

科学技術社会論 II
距離

岡本 吉央
okamotoy@uec.ac.jp

電気通信大学 大学院情報理工学研究科 情報・通信工学専攻

2014 年 10 月 14 日

目次

① 距離の定義

② 距離の応用

分類

検索

配置

③ 距離の計算

文字列間の Levenshtein 距離

曲線間の Frechét 距離

この2コマの目標

「距離」にまつわる話

- ▶ 距離とは何か？ (定義)
- ▶ 距離をどう使うのか？ (応用)
- ▶ 距離をどう計算するのか？ (計算)

自己紹介

岡本 吉央 (おかもと よしお)

(愛知県出身)

- ▶ 1995年：東京大学 理科一類 入学
- ▶ 1999年：東京大学 教養学部 基礎科学科第二 卒業
- ▶ 2001年：東京大学 大学院総合文化研究科 広域科学専攻
修士課程 修了
- ▶ 2005年：スイス連邦工科大学チューリヒ校 情報科学科
大学院課程 修了, Ph.D.
- ▶ 2005年：豊橋技術科学大学 助手
- ▶ 2007年：同 助教
- ▶ 2007年：東京工業大学 特任准教授
- ▶ 2010年：北陸先端科学技術大学院大学 特任准教授
- ▶ 2012年：電気通信大学 准教授, 現在に至る

専門：離散数学, 離散アルゴリズム, 離散最適化, それらの応用

宣伝？

(特集) 社会と人間行動を照らすゲーム理論 (座談会) 岡本吉央×坂井豊貴×船木由喜彦×横尾真

経済セミナー 10・11
2014
No.680
THE KEIZAI SEMINAR

特集 *Feature*
社会と人間行動を照らすゲーム理論
武藤滋夫◎ゲーム理論の展開と経済学
関口格◎繰り返しゲーム——協調的行動の手引き
西條辰義◎メカニズム・デザインの新たな挑戦
藤中裕二+若山琢磨◎非分割財交換の分析——TTCアルゴリズムの適用
佐野隆司◎オークションの理論と現実への応用
——正しい情報を引き出すメカニズム

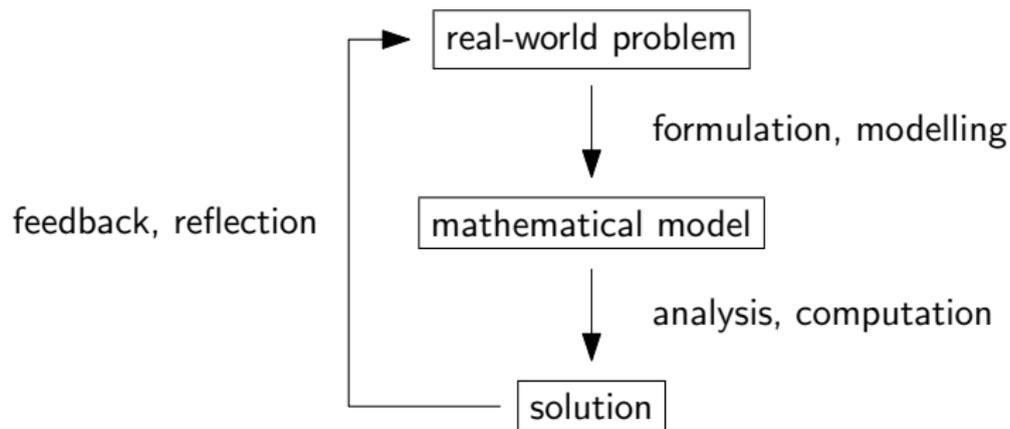
座談会 *Discussion*
数学セミナー+経済セミナー合同企画
ゲーム理論で変わる社会
岡本吉央 坂井豊貴 船木由喜彦 横尾真

トピックス *Topics*
橋本俊詔 [最終講義]
格差論、幸福論、経済政策論の軌跡
現代の経済政策を巡る議論 (パネルディスカッション) 八田達夫×太田聰一×八木匡

日本評論社『経済セミナー』
2014年10・11月号
日本評論社『数学セミナー』
2014年10月号

座談会
「ゲーム理論で変わる社会」

数学の使い方



勉強しなくてはいけないのは

- ▶ 道具としての数学
- ▶ 数学の使い方

目次

① 距離の定義

② 距離の応用

分類

検索

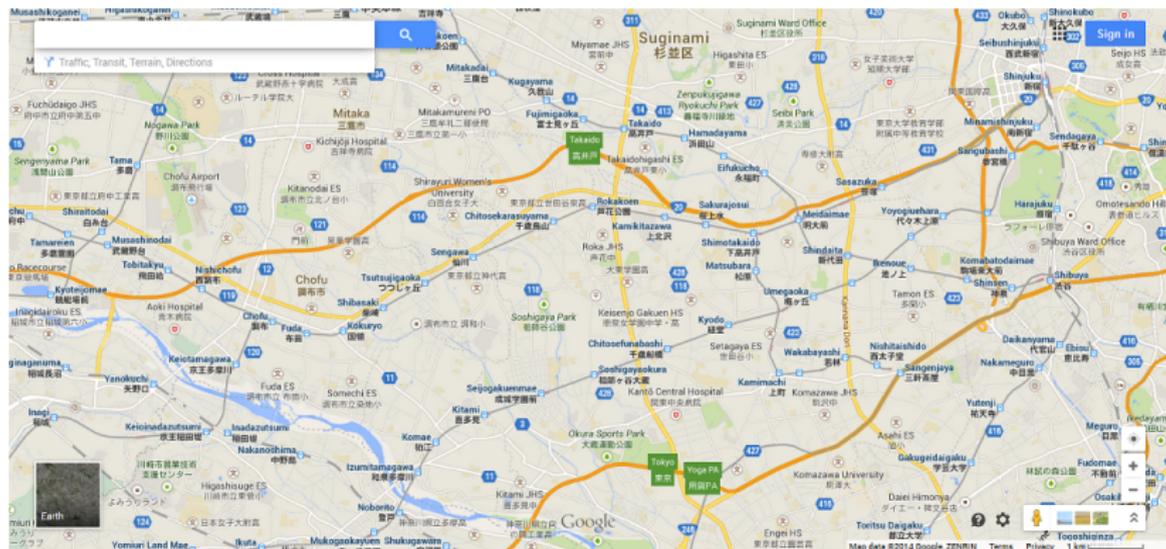
配置

③ 距離の計算

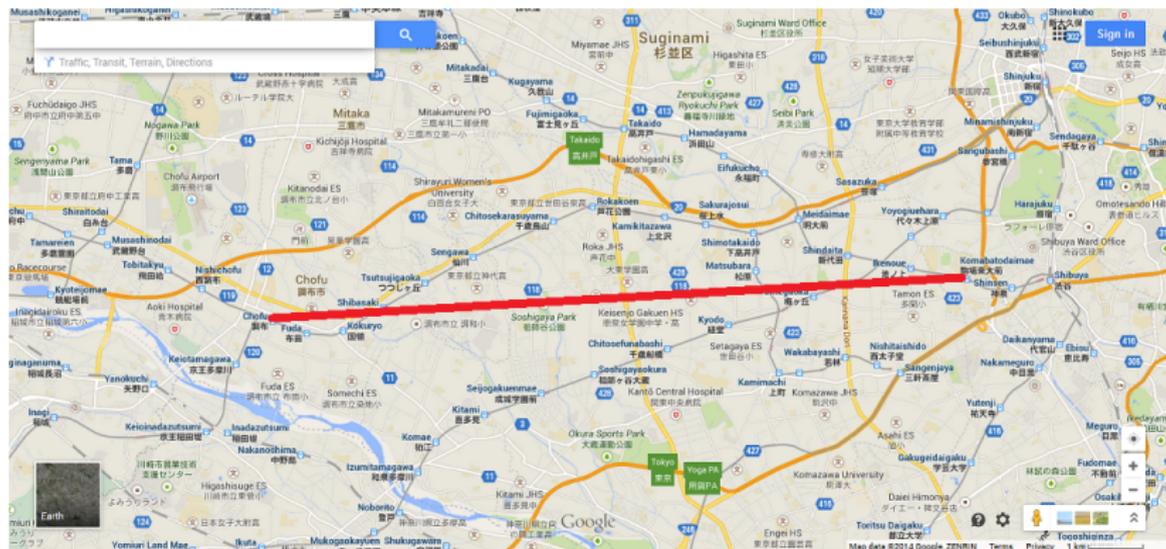
文字列間の Levenshtein 距離

曲線間の Frechét 距離

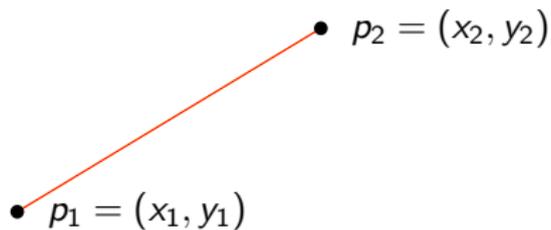
距離と言われて思い浮かべるもの



距離と言われて思い浮かべるもの



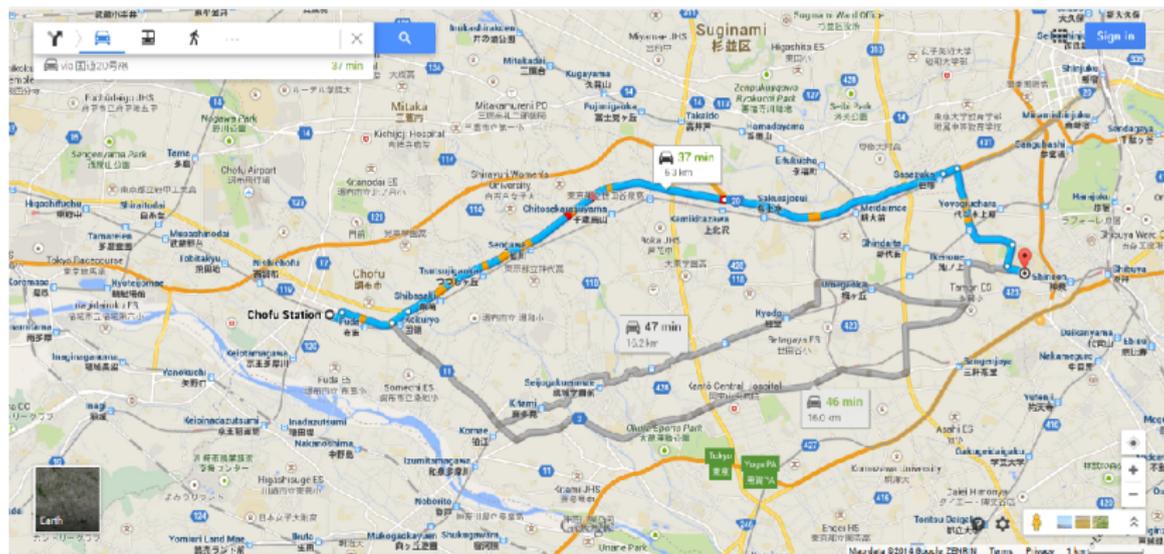
距離の公式



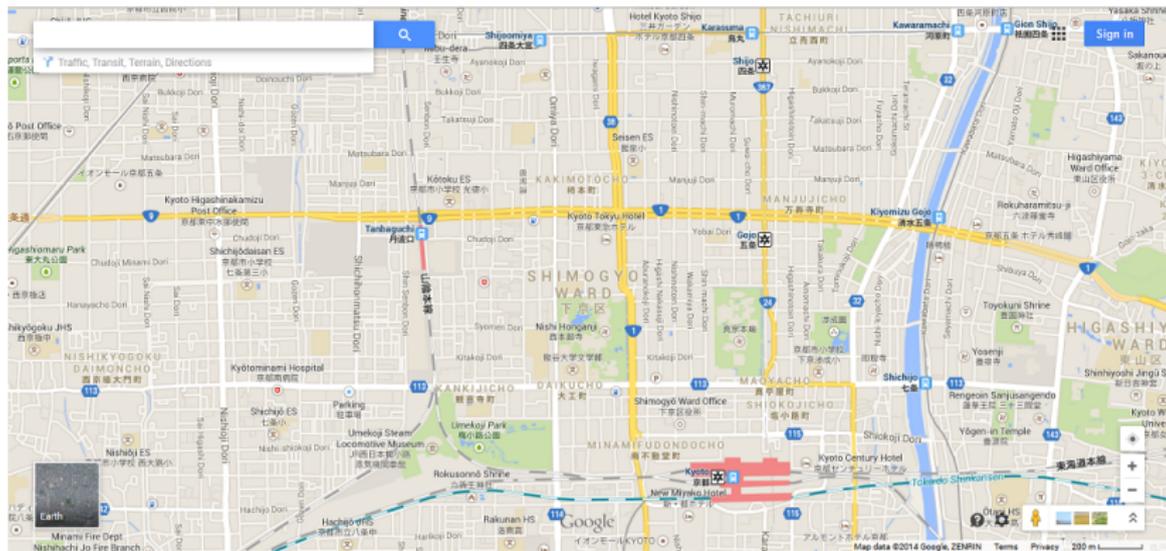
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

(よく「ユークリッド距離」と呼ばれる)

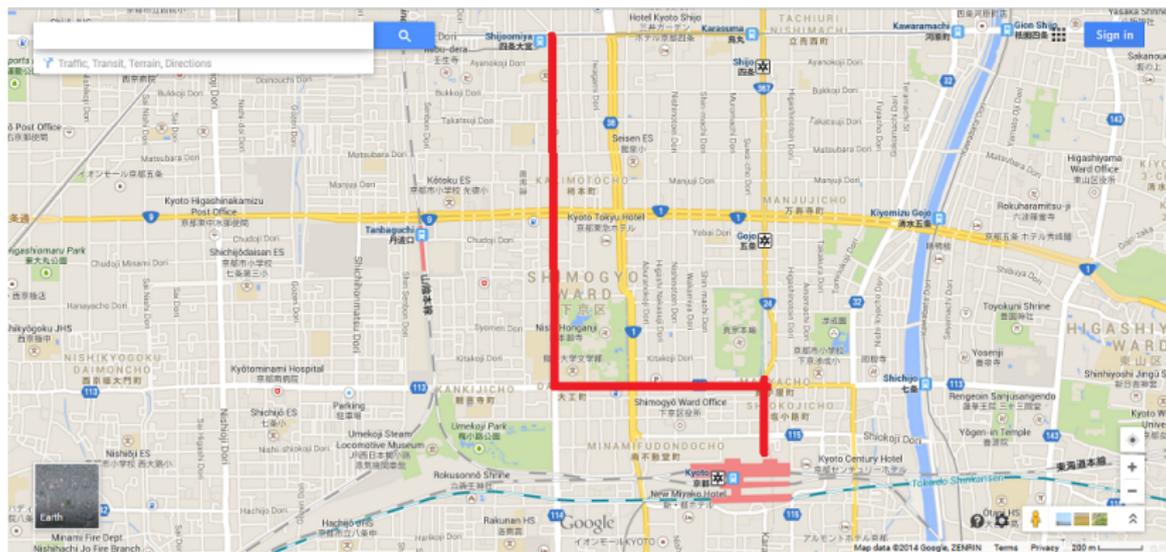
距離と言われて思い浮かべるもの



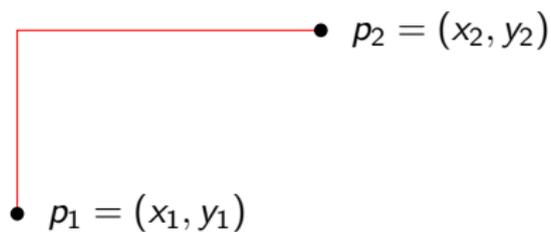
京都



京都



京都 — 距離の公式



$$|x_1 - x_2| + |y_1 - y_2|$$

(よく「マンハッタン距離」と呼ばれる)

