KU LEUVEN

Submodularity

With applications to Machine Learning

Jorik Jooken Michiel Baptist
KU Leuven
March 2019

0 Outline

1 Outline

1 Machine Learning

Many problems in machine learning boil down to:

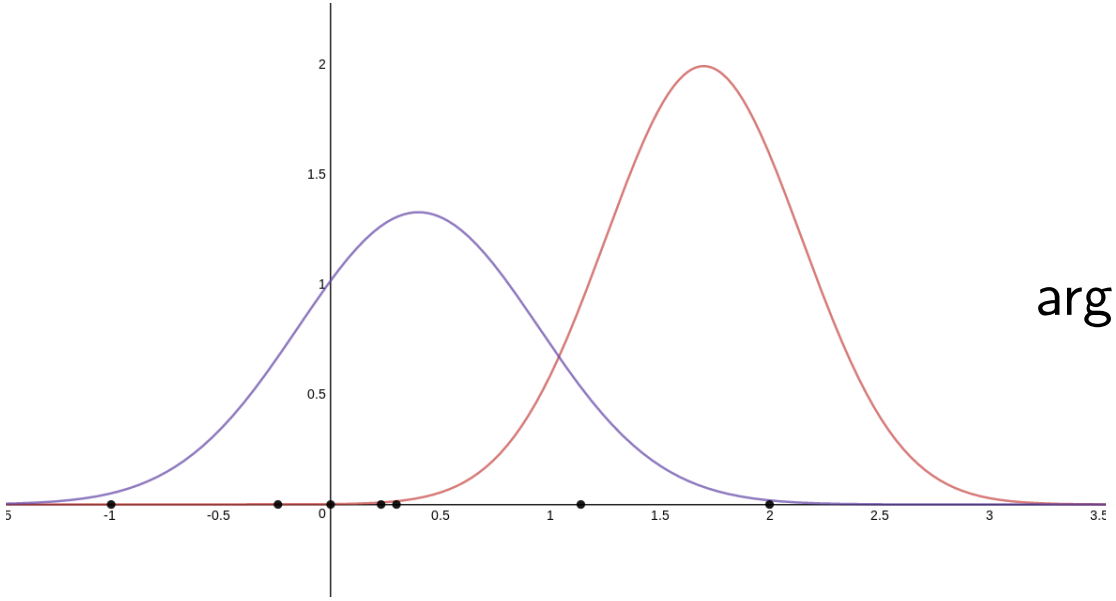
$$\arg\max_{\theta} F(\theta) \tag{1}$$

1 Machine Learning

Many problems in machine learning boil down to:

$$\arg\max_{\theta} F(\theta) \tag{1}$$

Example: maximum likelihood



$$\arg\max_{\theta} \sum_{i=1}^{n} \log(p(x_i|\theta))$$

1 Machine Learning

Many problems in machine learning boil down to:

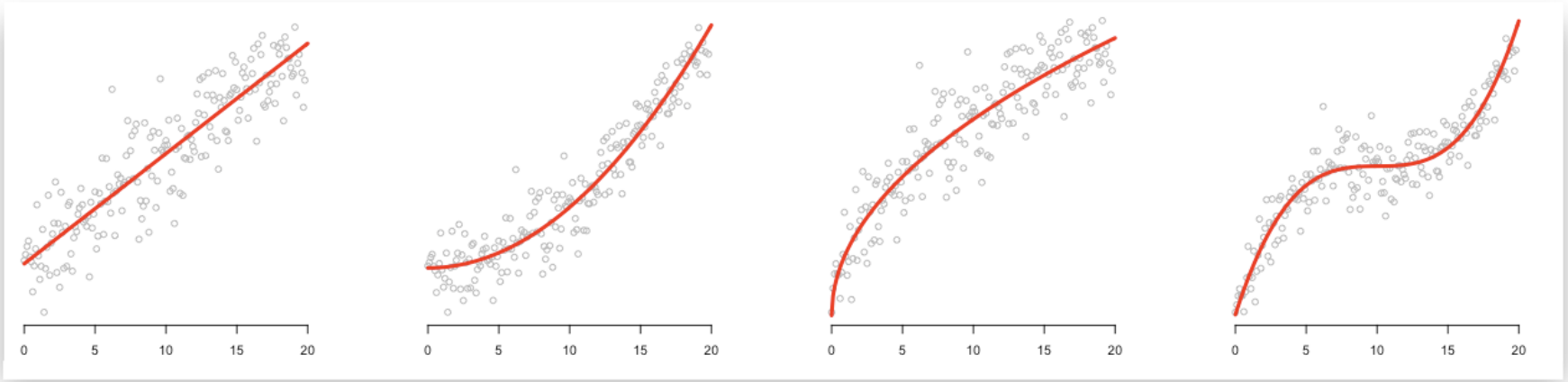
$$\arg\min_{\theta} L(\theta) \tag{2}$$

1 Machine Learning

Many problems in machine learning boil down to:

$$\arg\min_{\theta} L(\theta) \tag{2}$$

Example: least squares regression



$$\arg\min_{\theta} \sum_{i=1}^{n} \|y_i - f_\theta(x_i)\|^2 \tag{3}$$

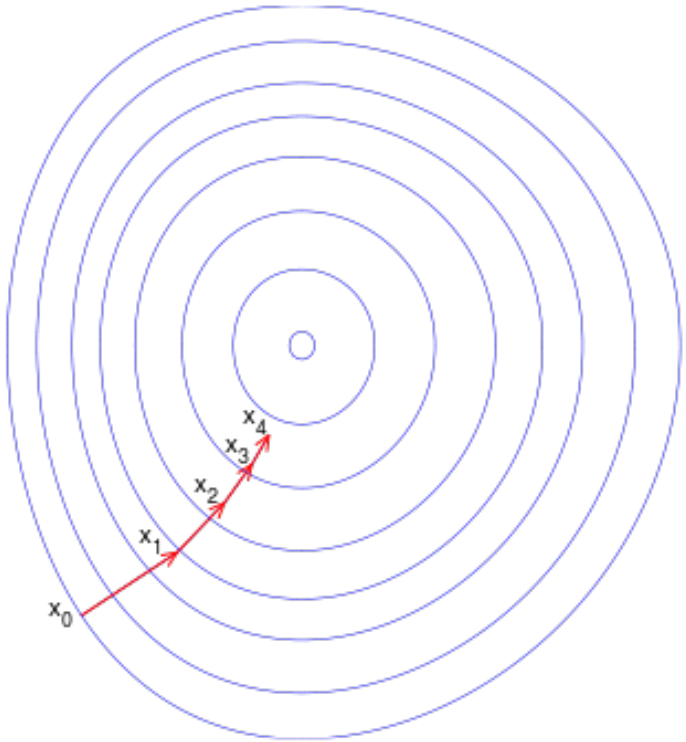
1 Role of convexity

How can we search in an infinite search space?

1 Role of convexity

How can we search in an infinite search space?

Example: "Hello world of optimisation"



[4]

$$x_{i+1} = x_i - \gamma \nabla F(x_i)$$

1 Role of convexity

Sometimes, this $L(\theta)$ is convex!

1 Role of convexity

Sometimes, this $L(\theta)$ is convex!

Why is this important?

1 Role of convexity

Sometimes, this $L(\theta)$ is convex!

Why is this important?

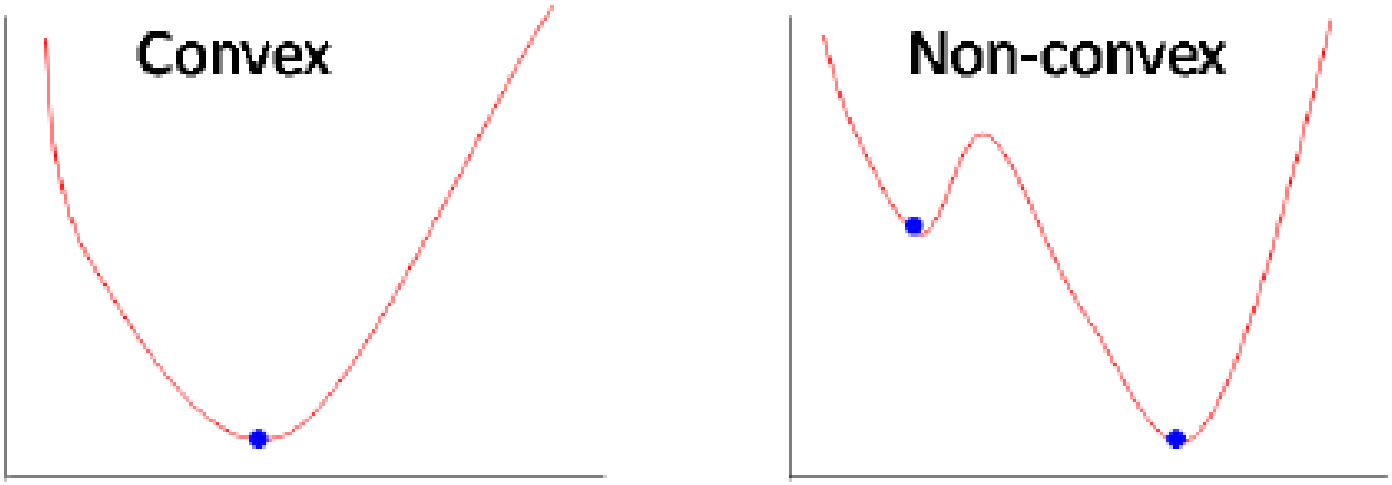


Figure: [12]

1 Role of submodularity

Not all problems are continuous!

