Inference: Integer Linear Programs

CS 6355: Structured Prediction

THE UNIVERSITY OF UTAH

So far in the class

- Thinking about structures
  - A graph, a collection of parts that are labeled jointly, a collection of decisions
- Algorithms for learning
  - Local learning
    - Learn parameters for individual components independently
    - Learning algorithm not aware of the full structure
  - Global learning
    - Learn parameters for the full structure
    - Learning algorithm "knows" about the full structure
- This section: ***Prediction***
  - Sets structured prediction apart from binary/multiclass

Inference

- What is inference?
  - An overview of what we have seen before
  - Combinatorial optimization
  - Different views of inference
- Graph algorithms
  - Dynamic programming, greedy algorithms, search
- Integer programming
- Heuristics for inference
  - Sampling
- Learning to search

The big picture

- MAP Inference is combinatorial optimization
- Combinatorial optimization problems can be written as integer linear programs (ILP)
  - The conversion is not always trivial
  - Allows injection of “knowledge” into the inference in the form of constraints
- Different ways of solving ILPs
  - Commercial solvers: CPLEX, Gurobi, etc
  - Specialized solvers if you know something about your problem
    - Incremental ILP, Lagrangian relaxation, etc
  - Can approximate to linear programs and hope for the best
- Integer linear programs are NP hard in general
  - No free lunch

Today's Agenda

- Linear and integer linear programming
  - What are they?
  - The geometric perspective

- ILPs for inference
  - Simple example: Multiclass classification
  - More general structures

Detour: Linear programming

- Minimizing a linear objective function subject to a finite number of linear constraints (equality or inequality)

- Very widely applicable
  - Operations research, micro-economics, management

- Historical note/anecdote
  - Developed during world war 2 to optimize army expenditure
    - Nobel Prize in Economics 1975
  - "Programming" not the same as computer programming
    - "*Program*" referred to military schedules and programming referred to optimizing the program

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

Minimize total cost

such that

At least 5 units of vitamin Z,

At least 3 units of nutrient X,

The number of units purchased is not negative

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

$$\min \quad 2c + 6s + 0.3d$$

Minimize total cost

such that

At least 5 units of vitamin Z,

At least 3 units of nutrient X,

The number of units purchased is not negative

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

$$\min \quad 2c + 6s + 0.3d$$

Minimize total cost

such that

$$4c + 10s + 0.01d \geq 5$$

At least 5 units of vitamin Z,

At least 3 units of nutrient X,

The number of units purchased is not negative

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

$$\min \quad 2c + 6s + 0.3d$$

Minimize total cost

$$\text{such that}$$

$$4c + 10s + 0.01d \geq 5$$

At least 5 units of vitamin Z,

$$0.4c + 4s + 2d \geq 3$$

At least 3 units of nutrient X,

The number of units purchased is not negative

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

$$\min \quad 2c + 6s + 0.3d$$

Minimize total cost

such that

$$4c + 10s + 0.01d \geq 5$$

At least 5 units of vitamin Z,

$$0.4c + 4s + 2d \geq 3$$

At least 3 units of nutrient X,

$$c \geq 0, s \geq 0, d \geq 0.$$

The number of units purchased is not negative

Linear programming

In general

$$
\begin{aligned}
\max \quad & \mathbf{c}^T\mathbf{x} \\
\text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\
& \mathbf{x} \geq 0.
\end{aligned}
$$

linear (objective $\mathbf{c}^T\mathbf{x}$)

linear (constraint $A\mathbf{x} \leq \mathbf{b}$)

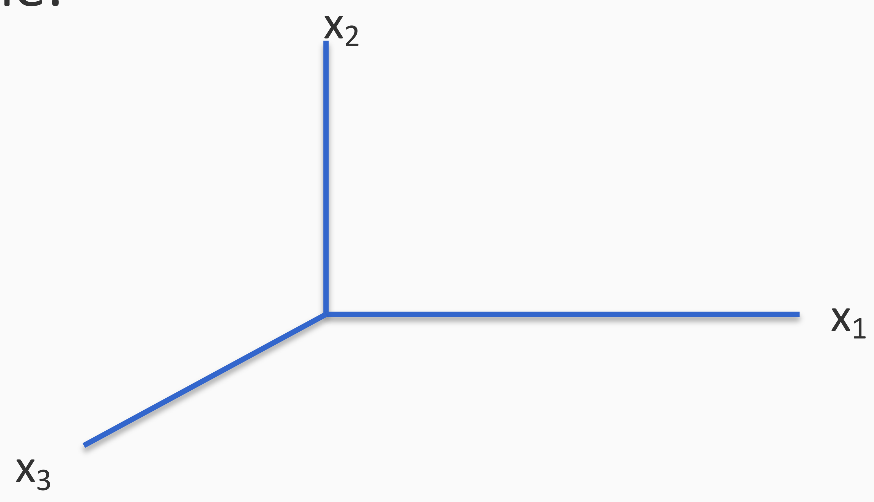
Linear programming

In general

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

This is a continuous optimization problem

- And yet, there are only a finite set of possible solutions
- For example:



Linear programming

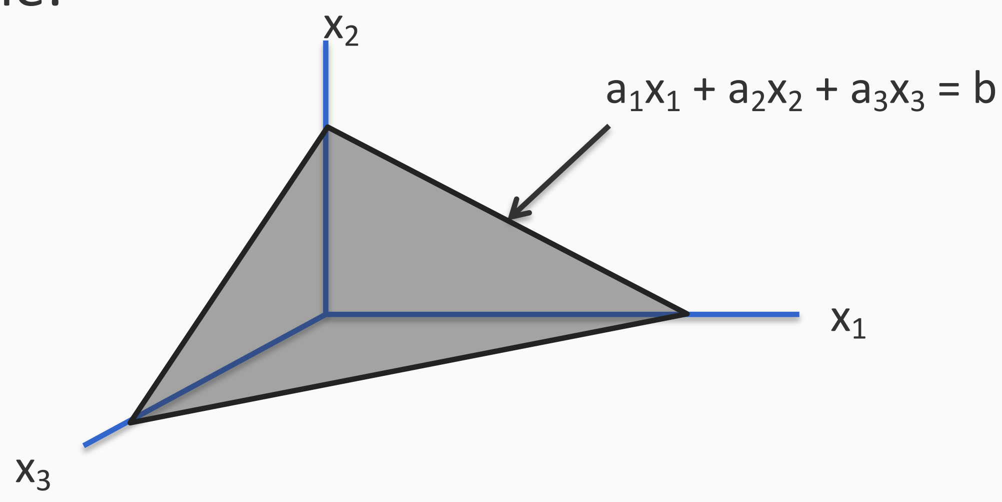
In general

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

This is a continuous optimization problem

– And yet, there are only a finite set of possible solutions

– For example:



Linear programming

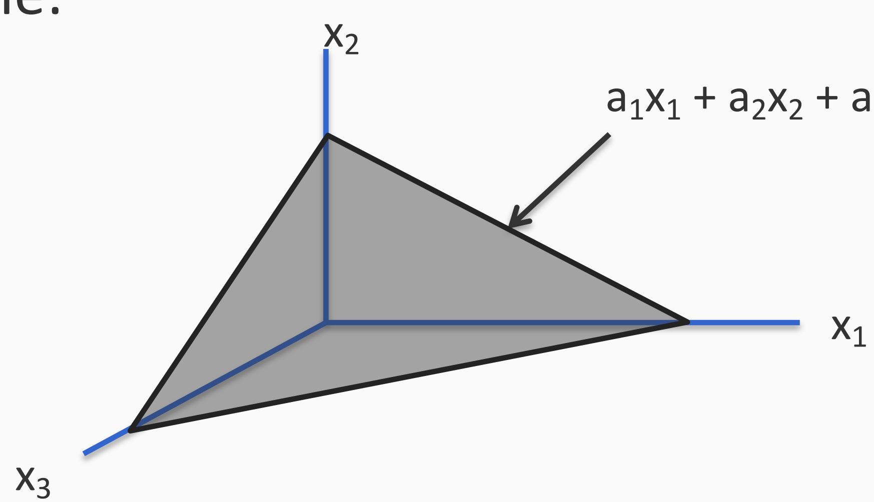
In general

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

This is a continuous optimization problem

- And yet, there are only a finite set of possible solutions
- For example:

Suppose we had to maximize any $\mathbf{c}^T\mathbf{x}$ on this region



Linear programming

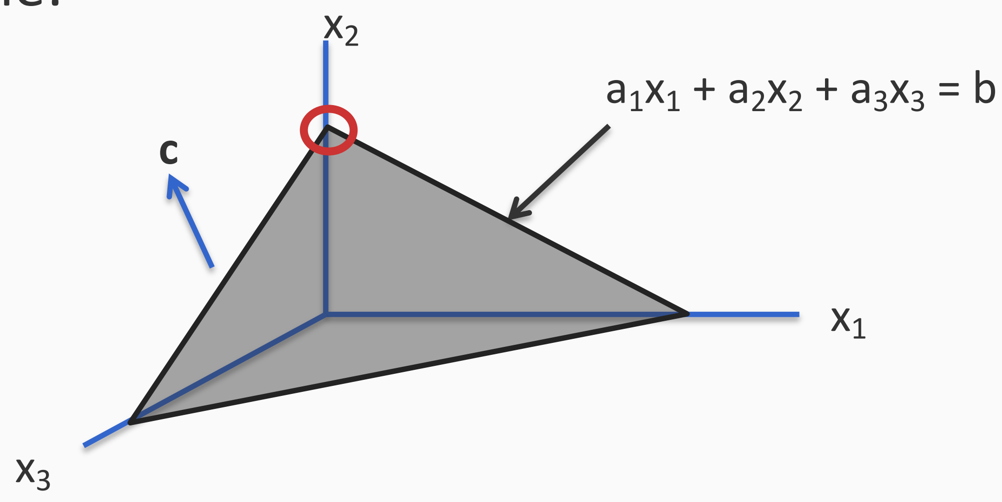
In general

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

This is a continuous optimization problem

- And yet, there are only a finite set of possible solutions
- For example:

Suppose we had to maximize any $\mathbf{c}^T\mathbf{x}$ on this region



Linear programming

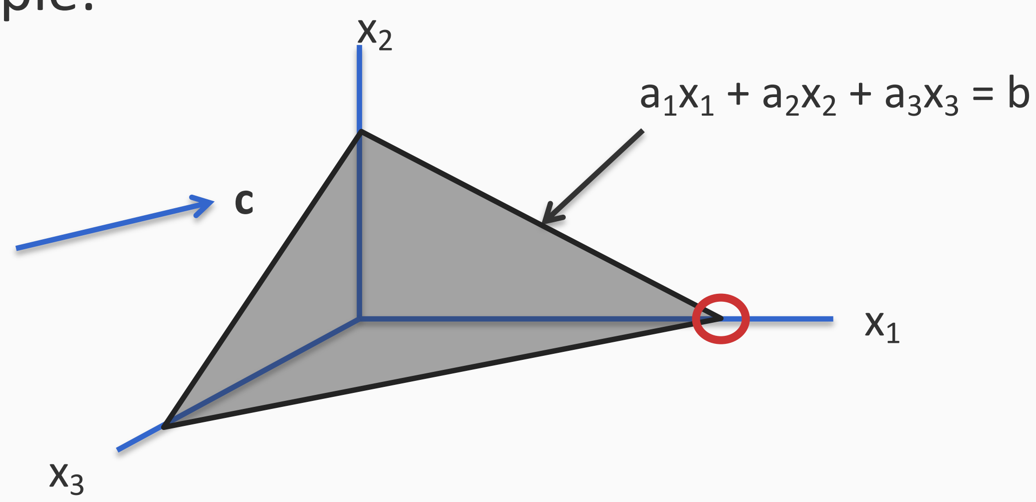
In general

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge 0. \end{aligned}$$

This is a continuous optimization problem

– And yet, there are only a finite set of possible solutions
– For example:

Suppose we had to maximize any $\mathbf{c}^T\mathbf{x}$ on this region



Linear programming

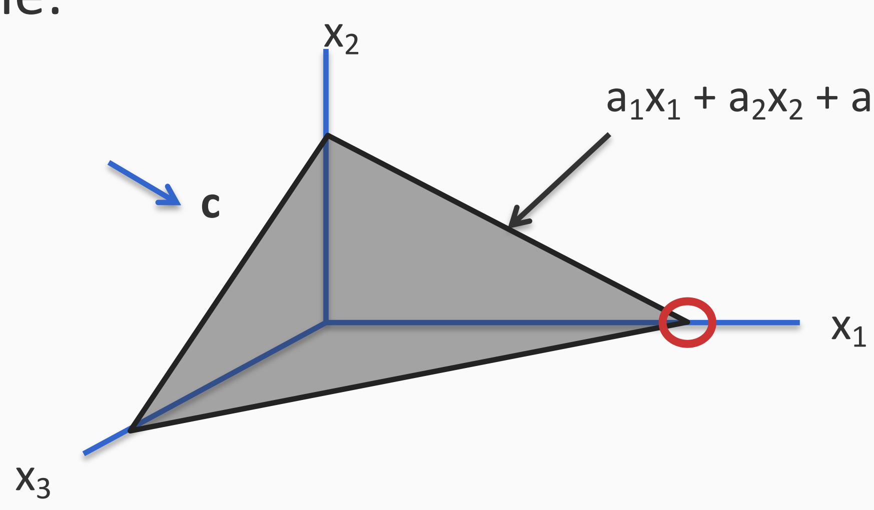
In general

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

This is a continuous optimization problem

- And yet, there are only a finite set of possible solutions
- For example:

Suppose we had to maximize any $\mathbf{c}^T\mathbf{x}$ on this region



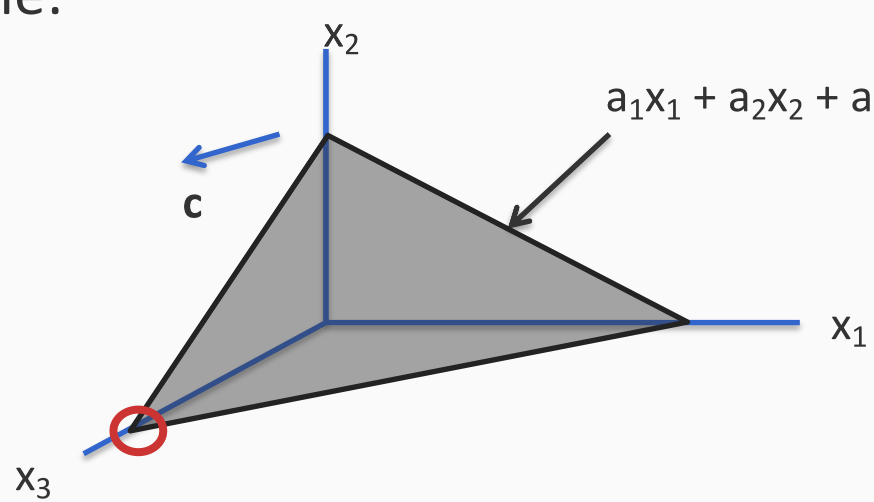
Linear programming

In general
$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge 0. \end{aligned}$$

This is a continuous optimization problem

- And yet, there are only a finite set of possible solutions
- For example:

Suppose we had to maximize any $\mathbf{c}^T\mathbf{x}$ on this region



Linear programming

In general

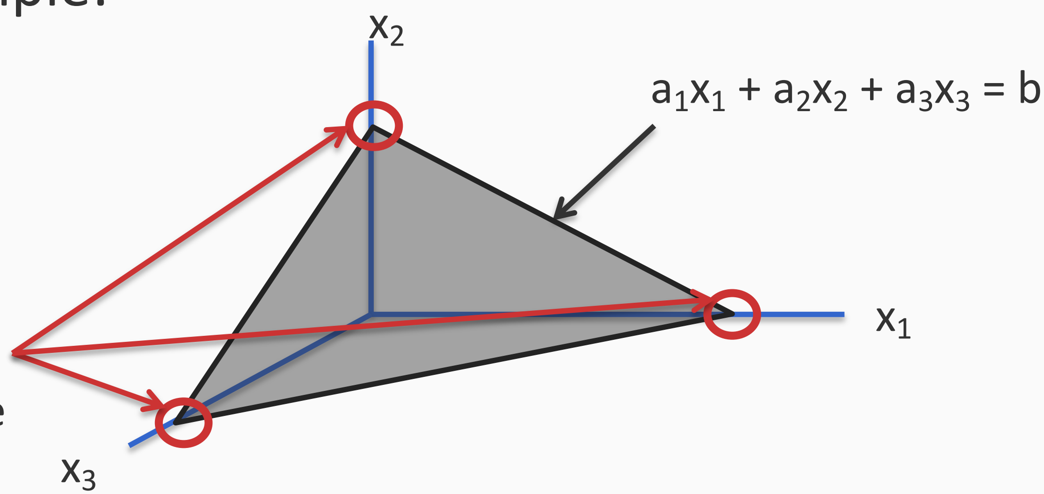
$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

This is a continuous optimization problem

- And yet, there are only a finite set of possible solutions
- For example:

Suppose we had to maximize any $\mathbf{c}^T\mathbf{x}$ on this region

These three vertices are the only possible solutions!