距離によって何ができる?: 分類



<http://ja.wikipedia.org/wiki/東京都区部>

距離によって何ができる?: 検索



距離は有用なのだから…

距離は有用なのだから、いろいろなものの間の「距離」を測りたい

それができれば、次もできそう

- ▶ 分類
- ▶ 検索
- ▶ 配置

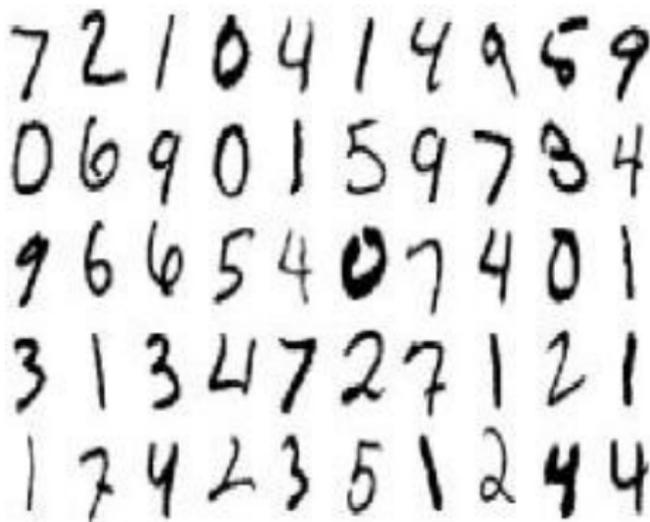
別の言い方

分類／検索／配置を行いたいときは、
「距離」が測れれば、うまくいきそう

⇒ 距離に基づく統一的方法論を目指せる

(数理の力)

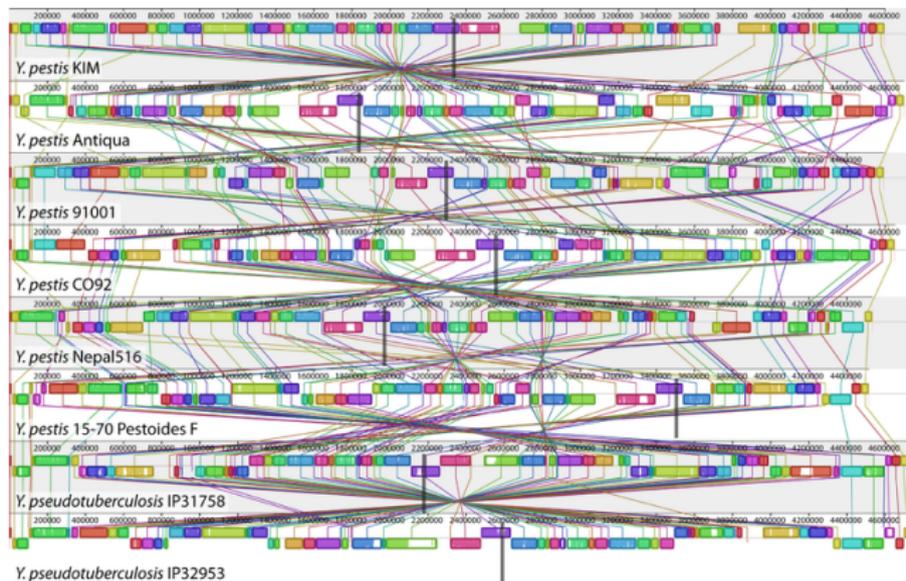
距離を測りたいもの：画像



MNIST Database of handwritten digits

⇒ 画像認識，画像検索

距離を測りたいもの：ゲノム

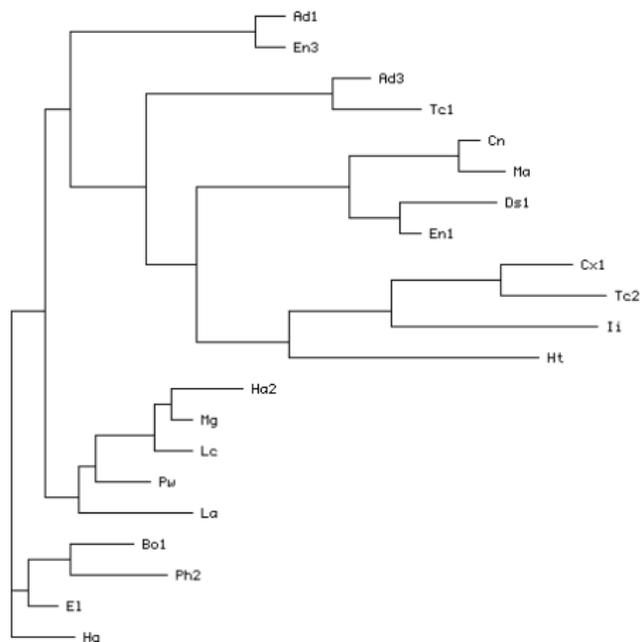


http://en.wikipedia.org/wiki/Comparative_genomics

⇒ 系統樹復元，遺伝形質発見

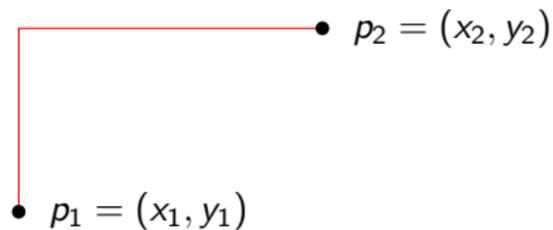
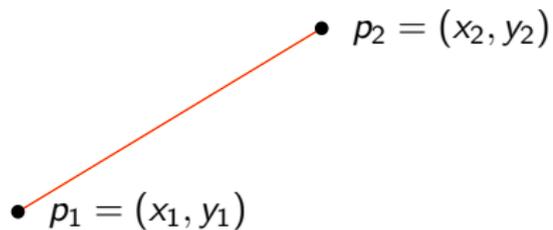
距離を測りたいもの：文書

カンタベリー物語の「系統樹」



http://www.ahds.ac.uk/_print_/history/creating/case-studies/canterbury/index.htm

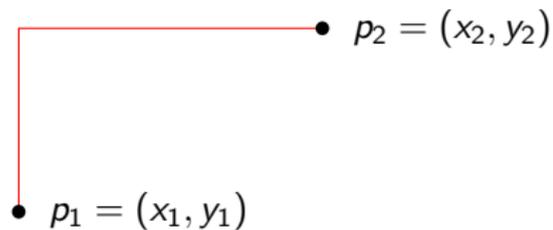
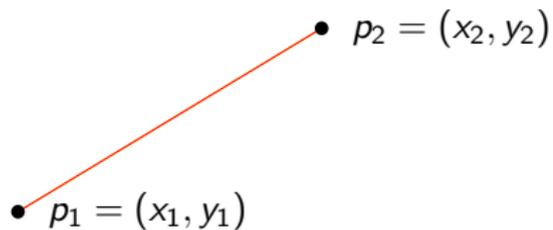
2つの距離が共通して持つ性質：非負性



非負性

$$d(p_1, p_2) \geq 0$$

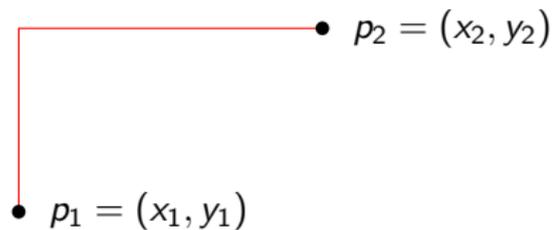
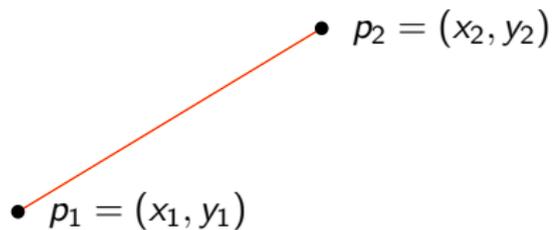
2つの距離が共通して持つ性質：同一性



同一性

$$d(p_1, p_2) = 0 \quad \Leftrightarrow \quad p_1 = p_2$$

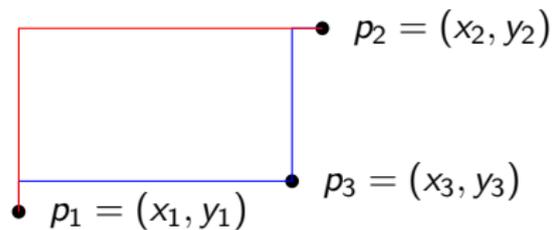
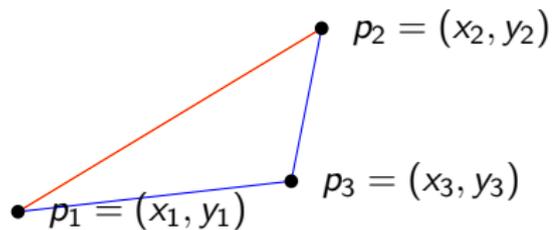
2つの距離が共通して持つ性質：対称性



対称性

$$d(p_1, p_2) = d(p_2, p_1)$$

2つの距離が共通して持つ性質：三角不等式



三角不等式

$$d(p_1, p_2) \leq d(p_1, p_3) + d(p_3, p_2)$$

距離の定義

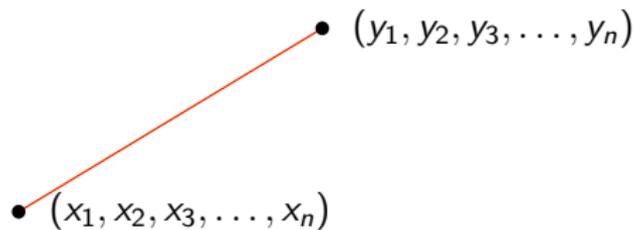
距離の定義

2つの「もの」 x と y の間の距離 $d(x, y)$ とは次を満たすもののこと

- 1 $d(x, y) \geq 0$
- 2 $d(x, y) = 0 \Leftrightarrow x = y$
- 3 $d(x, y) = d(y, x)$
- 4 $d(x, y) \leq d(x, z) + d(z, y)$

これを満たす $d(x, y)$ はどれも「距離」である

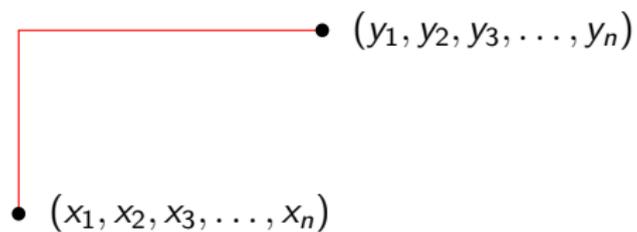
高次元ユークリッド距離



$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

先ほどの4つの条件を満たす

高次元マンハッタン距離



$$\sum_{i=1}^n |x_i - y_i|$$

先ほどの4つの条件を満たす

編集に基づく距離

k i t t e n	
s i t t e n	replace
s i t t i n	replace
s i t t i n g	insert

編集操作 :

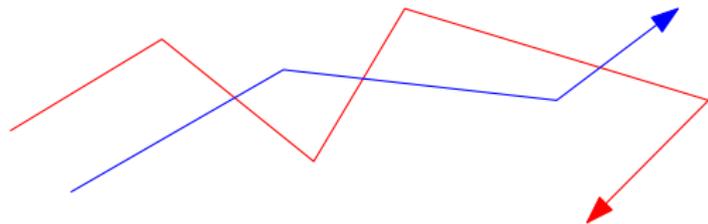
- ▶ 置換
- ▶ 挿入
- ▶ 削除

Levenshtein 距離 = 一方から他方を得るための編集回数の**最小値**

$$d(\text{kitten}, \text{sitting}) = 3$$

曲線間の Frechét 距離

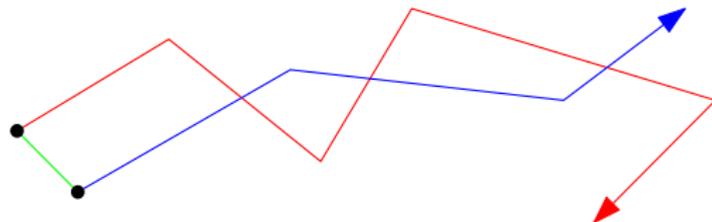
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

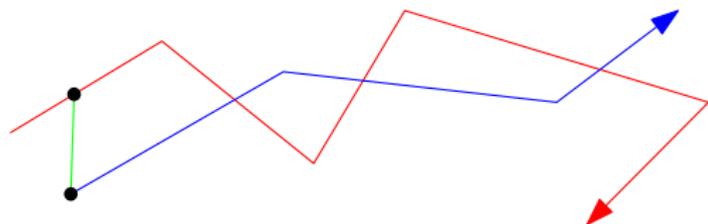
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

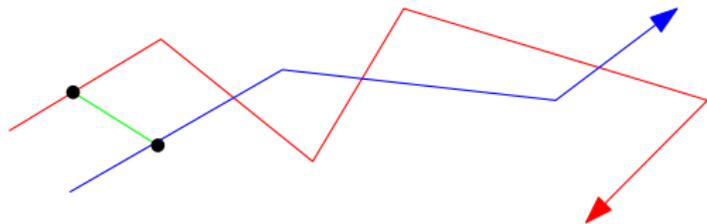
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

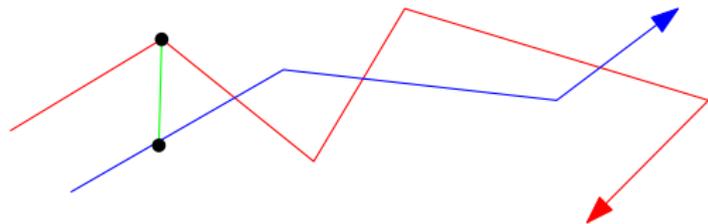
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

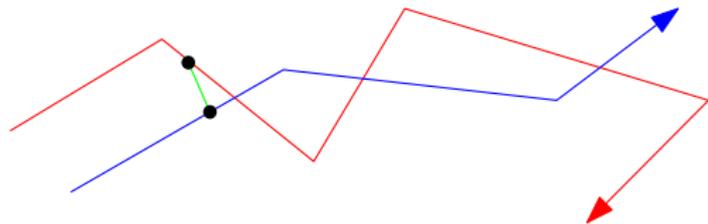
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