1 Role of submodularity

Not all problems are continuous!

Discrete optimisation:

1 Role of submodularity

Not all problems are continuous!

Discrete optimisation:

- search space is often finite!

1 Role of submodularity

Not all problems are continuous!

Discrete optimisation:

- search space is often finite!
- search space is often exponential!

1 Role of submodularity

Not all problems are continuous!

Discrete optimisation:

- search space is often finite!
- search space is often exponential!

Question: can we leverage a "discrete convexity" structure?

1 Role of submodularity

Not all problems are continuous!

Discrete optimisation:

- search space is often finite!
- search space is often exponential!

Question: can we leverage a "discrete convexity" structure?

Submodularity

2 Outline

2 Definition: Set functions

Given a (finite) ground set V:

$$F : 2^V \to \mathbb{R} \tag{4}$$

2 Definition: Set functions

Given a (finite) ground set V:

$$F : 2^V \to \mathbb{R} \tag{4}$$

Example: sleep or caffeine?

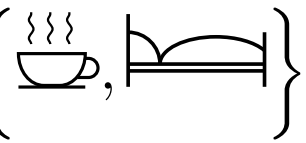
$$V = \left\{ \text{☕}, \text{🛏} \right\}$$

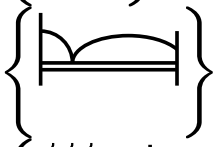
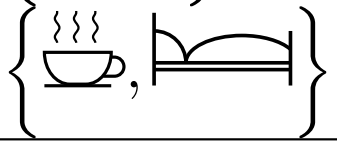
2 Definition: Set functions

Given a (finite) ground set V:

$$F : 2^V \to \mathbb{R} \tag{4}$$

Example: sleep or caffeine?

$V =$ 

$S \in 2^V$	$F(S)$
$\emptyset$	0
	3
	2
	-3

2 Definition: Submodularity

Given a $F : 2^V \rightarrow \mathbb{R}$, F is **submodular** iff:

2 Definition: Submodularity

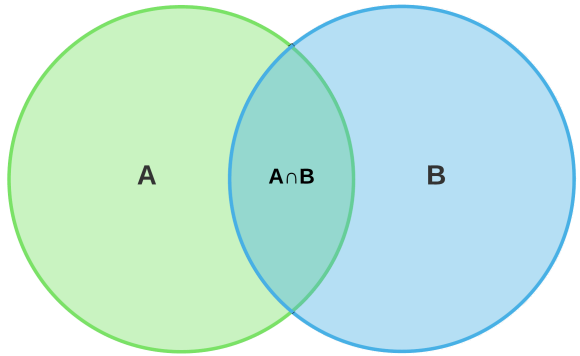
Given a $F : 2^V \rightarrow \mathbb{R}$, F is **submodular** iff:

$$\forall A, B \in 2^V : F(A) + F(B) \geq F(A \cup B) + F(A \cap B) \tag{5}$$

2 Definition: Submodularity

Given a $F : 2^V \rightarrow \mathbb{R}$, F is **submodular** iff:

$$\forall A, B \in 2^V : F(A) + F(B) \geq F(A \cup B) + F(A \cap B) \tag{5}$$



$$F\left(\right) + F\left(\right) \geq F\left(\right) + F\left(\right)$$

2 Definition: Submodularity

Given a $F : 2^V \rightarrow \mathbb{R}$, F is **submodular** iff:

2 Definition: Submodularity

Given a $F : 2^V \rightarrow \mathbb{R}$, F is **submodular** iff:

- $\forall A, B \in 2^V$ such that $A \subseteq B$
- $\forall s \in 2^V \setminus B$

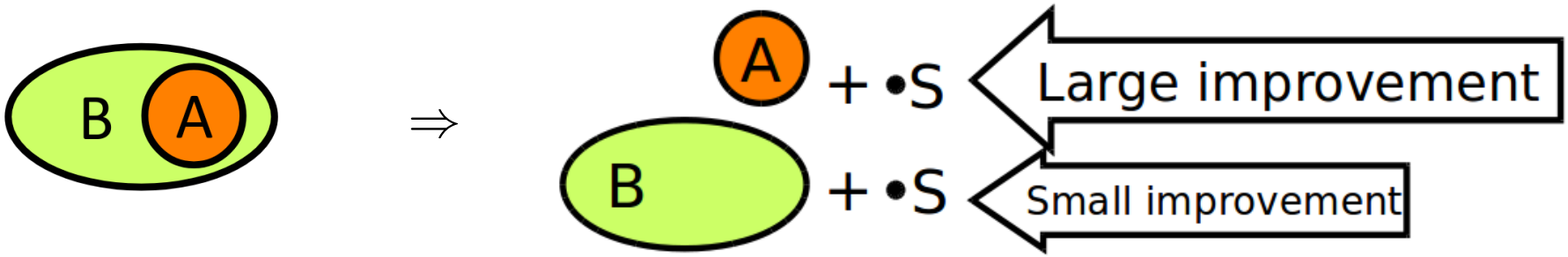
$$F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B) \tag{6}$$

2 Definition: Submodularity

Given a $F : 2^V \rightarrow \mathbb{R}$, F is **submodular** iff:

- $\forall A, B \in 2^V$ such that $A \subseteq B$
- $\forall s \in 2^V \setminus B$

$$F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B) \tag{6}$$



Referred to as "diminishing returns"

Example from: [14]

2 Submodularity: example

Previous example: sleep or caffeine?

2 Submodularity: example

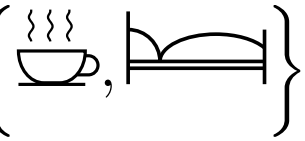
Previous example: sleep or caffeine?

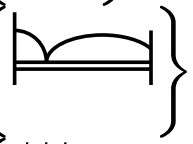
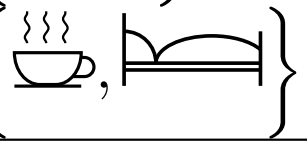
$V = \{$  $\}$

$S \in 2^V$	$F(S)$
$\emptyset$	0
$\{$  $\}$	3
$\{$  $\}$	2
$\{$  $\}$	-3

2 Submodularity: example

Previous example: sleep or caffeine?

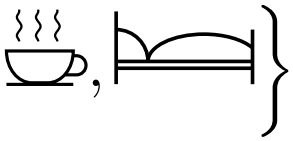
$V = \{$  $\}$

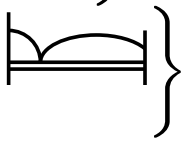
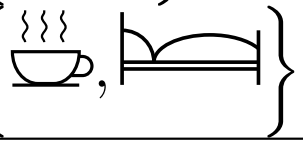
$S \in 2^V$	$F(S)$
$\emptyset$	0
$\{$  $\}$	3
$\{$  $\}$	2
$\{$  $\}$	-3

Take: $A = \emptyset \subset B =$ 

2 Submodularity: example

Previous example: sleep or caffeine?

$V = \{$  $\}$

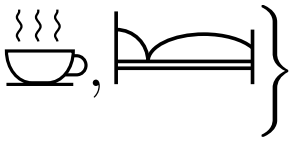
$S \in 2^V$	$F(S)$
$\emptyset$	0
$\{$  $\}$	3
$\{$  $\}$	2
$\{$  $\}$	-3

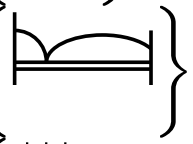
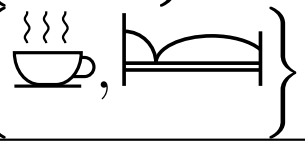
Take: $A = \emptyset \subset B =$ $\{$  $\}$

- Adding  to A: $F\left(A \cup \{$  $\}\right) - F(A) = 2$

2 Submodularity: example

Previous example: sleep or caffeine?

