An Ode to Linear Programming

The plan

- Why talk about Linear Programming?
- A light introduction to Linear Programming
- The fun part: Mixed Integer Linear Programming
- Applications & Limitations
- Workshop: 4 levels of Linear Programming

Why this talk?

Hello :)

- Mathias Brandewinder
- Using F# as my primary language for nearly 10 years
- Background: economics, operations research

Why Linear Programming?

- Focused on F# + ML in recent years
- "We need to solve this problem with ML or AI"
- ... Do you really?
- LP, MILP are tragically under-appreciated
- Google OR Tools: affordable LP / MILP library

Linear Programming

What is Linear Programming

- Dates back to the 40s
- Solver: Simplex algorithm (Dantzig)
- Linear objective function, linear constraints

An example: the Perfect Burrito

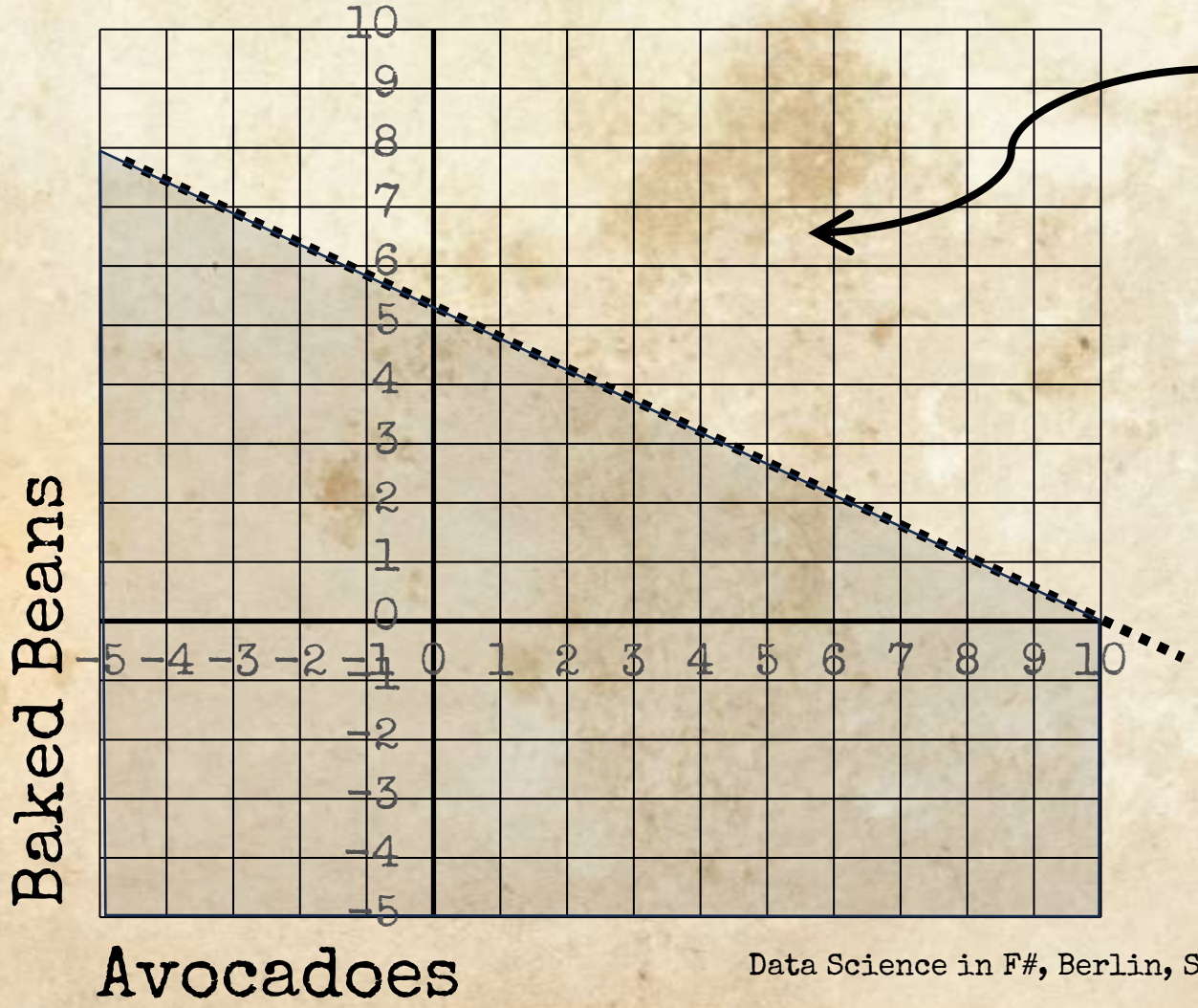
- We want to cook a vegan burrito
- We want to meet daily dietary requirements
- All we have is Avocadoes and Baked Beans

