Optimization for Machine Learning

Lecture 5: Constraints, Discrete Optimization I

December 1, 2022

TC2 - Optimisation

Université Paris-Saclay

Anne Auger and Dimo Brockhoff

Inria Saclay – Ile-de-France

Course Overview

Date		Topic
Thu, 3.11.2022	DB	Introduction
Thu, 10.11.2022	AA	Continuous Optimization I: differentiability, gradients, convexity, optimality conditions
Thu, 17.11.2022	AA	Continuous Optimization II: constrained optimization, gradient-based algorithms, stochastic gradient
Thu, 24.11.2022	AA	Continuous Optimization III: stochastic algorithms, derivative-free optimization written test / « contrôle continue »
Thu, 1.12.2022	DB	Constrained optimization, Discrete Optimization I: graph theory, greedy algorithms
Thu, 8.12.2022	DB	Discrete Optimization II: dynamic programming, branch&bound
Thu 15.12.2022	DB	Written exam
		classes from 13h30 – 16h45 (2nd break at end)

Constrained Optimization

Small exercises on whiteboard

Equality Constraint

Objective:

Generalize the necessary condition of $\nabla f(x) = 0$ at the optima of f *when f is in $\mathcal{C}^1$, i.e. is differentiable and its differential is continuous*

Theorem:

Be $f: \mathbb{R}^n \to \mathbb{R}$, $g: \mathbb{R}^n \to \mathbb{R}$ in $\mathcal{C}^1$.

Let $a \in \mathbb{R}^n$ satisfy

$$\begin{cases} f(a) = \min \{f(x) \mid x \in \mathbb{R}^n, g(x) = 0\} \\ g(a) = 0 \end{cases}$$

i.e. a is optimum of the problem

If $\nabla g(a) \neq 0$, then there exists a constant $\lambda \in \mathbb{R}$ called *Lagrange multiplier*, such that

$$\underbrace{\nabla f(a) + \lambda \nabla g(a) = 0}_{\text{i.e. gradients of } f \text{ and } g \text{ in } a \text{ are colinear}} \quad \text{Euler} - \text{Lagrange equation}$$

Geometrical Interpretation Using an Example

Exercise:

Consider the problem

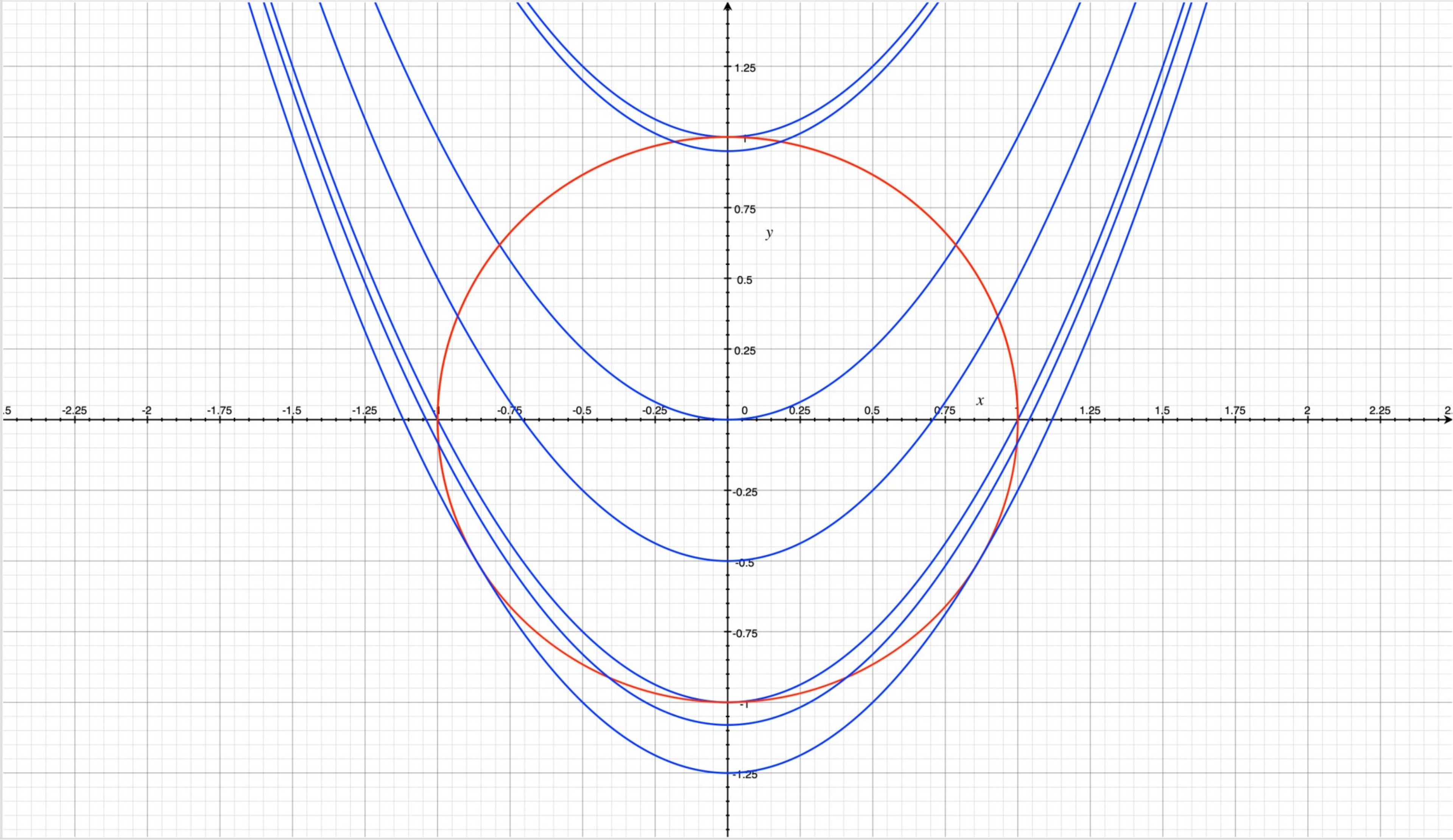
$$\min \ \{ f(x,y) \mid (x,y) \in \mathbb{R}^2, g(x,y) = 0\}$$

$f(x,y) = y - x^2 \qquad g(x,y) = x^2 + y^2 - 1 = 0$

1) Plot the level sets of f, plot $g = 0$
2) Compute ∇f and ∇g
3) Find the solutions with $\nabla f + \lambda \nabla g$ = 0

   equation solving with 3 unknowns (x, y, λ)
4) Plot the solutions of 3) on top of the level set graph of 1)

Answer



1.25
0.75
y
0.5
0.25
x
-2.25
-2
-1.75
-1.5
-1.25
-0.75
-0.5
-0.25
0
0.25
0.5
0.75
1.25
1.5
1.75
2
2.25
-0.25
-0.5
-0.75
-1
-1.25

Answer

- $(x_1, y_1, \lambda_1) = \left(0, 1, -\frac{1}{2}\right)$ [max global]
- $= \left(0, -1, \frac{1}{2}\right)$ [max local]
- $= \left(\sqrt{\frac{3}{4}}, -\frac{1}{2}, 1\right)$ [min global]
- $= \left(-\sqrt{\frac{3}{4}}, -\frac{1}{2}, 1\right)$ [min global]

Note:

Here we see clearly that the previous conditions are necessary conditions but not sufficient conditions.

Interpretation of Euler-Lagrange Equation

Intuitive way to retrieve the Euler-Lagrange equation:

- In a local minimum a of a constrained problem, the hypersurfaces (or level sets) $f = f(a)$ and $g = 0$ are necessarily tangent (otherwise we could decrease f by moving along $g = 0$).
- Since the gradients $\nabla f(a)$ and $\nabla g(a)$ are orthogonal to the level sets $f = f(a)$ and $g = 0$, it follows that $\nabla f(a)$ and $\nabla g(a)$ are colinear.