Linear programming

In general

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge 0. \end{aligned}$$

This is a continuous optimization problem

- And yet, there are only a finite set of possible solutions
- The constraint matrix defines a convex polytope
- Only the vertices or faces of the polytope can be solutions

Geometry of linear programming

The constraint matrix defines a polytope that contains allowed solutions (possibly not closed)

$$
\begin{aligned}
\max \quad & \mathbf{c}^T \mathbf{x} \\
\text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\
& \mathbf{x} \geq 0.
\end{aligned}
$$

Geometry of linear programming

The constraint matrix defines a polytope that contains allowed solutions (possibly not closed)

One of the constraints: $A_i^T \boldsymbol{x} \leq b_i$

$$
\begin{aligned}
\max \quad & \mathbf{c}^T \mathbf{x} \\
\text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\
& \mathbf{x} \geq 0.
\end{aligned}
$$

Geometry of linear programming

The constraint matrix defines a polytope that contains allowed solutions (possibly not closed)

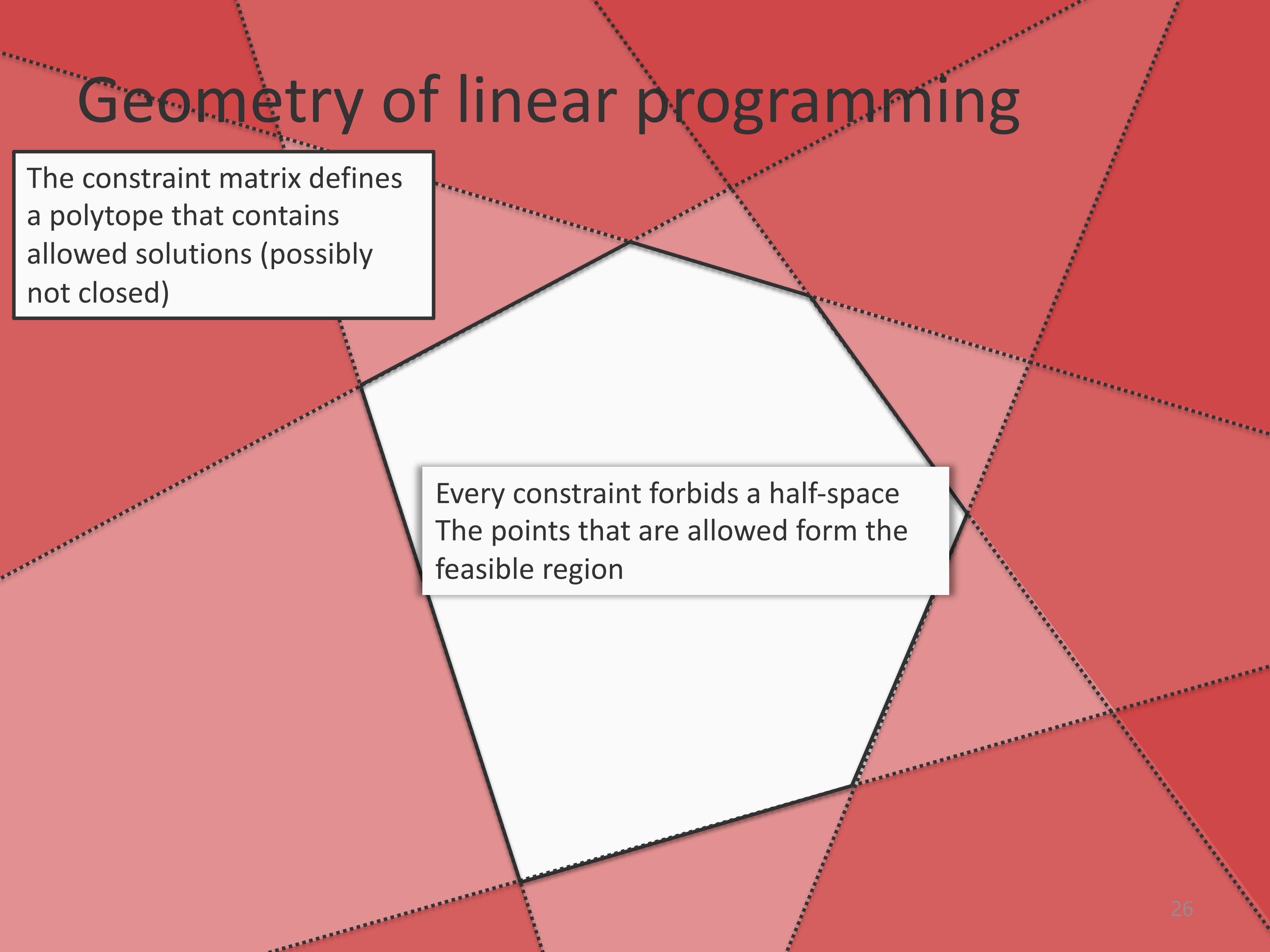
One of the constraints: $A_i^T \boldsymbol{x} \leq b_i$

Points in the shaded region can are not allowed by this constraint

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

Geometry of linear programming

The constraint matrix defines a polytope that contains allowed solutions (possibly not closed)

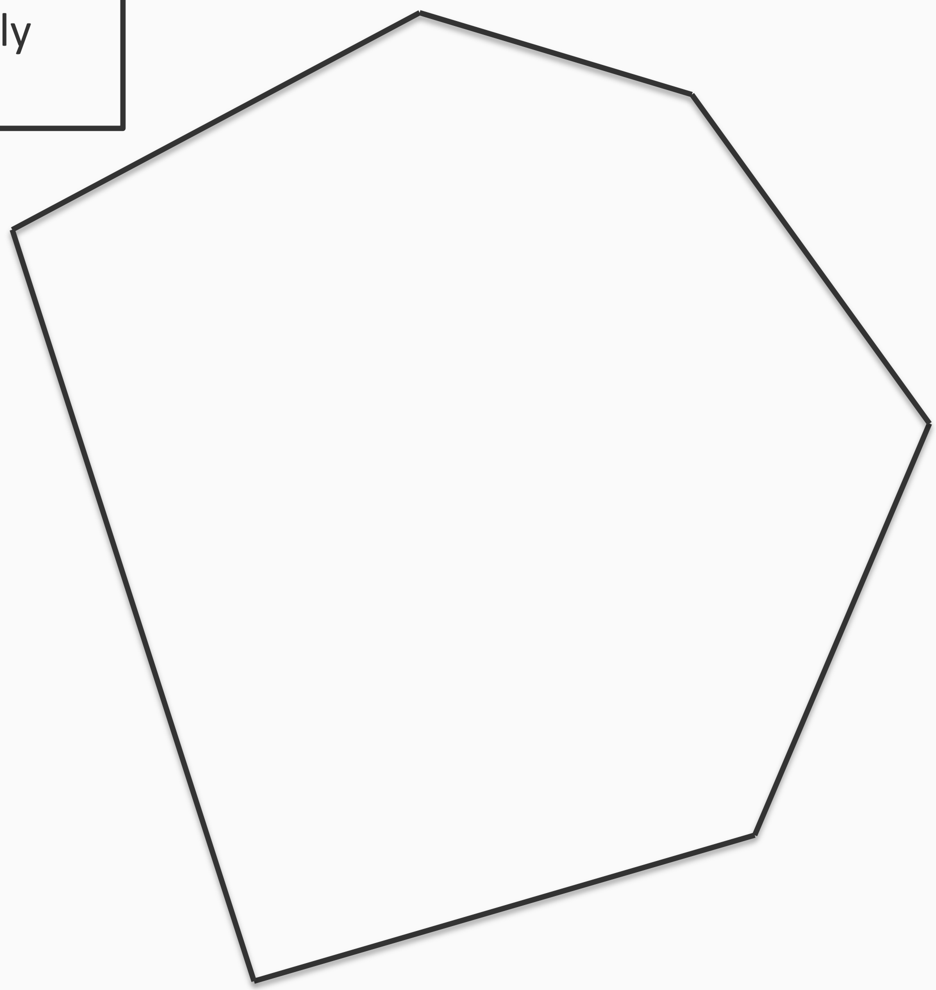


Every constraint forbids a half-space
The points that are allowed form the feasible region