The Perfect Burrito: Potassium

- 1 serving of Avocado (1/2 avocado) ~ 10% daily needs
- 1 serving of Baked Beans (1 cup) ~ 19% daily needs

- 0.10 * A(vocadoes) + 0.19 * B(aked Beans) >= 1.00
- 0.19 * B >= 1.00 - 0.10 * A
- B >= 5.26 - 0.52 * A
  - A = 0.0 => B ~ 5.26
  - B = 0.0 => A ~ 10.00

Potassium requirements



More requirements...

- Vitamin C
- 1 serving of Avocado (1/2 avocado) ~ 11% daily needs
- 1 serving of Baked Beans (1 cup) ~ 3% daily needs
- 0.11 * Avocadoes + 0.03 * Baked Beans >= 1.0

- Burrito Weight <= 0.5 kilograms
- Burrito Volume <= 10 cm3

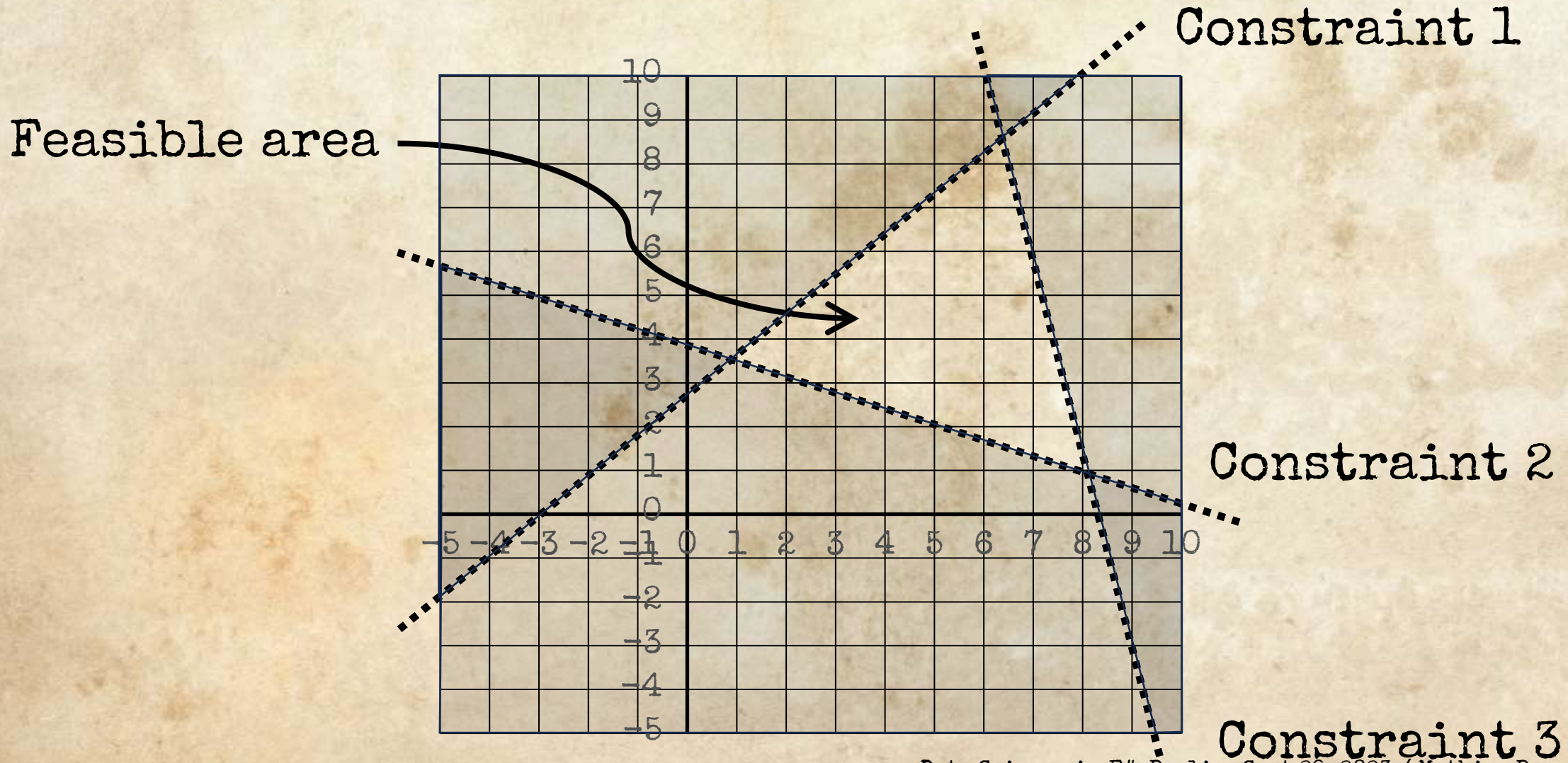
A pattern emerges

- `Coeff_A * A + Coeff_B * B <= Constant`

- Linear equation / Linear Combination
- Sum of Coefficients * Variables
- Defines a half-plane

