Intro to optimisation

Dhruva V. Raman

Optimisation theory

Extremising quantities subject to constraints

Maximise profits over a portfolio of investments (money, risk constraints)

Minimise classification error by changing network weights (unconstrained)

Minimise delivery times subject to locations, etc

Minimise wind resistance subject to shape constraints

Eat cheaply

Only eat 🍔🥑🧀

Prices

🍔= £3.99

🥑= £1.49

🧀= £4.99

Optimisation variables

$\underline{x} = [$🍔, 🥑, 🧀$]$

e.g. $\underline{x} = [3, 4, 1]$

Objective / loss / cost function

… = money spent

$$f(\underline{x}) = \underline{c}^T \underline{x}$$

$$c^T = [3.99, 1.49, 4.99]$$

$$\min_{\underline{x} \in \mathbb{R}^3} f(\underline{x})$$

“Minimum value of f in *set* $\mathbb{R}^3$”

Gives *output* (price)

e.g. ‘£2.99’

$$\underline{x}^* = \arg\min_{\underline{x} \in \mathbb{R}^3} f(\underline{x})$$

“*Vector* that induces minimum value”

Gives *argument* (input variables) that optimise price

e.g. $\underline{x}^* = [2, 1, 3]$

What’s the solution?

$$f(\underline{x}) = \underline{c}^T \underline{x}$$

$$c^T = [3.99, 1.49, 4.99]$$

$$k^* = \min_{\underline{x} \in \mathbb{R}^3} f(\underline{x})?$$

$$\underline{x}^* = \arg\min_{\underline{x} \in \mathbb{R}^3} f(\underline{x})?$$

$$k^* = -\infty$$

$$\underline{x}^* = [-\infty, -\infty, -\infty]$$

Error : unbounded solution

- Need to add constraints!

Adding an inequality constraint

$$f(\underline{x}) = \underline{c}^T \underline{x}$$

$$c^T = [3.99, 1.49, 4.99]$$

$$\underline{x} = [🍔, 🥑, 🧀]$$

Kcal $\geq$ 2000

Calorific content per unit

🍔= 450

🥑= 300

🧀= 700

$$\tilde{g}_1(\underline{x}) = [450, 300, 700] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \geq 2000$$

Standard form of inequality constraints

… should be ≤ 0

$$g_1(\underline{x}) = 2000 - \tilde{g}_1(\underline{x})$$

$$g_1(\underline{x}) \leq 0$$

Adding another inequality constraint

$$f(\underline{x}) = \underline{c}^T \underline{x}$$

$$c^T = [3.99, 1.49, 4.99]$$

$$\underline{x} = [🍔, 🥑, 🧀]$$

Protein > 100g

Protein per unit

🍔= 9

🥑= 0

🧀= 4

$$g_2(\underline{x}) = 100 - [9, 0, 4] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq 0$$

$$\underline{g}(\underline{x}) = \begin{bmatrix} g_1(\underline{x}) \\ g_2(\underline{x}) \end{bmatrix}$$

$$= \begin{bmatrix} 2000 \\ 100 \end{bmatrix} - \begin{bmatrix} 450 & 300 & 700 \\ 9 & 0 & 4 \end{bmatrix}$$

$$\leq 0$$

Affine constraint

(not linear due to constant term)

$g(\underline{x}) : \mathbb{R}^3 \to \mathbb{R}^2$ (vector-valued function)

Adding an equality constraint

$$f(\underline{x}) = \underline{c}^T \underline{x}$$

$$c^T = [3.99, 1.49, 4.99]$$

$$\underline{x} = [🍔, 🥑, 🧀]$$

Burgers = 1

$$h_1(\underline{x}) = [1, 0, 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

- Can add more equality constraints

General form of optimisation problem

$$\min_{\underline{x} \in \mathbb{R}^n} f(\underline{x})$$

subject to

$g(\underline{x}) \leq 0$ (inequality constraints)

$h(\underline{x}) = 0$ (equality constraints)

g and h are vector-valued functions

$g : \mathbb{R}^n \to \mathbb{R}^m$ means m inequality constraints

(similar for h)

General form of linear optimisation problem

$$\min_{\underline{x} \in \mathbb{R}^n} f(\underline{x}) = \underline{c}^T \underline{x}$$

subject to

$A\underline{x} \leq b$

No equality constraints?

$$g(\underline{x}) = \underline{b} - A\underline{x} \leq 0$$

Equality constraint expressible as two inequality constraints e.g.