$V = \{$  $\}$

$S \in 2^V$	$F(S)$
$\emptyset$	0
$\{$  $\}$	3
$\{$  $\}$	2
$\{$  $\}$	-3

Take: $A = \emptyset \subset B =$ 

- Adding  to A: $F\left(A \cup \{$  $\}\right) - F(A) = 2$
- Adding  to B: $F\left(B \cup \{$  $\}\right) - F(B) = -6$

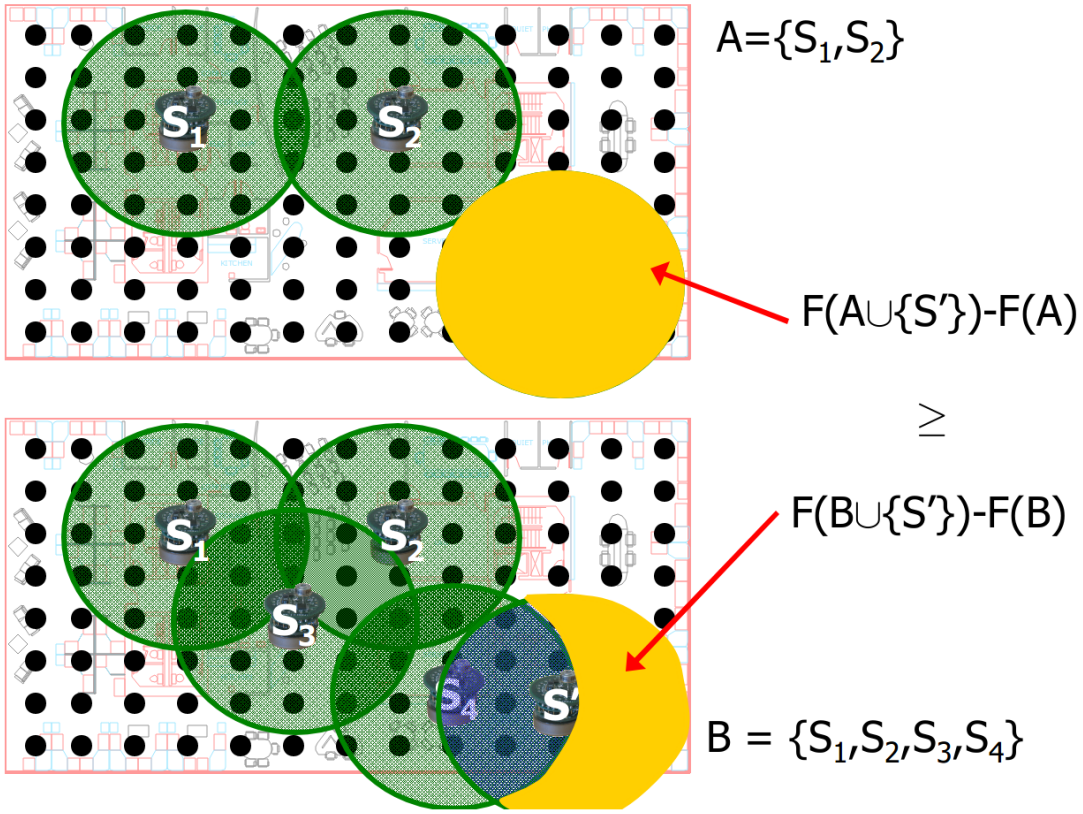
2 Submodularity in ML

Submodularity arises naturally in many problems.

2 Submodularity in ML

Submodularity arises naturally in many problems.

Example: Set coverage



Example from: [14]

2 Submodularity in ML

Submodularity arises naturally in many problems.

2 Submodularity in ML

Submodularity arises naturally in many problems.
Example: Graph cut

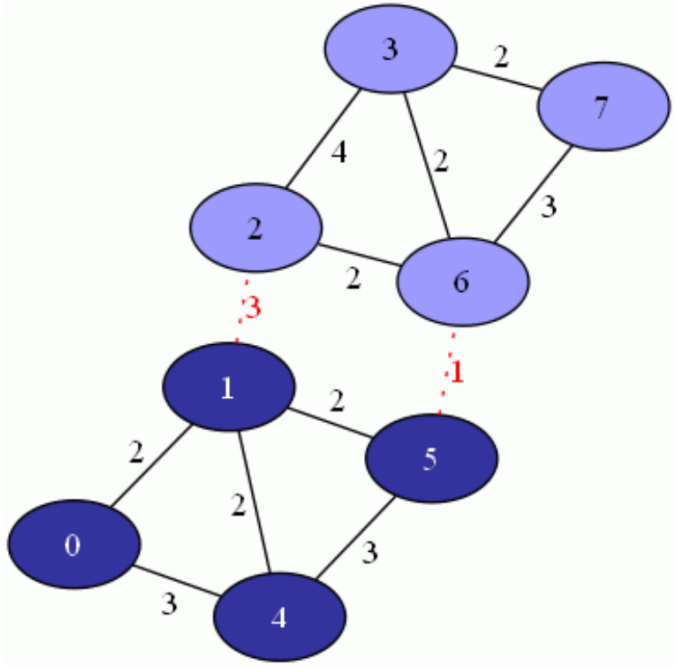


Figure: from [5]

2 Submodularity in ML

Submodularity arises naturally in many problems.

2 Submodularity in ML

Submodularity arises naturally in many problems.
Example: Text summarisation [16]

> I have a cat. I like my cat. My cat is very cool. My cat likes to eat cat snacks. My cat has brown and black fur. I have a dog too. He likes his toys. I like taking him on walks. I also like my dog.

(7)

2 Submodularity in ML

Submodularity arises naturally in many problems.
Example: Text summarisation [16]

I have a cat. I like my cat. My cat is very cool. My cat likes to eat cat snacks. My cat has brown and black fur. I have a dog too. He likes his toys. I like taking him on walks. I also like my dog.

$\rightarrow$

I have a cat. I like my cat. I have a dog too. I also like my dog.

(7)

2 Submodularity in ML

Submodularity arises naturally in many problems.
Example: Text summarisation [16]

I have a cat. I like my cat. My cat is very cool. My cat likes to eat cat snacks. My cat has brown and black fur. I have a dog too. He likes his toys. I like taking him on walks. I also like my dog.

$\rightarrow$

I have a cat. I like my cat. I have a dog too. I also like my dog.

- $V =$ set of all sentences.

(7)

2 Submodularity in ML

Submodularity arises naturally in many problems.
Example: Text summarisation [16]

I have a cat. I like my cat. My cat is very cool. My cat likes to eat cat snacks. My cat has brown and black fur. I have a dog too. He likes his toys. I like taking him on walks. I also like my dog.

$\rightarrow$

I have a cat. I like my cat. I have a dog too. I also like my dog.

- $S =$ summary sentences.

- $V =$ set of all sentences.

(7)

2 Submodularity in ML

Submodularity arises naturally in many problems.
Example: Text summarisation [16]

I have a cat. I like my cat. My cat is very cool. My cat likes to eat cat snacks. My cat has brown and black fur. I have a dog too. He likes his toys. I like taking him on walks. I also like my dog.

$\rightarrow$

I have a cat. I like my cat. I have a dog too. I also like my dog.

- $S =$ summary sentences.
- $V =$ set of all sentences.

$$F(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S) \tag{7}$$

2 Submodularity in ML

Submodularity arises naturally in many problems.

2 Submodularity in ML

Submodularity arises naturally in many problems.

More examples:

- Feature selection
- Budgeting problems
- ...

3 Outline

3 Minimising submodular functions

Algorithms overview:

3 Minimising submodular functions

Algorithms overview:

Grötschel, Lovasz, Schrijver 1981 [11]

Use the ellipsoid algorithm to minimise submodular functions

- Problem: $\mathcal{O}(n^8 \log^2(n))$

3 Minimising submodular functions

Algorithms overview:

Grötschel, Lovasz, Schrijver 1981 [11]

Use the ellipsoid algorithm to minimise submodular functions

- Problem: $\mathcal{O}(n^8 \log^2(n))$

Satoru Fujishige 2005 [9]

Use minimum norm algorithm to minimise submodular functions.

- Unknown runtime
- However: many practical problems solved

3 Maximising submodular functions

Most common maximisation problem:

$$A^* = \arg\max_{|A| \leq k} F(A) \tag{8}$$

Called subset selection

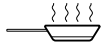
- Extensively studied in literature
- We only consider this problem

3 Maximising submodular functions

Example: I'm very hungry $k = 2$

3 Maximising submodular functions

Example: I'm very hungry $k = 2$

$V = \{$  ,  ,  $\}$

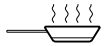
 $=$ Steak

 $=$ Pizza

 $=$ Cake

3 Maximising submodular functions

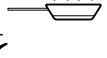
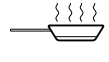
Example: I'm very hungry $k = 2$

$V = \{$  $\}$

 = Steak

 = Pizza

 = Cake

$S \in 2^V$	$F(S)$
$\emptyset$	0
$\{$  $\}$	3
$\{$  $\}$	4
$\{$  $\}$	3
$\{$  $\}$	4
$\{$  $\}$	6
$\{$  $\}$	5
$\{$  $\}$	1

3 Maximising submodular functions

So what has been suggested?

