{if } a^\top x_i - b \geq 0, \\ b - a^\top x_i, & \text{if } a^\top x_i - b < 0. \end{cases}$$

Using slack variables u_i, $i = 1, \dots, M$, and v_i, $i = 1, \dots, N$, to represent e^- and e^+, the optimal classifier is obtained from:

$$\begin{aligned}
(LC): \quad \text{min.} \quad & \sum_{i=1}^{M} u_i + \sum_{i=1}^{N} v_i \\
\text{subject to:} \quad & a^\top x_i - b - u_i \leq 0, i = 1, \dots, M \\
& a^\top x_i - b + v_i \geq 0, i = 1, \dots, N \\
& ||a||_2 = 1 \\
& u_i \geq 0, i = 1, \dots, M; v_i \geq 0, i = 1, \dots, N; a \in \mathbb{R}^n, b \in \mathbb{R}.
\end{aligned}$$

Classification

In practice, we can enforce a slab $S = \left\{-1 \leq a^\top x_i - b \leq 1\right\}$ as a buffer to trade off the robustness of the classifier to outliers.

Accordingly, we redefine our error measures as follows.

$$e^-(x_i \in I^-; a, b) := \begin{cases} 0, \text{ if } a^\top x_i - b \leq -1, \\ a^\top x_i - b, \text{ if } a^\top x_i - b > -1. \end{cases}$$

$$e^+(x_i \in I^+; a, b) := \begin{cases} 0, \text{ if } a^\top x_i - b \geq 1, \\ b - a^\top x_i, \text{ if } a^\top x_i - b < 1. \end{cases}$$

Classification

In practice, we can enforce a slab $S = \left\{-1 \leq a^\top x_i - b \leq 1\right\}$ as a buffer to trade off the robustness of the classifier to outliers.

Accordingly, we redefine our error measures as follows.

$$e^-(x_i \in I^-; a, b) := \begin{cases} 0, \text{ if } a^\top x_i - b \leq -1, \\ a^\top x_i - b, \text{ if } a^\top x_i - b > -1. \end{cases}$$

$$e^+(x_i \in I^+; a, b) := \begin{cases} 0, \text{ if } a^\top x_i - b \geq 1, \\ b - a^\top x_i, \text{ if } a^\top x_i - b < 1. \end{cases}$$

e^- and e^+ include misclassifications and correct classifications that lie within S. The latter are know as support vectors.

The width of S is given by $2/||a||_2$, which is the distance between the hyperplanes $a^\top x_i - b = -1$ and $a^\top x_i - b = 1$.

Classification

The robust version of LC incorporating this buffer becomes

$$\begin{aligned}
\text{min.} \quad & \sum_{i=1}^{M} u_i + \sum_{i=1}^{N} v_i + \gamma ||a||_2^2 \\
\text{subject to:} \quad & a^\top x_i - b - u_i \leq -1, \;\; i = 1, \dots, M \\
& a^\top x_i - b + v_i \geq -1, \;\; i = 1, \dots, N \\
& u_i \geq 0, i = 1, \dots, M; v_i \geq 0, i = 1, \dots, N; \\
& a \in \mathbb{R}^n, b \in \mathbb{R}.
\end{aligned}$$

Classification

The robust version of LC incorporating this buffer becomes

$$\begin{aligned}
\text{min.} \quad & \sum_{i=1}^{M} u_i + \sum_{i=1}^{N} v_i + \gamma ||a||_2^2 \\
\text{subject to:} \quad & a^\top x_i - b - u_i \leq -1, \; i = 1, \dots, M \\
& a^\top x_i - b + v_i \geq -1, \; i = 1, \dots, N \\
& u_i \geq 0, i = 1, \dots, M; v_i \geq 0, i = 1, \dots, N; \\
& a \in \mathbb{R}^n, b \in \mathbb{R}.
\end{aligned}$$

Remarks:

- The parameter γ controls the trade-off between the width of the slab S and the number of observations within the slab.

Classification

The robust version of LC incorporating this buffer becomes

$$
\begin{aligned}
\text{min.} \quad & \sum_{i=1}^{M} u_i + \sum_{i=1}^{N} v_i + \gamma ||a||_2^2 \\
\text{subject to:} \quad & a^\top x_i - b - u_i \leq -1, \; i = 1, \dots, M \\
& a^\top x_i - b + v_i \geq -1, \; i = 1, \dots, N \\
& u_i \geq 0, i = 1, \dots, M; v_i \geq 0, i = 1, \dots, N; \\
& a \in \mathbb{R}^n, b \in \mathbb{R}.
\end{aligned}
$$

Remarks:

- The parameter γ controls the trade-off between the width of the slab S and the number of observations within the slab.
- This quadratic programming problem is known in the machine learning literature as support vector machine (SVM).

Classification