Generalization to More than One Constraint

Theorem

- Assume $f: \mathbb{R}^n \to \mathbb{R}$ and $g_k: \mathbb{R}^n \to \mathbb{R}$ $(1 \leq k \leq p)$ are $\mathcal{C}^1$.
- Let a be such that

$$\begin{cases} f(a) = \min \{f(x) \mid x \in \mathbb{R}^n, \quad g_k(x) = 0, \quad 1 \leq k \leq p\} \\ g_k(a) = 0 \ \text{ for all } 1 \leq k \leq p \end{cases}$$

- If $\left(\nabla g_k(a)\right)_{1 \leq k \leq p}$ are linearly independent, then there exist p real constants $(\lambda_k)_{1 \leq k \leq p}$ such that

$$\nabla f(a) + \sum_{k=1}^{p} \lambda_k \nabla g_k(a) = 0$$

↑ Lagrange multiplier

again: a does not need to be global but local minimum

The Lagrangian

- Define the Lagrangian on $\mathbb{R}^n \times \mathbb{R}^p$ as

$$\mathcal{L}(x, \{\lambda_k\}) = f(x) + \sum_{k=1}^{p} \lambda_k g_k(x)$$

- To find optimal solutions, we can solve the optimality system

$$\begin{cases} \text{Find } (x, \{\lambda_k\}) \in \mathbb{R}^n \times \mathbb{R}^p \text{ such that } \nabla f(x) + \sum_{k=1}^{p} \lambda_k \nabla g_k(x) = 0 \\ g_k(x) = 0 \;\text{ for all } 1 \leq k \leq p \end{cases}$$

$$\Leftrightarrow \begin{cases} \text{Find } (x, \{\lambda_k\}) \in \mathbb{R}^n \times \mathbb{R}^p \text{ such that } \nabla_x \mathcal{L}(x, \{\lambda_k\}) = 0 \\ \nabla_{\lambda_k} \mathcal{L}(x, \{\lambda_k\})(x) = 0 \;\text{ for all } 1 \leq k \leq p \end{cases}$$

Inequality Constraint: Definitions

Let $\mathcal{U} = \{x \in \mathbb{R}^n \mid g_k(x) = 0 \text{ (for } k \in E),\ g_k(x) \leq 0 \text{ (for } k \in I)\}$.

Definition:

The points in $\mathbb{R}^n$ that satisfy the constraints are also called *feasible* points.

Definition:

Let $a \in \mathcal{U}$, we say that the constraint $g_k(x) \leq 0$ (for $k \in I$) is *active* in a if $g_k(a) = 0$.

Inequality Constraint: Karush-Kuhn-Tucker Theorem

Theorem (Karush-Kuhn-Tucker, KKT):

Let $f: \mathbb{R}^n \to \mathbb{R}$, $g_k: \mathbb{R}^n \to \mathbb{R}$, all $\mathcal{C}^1$

Furthermore, let $a \in \mathbb{R}^n$ satisfy

$$\begin{cases} f(a) = \min(f(x) \mid x \in \mathbb{R}^n, g_k(x) = 0 \text{ (for } k \in E), g_k(x) \leq 0 \text{ (for } k \in \mathrm{I}) \\ g_k(a) = 0 \text{ (for } k \in E) \\ g_k(a) \leq 0 \text{ (for } k \in I) \end{cases}$$

also works again for a being a local minimum

Let I_a^0 be the set of constraints that are active in a. Assume that $\left(\nabla g_k(a)\right)_{k \in E \cup I_a^0}$ are linearly independent.