Geometry of linear programming

The constraint matrix defines a polytope that contains allowed solutions (possibly not closed)

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge 0. \end{aligned}$$

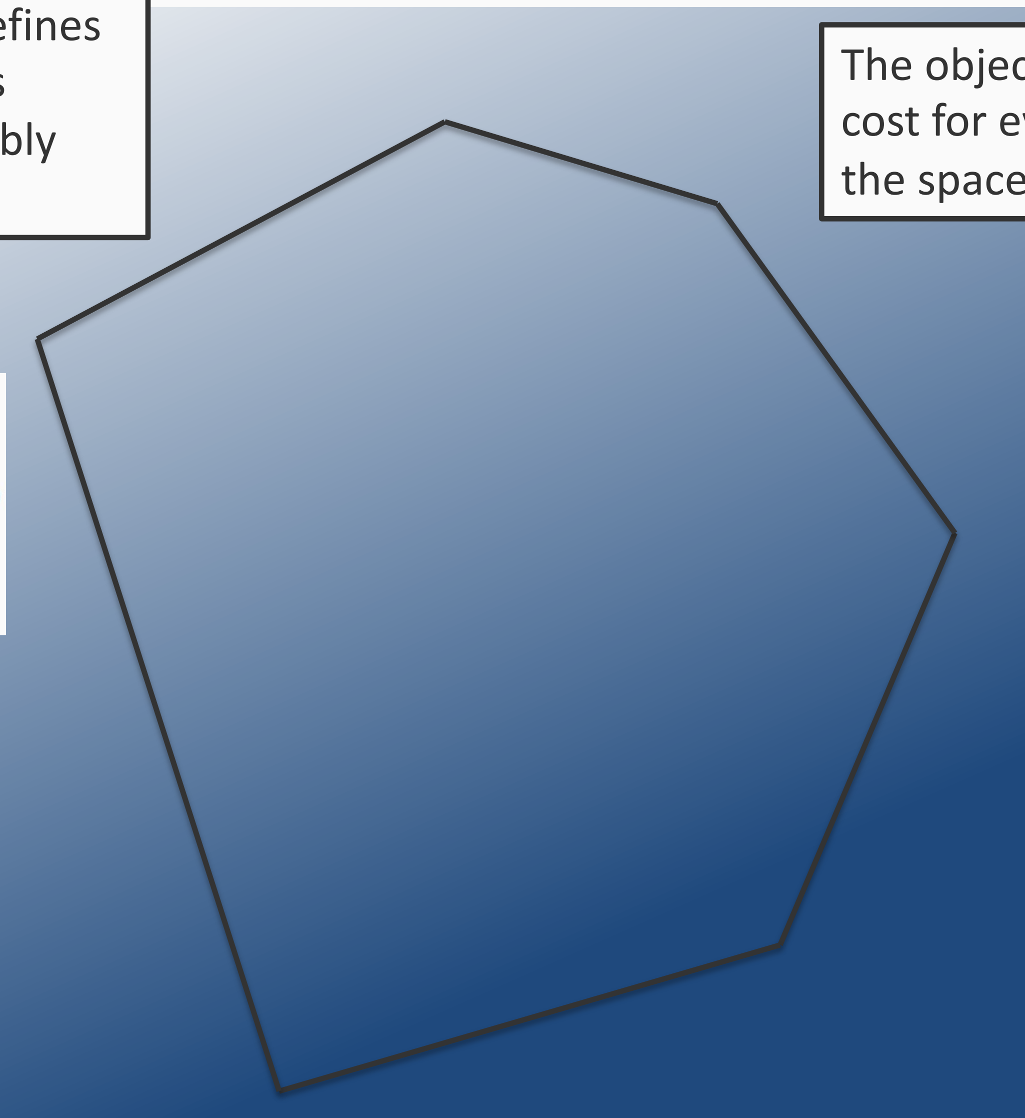


Geometry of linear programming

The constraint matrix defines a polytope that contains allowed solutions (possibly not closed)

The objective defines cost for every point in the space

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

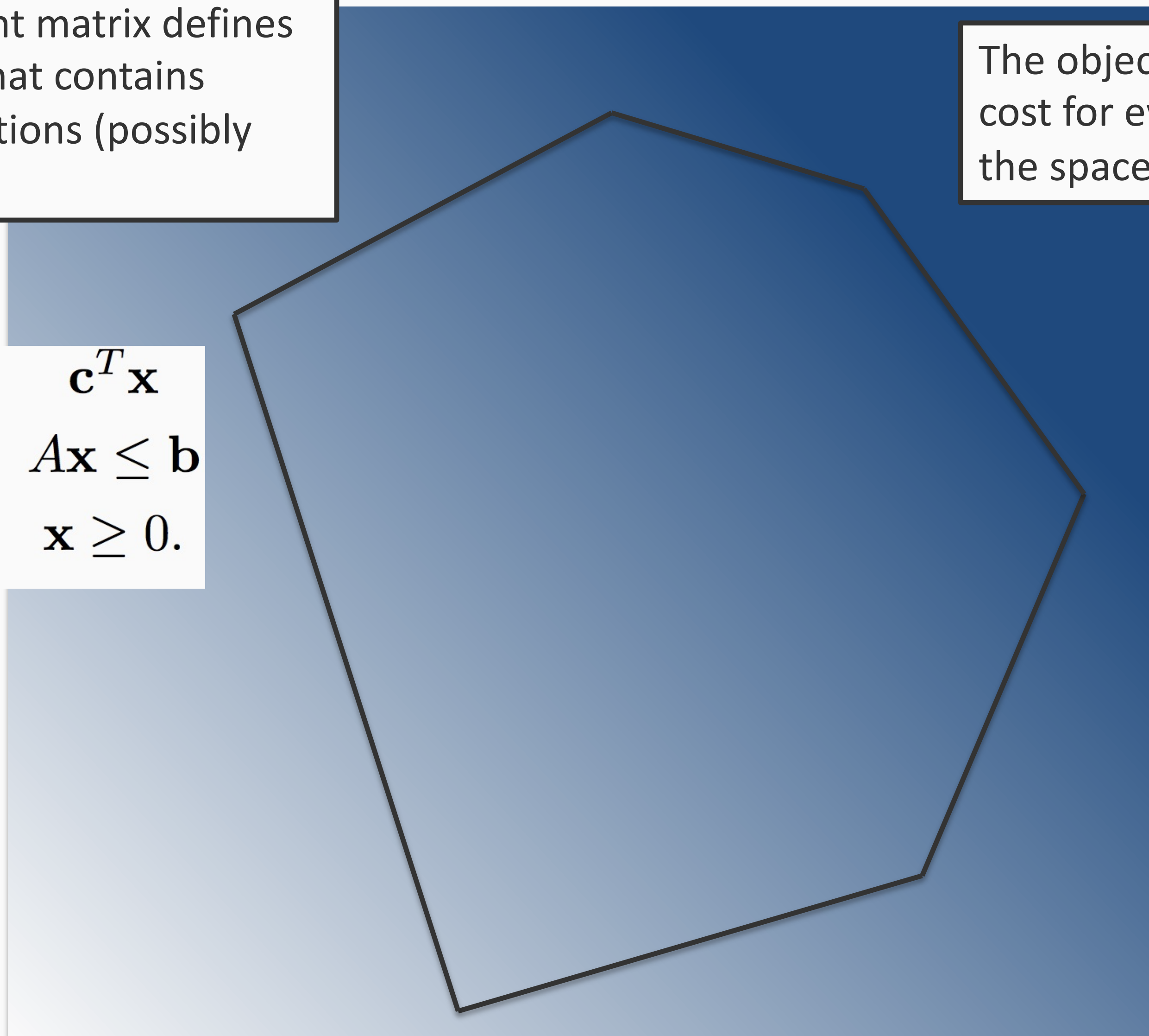


Geometry of linear programming

The constraint matrix defines a polytope that contains allowed solutions (possibly not closed)