Heesang Lee, George L. Nemhauser, Yinhua Wang 1996 [15]

Suggested a mixed integer programming approach for submodular optimisation.

B. Goldengorin et al. 1999 [10]

Suggested a branch and bound approach for submodular optimisation.

Problem: worst case exponential!

3 Maximising submodular functions

What about an approximate solution?

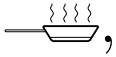
Let's try a **greedy** approach:

Algorithm 1: Greedy submodular optimisation

1 $A_0 = \emptyset$;
2 **for** $i = 1$ *until* k **do**
3 $\quad s_i := \arg\max_{s \in V \setminus A_{i-1}} F(A_{i-1} \cup \{s\}) - F(A_{i-1})$;
4 $\quad A_i = A_{i-1} \cup \{s_i\}$
5 **end**

3 Maximising submodular functions

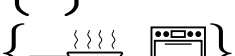
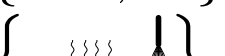
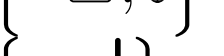
Example: I'm very hungry $k = 2$

$V = \{$  $\}$

 = Steak

 = Pizza

 = Cake

$S \in 2^V$	$F(S)$
$\emptyset$	0
$\{$  $\}$	3
$\{$  $\}$	4
$\{$  $\}$	3
$\{$  $\}$	4
$\{$  $\}$	6
$\{$  $\}$	5
$\{$  $\}$	1

3 Maximising submodular functions

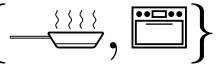
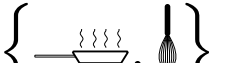
Example: I'm very hungry $k = 2$

$V = \{$  $\}$

 = Steak

 = Pizza

 = Cake

$S \in 2^V$	$F(S)$
$\emptyset$	0
$\{$  $\}$	3
$\{$  $\}$	4
$\{$  $\}$	3
$\{$  $\}$	4
$\{$  $\}$	6
$\{$  $\}$	5
$\{$  $\}$	1

Step 0: $A_0 = \emptyset$

3 Maximising submodular functions

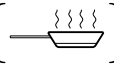
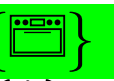
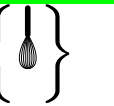
Example: I'm very hungry $k = 2$

$V =$ 

 = Steak

 = Pizza

 = Cake

$S \in 2^V$	$F(S)$
$\emptyset$	0
	3
	4
	3
	4
	6
	5
	1

Step 0: $A_0 = \emptyset$

Step 1: $A_1 = A_0 \cup$ 

3 Maximising submodular functions

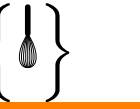
Example: I'm very hungry $k = 2$

$V =$ 

 = Steak

 = Pizza

 = Cake

$S \in 2^V$	$F(S)$
$\emptyset$	0
	3
	4
	3
	4
	6
	5
	1

Step 0: $A_0 = \emptyset$

Step 1: $A_1 = A_0 \cup$ 

3 Maximising submodular functions

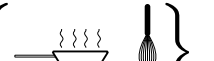
Example: I'm very hungry $k = 2$

$V = \{$  $\}$

 = Steak

 = Pizza

 = Cake

$S \in 2^V$	$F(S)$
$\emptyset$	0
$\{$  $\}$	3
$\{$  $\}$	4
$\{$  $\}$	3
$\{$  $\}$	4
$\{$  $\}$	6
$\{$  $\}$	5
$\{$  $\}$	1

Step 0: $A_0 = \emptyset$

Step 1: $A_1 = A_0 \cup \{$  $\}$

Step 2: $A_2 = A_1 \cup \{$  $\}$

3 Maximising submodular functions

Greedy maximisation results in suboptimal results.

3 Maximising submodular functions

Greedy maximisation results in suboptimal results.

- Is it useless?

3 Maximising submodular functions

Greedy maximisation results in suboptimal results.

- Is it useless?

Nemhauser et al 1978 [18]

For a monotic submodular function F it holds that

$$F(A_{greedy}) \geq \underbrace{(1 - \frac{1}{e})}_{\approx 63\%} F(A^*) \tag{9}$$

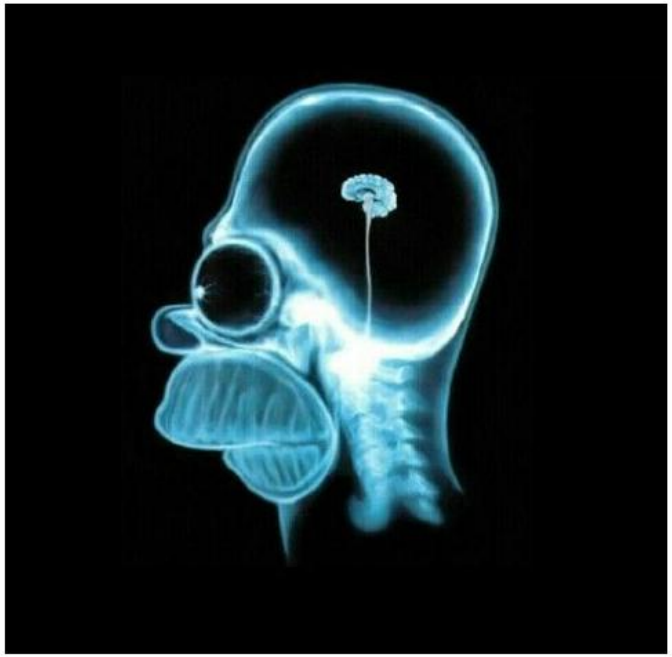
4 Outline

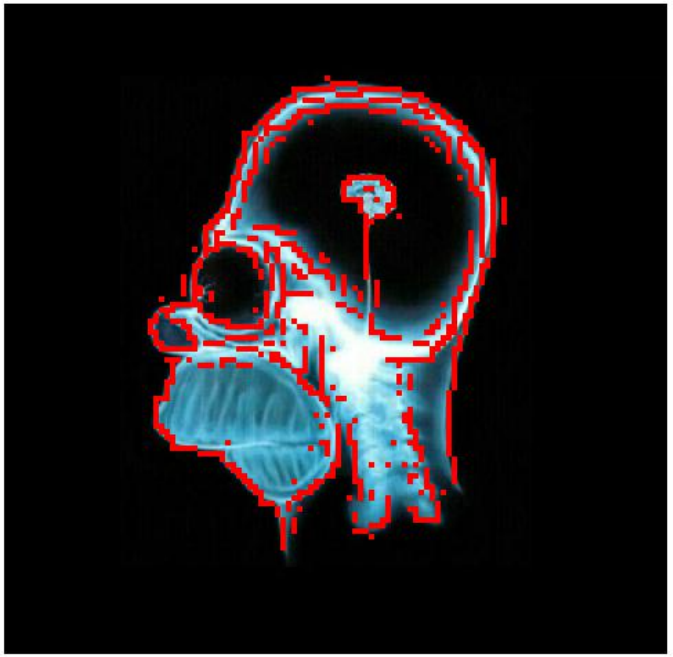
4 Finding structure in images

Goal: Finding structure in images

4 Finding structure in images

Goal: Finding structure in images





4 Finding structure in images

How does it work?

1. Compress the image
2. Image to graph
3. Find max cut

4 Finding structure in images

How does it work?

1. Compress the image
2. Image to graph
3. Find max cut

4 Finding structure in images

Step 1: **compression**

$(0, 6, 3)$	$(9, 9, 9)$	$(0, 0, 0)$	$(0, 0, 0)$
$(5, 1, 0)$	$(2, 0, 4)$	$(0, 0, 0)$	$(0, 0, 0)$
$(0, 0, 0)$	$(3, 2, 0)$	$(2, 0, 1)$	$(5, 0, 0)$
$(0, 0, 0)$	$(1, 2, 8)$	$(0, 4, 3)$	$(1, 0, 4)$

4 Finding structure in images

Step 1: **compression**

$(0, 6, 3)$	$(9, 9, 9)$	$(0, 0, 0)$	$(0, 0, 0)$
$(5, 1, 0)$	$(2, 0, 4)$	$(0, 0, 0)$	$(0, 0, 0)$
$(0, 0, 0)$	$(3, 2, 0)$	$(2, 0, 1)$	$(5, 0, 0)$
$(0, 0, 0)$	$(1, 2, 8)$	$(0, 4, 3)$	$(1, 0, 4)$

4 Finding structure in images

Step 1: **compression**

$(0,6,3)$	$(9,9,9)$	$(0,0,0)$	$(0,0,0)$
$(5,1,0)$	$(2,0,4)$	$(0,0,0)$	$(0,0,0)$
$(0,0,0)$	$(3,2,0)$	$(2,0,1)$	$(5,0,0)$
$(0,0,0)$	$(1,2,8)$	$(0,4,3)$	$(1,0,4)$

$\xrightarrow{\frac{1}{4}\sum}$

$(4,4,4)$	$(0,0,0)$
$(1,1,2)$	$(2,1,1)$

4 Finding structure in images

Example: $k = 64$

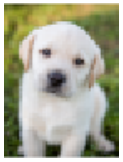
4 Finding structure in images

Example: $k = 64$

4 Finding structure in images

Example: $k = 64$

$\xrightarrow{\frac{1}{64^2}\sum}$



4 Finding structure in images

How does it work?