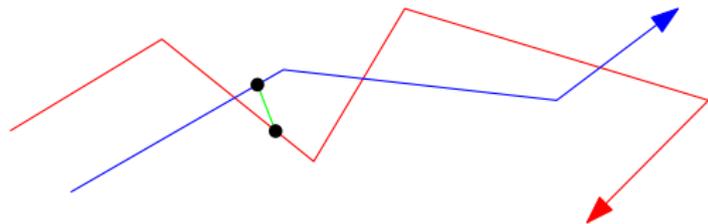
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

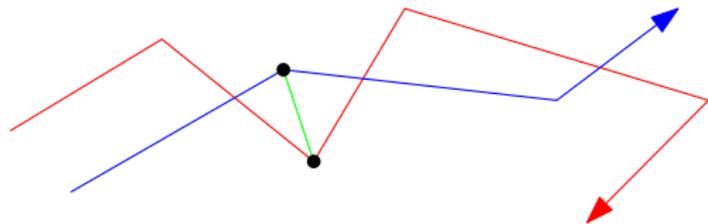
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

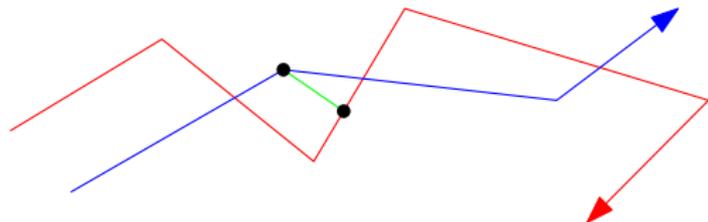
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

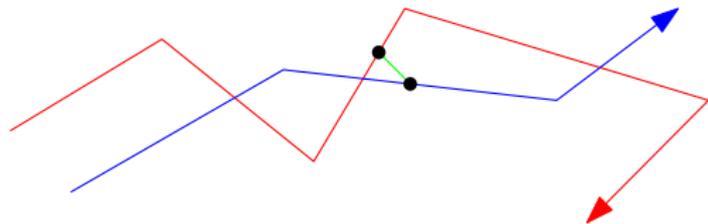
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

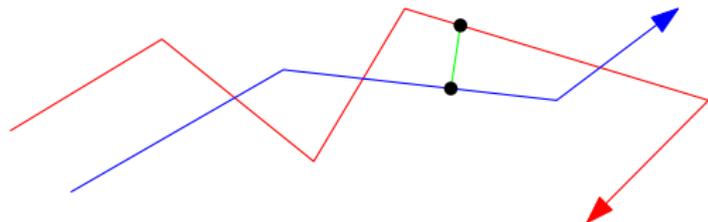
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

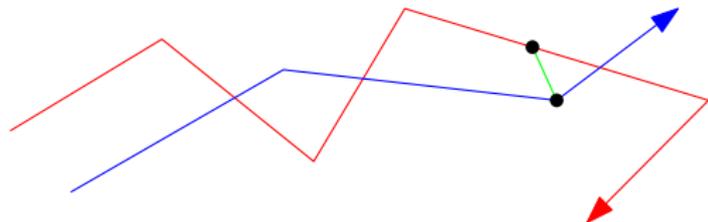
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

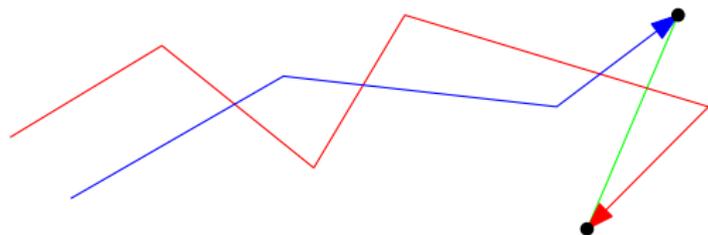
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

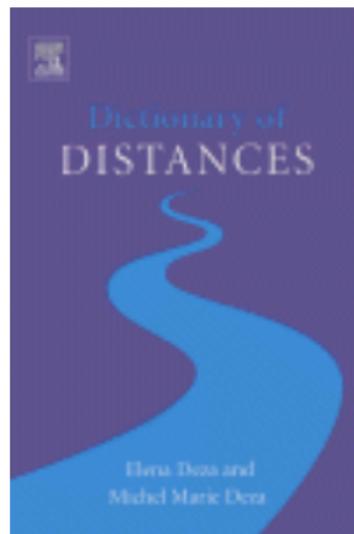
曲線間の Frechét 距離

曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

距離の辞典



Elena Deza and Michel-Marie Deza,
Dictionary of Distances,
Elsevier, 2006.

目次

- ① 距離の定義
- ② 距離の応用
 - 分類
 - 検索
 - 配置
- ③ 距離の計算
 - 文字列間の Levenshtein 距離
 - 曲線間の Frechét 距離

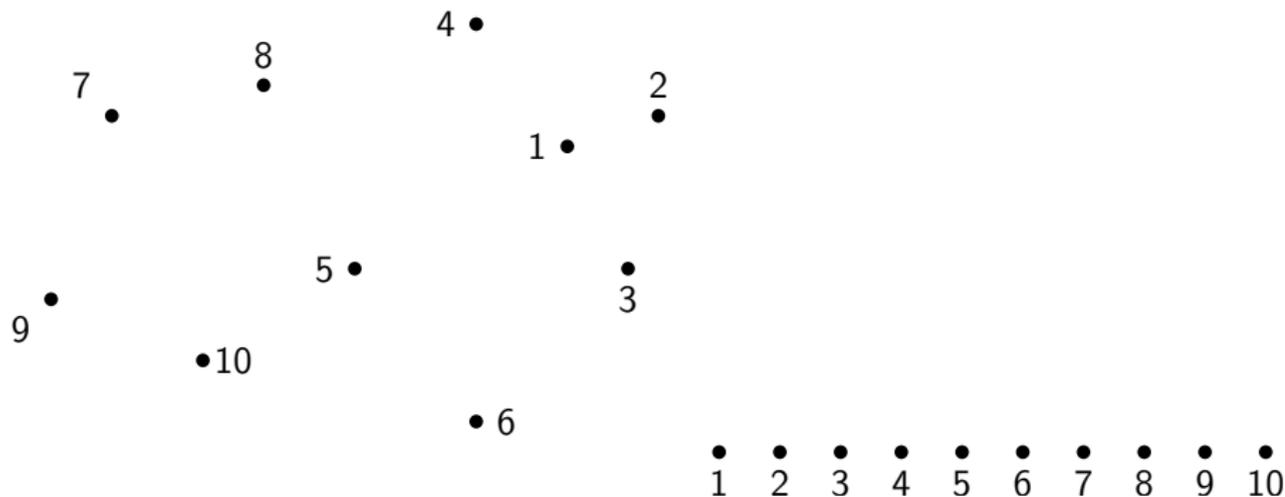
距離を用いた分類



応用例

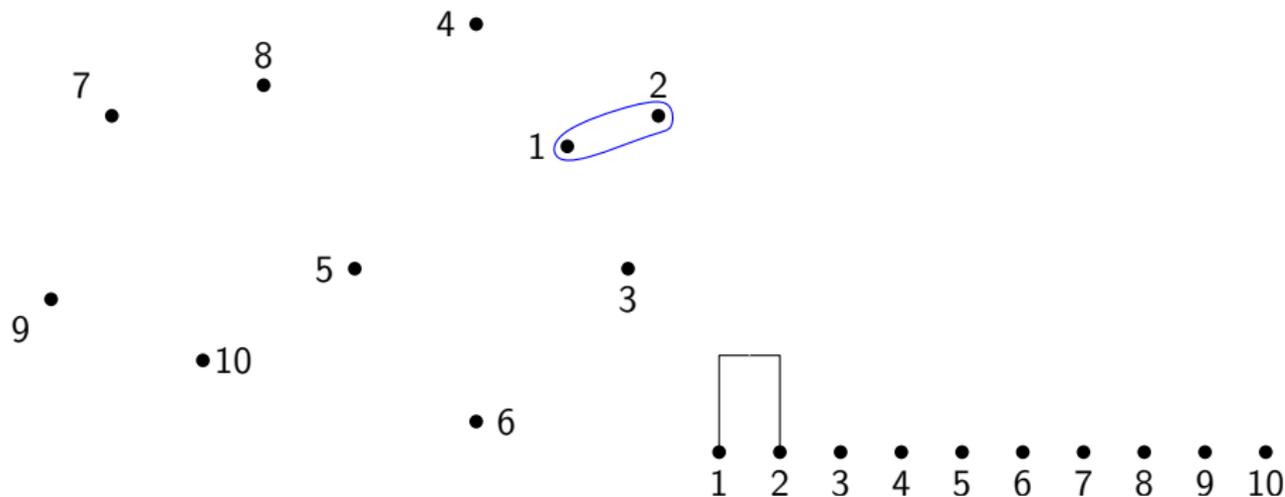
- ▶ 階層的クラスタリング
- ▶ 系統樹作成

樹形図 (デンドログラム) の作成法



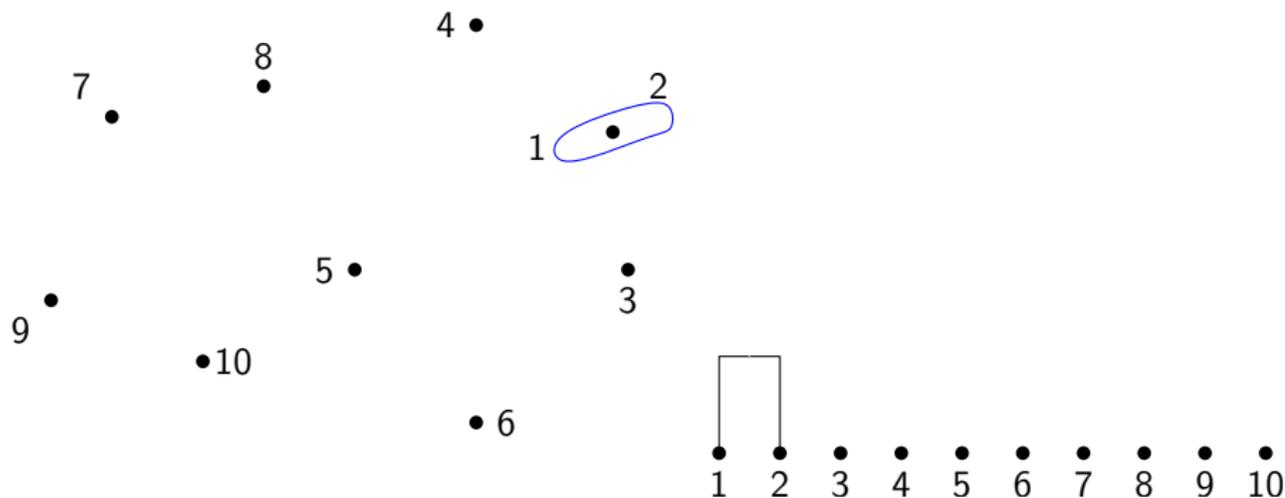
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



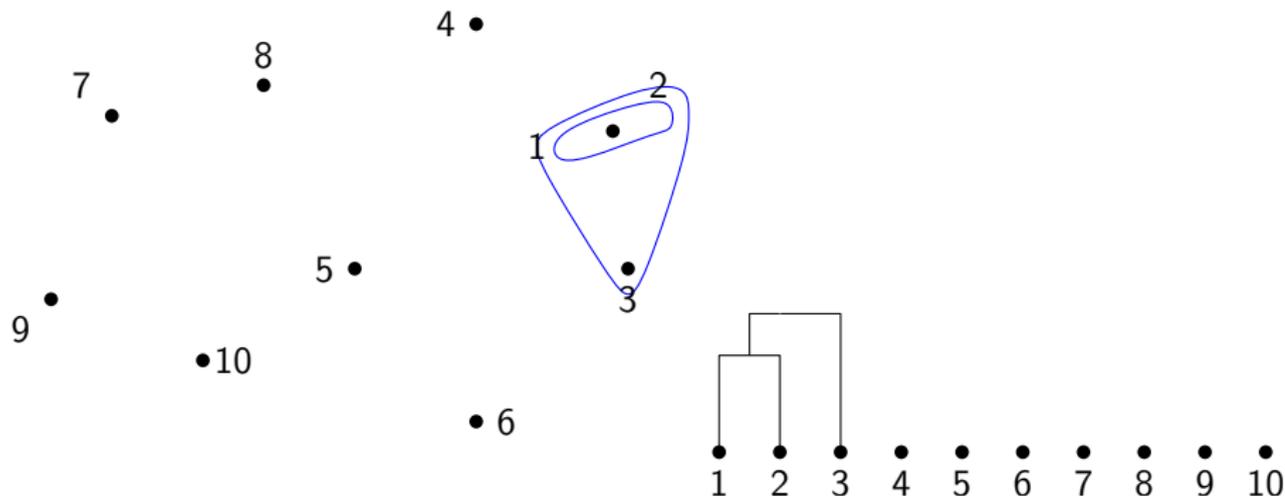
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



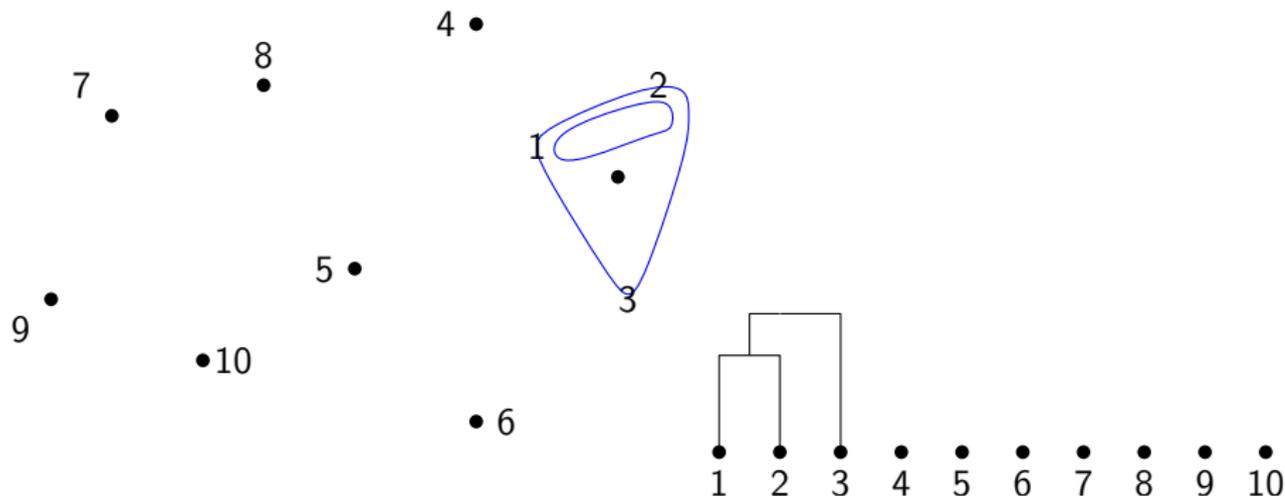
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