The objective defines cost for every point in the space

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

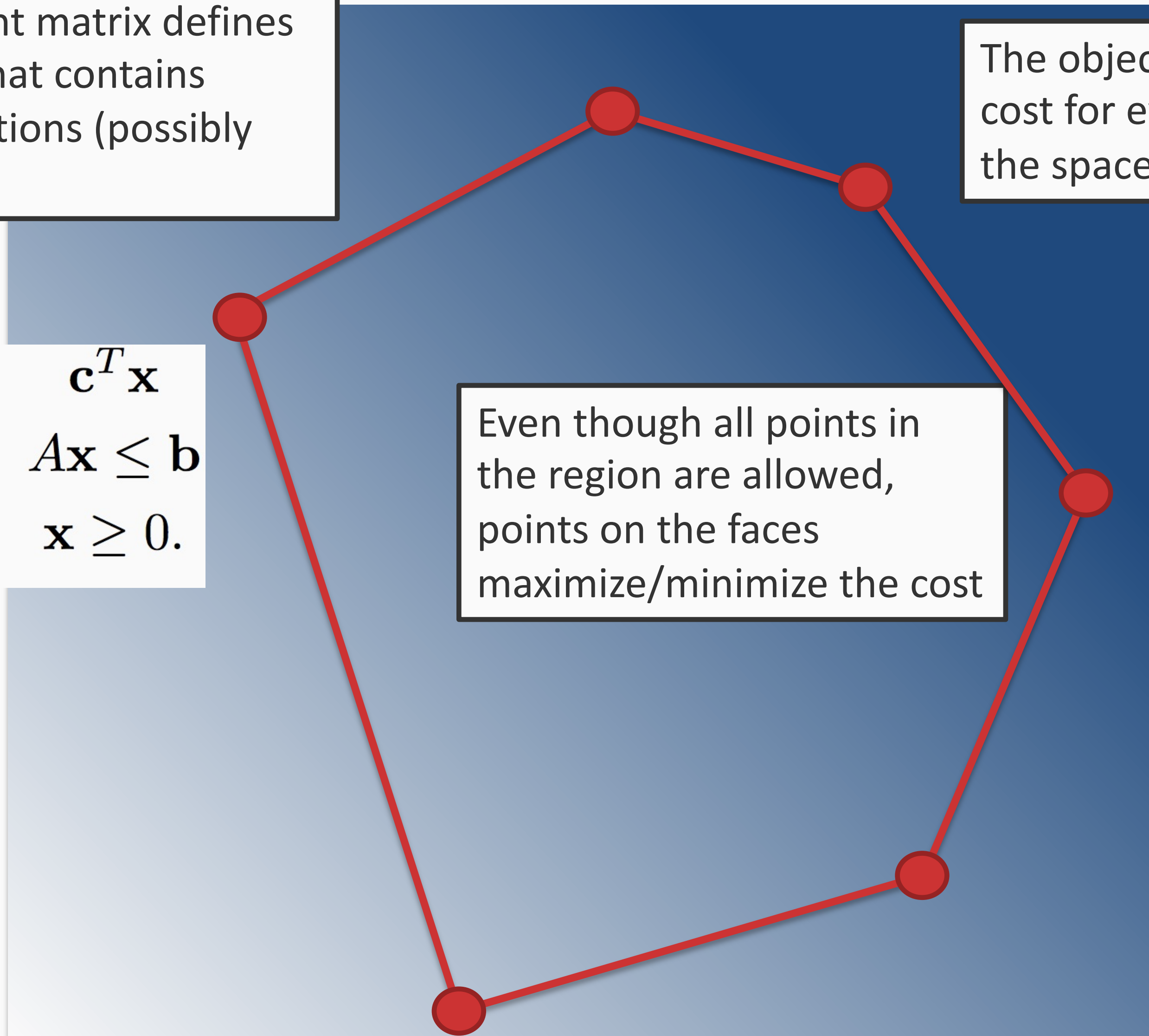


Geometry of linear programming

The constraint matrix defines a polytope that contains allowed solutions (possibly not closed)

The objective defines cost for every point in the space

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$



Even though all points in the region are allowed, points on the faces maximize/minimize the cost

Linear programming

- In general

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

- This is a continuous optimization problem
  - And yet, there are only a finite set of possible solutions
  - The constraint matrix defines a *convex* polytope
  - Only the vertices or faces of the polytope can be solutions

- Linear programs can be solved in polynomial time

Questions?

Integer linear programming

- In general

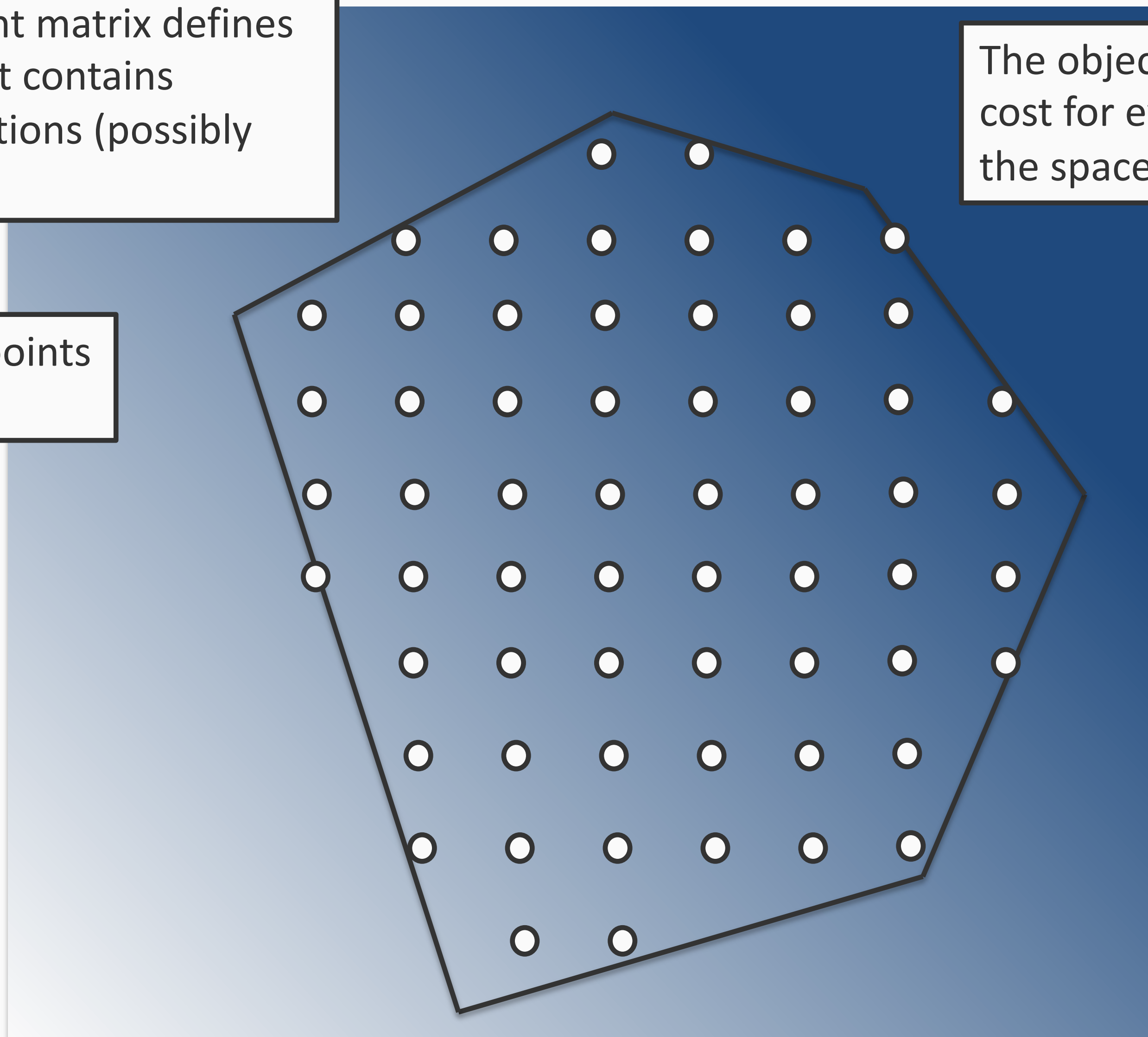
$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \\ & \text{Each } x_i \text{ is an integer.} \end{aligned}$$

Geometry of integer linear programming

The constraint matrix defines polytope that contains allowed solutions (possibly not closed)

The objective defines cost for every point in the space

Only integer points allowed



Integer linear programming

- In general

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \\ & \boxed{\text{Each } x_i \text{ is an integer.}} \end{aligned}$$

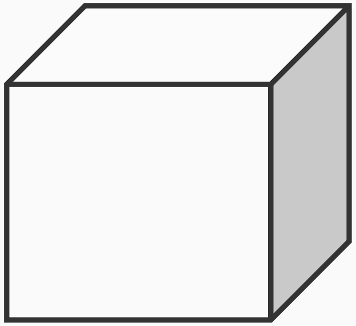
- Solving integer linear programs in general can be NP-hard!
- LP-relaxation: Drop the integer constraints and hope for the best