1. Compress the image
2. Image to graph
3. Find max cut

4 Finding structure in images

How does it work?

1. Compress the image
2. Image to graph
3. Find max cut

4 Finding structure in images

Step 2: **image to graph**

We want to leverage tools from graph theory:

4 Finding structure in images

Step 2: **image to graph**

We want to leverage tools from graph theory:

1. Construct graph from pixels

4 Finding structure in images

Step 2: **image to graph**

We want to leverage tools from graph theory:

1. Construct graph from pixels
2. Define weights for each edge

4 Finding structure in images

Step 2a: from **pixels** to **graph**

Connect every pixel with all its surrounding neighbours.

4 Finding structure in images

Step 2a: from **pixels** to **graph**

Connect every pixel with all its surrounding neighbours.

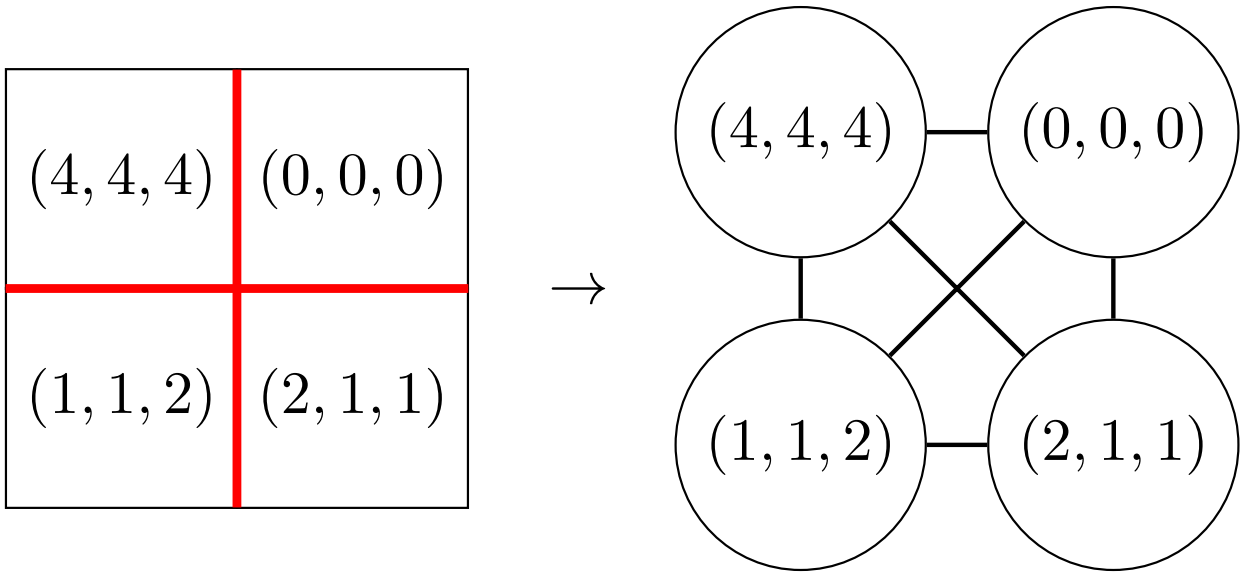
- We choose $N(p_{ij}) = \{p_{kl} || i-k| = 1 \vee |j-l| = 1\}$

4 Finding structure in images

Step 2a: from **pixels** to **graph**

Connect every pixel with all its surrounding neighbours.

- We choose $N(p_{ij}) = \{p_{kl} || i-k| = 1 \vee |j-l| = 1\}$



4 Finding structure in images

Step 2b: choosing the **weights**

4 Finding structure in images

Step 2b: choosing the **weights**

- This will decide how pixels are related

4 Finding structure in images

Step 2b: choosing the **weights**

- This will decide how pixels are related
- In our application: Euclidean distance of colours

4 Finding structure in images

Step 2b: choosing the **weights**

- This will decide how pixels are related
- In our application: Euclidean distance of colours
- Threshold low values

4 Finding structure in images

Step 2b: choosing the **weights**

- This will decide how pixels are related
- In our application: Euclidean distance of colours
- Threshold low values

$$w(e) = \begin{cases} 0, & \text{if } \|p_1 - p_2\| < \alpha \\ \|p_1 - p_2\|, & \text{otherwise} \end{cases} \tag{10}$$

4 Finding structure in images

Step 2b: choosing the **weights**

- This will decide how pixels are related
- In our application: Euclidean distance of colours
- Threshold low values

$$w(e) = \begin{cases} 0, & \text{if } \|p_1 - p_2\| < \alpha \\ \|p_1 - p_2\|, & \text{otherwise} \end{cases} \tag{10}$$

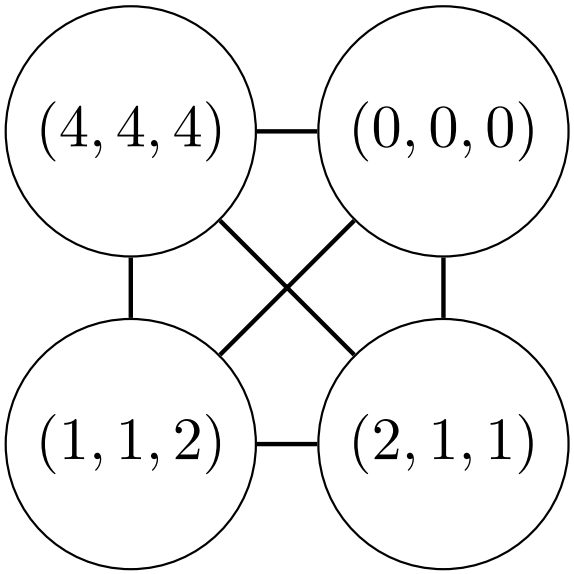
- α can be adjusted according to image

4 Finding structure in images

Step 2b: choosing the **weights**

4 Finding structure in images

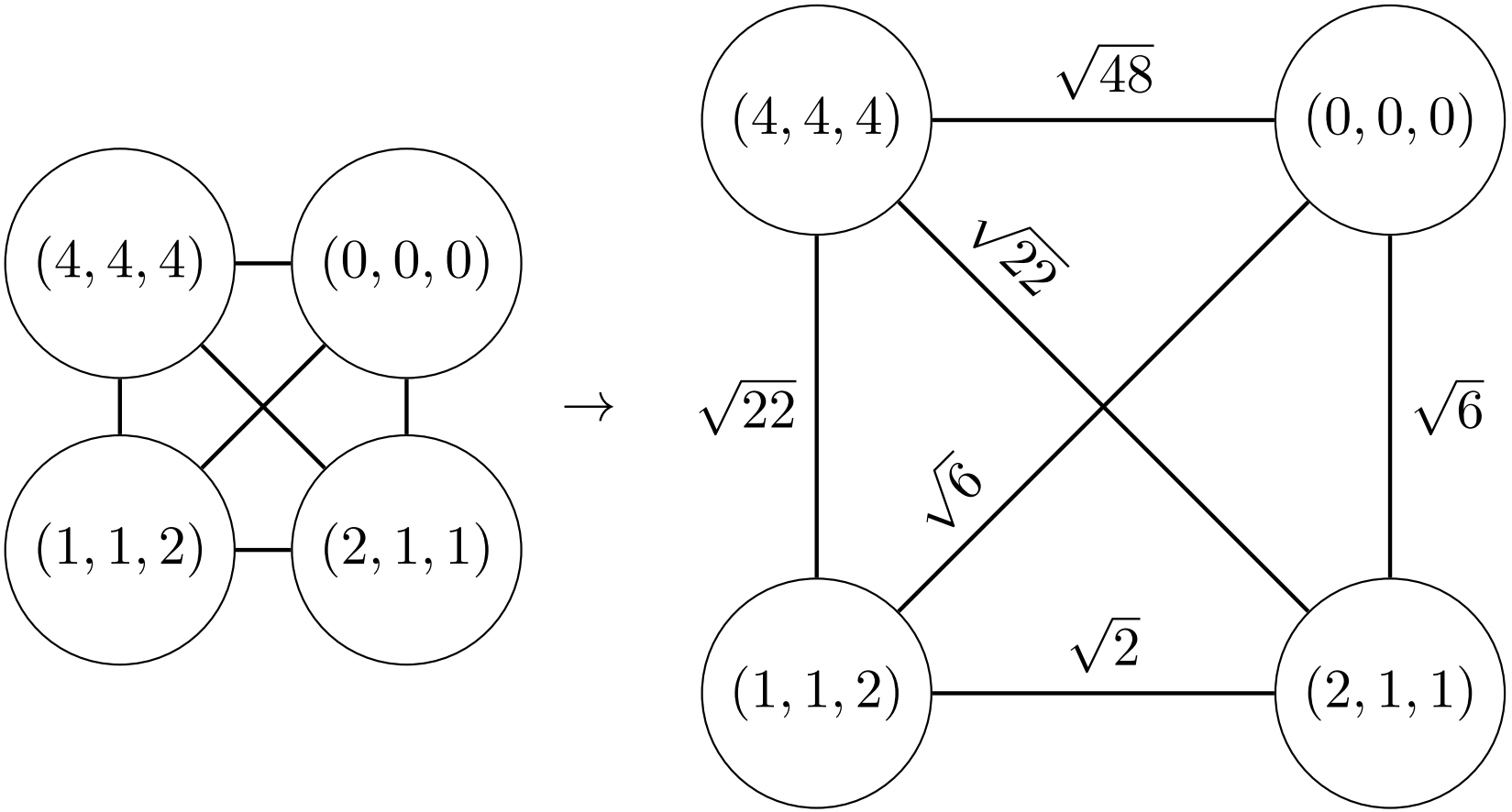
Step 2b: choosing the **weights**
Running example: $\alpha = 2$