Then there exist $(\lambda_k)_{1 \leq k \leq p}$ that satisfy

$$\begin{cases} \nabla f(a) + \sum_{k=1}^{p} \lambda_k \nabla g_k(a) = 0 \\ g_k(a) = 0 \text{ (for } k \in E) \\ g_k(a) \leq 0 \text{ (for } k \in I) \\ \lambda_k \geq 0 \text{ (for } k \in I_a^0) \\ \lambda_k g_k(a) = 0 \text{ (for } k \in E \cup I) \end{cases}$$

Inequality Constraint: Karush-Kuhn-Tucker Theorem

Theorem (Karush-Kuhn-Tucker, KKT):

Let $f: \mathbb{R}^n \to \mathbb{R}$, $g_k: \mathbb{R}^n \to \mathbb{R}$, all $\mathcal{C}^1$

Furthermore, let $a \in \mathbb{R}^n$ satisfy

$$\begin{cases} f(a) = \min(f(x) \mid x \in \mathbb{R}^n, g_k(x) = 0 \ (\text{for } k \in E), g_k(x) \le 0 \ (\text{for } k \in \text{I}) \\ g_k(a) = 0 \ (\text{for } k \in E) \\ g_k(a) \le 0 \ (\text{for } k \in I) \end{cases}$$

Let I_a^0 be the set of constraints that are active in a. Assume that $\left(\nabla g_k(a)\right)_{k \in E \cup I_a^0}$ are linearly independent.

Then there exist $(\lambda_k)_{1 \le k \le p}$ that satisfy

$$\begin{cases} \nabla f(a) + \sum_{k=1}^{p} \lambda_k \nabla g_k(a) = 0 \\ g_k(a) = 0 \ (\text{for } k \in E) \\ g_k(a) \le 0 \ (\text{for } k \in I) \\ \lambda_k \ge 0 \ (\text{for } k \in I_a^0) \\ \lambda_k g_k(a) = 0 \ (\text{for } k \in E \cup I) \end{cases}$$

either active constraint or $\lambda_k = 0$

Discrete Optimization

Discrete Optimization

Context discrete optimization:

- discrete variables
- or optimization over discrete structures (e.g. graphs)
- search space often finite, but typically too large for enumeration
- → need for smart algorithms

Algorithms for discrete problems:

- typically problem-specific
- but some general concepts are repeatedly used:
  - greedy algorithms
  - [branch and bound]
  - dynamic programming
  - randomized search heuristics

before 2 excursions:
the O-notation
& graph theory

Motivation for this Part:

- get an idea of the most common algorithm design principles

Excursion: The O-Notation

Excursion: The O-Notation

Motivation:

- we often want to characterize how quickly a function f(x) grows asymptotically
- e.g. when we say an algorithm takes quadratically many steps (in the input size) to find the optimum of a problem with n (binary) variables, it is most likely not exactly n^2, but maybe n^2+1 or $(n+1)^2$

Big-O Notation

should be known, here mainly restating the definition:

Definition 1 We write $f(x) = O(g(x))$ iff there exists a constant $c > 0$ and an $x_0 > 0$ such that $|f(x)| \leq c \cdot g(x)$ holds for all $x > x_0$

we also view O(g(x)) as a set of functions growing at most as quick as g(x) and write $f(x) \in O(g(x))$

Big-O: Examples

- $f(x) + c = O(f(x))$ [if f(x) does not go to zero for x to infinity]
- $c \cdot f(x) = O(f(x))$
- $f(x) \cdot g(x) = O(f(x) \cdot g(x))$
- $3n^4 + n^2 - 7 = O(n^4)$

Intuition of the Big-O:

- if f(x) = O(g(x)) then g(x) gives an upper bound (asymptotically) for f excluding constants and lower order terms
- With Big-O, you should have '≤' in mind
- An algorithm that solves a problem in polynomial time is "efficient"
- An algorithm that solves a problem in exponential time is not
- But be aware:
  In practice, often the line between efficient and non-efficient lies around $n \log n$ or even n (or even $\log n$ in the big data context) and the constants **do** matter!!!