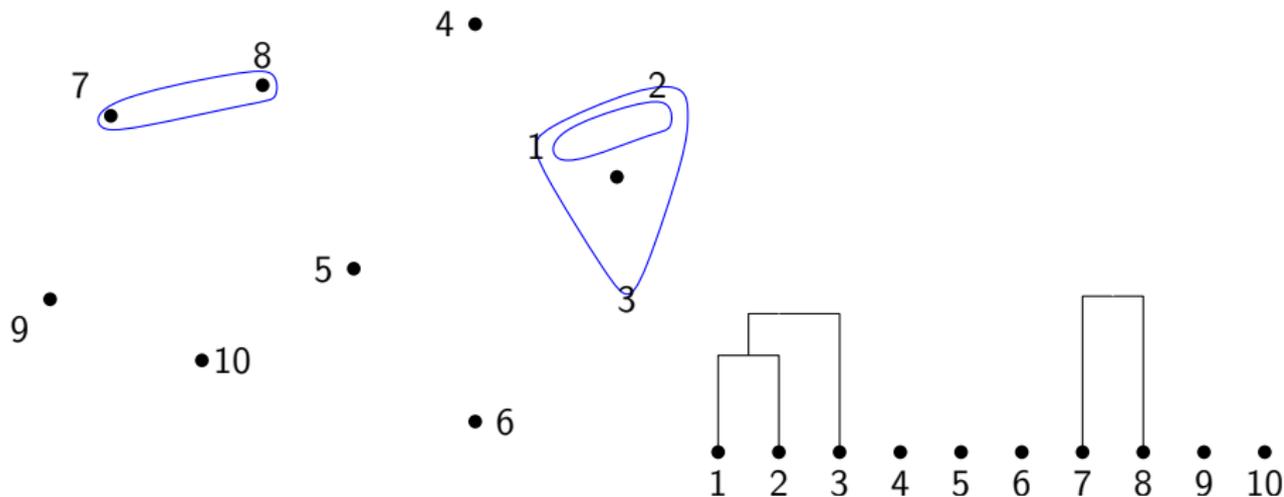
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



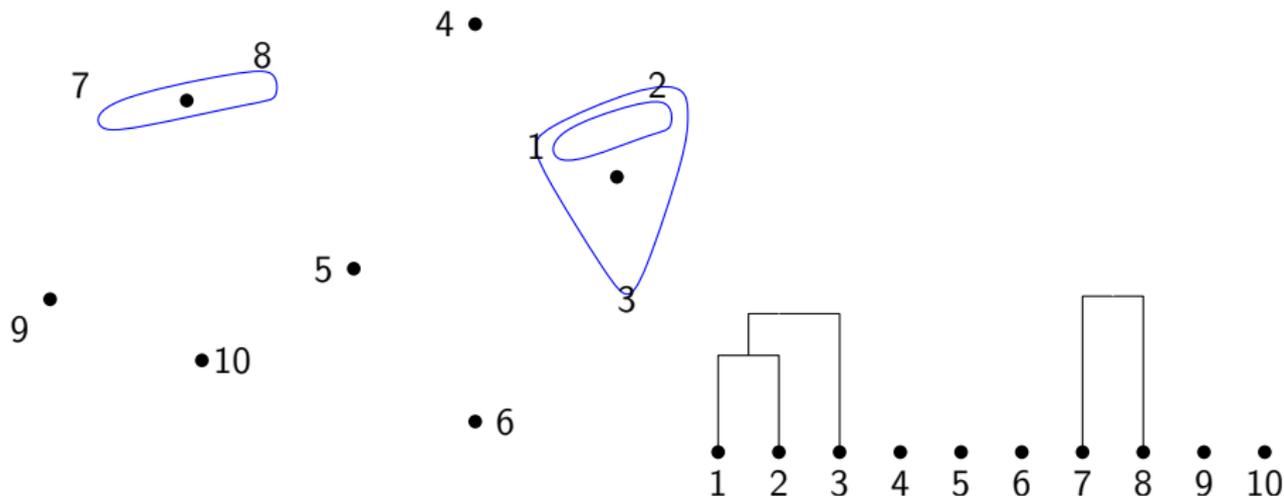
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



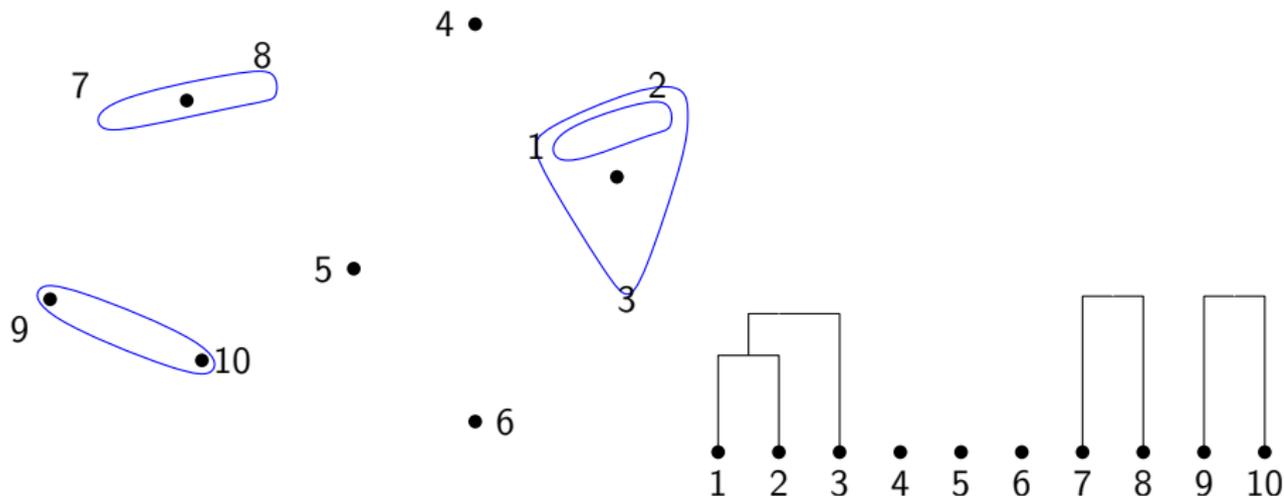
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



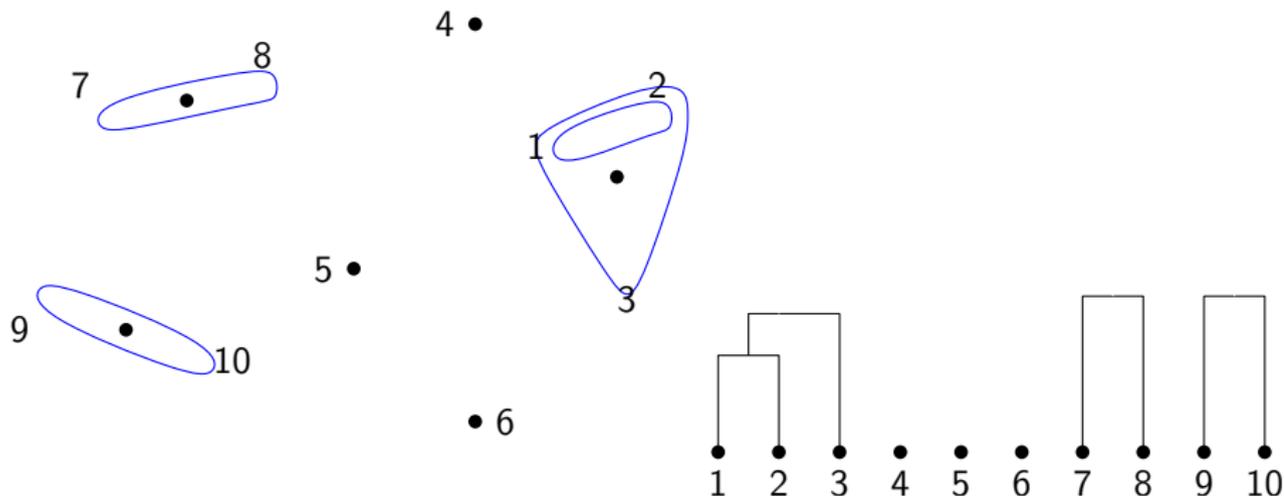
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



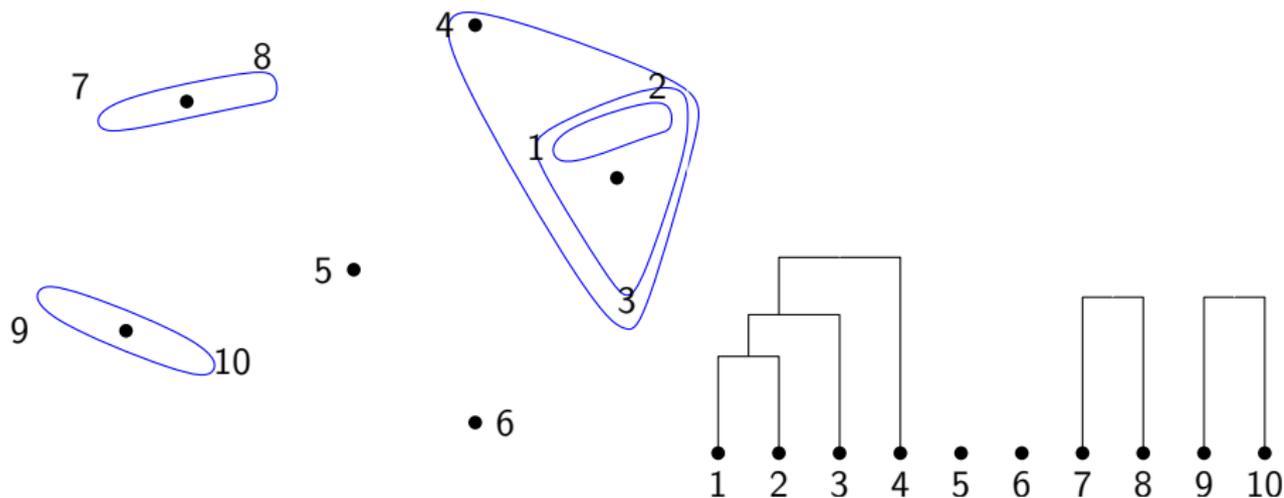
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



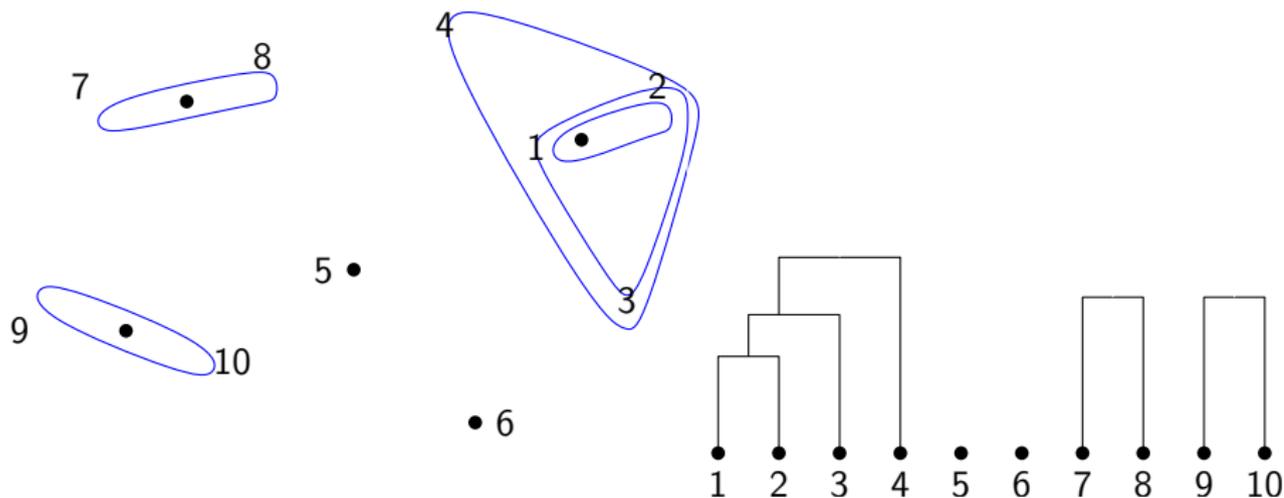
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



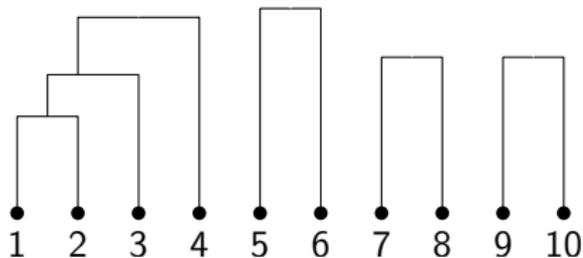
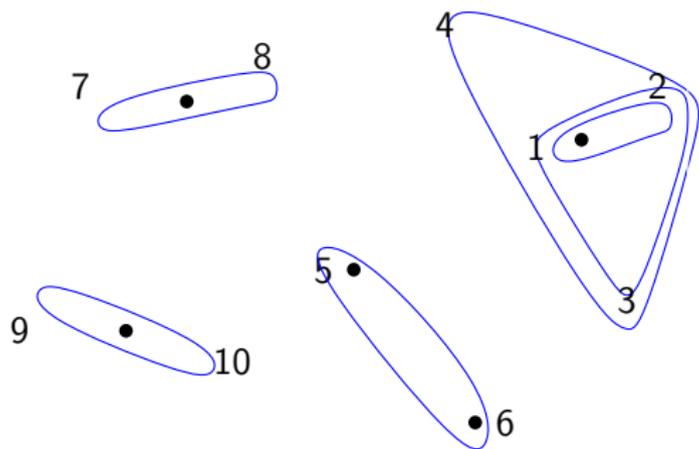
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



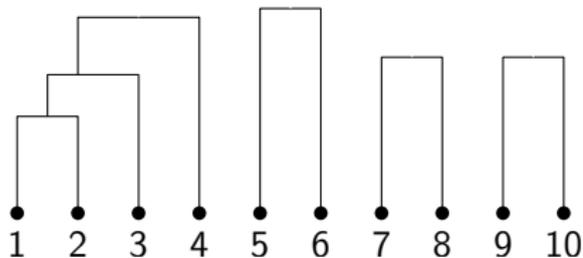
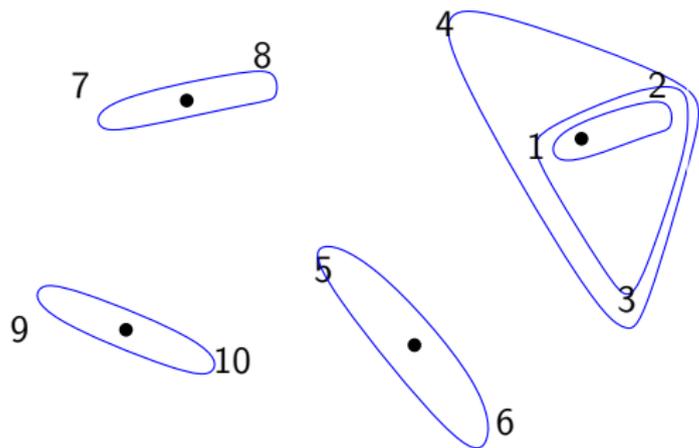
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