4 Finding structure in images

Step 2b: choosing the **weights**

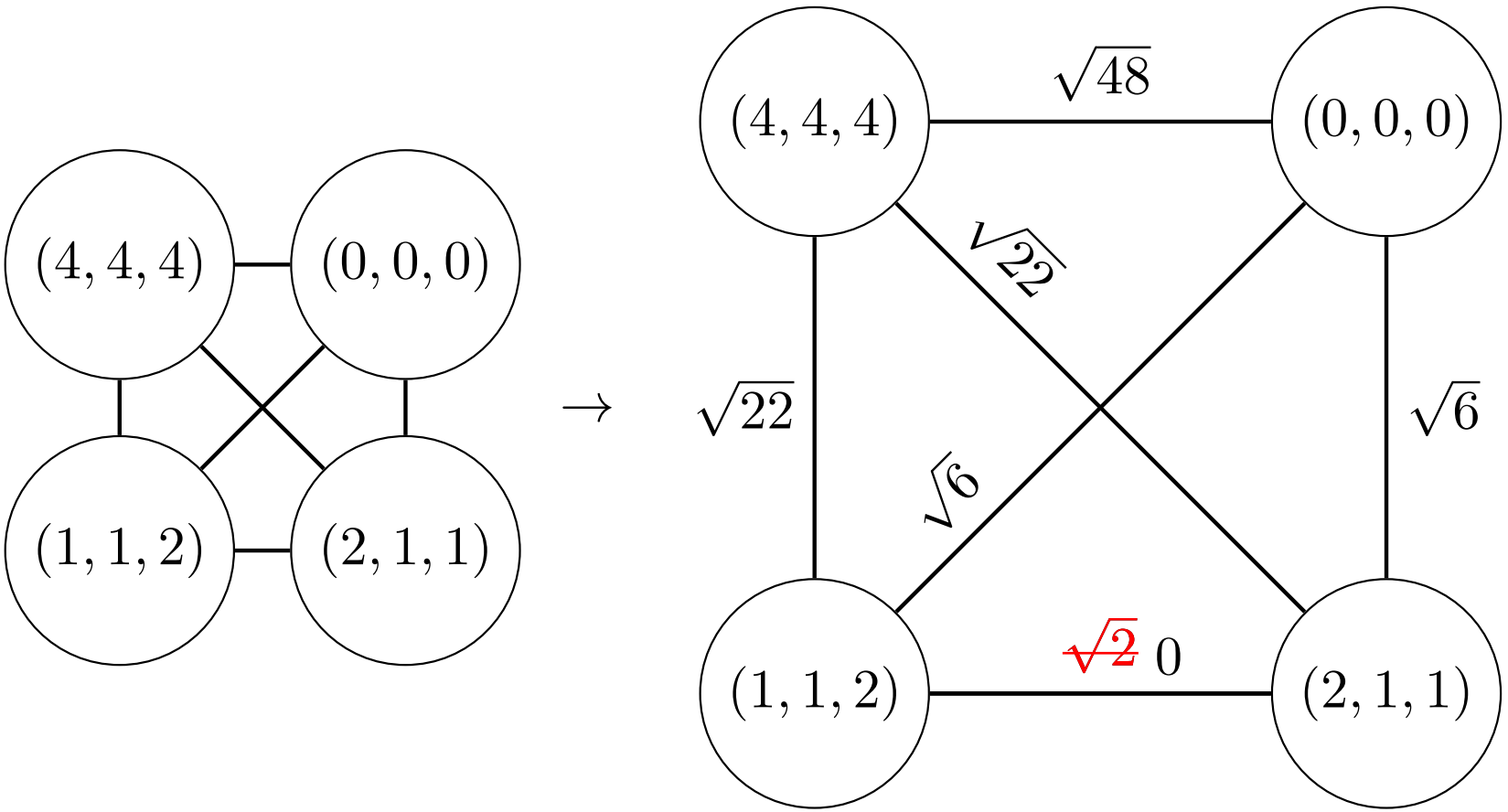
Running example: $\alpha = 2$



4 Finding structure in images

Step 2b: choosing the **weights**

Running example: $\alpha = 2$



4 Finding structure in images

How does it work?

1. Compress the image
2. Image to graph
3. Find max cut

4 Finding structure in images

How does it work?

1. Compress the image
2. Image to graph
3. Find max cut

4 Finding structure in images

Step 3: **using** the graph

4 Finding structure in images

Step 3: **using** the graph

How do we use the graph?

4 Finding structure in images

Step 3: **using** the graph

How do we use the graph?

- Edges with a high weight show a contrast

4 Finding structure in images

Step 3: **using** the graph

How do we use the graph?

- Edges with a high weight show a contrast
- Nodes with a high weight edge are on a "border"

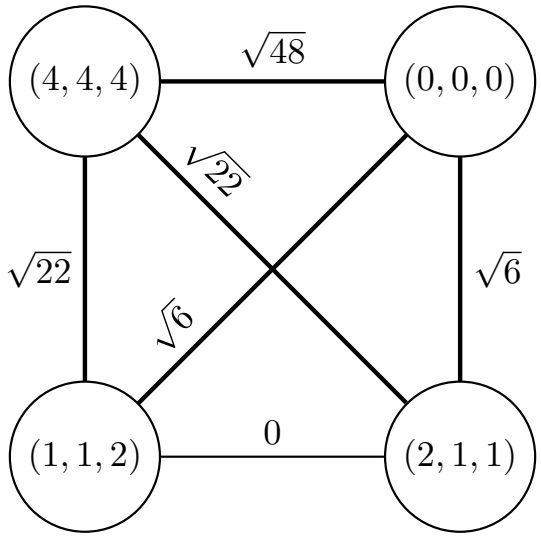
4 Finding structure in images

Step 3: **using** the graph

How do we use the graph?