$x + 4 = 0 \Leftrightarrow$

$x + 4 \geq 0$ and $x + 4 \leq 0$

Linear optimisation = Linear program

- Nothing to do with computers
- Everything to do with sounding cool to get (US defence) funding!

Our particular linear program

$$\min_{\underline{x} \in \mathbb{R}^n} f(\underline{x}) = \underline{c}^T \underline{x}$$

subject to

$$A\underline{x} \leq b$$

$$\underline{c}^T = [3.99, 1.49, 4.99]$$

Express following constraints as matrix:

- Kcal > 2000
- Protein > 100
- Burgers = 1

Protein per unit

🍔= 9

🥑= 0

🧀= 4

Calorific content per unit

🍔= 450

🥑= 300

🧀= 700

Our particular linear program

$\min_{\underline{x} \in \mathbb{R}^n} f(\underline{x}) = \underline{c}^T \underline{x}$ subject to $A\underline{x} \leq b$

$$\underline{c}^T = [3.99, 1.49, 4.99]$$

$$A\underline{x} = \begin{bmatrix} -450 & -300 & -700 \\ -9 & 0 & -4 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad b = \begin{bmatrix} -2000 \\ -100 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \left(\begin{array}{c} \text{- calories less than -2000} \\ \text{- protein < - 100} \\ \text{more than one burger} \\ \text{less than one burger} \\ \text{positivity constraint} \\ \text{positivity constraint} \\ \text{positivity constraint} \end{array} \right)$$

Massaging constraints is a key art of formulating optimisation problems

Specific massage depends on what your optimisation package needs!

Relax about the issues

$$\min_{\underline{x}\in\mathbb{R}^n} f(\underline{x}) = \underline{c}^T\underline{x}$$

subject to

$$A\underline{x} \leq b$$

- I can't buy 1.35 burgers!

Integer programs $x \in \mathbb{Z}^n$ … are horrible to solve

Continuous programs $x \in \mathbb{Z}^n$ … are easy…so round your solution!

Relaxing an optimisation

Approximate difficult problem with similar easy problem

- Art of relaxation is critical optimisation skill

Example code

Factory makes products A (£30 profit) and B (£20 profit)

- A: 2 hours of labour, 1kg of material
- B: 1 hour of labour, 2kg of material

Maximise profit in one day (8 hours) with $8kg$ material

Maximise $\begin{bmatrix} 30 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ subject to

$$2x_1 + x_2 \leq 8$$

$$x_1 + 2x_2 \leq 8$$

$$x_1, x_2 \geq 0$$

```python
import cvxpy as cp

# Define variables
x1 = cp.Variable(nonneg=True)
x2 = cp.Variable(nonneg=True)

# Define objective (maximize 30x1 + 20x2)
objective = cp.Maximize(30*x1 + 20*x2)

# Define constraints
constraints = [
    2*x1 + x2 <= 8,
    x1 + 2*x2 <= 8
]

# Define and solve the problem
problem = cp.Problem(objective, constraints)
problem.solve()

# Display results
print("Status:", problem.status)
print("Optimal value (profit):", problem.value)
print(f"Optimal x1 = {x1.value:.2f}")
print(f"Optimal x2 = {x2.value:.2f}")
```

```
Status: optimal
Optimal value (profit): 133.33333321833703
Optimal x1 = 2.67
Optimal x2 = 2.67
```

- CVX.jl very similar. JuMP.jl as well.

Infeasibility

$$\min_{\underline{x}} f(\underline{x})$$

subject to

$$\begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix} \underline{x} = \underline{0}$$

What can you say about the kernel of A?

What can you say about the feasible set of the program?

- Rank 2 (invertible)
- Rank-nullity theorem means nullity 0
- $Ker(A) = \{\underline{0}\}$

Feasible set is $\{0\}$: we've solved the optimisation!