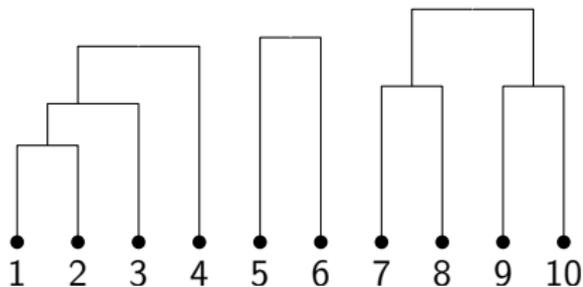
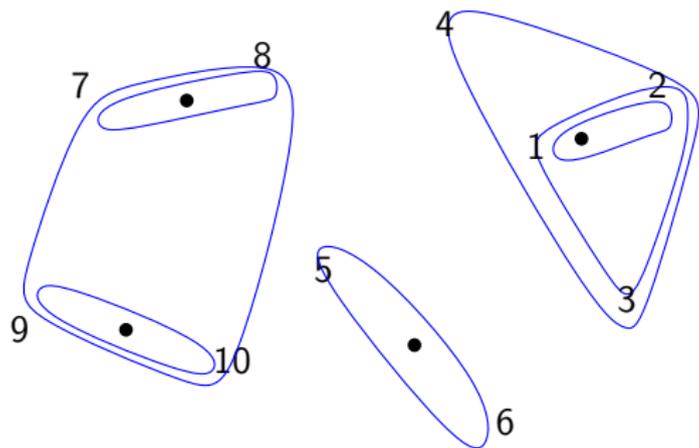
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



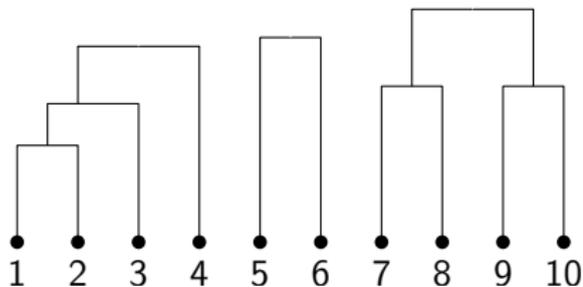
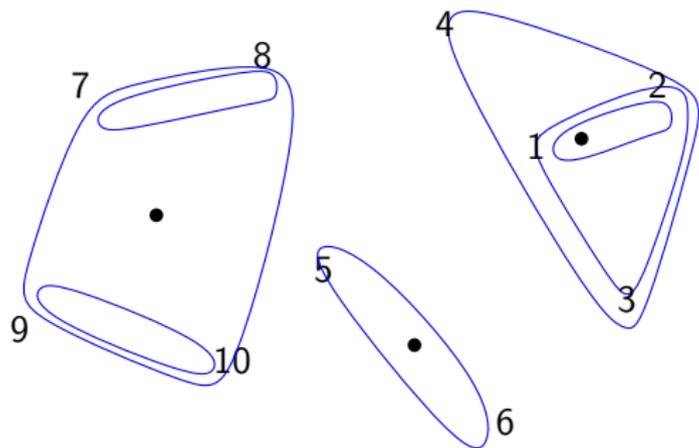
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



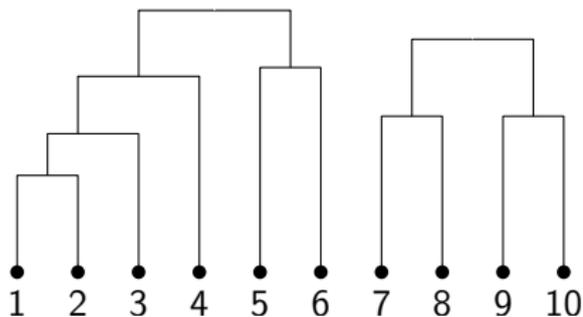
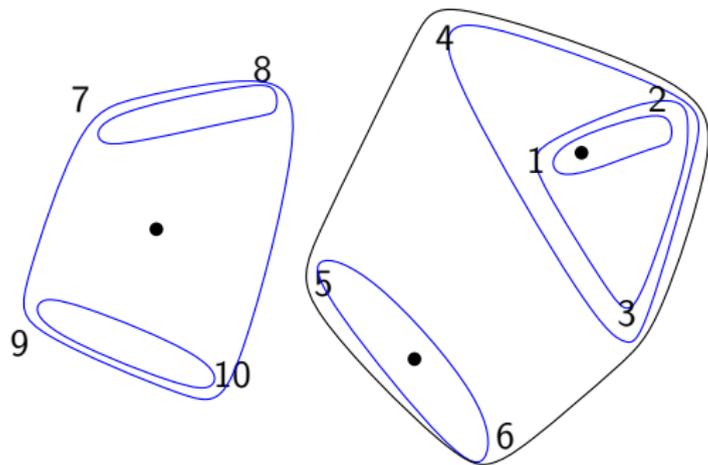
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



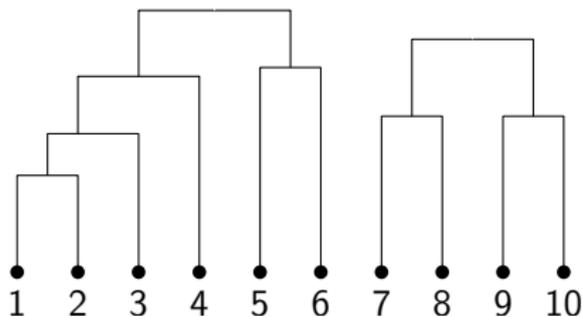
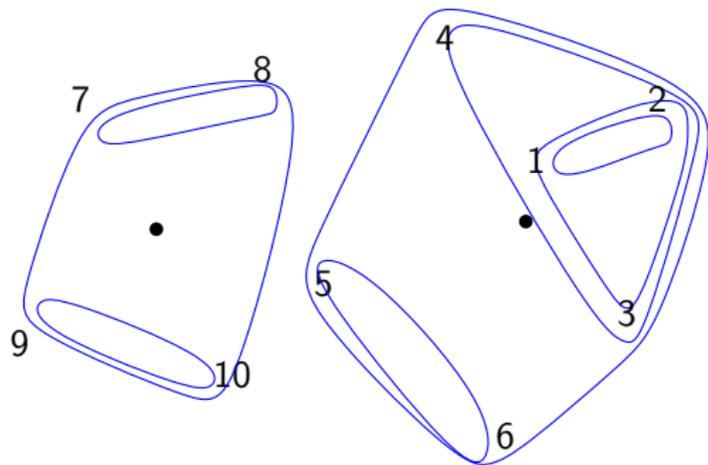
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



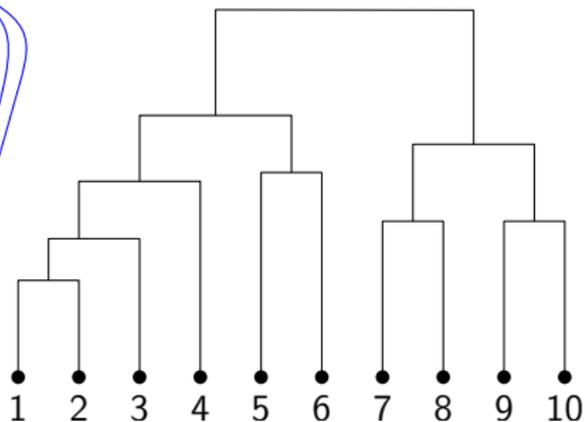
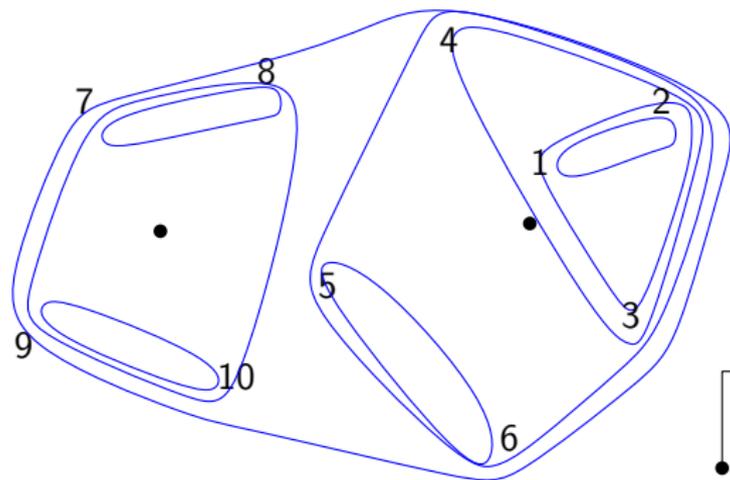
- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

樹形図 (デンドログラム) の作成法



- ▶ 最も近い2つのものをまとめる
- ▶ それらをその重心で置き換える

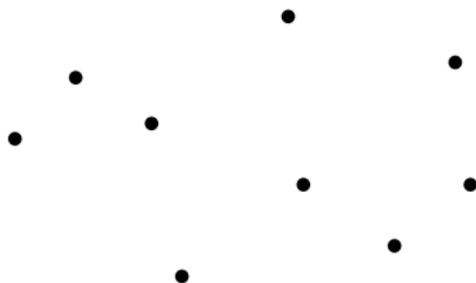
距離を用いた検索



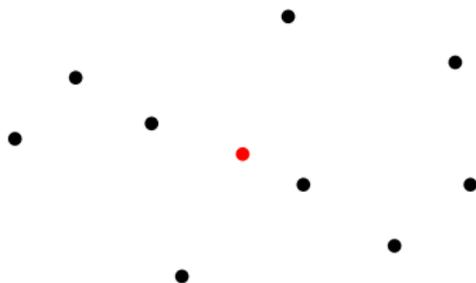
応用例

- ▶ 店舗検索
- ▶ 商品推薦

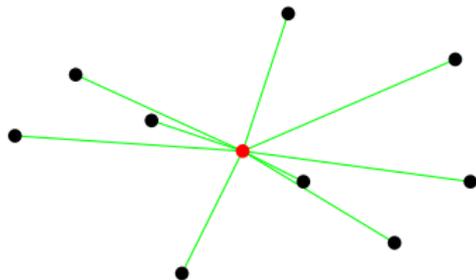
一番近い点を見つけるには、どうすればよいか？



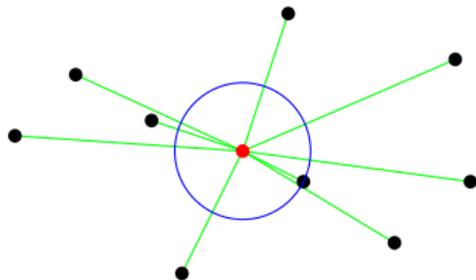
一番近い点を見つけるには、どうすればよいか？



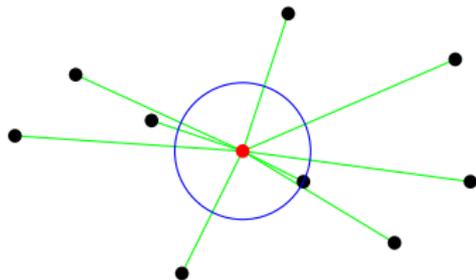
一番近い点を見つけるには、どうすればよいか？



一番近い点を見つけるには、どうすればよいか？



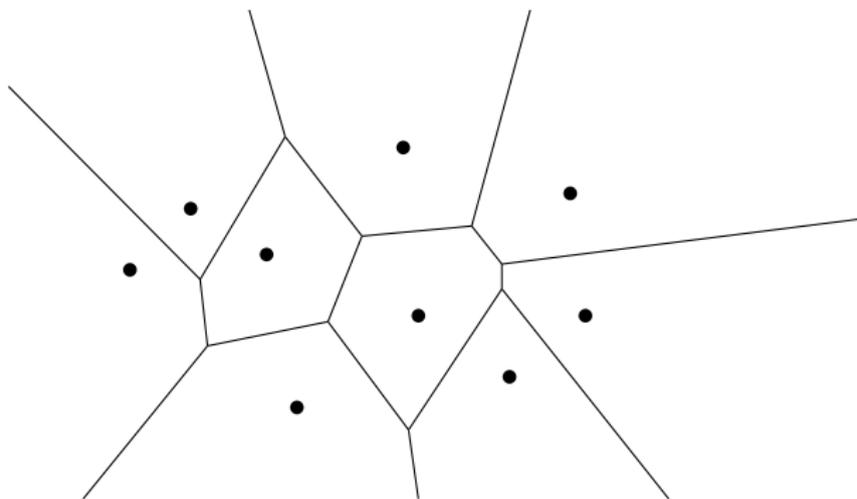
一番近い点を見つけるには、どうすればよいか?: 効率性



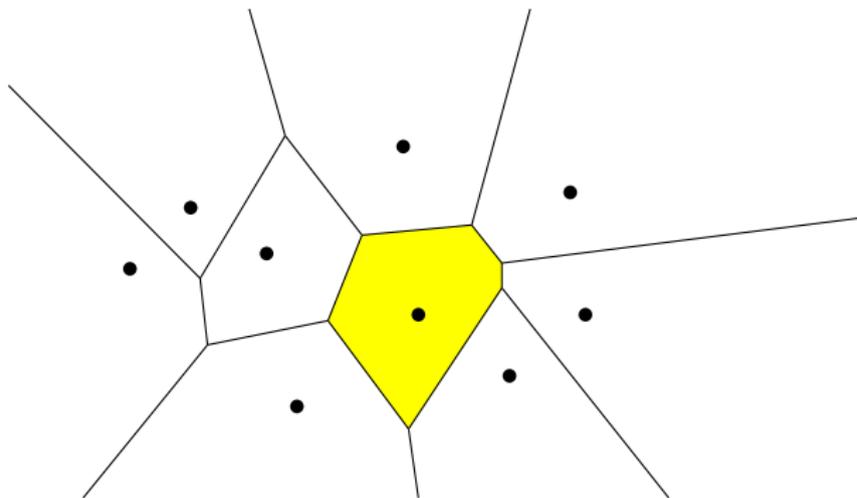
→ すべての点を見る必要がある

($O(n)$ 時間アルゴリズム)