- Note: ... >= Constant can be converted to same form

Feasible region

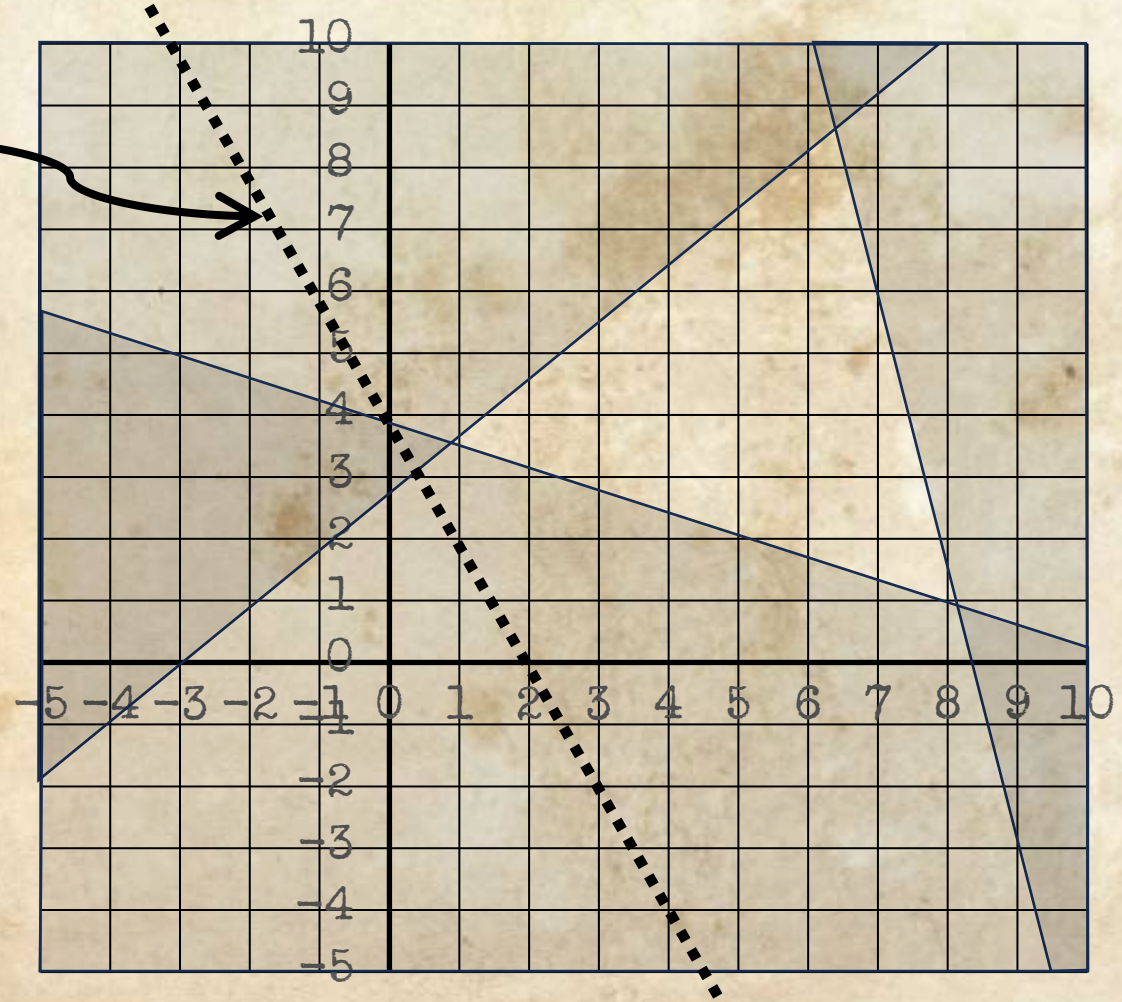


Cheapest acceptable burrito

- Cost = 0.5 * Avocado + 0.25 * Beans
- If cost = $1.0
- 1.0 = 0.5 * A + 0.25 * B
- B = 4.0 - 2.0 * A