Excursion: The O-Notation

Further definitions to generalize from ‘≤’ to ‘≥‘ and ‘=‘:

- $f(x) = \Omega(g(x))$ if $g(x) = O(f(x))$
- $f(x) = \Theta(g(x))$ if $f(x) = O(g(x))$ and $g(x) = O(f(x))$

Note: extensions to ‘<‘ and ‘>’ exist as well, but are not needed here.

Example:

- Algo A solves problem P in time $O(n)$
- Algo B solves problem P in time $O(n^2)$
- which one is faster?

only proving upper bounds to compare algorithms is not sufficient!

Excursion: The O-Notation

Further definitions to generalize from '≤' to '≥' and '=':

- $f(x) = \Omega(g(x))$ if $g(x) = O(f(x))$
- $f(x) = \Theta(g(x))$ if $f(x) = O(g(x))$ and $g(x) = O(f(x))$

Note: extensions to '<' and '>' exist as well, but are not needed here.

Example:

- Algo A solves problem P in time $O(n)$
- Algo B solves problem P in time ~~$O(n^2)$~~ $\Omega(n^2)$
- which one is faster?

only proving upper bounds to compare algorithms is not sufficient!

Exercise O-Notation

❶ Please order the following functions in terms of their asymptotic behavior (from smallest to largest):

- $\exp(n^2)$
- log n
- ln n / ln ln n
- n
- n log n
- exp(n)
- ln n!

❷ Pick one pair of runtimes and give a formal proof for the relation.

Exercise O-Notation (Solution)

Correct ordering:

$\frac{\ln(n)}{\ln(\ln(n))} = O(\log n)$ $\quad$ $\log n = O(n)$ $\quad$ $n = O(n \log n)$

$n \log n = \Theta(\ln(n!))$ $\quad$ $\ln(n!) = O(e^n)$ $\quad$ $e^n = O(e^{n^2})$

but for example $e^{n^2} \neq O(e^n)$

One exemplary proof:

$\frac{\ln(n)}{\ln(\ln(n))} = O(\log n)$:

- $$\left|\frac{\ln(n)}{\ln(\ln(n))}\right| = \frac{\log(n)}{\log(e)\ln(\ln(n))} \leq \frac{3\log(n)}{\ln(\ln(n))} \leq 3\log(n)$$

  first $\leq$: for $n > 1$; second $\leq$: for $n > 15$

Exercise O-Notation (Solution)

One additional proof: ln n! = O(n log n)

- Stirling's approximation: $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ or even

$$\sqrt{2\pi}n^{n+1/2}e^{-n} \leq n! \leq e n^{n+\frac{1}{2}}e^{-n}$$

- $\ln n! \leq \ln(e n^{n+\frac{1}{2}}e^{-n}) = 1 + \left(n+\frac{1}{2}\right)\ln n - n$

$$\leq \left(n+\frac{1}{2}\right)\ln n \leq 2n\ln n = 2n\,\frac{\log n}{\log e} = c\cdot n\log n$$

okay for $c = 2/\log e$ and all $n \in \mathbb{N}$

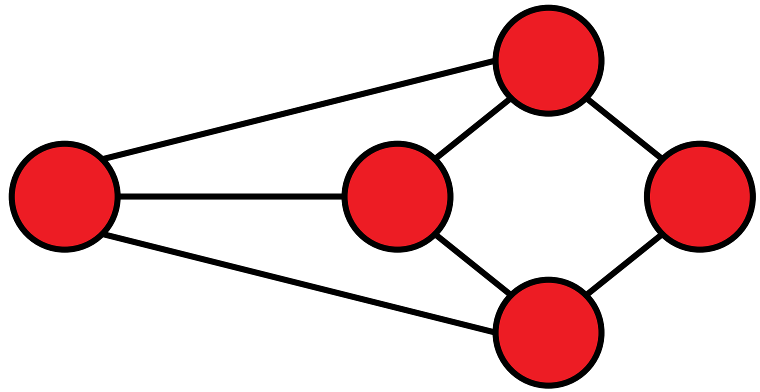
- n ln n = O(ln n!) proven in a similar vein