- Edges with a high weight show a contrast
- Nodes with a high weight edge are on a "border"

Running example: top left node



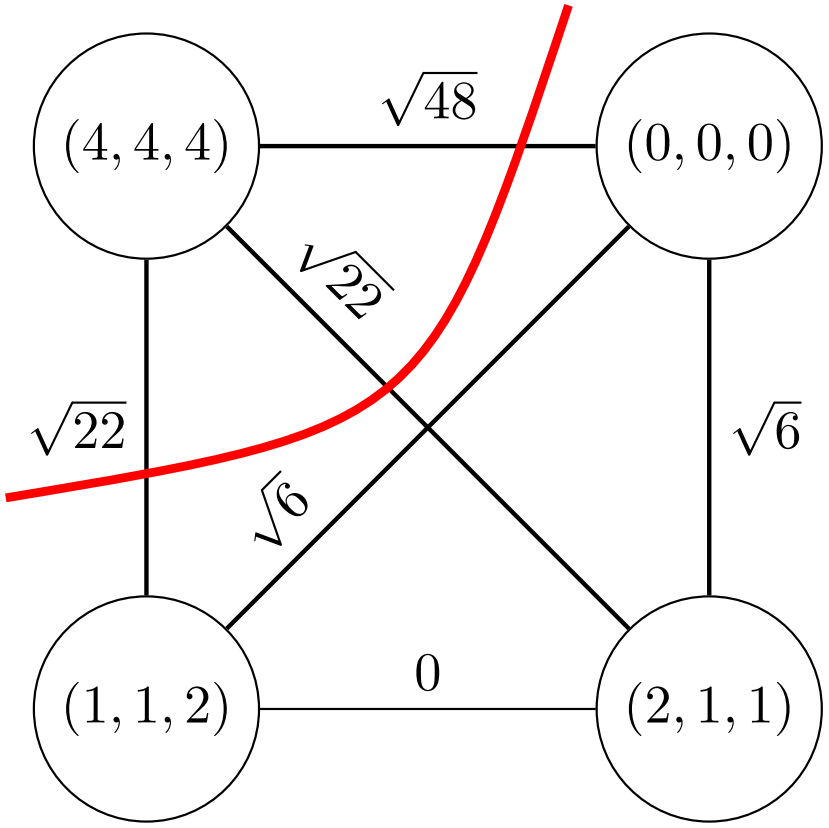
4 Finding structure in images

Step 3: **using** the graph

4 Finding structure in images

Step 3: **using** the graph

Running example:



4 Finding structure in images

Step 3: **using** the graph

Maximum cut is a discrete maximisation problem!

4 Finding structure in images

Step 3: **using** the graph

Maximum cut is a discrete maximisation problem!

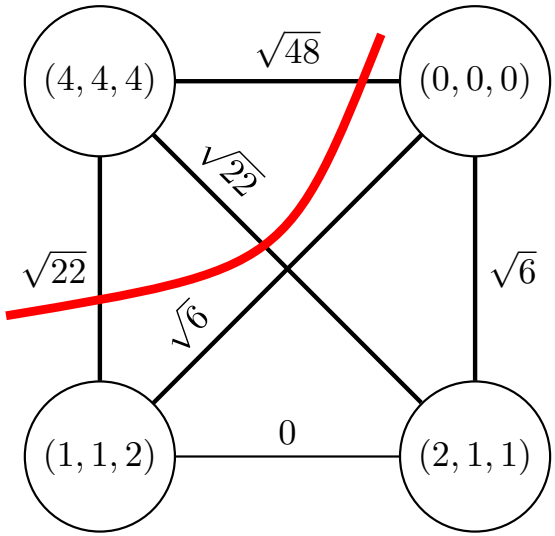
- Exponential many cuts possible

4 Finding structure in images

Step 3: **using** the graph

Maximum cut is a discrete maximisation problem!

- Exponential many cuts possible

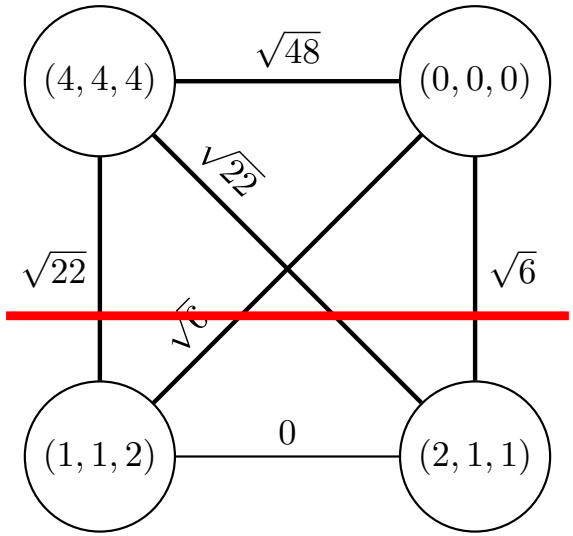


4 Finding structure in images

Step 3: **using** the graph

Maximum cut is a discrete maximisation problem!

- Exponential many cuts possible

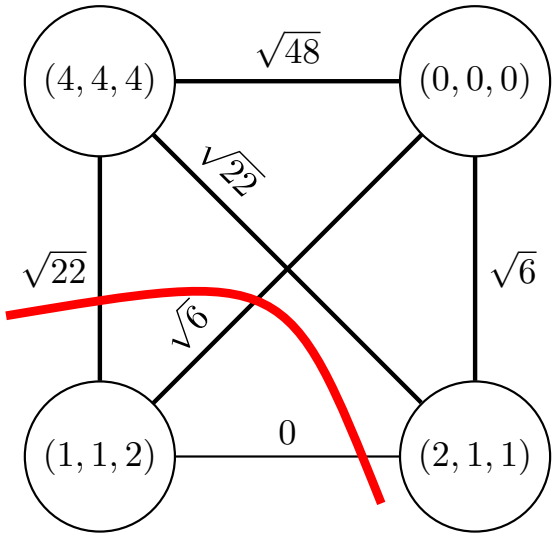


4 Finding structure in images

Step 3: **using** the graph

Maximum cut is a discrete maximisation problem!

- Exponential many cuts possible

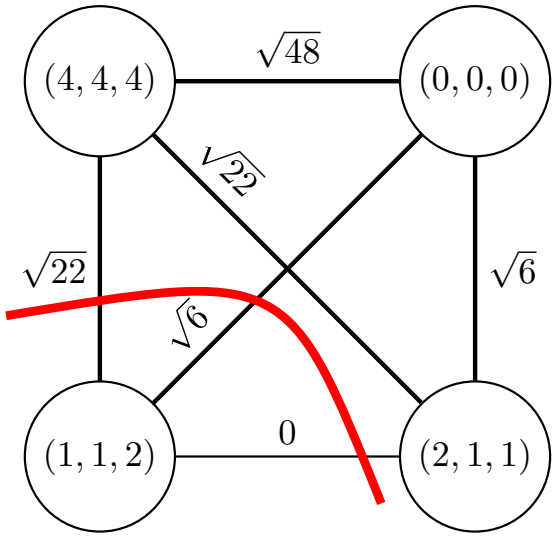


4 Finding structure in images

Step 3: **using** the graph

Maximum cut is a discrete maximisation problem!

- Exponential many cuts possible



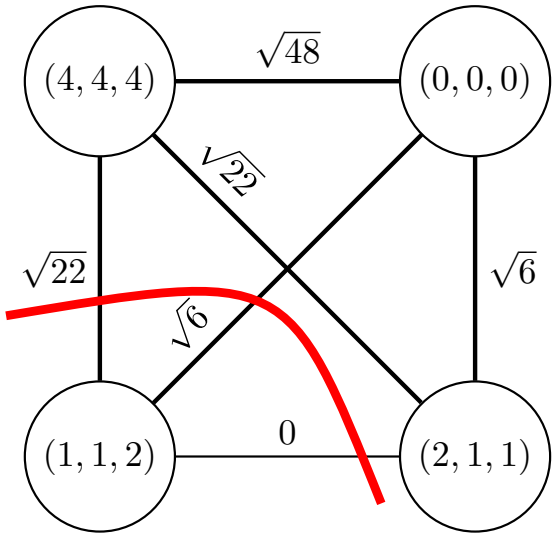
All is lost?

4 Finding structure in images

Step 3: **using** the graph

Maximum cut is a discrete maximisation problem!

- Exponential many cuts possible



All is lost?

Max cut is submodular!

4 Finding structure in images

Step 3: **using** the graph

Practical aspects:

4 Finding structure in images

Step 3: **using** the graph

Practical aspects:

- Implementation in MatLab

4 Finding structure in images

Step 3: **using** the graph

Practical aspects:

- Implementation in MatLab
- Used the submodular function optimisation (sfo) toolbox written by A. Krause [13].