0-1 integer linear programming

- In general

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \\ & \boxed{\mathbf{x} \in \{0,1\}^n} \end{aligned}$$

- An instance of integer linear programs
  - Still NP-hard
- Geometry: We are only considering points that are vertices of the Boolean hypercube

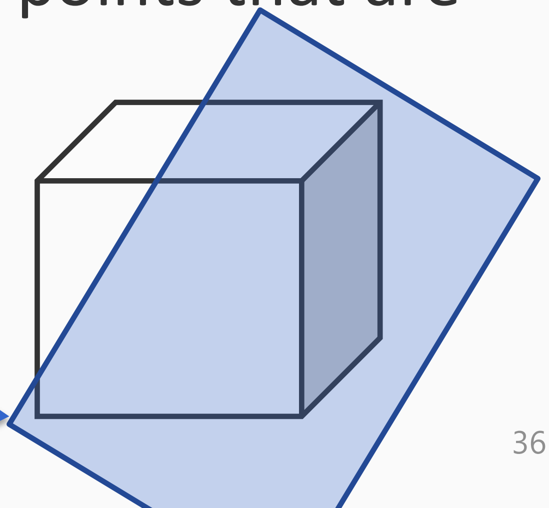


0-1 integer linear programming

- In general

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \\ & \boxed{\mathbf{x} \in \{0,1\}^n} \end{aligned}$$

- An instance of integer linear programs
  – Still NP-hard
- Geometry: We are only considering points that are vertices of the Boolean hypercube
  – Constraints prohibit certain vertices



Eg: Only points within this region are allowed

0-1 integer linear programming

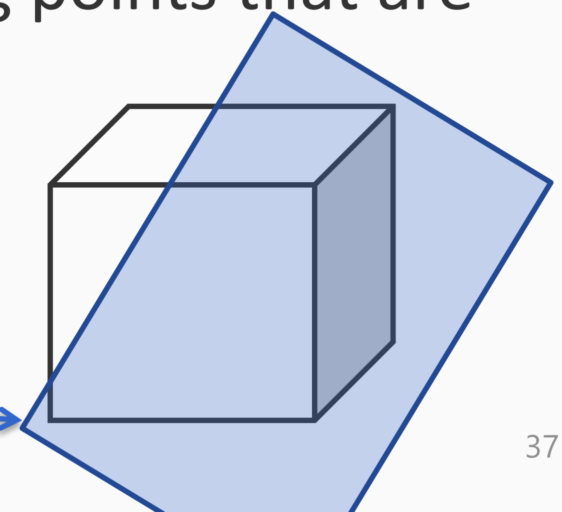
- In general

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge 0 \\ & \boxed{\mathbf{x} \in \{0,1\}^n} \end{aligned}$$

Solution can be an interior point of the constraint set defined by $Ax \le b$

- An instance of integer linear programs
  - Still NP-hard
- Geometry: We are only considering points that are vertices of the Boolean hypercube
  - Constraints prohibit certain vertices

Eg: Only points within this region are allowed



Questions?

Back to structured prediction

- Recall that we are solving $\text{argmax}_{\mathbf{y}}\, w^T \phi(x, \mathbf{y})$
  - The goal is to produce a graph

- The set of possible values that **y** can take is finite, but large

- General idea: Frame the argmax problem as a 0-1 integer linear program
  - Allows addition of arbitrary constraints

Thinking in ILPs

Let's start with multi-class classification

$$\underset{y \in \{A,B,C\}}{\operatorname{argmax}} \, w^T \phi(x, y) = \underset{y \in \{A,B,C\}}{\operatorname{argmax}} \, \text{score}(y)$$

Introduce decision variables for each label

- z_A = 1 if output = A, 0 otherwise
- z_B = 1 if output = B, 0 otherwise
- z_C = 1 if output = C, 0 otherwise

Thinking in ILPs

Let's start with multi-class classification

$$\underset{y\in\{A,B,C\}}{\text{argmax}}\, w^T\phi(x,y) = \underset{y\in\{A,B,C\}}{\text{argmax}}\, \text{score}(y)$$

Introduce decision variables for each label

- z_A = 1 if output = A, 0 otherwise
- z_B = 1 if output = B, 0 otherwise
- z_C = 1 if output = C, 0 otherwise

$$\max_{\mathbf{z}} \quad z_A\text{score}(A) + z_B\text{score}(B) + z_C\text{score}(C)$$

Maximize the score

s.t.

Pick exactly one label

$$z_A, z_B, z_C \in \{0, 1\}.$$

Thinking in ILPs

Let's start with multi-class classification

$$\underset{y\in\{A,B,C\}}{\operatorname{argmax}}\, w^T\phi(x,y) = \underset{y\in\{A,B,C\}}{\operatorname{argmax}}\, \text{score}(y)$$

Introduce decision variables for each label

- z_A = 1 if output = A, 0 otherwise
- z_B = 1 if output = B, 0 otherwise
- z_C = 1 if output = C, 0 otherwise

$$\begin{aligned} \max_{\mathbf{z}} \quad & z_A \text{score}(A) + z_B \text{score}(B) + z_C \text{score}(C) \\ \text{s.t.} \quad & z_A + z_B + z_C = 1 \\ & z_A, z_B, z_C \in \{0,1\}. \end{aligned}$$

Maximize the score

Pick exactly one label

An assignment to the **z** vector gives us a **y**

Thinking in ILPs

Let's start with multi-class classification

$$\underset{y\in\{A,B,C\}}{\text{argmax}}\, w^T\phi(x,y) = \underset{y\in\{A,B,C\}}{\text{argmax}}\, \text{score}(y)$$

Intr

- z
- z
- z

We have taken a trivial problem (finding a highest scoring element of a list) and converted it into a representation that is NP-hard in the worst case!

Lesson: Don't solve multiclass classification with an ILP solver

$$\begin{aligned} \max_{\mathbf{z}} \quad & z_A \text{score}(A) + z_B \text{score}(B) + z_C \text{score}(C) \\ \text{s.t.} \quad & z_A + z_B + z_C = 1 \\ & z_A, z_B, z_C \in \{0,1\}. \end{aligned}$$

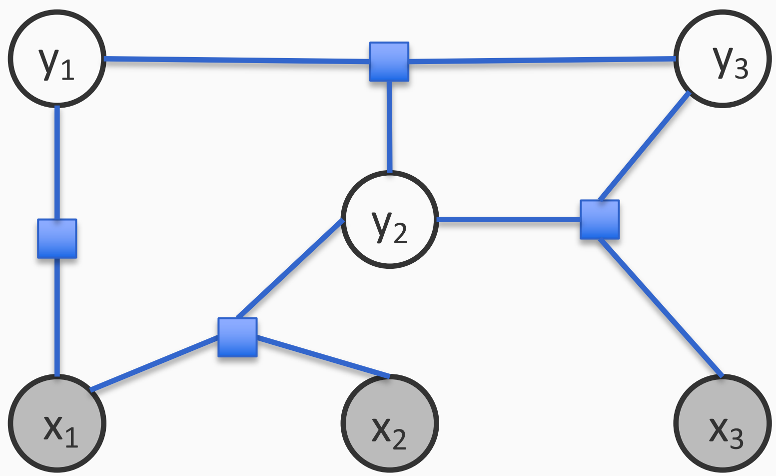
Maximize the score

Pick exactly one label

An assignment to the **z** vector gives us a **y**

Questions?