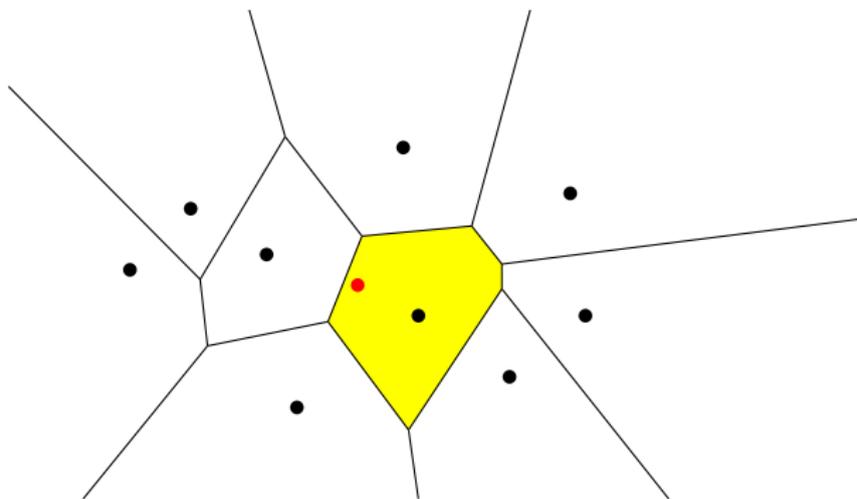
Voronoi 図



Voronoi 図



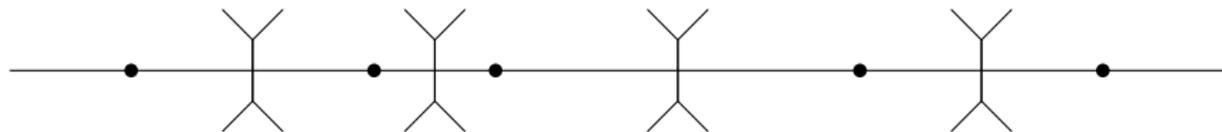
Voronoi 図



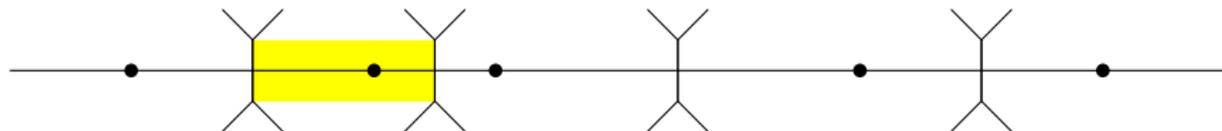
Voronoi 図 : 1 次元の場合



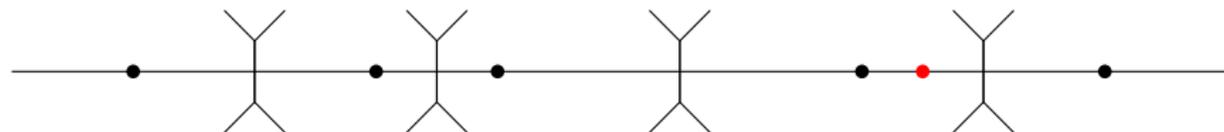
Voronoi 図 : 1 次元の場合



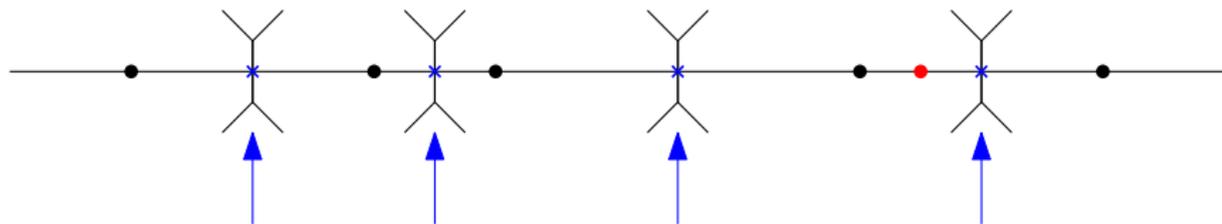
Voronoi 図 : 1 次元の場合



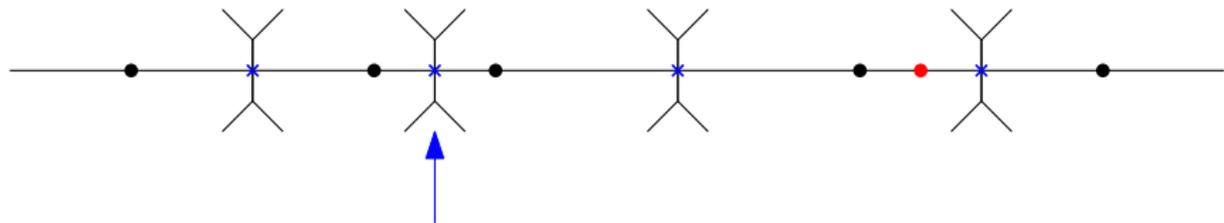
Voronoi 図 : 1 次元の場合



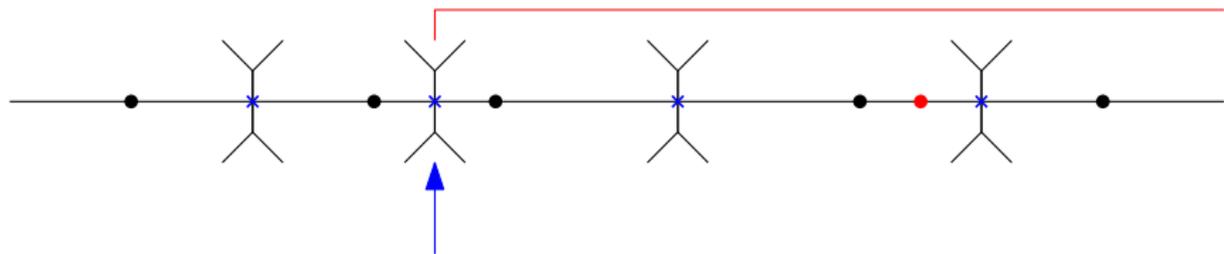
Voronoi 図 : 1 次元の場合



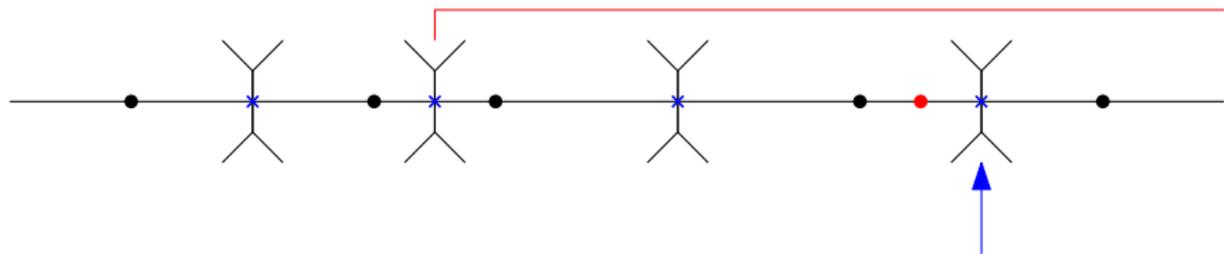
Voronoi 図 : 1 次元の場合



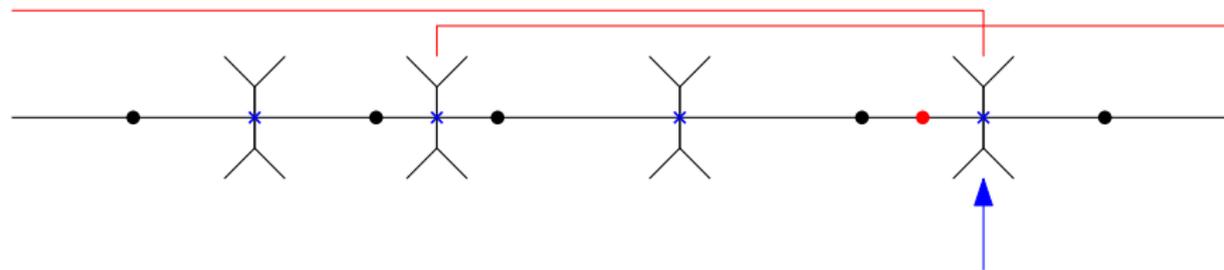
Voronoi 図 : 1 次元の場合



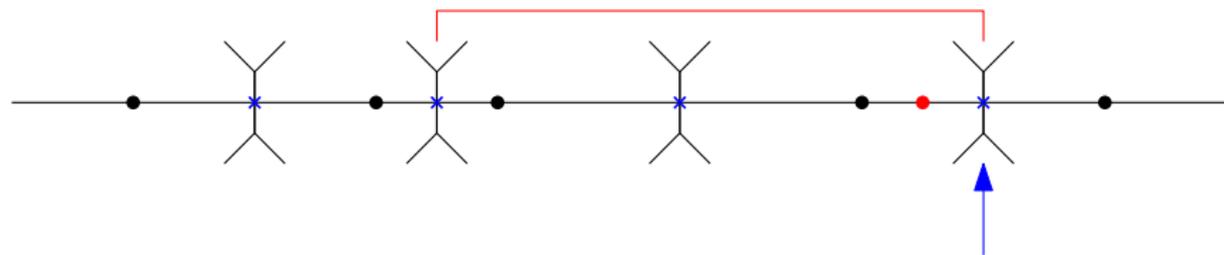
Voronoi 図 : 1 次元の場合



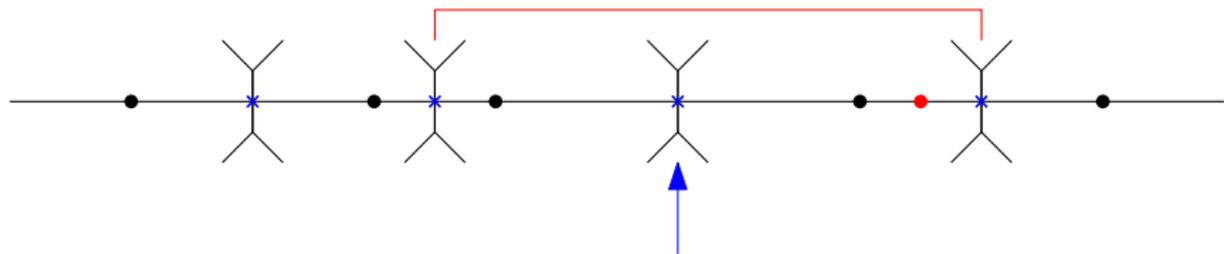
Voronoi 図 : 1 次元の場合



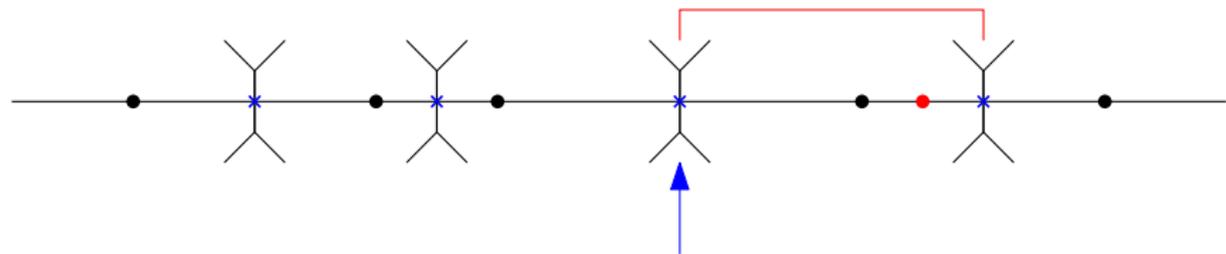
Voronoi 図 : 1 次元の場合



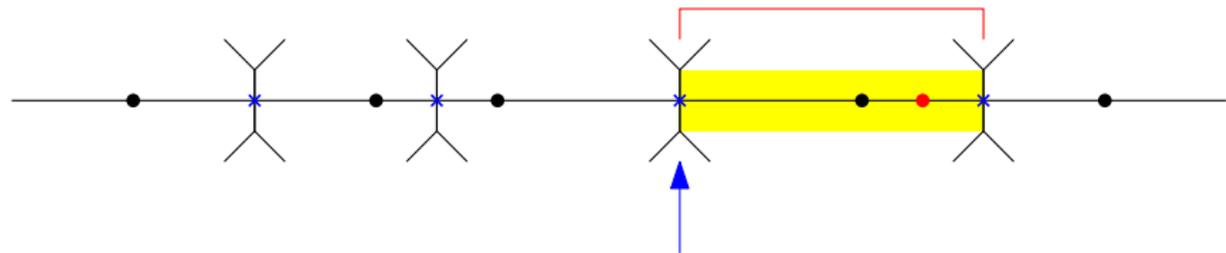
Voronoi 図 : 1 次元の場合



Voronoi 図 : 1 次元の場合



Voronoi 図 : 1 次元の場合

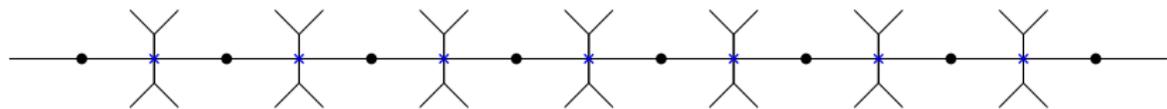


Voronoi 図 : 1 次元の場合



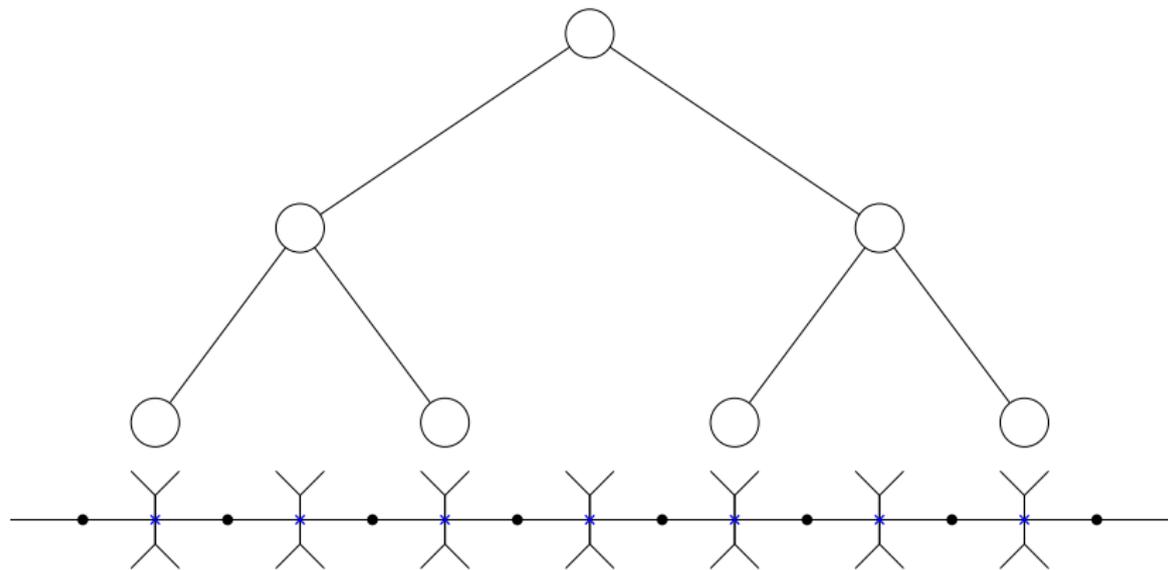
二分探索 ($O(\log n)$ 時間アルゴリズム)