4 Finding structure in images

Step 3: **using** the graph

Practical aspects:

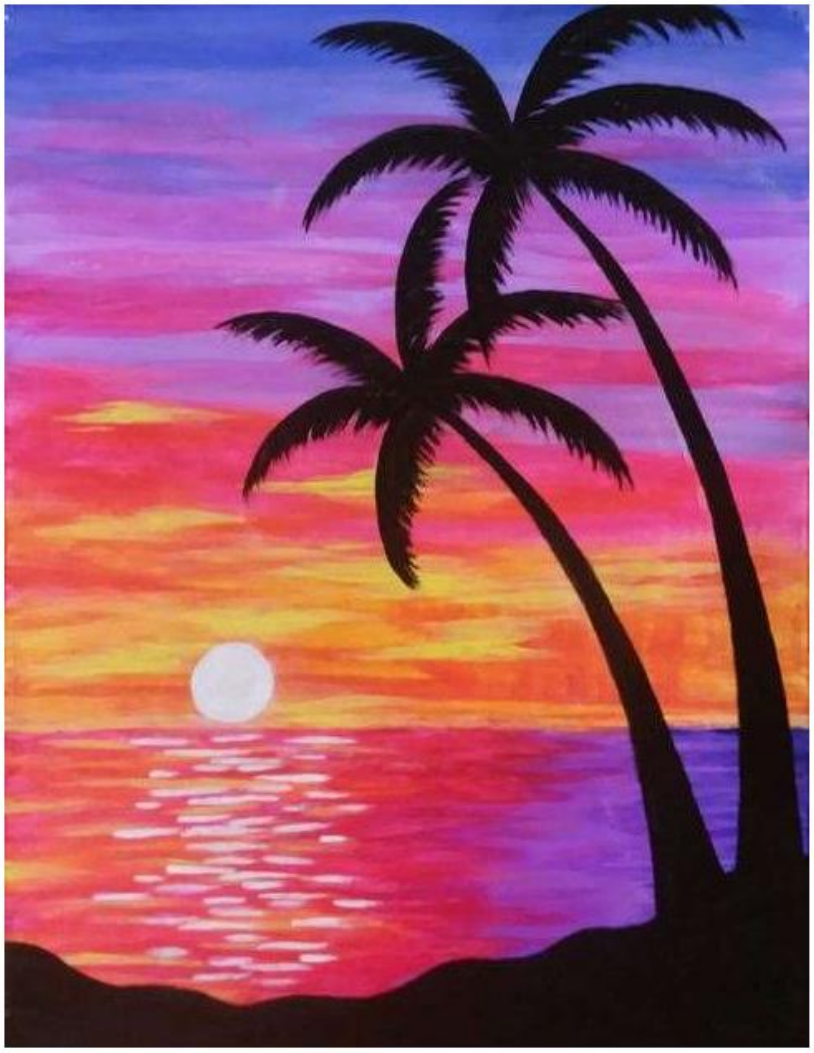
- Implementation in MatLab
- Used the submodular function optimisation (sfo) toolbox written by A. Krause [13].
- Implementation available on GitHub.
- https://github.com/MichielBaptist/Structure-Finding-Submodularity

5 Outline

5 Experimental results

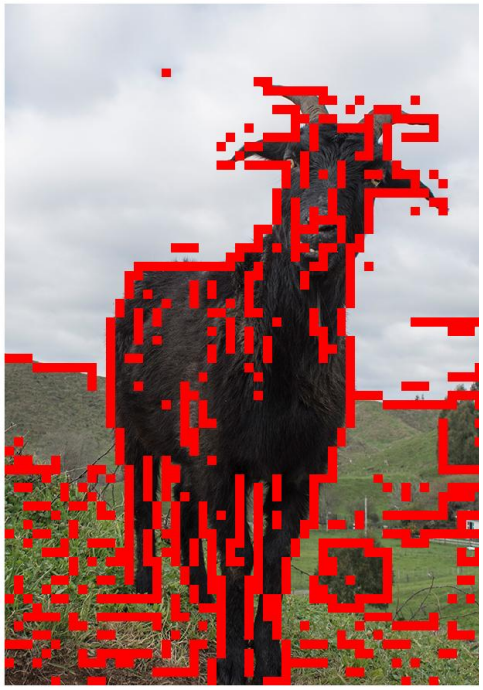
5 Experimental results

5 Experimental results

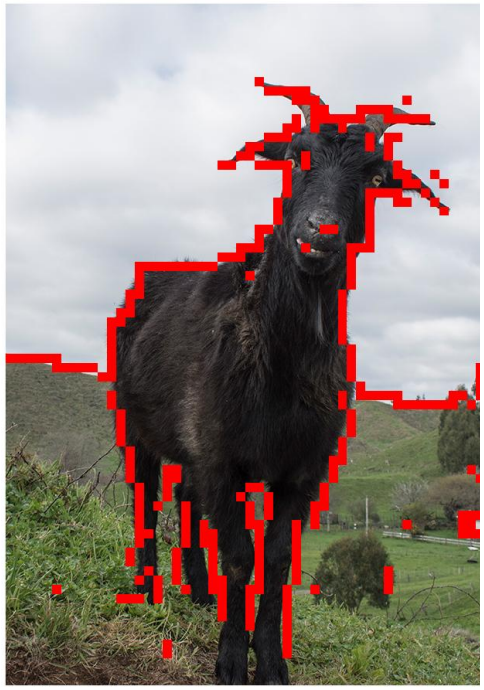


5 Experimental results

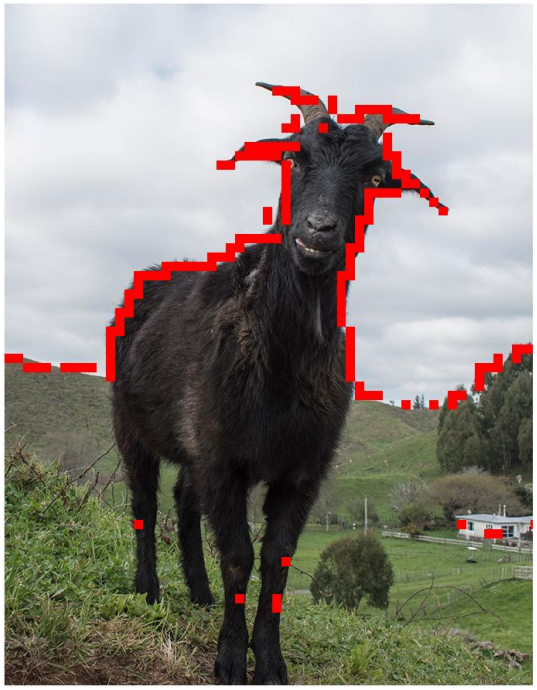
5 Experimental results



$\alpha = 0.1$



$\alpha = 0.3$



$\alpha = 0.5$

6 Outline

6 Lessons learned

Conclusion:

6 Lessons learned

Conclusion:

1. Convexity $\approx$ submodularity

6 Lessons learned

Conclusion:

1. Convexity $\approx$ submodularity
2. Submodularity in machine learning

6 Lessons learned

Conclusion:

1. Convexity $\approx$ submodularity
2. Submodularity in machine learning
3. Efficient algorithms available

6 Lessons learned

Conclusion:

1. Convexity $\approx$ submodularity
2. Submodularity in machine learning
3. Efficient algorithms available

Main takeaway: leverage structure of problem!

Thank you for your attention!

Questions?

Picture references:

- Cute dog: [2]
- Panda: [7]
- Homer's brain: [6]
- Beach: [8]
- Goat: [3]

6 References I

Image of 3d convex function.

https://www.matroid.com/blog/post/the-hard-thing-about-deep-learning.

Image of cute dog.

https://www.medicalnewstoday.com/articles/322868.php.

Image of goat.

https://landusenz.org.nz/goats-meat/.

Image of gradient descent example.

https://en.wikipedia.org/wiki/Gradient_descent.

6 References II

Image of graph cut.

https://www.boost.org/doc/libs/1_68_0/libs/graph/doc/stoer_wagner_min_cut.html.

Image of homer.

https://www.pinterest.dk/pin/556757572658884856/.

Image of panda.

https://tinypng.com/.

Image of sunset.

https://www.pinotspalette.com/bayshore/event/233562.

6 References III

Satoru Fujishige.

Submodular Functions and Optimization.

Elsevier Science, second edition edition, 2005.

Boris Goldengorin, Gerard Sierksma, Gert A. Tijssen, and Michael Tso.

The data-correcting algorithm for the minimization of supermodular functions.

Management Science, 45:1539–1551, 11 1999.

6 References IV

M. Grötschel, L. Lovász, and A. Schrijver.

The ellipsoid method and its consequences in combinatorial optimization.

Combinatorica, 1(2):169–197, Jun 1981.

Jiayue He, Jennifer Rexford, and Mung Chiang.

Design for Optimizability: Traffic Management of a Future Internet, pages 3–18.

01 2010.

Andreas Krause.

Sfo: A toolbox for submodular function optimization.

J. Mach. Learn. Res., 11, 03 2010.

6 References V

Andreas Krause and Carlos Guestrin.

Beyond convexity - submodularity in machine learning.

tutorial, 2008.

Heesang Lee, George L. Nemhauser, and Yinhua Wang.

Maximizing a submodular function by integer programming: Polyhedral results for the quadratic case.

European Journal of Operational Research, 94(1):154 – 166, 1996.

6 References VI

Hui Lin and Jeff Bilmes.

A class of submodular functions for document summarization.

In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 510–520, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

L. Lovász.

Submodular functions and convexity, pages 235–257.

Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.

6 References VII

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher.

An analysis of approximations for maximizing submodular set functions—i.

Mathematical Programming, 14(1):265–294, Dec 1978.