ILP for a general conditional models



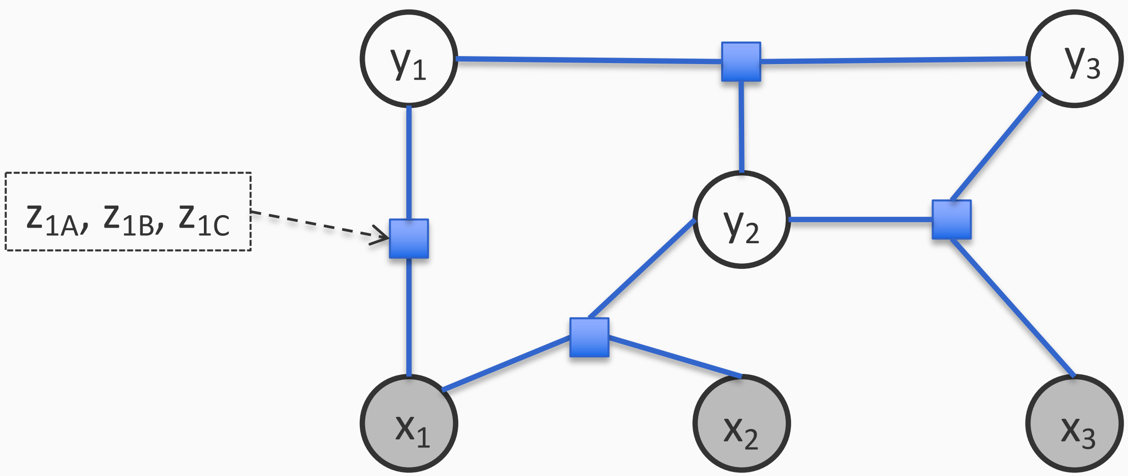
Suppose each y_i can be A, B or C

Introduce one decision variable for each part being assigned labels

Our goal

$$\max_{y_1, y_2, y_3} \mathbf{w}^T \phi(x_1, y_1) + w^T \phi(y_1, y_2, y_3) + w^T \phi(x_3, y_1, y_3) + w^T \phi(x_1, x_2, y_2)$$

ILP for a general conditional models



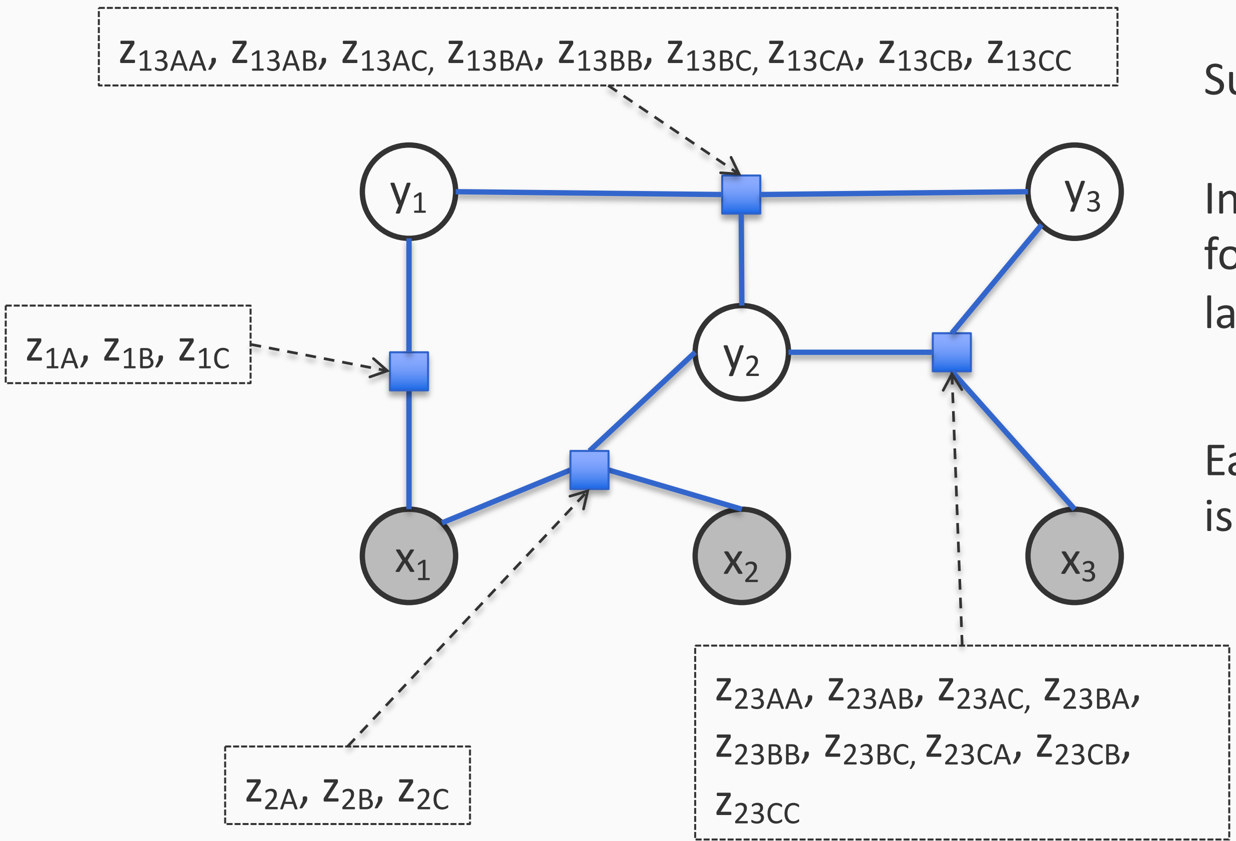
Suppose each y_i can be A, B or C

Introduce one decision variable for each part being assigned labels

Our goal

$$\max_{y_1,y_2,y_3} \mathbf{w}^T\phi(x_1,y_1) + w^T\phi(y_1,y_2,y_3) + w^T\phi(x_3,y_1,y_3) + w^T\phi(x_1,x_2,y_2)$$

ILP for a general conditional models



Suppose each y_i can be A, B or C

Introduce one decision variable for each part being assigned labels

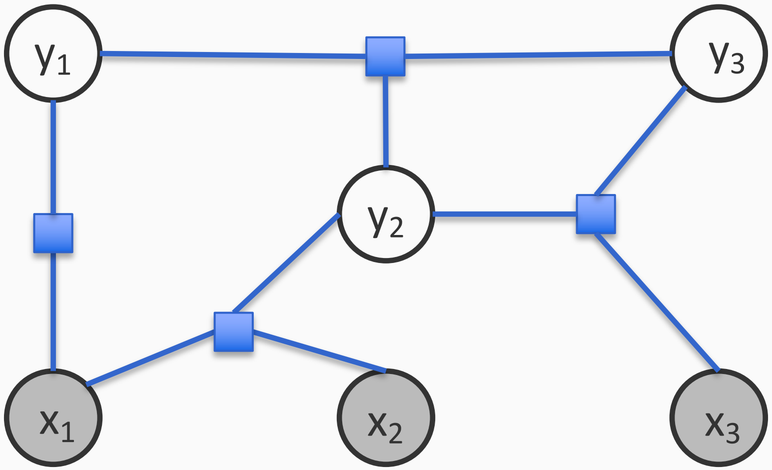
Each of these decision variables is associated with a score

Our goal

$$\max_{y_1,y_2,y_3} \mathbf{w}^T \phi(x_1, y_1) + w^T \phi(y_1, y_2, y_3) + w^T \phi(x_3, y_1, y_3) + w^T \phi(x_1, x_2, y_2)$$

Questions?

ILP for a general conditional models



Suppose each y_i can be A, B or C

Introduce one decision variable for each part being assigned labels

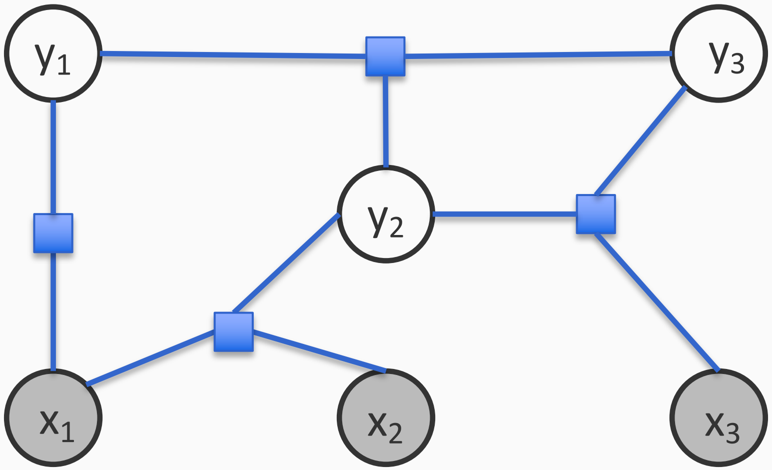
Each of these decision variables is associated with a score

$$\sum_l z_{1l} s_{1l} + \sum_l z_{2l} s_{2l} + \sum_{l,l'} z_{13ll'} s_{13ll'} + \sum_{l,l'} z_{23ll'} s_{23ll'}$$

Our goal

$$\max_{y_1, y_2, y_3} \mathbf{w}^T \phi(x_1, y_1) + w^T \phi(y_1, y_2, y_3) + w^T \phi(x_3, y_1, y_3) + w^T \phi(x_1, x_2, y_2)$$