Voronoi 図 : 1 次元の場合

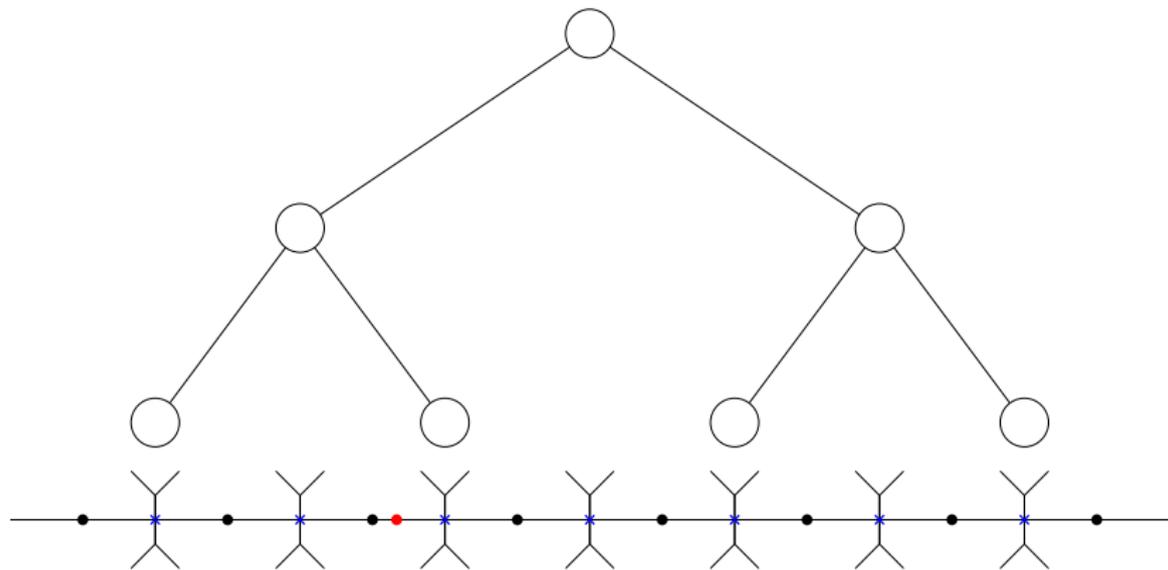


二分探索 ($O(\log n)$ 時間アルゴリズム)

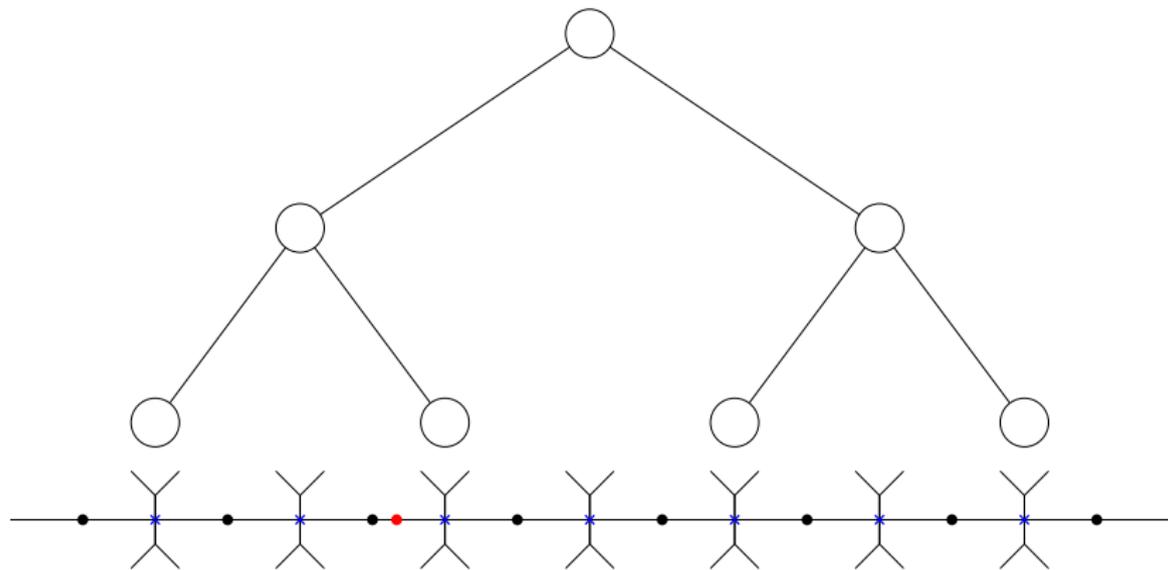
Voronoi 図 : 1 次元の場合

二分探索 ($O(\log n)$ 時間アルゴリズム)

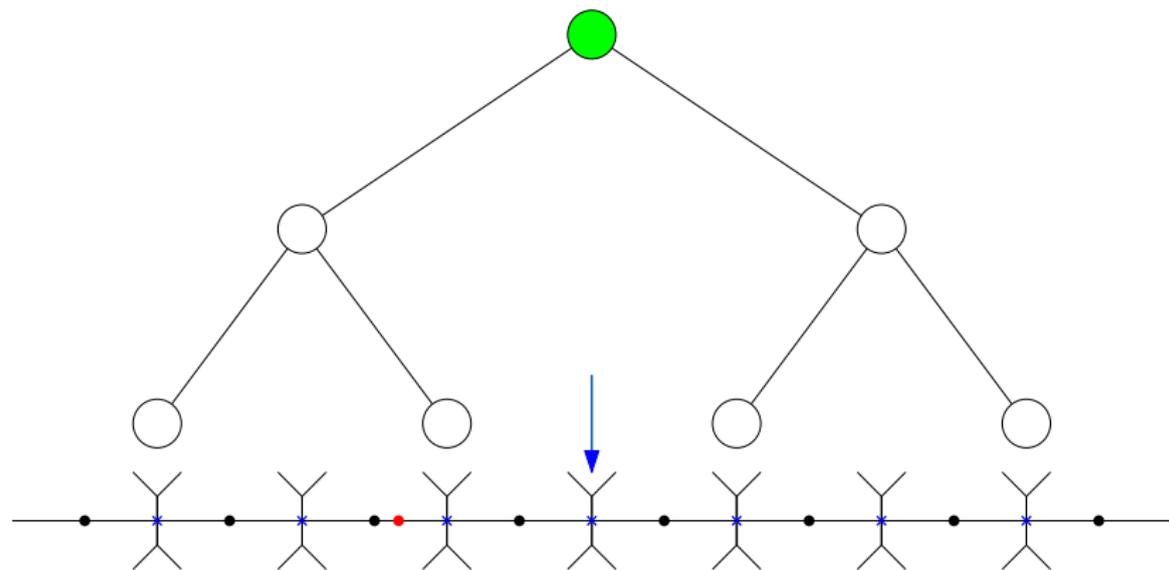
Voronoi 図 : 1 次元の場合

二分探索 ($O(\log n)$ 時間アルゴリズム)

Voronoi 図 : 1 次元の場合

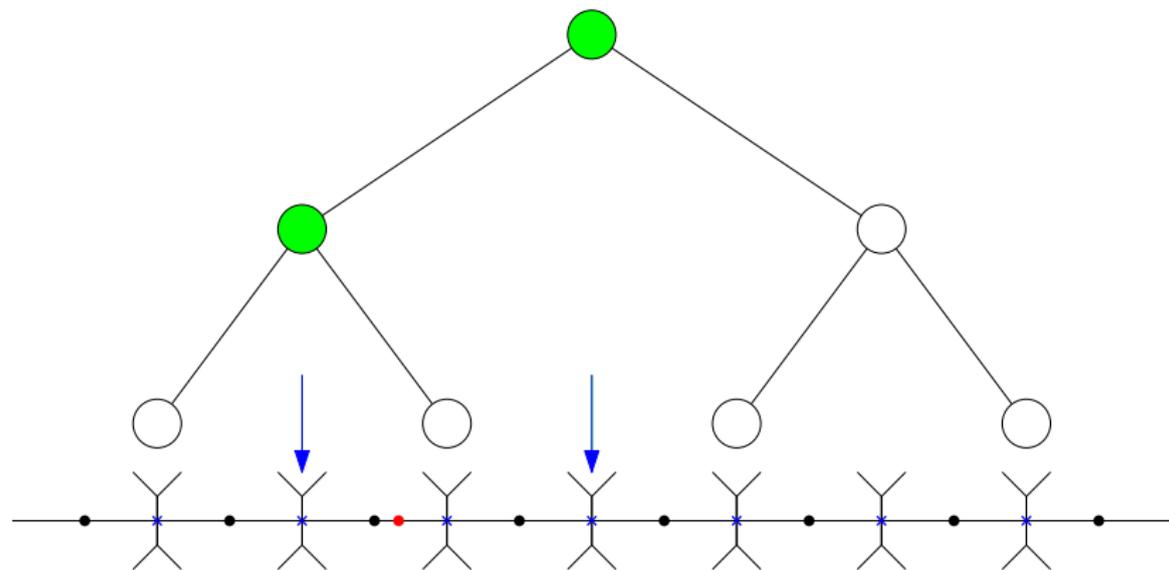
二分探索 ($O(\log n)$ 時間アルゴリズム)

Voronoi 図 : 1次元の場合

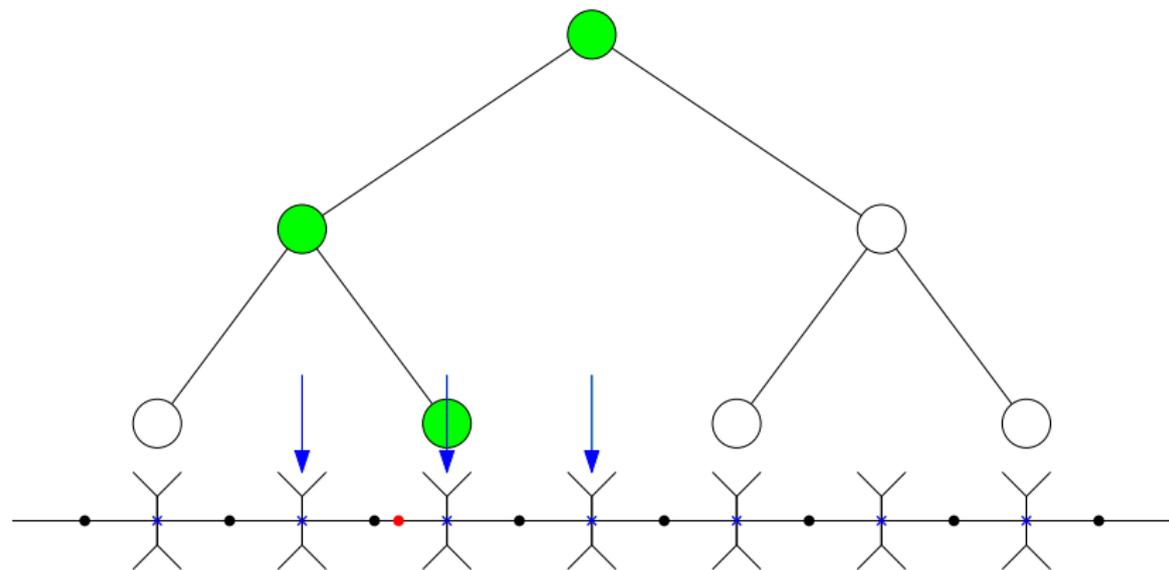


二分探索 ($O(\log n)$ 時間アルゴリズム)

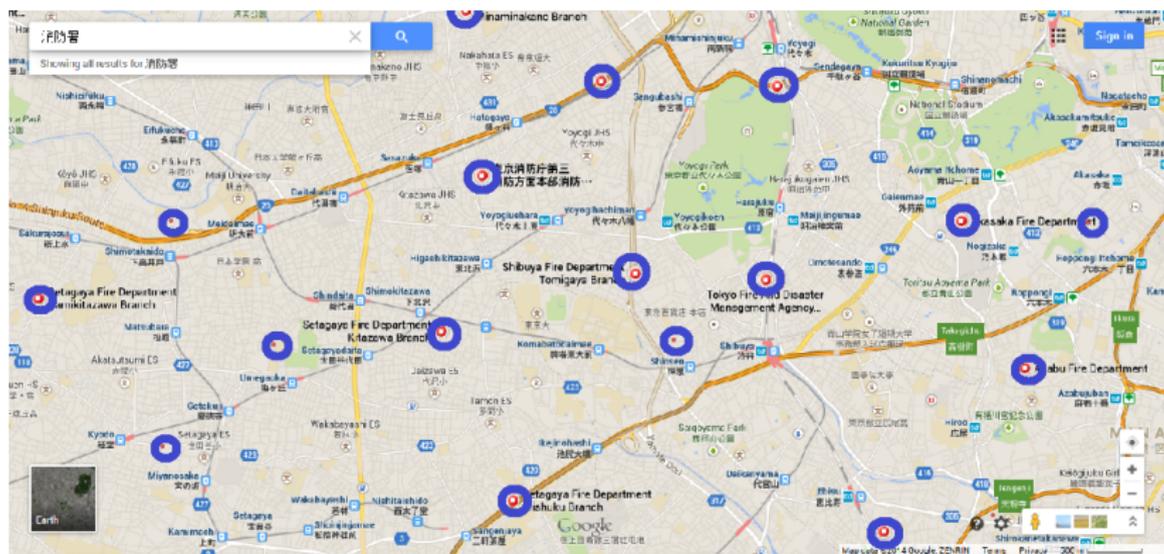
Voronoi 図 : 1 次元の場合

二分探索 ($O(\log n)$ 時間アルゴリズム)

Voronoi 図 : 1次元の場合

二分探索 ($O(\log n)$ 時間アルゴリズム)