Minimum feasible cost

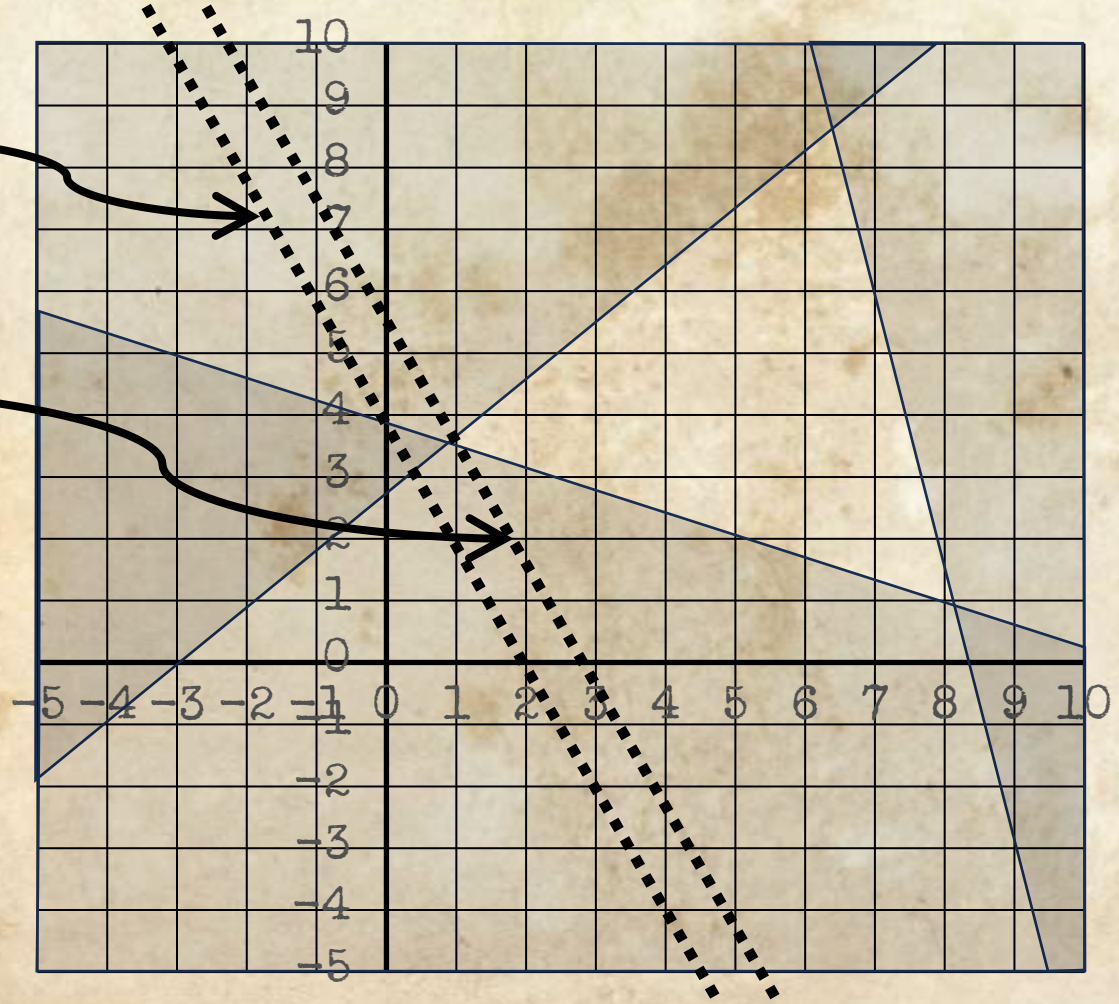
Cost = $1.0



Minimum feasible cost

Cost = $1.0

Minimum feasible cost



From 2 to N variables

- Original: Coeff_A * A + Coeff_B * B <= Constant

- Generalized: N variables X_1, X_2, ... X_N
- Objective: $C_1 * X_1 + C_2 * X_2 + ... + C_N * X_N$
- Constraint : $A_1 * X_1 + A_2 * X_2 + ... + A_N * X_N <= Constant$

- Everything still works out the same

Linear Programming

- Setup the problem:
- Define N variables of interest
- Define 1 Linear Objective (what we want to maximize)
- Define Linear Constraints
- Solve! Give me values for the variables that work

Very powerful

- If it's a linear problem, Simplex will solve it
- Strong guarantees: Optimal, Infeasible, Unbounded
- Handles constraints directly
- Surprisingly fast
- Suitable for large scale problems

Mixed Integer Linear Programming

Sometimes it's not a float

- LP variables are assumed to be continuous / floats
- Not everything is a float!
- MILP: some variables MUST be integer values

MILP

- Mixture of continuous and integer variables
- Different algorithm
- From a code perspective, no major differences
- Same guarantees on solution
- Slower (integer variables are expensive)

The Perfect Burrito: Potassium

- 1 serving of Avocado (1/2 avocado) ~ 10% daily needs
- 1 serving of Baked Beans (1 cup) ~ 19% daily needs

- 0.10 * A(vocadoes) + 0.19 * B(aked Beans) >= 1.00

Code example (F#, Google OR Tools)

```
open Google.OrTools.LinearSolver
// Mixed Integer Solver
let solver = Solver.CreateSolver("SCIP")

let var1 = solver.MakeNumVar(0.0, infinity, "var1")
