# 6. Modeling Languages: Overview

# Introduction

"Pyomo is a Python package that supports the formulation and analysis of mathematical models for complex optimization applications. This capability is commonly associated with commercially available algebraic modeling languages (AMLs) such as AMPL, AIMMS, and GAMS."

## Installation and Usage

If you are using Anaconda:

- `conda install -c conda-forge pyomo`
- `conda install -c conda-forge ipopt coincbc glpk`

If you are not using Anaconda:

- `pip3 install pyomo`

Usage:

- `import pyomo.environ as *`
- `from pyomo.opt import SolverFactory`

# Modeling components

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq b_i, \ i \in I. \end{aligned}$$

What do we need?

1. Model
2. Sets
3. Parameters
4. Variables
5. Objective
6. Constraints
7. Interaction with solvers

# Concrete Model vs. Abstract Model

**Abstract Model**

$$\min_x \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \;\; i \in I$$

$$x_j \geq 0, \; j \in J$$

```
m = AbstractModel()
```

**Concrete Model**

$$\min_x \quad 2x_1 + 3x_2$$

$$\text{s.t.} \quad 3x_1 + 4x_2 \geq 1$$

$$x_1, x_2 \geq 0$$

```
m = ConcreteModel()
```

## Sets

Initialization:

```
m.I = Set()
```

Useful arguments:

- dimen: Dimension of the members of the set.
- initialize: An iterable containing the initial members of the Set, or function that returns an iterable of the initial members the set.

Example:

```
m.I = Set(dimen=2,initialize=[(1,1),(1,2)])
```

Operations:

- `m.I = m.A | m.D # union`
- `m.J = m.A & m.D # intersection`
- `m.K = m.A - m.D # difference`
- `m.L = m.A ^ m.D # exclusive-or`

## Parameters

Initialization:

```
m.A = Set()
m.B = Set()
m.P = Param(m.A, m.B)
```

Useful arguments:

- default: The default value if no other specification is available.

Example:

```
m.S = Param(m.A, m.A, default=0)
```

# Variables

Initialization:

```
m.A = Set()
m.B = Set()
m.x = Var(m.A, m.B)
```

Useful arguments:

- bounds: A function (or Python object) that gives a (lower, upper) bound pair for the variable
- domain: A set that is a super-set of the values the variable can take on.

Example:

```
m.x = Var(m.A, domain=PositiveIntegers, bounds=(0,6))
```

## Objective and Constraints

Initialization of the Objective:

```
def ObjRule(m):
    return sum(m.x[a] for a in m.A) + m.y
m.Obj = Objective(rule=ObjRule, sense=maximize)
```

Initialization of a typical constraint:

```
def Cons1_rule(m, a):
    return m.P[a,a]*m.x[a] <= a
m.Cons1 = Constraint(m.A, rule=Cons1_rule)
```

## Instantiate models using dictionaries

Example:

```
m.I = Set()
m.p = Param()
m.q = Param(m.I)
m.r = Param(m.I, m.I, default=0)
data = {None:  {
'I': {None:  [1, 2, 3]},
'p':  {None:  100},
'q':  {1:10, 2:20, 3:30},
'r':  {(1,1):110, (1,2):120, (2,3):230}}}
i = m.create_instance(data)
```

## Solving and interaction with solvers

Example:

```
i = m.create_instance(data)
opt = SolverFactory('ipopt')
opt.solve(i)
```

Useful arguments of the solve method:

- tee: Boolean argument which controls if the solver output is printed.
- warmstart: Boolean argument which controls if the solver is warm started using the values given in the variables.
- timelimit: Time in seconds after which the solver is told to stop computing and return his current optimal solution.

## Using pyomo with GAMS

Pyomo instance → GAMS → Solver → GAMS → Pyomo Instance

Example:

```
i = m.create_instance(data)
opt = SolverFactory('gams')
opt.solve(i)
```

Useful arguments of the solve method:

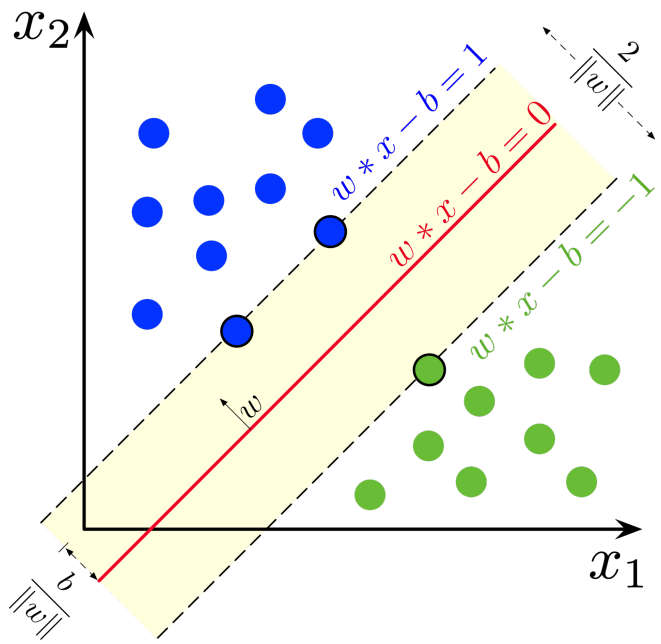
- tee: Boolean argument which controls if the solver output is printed.
- solver: Solver used for the computation.
- warmstart: Boolean argument which controls if the solver is warm started using the values given in the variables.
- add_options: List of additional lines to write directly into model file before the solve statement.
- mtype: Model type.

# Warmstarting and Retrieving Variable Values

Example:

```
m.a = Var()
i = m.create_instance(data)
i.a.value = 2
opt = SolverFactory('gams')
opt.solve(i, warmstart=True)
value_a_after_computation = i.a.value
```

# Example: Soft Margin SVM




$$\min_{b\in\mathbb{R}, w\in\mathbb{R}^n, \xi\in\mathbb{R}^m} \quad \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \quad y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i, \;\; i \in I$$

$$\xi_i \geq 0, \; i \in I.$$