ILP for a general conditional models



Suppose each y_i can be A, B or C

Introduce one decision variable for each part being assigned labels

Each of these decision variables is associated with a score

Not all decisions can exist together.
Eg: z_{13AB} implies z_{1A} and z_{3B}

$$\max_{\mathbf{z}} \quad \sum_l z_{1l} s_{1l} + \sum_l z_{2l} s_{2l} + \sum_{l,l'} z_{13ll'} s_{13ll'} + \sum_{l,l'} z_{23ll'} s_{23ll'}$$
$$\text{s.t} \quad \text{Only valid output allowed}$$

Our goal

$$\max_{y_1, y_2, y_3} \mathbf{w}^T \phi(x_1, y_1) + w^T \phi(y_1, y_2, y_3) + w^T \phi(x_3, y_1, y_3) + w^T \phi(x_1, x_2, y_2)$$

Writing constraints as linear inequalities

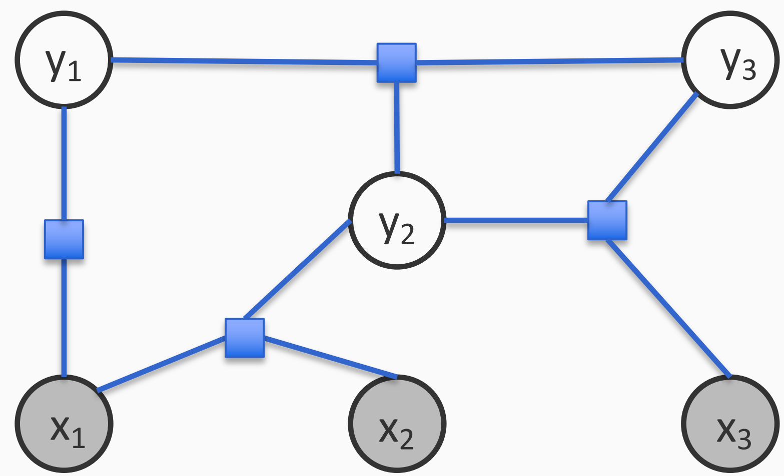
- Exactly one of z_A, z_B, z_C can be true
  $z_A + z_B + z_C = 1$
- At least m of z_A, z_B, z_C should be true
  $z_A + z_B + z_C$ ¸ m
- At most m of z_A, z_B, z_C should be true
  $z_A + z_B + z_C$ · m
- Implication: z_i) z_j
  - Convert to disjunction: : z_i Ç z_j
    - At least one of “not z_i” or z_j
  - $1 – z_i + z_j$ ¸ 1
    (i.e.) z_j ¸ z_i

Writing constraints as linear inequalities

- Exactly one of z_A, z_B, z_C can be true
  $z_A + z_B + z_C = 1$
- At least m of z_A, z_B, z_C should be true
  $z_A + z_B + z_C$ ¸ m
- At most m of z_A, z_B, z_C should be true
  $z_A + z_B + z_C$ · m
- Implication: z_i) z_j
  - Convert to disjunction: : z_i Ç z_j
    - At least one of “not z_i” or z_j
  - $1 - z_i + z_j$ ¸ 1
    (i.e.) z_j ¸ z_i

Generally: All Boolean formulas can be converted to constraints

Exercise: Convert the toy conditional model below to an ILP by hand

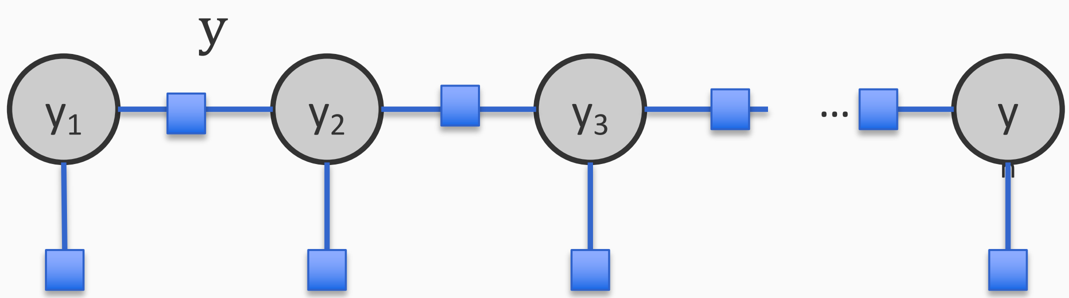


Integer linear programming for inference

- Easy to add additional knowledge
  - Specify them as Boolean formulas
  - Examples
    - “If y_1 is an A, then y_2 or y_3 should be a B or C”
    - “No more than two A’s allowed in the output”
- Many inference problems have “standard” mappings to ILPs
  - Sequences, parsing, dependency parsing
- Encoding of the problem makes a difference in solving time
  - The mechanical encoding may not be efficient to solve
- Generally: more complex constraints make solving harder

Exercise 1: Sequence labeling

Goal: Find argmax $w^T\phi(x, y)$ $\quad$ **y** = (y_1, y_2, $\cdots$, y_n)



$$\mathbf{w}^T\phi(\mathbf{x},\mathbf{y}) = \sum_i \left(\mathbf{w}^T\phi_T(y_i, y_{i+1}) + \mathbf{w}^T\phi_E(\mathbf{x}, y_i)\right)$$

How can this be written as an ILP?

Exercise 2: Alignment

Suppose we have two sequences

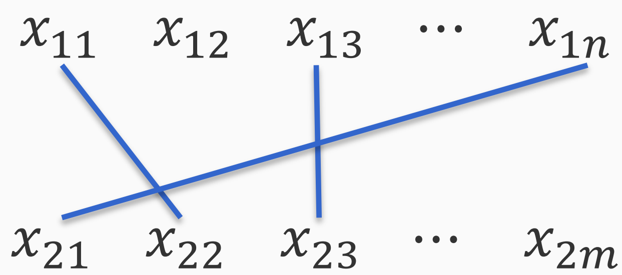
$$x_{11} \quad x_{12} \quad x_{13} \quad \cdots \quad x_{1n}$$

$$x_{21} \quad x_{22} \quad x_{23} \quad \cdots \quad x_{2m}$$

Each pair x_{1i}, x_{2j} is assigned a score s_{ij}.

Exercise 2: Alignment

Suppose we have two sequences



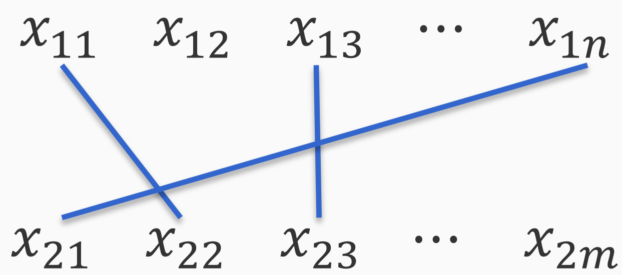
Each pair x_{1i}, x_{2j} is assigned a score s_{ij}.

The goal is to find edges between the two sequences such that the following conditions hold:

1. The total score of the selected edges is maximized
2. No more than one edge should be connected to any element of the second sequence.

Exercise 2: Alignment

Suppose we have two sequences



Each pair x_{1i}, x_{2j} is assigned a score s_{ij}.

The goal is to find edges between the two sequences such that the following conditions hold:

1. The total score of the selected edges is maximized
2. No more than one edge should be connected to any element of the second sequence.

How can this be written as an ILP?

ILP for inference: Remarks

- Any combinatorial optimization problem can be written as an ILP
  - Even the "easy"/polynomial ones
  - Given an ILP, checking whether it represents a polynomial problem is intractable in general
- ILPs are a general language for thinking about combinatorial optimization
  - The representation allows us to make general statements about inference
  - Important: Framing/writing down the inference problem is separate from solving it
- Off-the-shelf solvers for ILPs are quite good
  - Gurobi, CPLEX
  - Use an off the shelf solver only if you can't solve your inference problem otherwise

The big picture

- MAP Inference is combinatorial optimization

- Combinatorial optimization problems can be written as 0-1 integer linear programs
  - The conversion is not always trivial
  - Allows injection of “knowledge” into the inference in the form of constraints

- Different ways of solving ILPs
  - Commercial solvers: CPLEX, Gurobi, etc
  - Specialized solvers if you know something about your problem
    - Incremental ILP, Lagrangian relaxation, etc
  - Can relax to linear programs and hope for the best

- Integer linear programs are NP hard in general
  - No free lunch

Questions?