let var2 = solver.MakeIntVar(0, 10, "var2")
let var3 = solver.MakeBoolVar("var3")
```

Code example (F#, Google OR Tools)

```
// Setting up a constraint
// 0.0 <= 1.0 * var1 + 2.0 * var2 + 3.0 * var3 <= 100.0
let c = solver.MakeConstraint("constraint1")
c.SetLb(0.0)
c.SetUb(100.0)
c.SetCoefficient(var1, 1.0)
c.SetCoefficient(var2, 2.0)
c.SetCoefficient(var3, 3.0)
```

MILP opens up many possibilities

- Booleans: integers, 0 | 1
- Suppose A: 0 | 1, B: 0 | 1, can we represent X = A AND B?
- Careful! We CANNOT multiply A and B
- Constraints
  - X >= A + B - 1,
  - X <= A,
  - X <= B,
  - 0 <= X <= 1
- OR, NOT, XOR, IF, ...

Applications & Limitations

"If your only tool is a hammer,
everything looks like a nail"

... but what if it's a huge hammer?

Linear Programming is a Huge Hammer!

- Not everything needs “Machine Learning” or “AI”
- Very strong benefits: guaranteed global optimum
- Very strong benefits: constraints handling
- Very efficient solvers

- **If you recognize a LP in your model, your job is done**

Applications

- Many problems ARE Linear
  - Production schedule, resource allocation, network flow, ...
- Total cost, total weight, total volume moved, ...
- With some tricks, Mixed Integers LP can handle a lot
  - Integers add Boolean logic
- Linear can be a good enough approximation
- Parts of your model can be solved with an LP

Limitations

- Sometimes, it's really not linear
- Mostly non linear objective function
- Can sometimes get away with linear approximation
- Unsurprisingly, some MILPs will be slow

Thank you!

www.hachyderm.io/@brandewinder

www.brandewinder.com

Workshop tomorrow!