Excursion:
Basic Concepts of Graph Theory

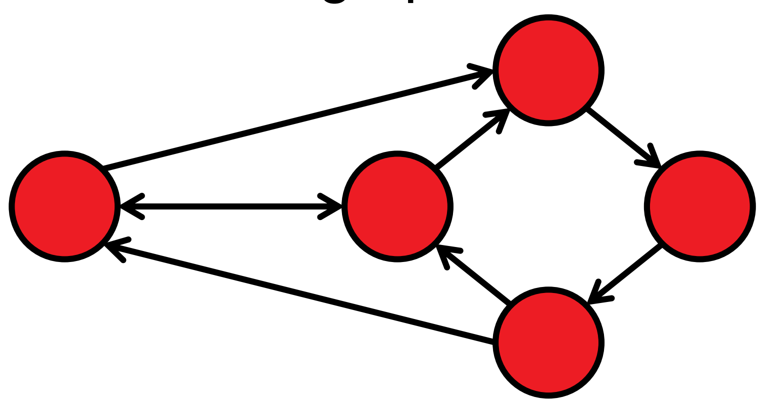
[following for example http://math.tut.fi/~ruohonen/GT_English.pdf]

Graphs

Definition 1 *An undirected graph* G *is a tupel* $G = (V, E)$ *of edges* $e = \{u, v\} \in E$ *over the vertex set* V *(i.e.,* $u, v \in V$*).*

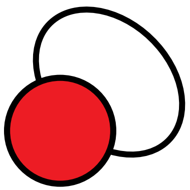


- vertices = nodes
- edges = lines
- Note: edges cover two *unordered* vertices (*undirected* graph)
  - if they are *ordered*, we call G a *directed* graph



Graphs: Basic Definitions

- G is called *empty* if E empty
- u and v are *end vertices* of an edge {u,v}
- Edges are *adjacent* if they share an end vertex
- Vertices u and v are *adjacent* if {u,v} is in E



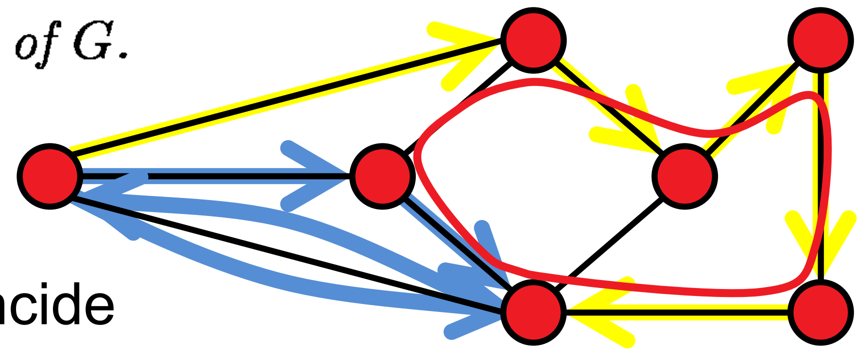
a loop

Walks, Paths, and Circuits

Definition 1 *A* walk *in a graph* $G = (V, E)$ *is a sequence*

$$v_{i_0}, e_{i_1} = (v_{i_0}, v_{i_1}), v_{i_1}, e_{i_2} = (v_{i_1}, v_{i_2}), \ldots, e_{i_k}, v_{i_k},$$

alternating vertices and adjacent edges of G.



A walk is

- *closed* if first and last node coincide
- a *trail* if each edge traversed at most once
- a *path* if each vertex is visited at most once

a closed path is called a *circuit* or *cycle*