Infeasibility

$$\min_{\underline{x}} f(\underline{x})$$

subject to

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 7 \end{bmatrix} \underline{x} = \underline{0}$$

What can you say about the kernel of A?

What can you say about the feasible set of the program?

- Rank 2 (singular)
- Rank-nullity theorem means nullity 1
- $Ker(A) = \{\underline{v} : v \propto \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}\}$

Feasible set is $Ker(A)$.

Quadratic (least-squares) programs

$$\underline{x}^T Q \underline{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Expand for yourself…

- $= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} x_1 + 3x_2 \\ 2x_1 + 5x_2 \end{bmatrix}$
- $= x_1^2 + 5x_2^2 + 5x_1 x_2$

Replace $Q = \begin{bmatrix} Q_{12} & Q_{21} \\ Q_{11} & Q_{22} \end{bmatrix}$

This is a quadratic form

$\underline{x}^T Q \underline{x} = \sum_{i=1,j=1}^{n} Q_{ij} x_i x_j$

- Always gives a scalar output

$$\min_{\underline{x}} f(\underline{x}) = \underline{x}^T Q \underline{x} + \underline{c}^T \underline{x}$$

subject to

$A\underline{x} \leq \underline{b}$

- Only linear constraints preserve convexity (which make solving easy)

Example: polynomial fitting

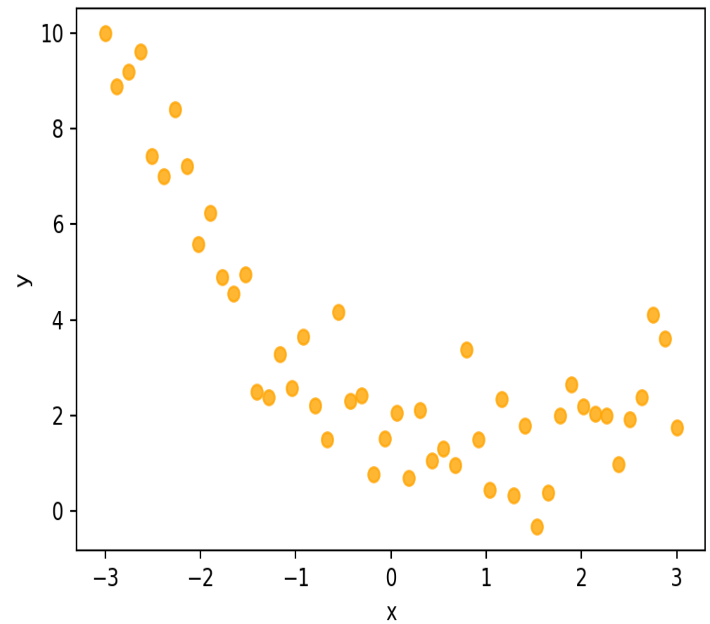
$$f(\underline{x}, \underline{w}) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

Data (each row is dot)

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \qquad \underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix}$$

Predictions

$$\underline{\hat{y}} = A\underline{w}$$



Least-squares regression:

$$\min_{\underline{w} \in \mathbb{R}^3} \|\underline{y} - A\underline{w}\|_2^2$$

$$\|\underline{v}\|_2^2 = \underline{v}^T \underline{v} = v_1^2 + v_2^2 + \dots$$

- minimise sum of squared residuals

Example: polynomial fitting

$$f(\underline{x}, \underline{w}) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

Predictions vs targets

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} \qquad \underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{bmatrix}$$

Predictions $\underline{\hat{y}} = A\underline{w}$

Least-squares regression:

$$\min_{\underline{w} \in \mathbb{R}^3} \|\underline{y} - A\underline{w}\|_2^2$$

Algebra practice: expand square!

$$\|\underline{y} - \underline{\hat{y}}\|_2^2 = (\underline{y} - \underline{\hat{y}})^T (\underline{y} - \underline{\hat{y}})$$

$$= \underline{y}^T \underline{y} - \underline{\hat{y}}^T \underline{y} - \underline{y}^T \underline{\hat{y}} + \underline{\hat{y}}^T \underline{\hat{y}}$$

$$= \underline{y}^T \underline{y} - (A\underline{w})^T \underline{y} - \underline{y}^T (A\underline{w}) + (A\underline{w})^T ($$

$$= \underline{y}^T \underline{y} - \underline{w}^T A^T \underline{y} - \underline{y}^T A\underline{w} + \underline{w}^T A^T A\underline{w}$$

$$= \underline{y}^T \underline{y} - 2\underline{y}^T A\underline{w} + \underline{w}^T A^T A\underline{w}$$

- Which terms can be ignored (independent of $\underline{w}$)?

Unconstrained quadratic program

Minimise $\underline{w}^T Q\underline{w} + \underline{cw}$ where $Q = A^T A$, $c = 2\underline{y}^T A$

When can we skip to neural network optimisation?

Linear/quadratic programs $\subset$ convex programs

Convex programming is unsung hero

Preferred to neural networks in:

- Robotics (e.g. Boston dynamics)
- Spacecraft trajectory planning/landing (e.g. reusable rockets)
- Car/plane stabilisation and control
- Finance
- Logistics

Convex programs are superior to neural networks

(just less widely applicable)

- Much much faster
- Accuracy and optimality guarantees
- Easy to code (good packages)

The zoology of optimisation

General form of constrained program:

$$\min_{\underline{x} \in \mathbb{R}^n} f(\underline{x})$$

subject to

$$g(\underline{x}) \leq 0, \qquad h(\underline{x}) = 0$$

Linear/quadratic program:

- Specific constraints on f, g, h
- Means better algorithms than generalist solvers (e.g. LBFGS, gradient descent)

More generally

- Many different types of program
- Challenge: find a nice form for f, g, h

Mixed-integer program (MIP), Semidefinite program (SDP), second-order cone program (SOCP), quadratically-constrained quadratic program (QCQP)…

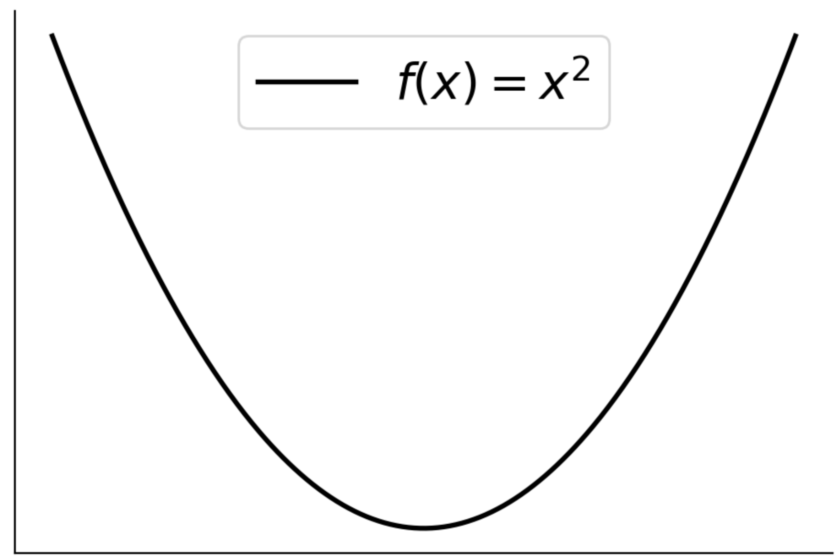
How do optimisers solve?

$$\min_{\underline{x} \in \mathbb{R}^n} f(\underline{x})$$

subject to

$$g(\underline{x}) \leq 0, \qquad h(\underline{x}) = 0$$

1. **Unconstrained** optimisation (forget g, h)



Derivative at minimum?

- $f'(x) = 0$ (flat)

- Is zero-derivative a necessary condition?
- Is zero-derivative a sufficient condition?

Interlude: the cold war

- Soviet union developed much of optimisation theory. Why?
- Centrally planned economy!

Problem: no (good) computers

Linear programming in the 1930s (nobel prizes):

- **Soviet union**: Kantorovich
- ~~**Soviet union**~~ **America**: Leontief

- Critical to WW2 logistics

Calculating missile trajectores is a quadratic program

Solving “optimal control” problems

Americans: Bellman equations

(globally optimal, difficult to compute)

Soviets: Pontryagin’s maximum principle

(locally optimal, easy to work)

- I don’t like missiles, but some of my favourite maths in Pontryagin