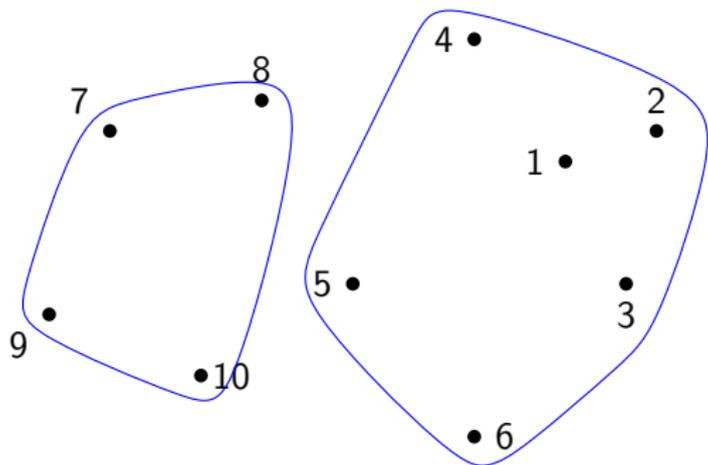
距離を用いた配置



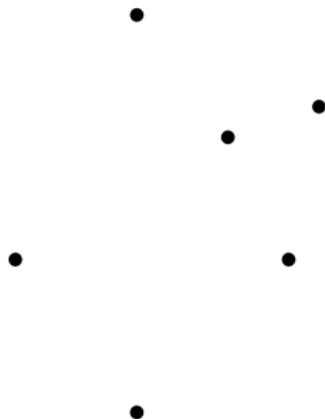
応用例

- ▶ 施設配置
- ▶ 要約

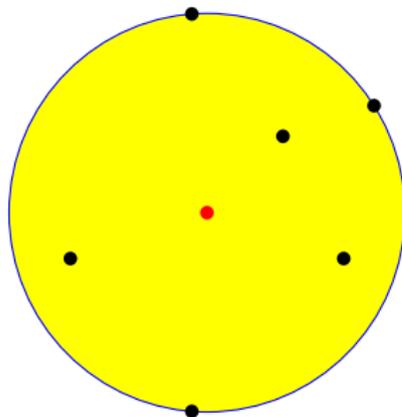
施設配置



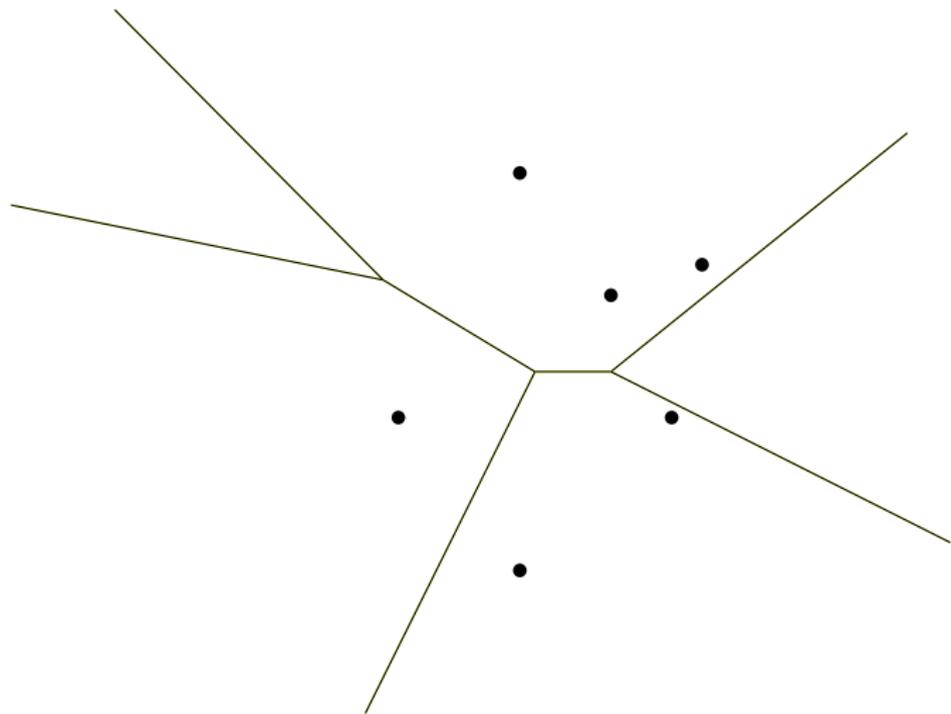
施設配置



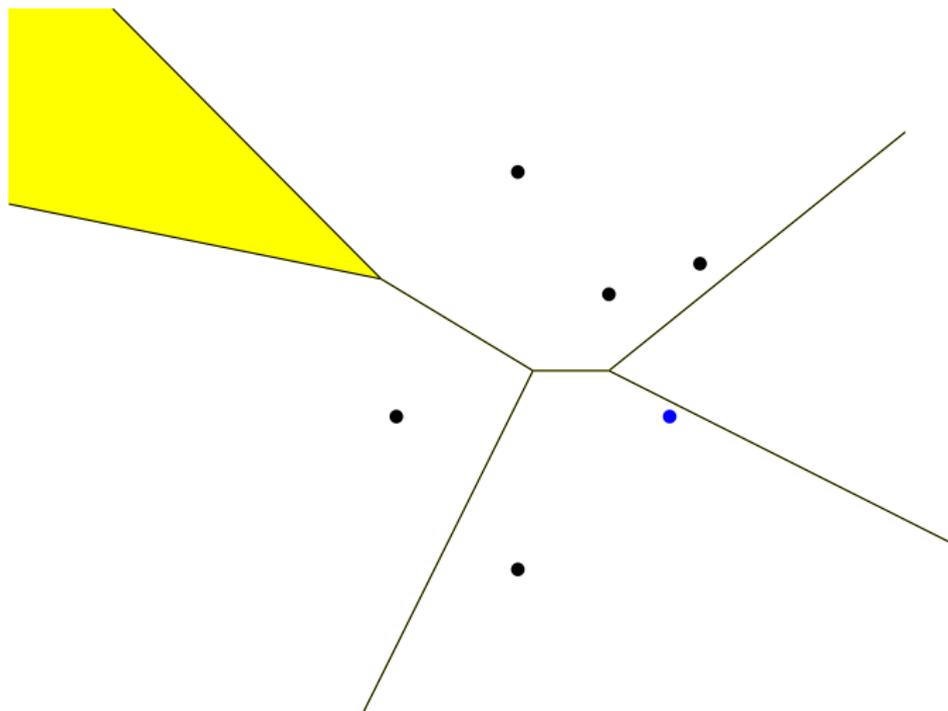
施設配置：最小包含円の中心



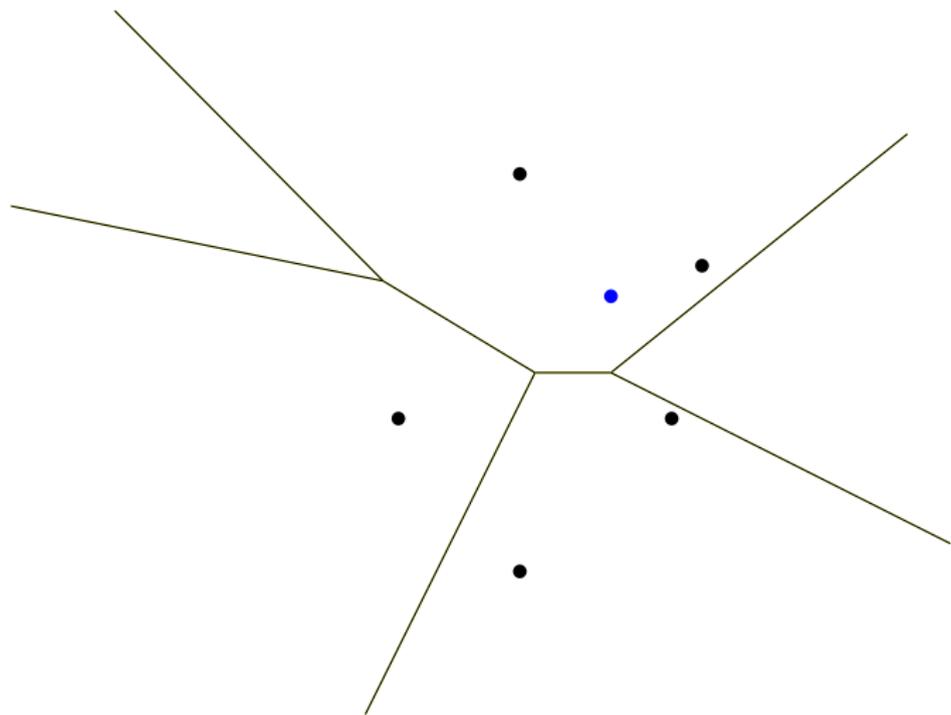
施設配置：最遠点 Voronoi 図



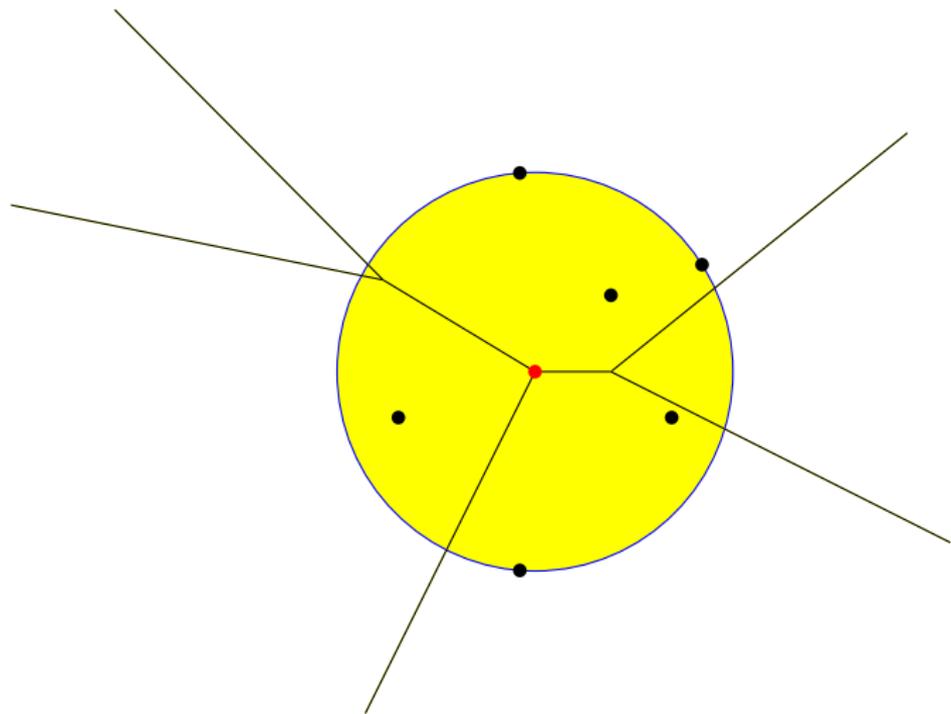
施設配置：最遠点 Voronoi 図



施設配置：最遠点 Voronoi 図



施設配置：最遠点 Voronoi 図と最小包含円



画像認識における要約の応用：固有顔 (eigenface)

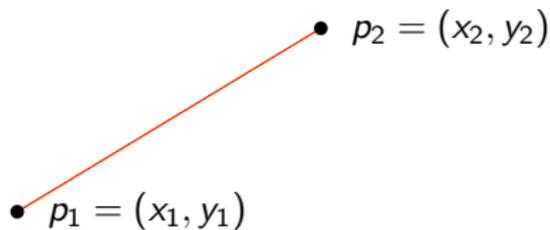


<http://en.wikipedia.org/wiki/Eigenface>

目次

- ① 距離の定義
- ② 距離の応用
 - 分類
 - 検索
 - 配置
- ③ 距離の計算
 - 文字列間の Levenshtein 距離
 - 曲線間の Frechét 距離

ユークリッド距離の計算

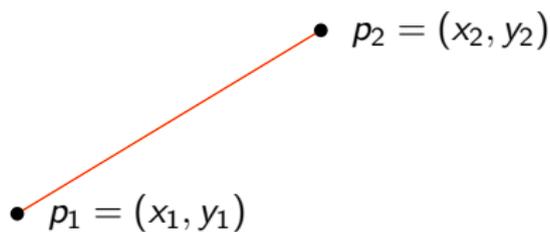


$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

(よく「ユークリッド距離」と呼ばれる)

- ▶ 計算は簡単 \rightsquigarrow \therefore 応用も膨らむ

ユークリッド距離の計算



$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

(よく「ユークリッド距離」と呼ばれる)

- ▶ 計算は簡単 \rightsquigarrow \therefore 応用も膨らむ

計算から見た価値観

簡単に計算できるような距離はよい距離

\rightsquigarrow 距離の計算は簡単にできるのか？

文字列間の Levenshtein 距離

— バイオインフォマティクスでよく使われる

編集に基づく距離

k i t t e n	
s i t t e n	replace
s i t t i n	replace
s i t t i n g	insert

編集操作 :

- ▶ 置換
- ▶ 挿入
- ▶ 削除

Levenshtein 距離 = 一方から他方を得るための編集回数の**最小値**

計算に向けて：漸化式を書いてみる (1)

2つの文字列 S, T の間の Levenshtein 距離は？

Case 1 : S が空の場合

距離 = T の文字数

Case 1' : T が空の場合

距離 = S の文字数

例

▶ S :

▶ T : kitten

S に k, i, t, t, e, n を挿入 \rightsquigarrow 6回の操作が必要十分

計算に向けて：漸化式を書いてみる (2)

2つの文字列 S, T の間の Levenshtein 距離は？

Case 2 : S と T が空でない場合

次の4つの場合を考えないといけない

- ▶ S の最後の文字が T の最後の文字と同じ場合の置換
- ▶ S の最後の文字が T の最後の文字が違う場合の置換
- ▶ S の最後の文字を削除して、 T が得られる場合
- ▶ T の最後の文字を削除して、 S が得られる場合

距離はこの4つの場合の最小値

計算に向けて：漸化式を書いてみる (2-1)

2つの文字列 S, T の間の Levenshtein 距離は？

Case 2-1 : S の最後の文字 = T の最後の文字の場合の置換

距離 = S の最後の文字を取り除いた文字列と
 T の最後の文字を取り除いた文字列の間の距離

例

- ▶ S : kitten
- ▶ T : sittin

最後の n はそのまま \rightsquigarrow kitte と sitti の距離を計算すれば十分

計算に向けて：漸化式を書いてみる (2-2)

2つの文字列 S, T の間の Levenshtein 距離は？

Case 2-2 : S の最後の文字 $\neq T$ の最後の文字の場合の置換

距離 = S の最後の文字を取り除いた文字列と
 T の最後の文字を取り除いた文字列の間の距離 + 1

例

- ▶ S : kitten
- ▶ T : sittig

最後の n を g に置換

↪ あとは、kitte と sitti の距離を計算すれば十分

計算に向けて：漸化式を書いてみる (2-3)

2つの文字列 S, T の間の Levenshtein 距離は？

Case 2-3 : S に対する削除で T が得られる

距離 = S の最後の文字を取り除いた文字列と T の間の距離 +1

例

- ▶ S : kitten
- ▶ T : sittig

最後の n を削除

↪ あとは, kitte と sittig の距離を計算すれば十分

計算に向けて：漸化式を書いてみる (2-4)

2つの文字列 S, T の間の Levenshtein 距離は？

Case 2-4 : T に対する削除で S が得られる

距離 = T の最後の文字を取り除いた文字列と S の間の距離 +1

例

- ▶ S : kitten
- ▶ T : sittig

最後の g を削除

↪ あとは、kitten と sittii の距離を計算すれば十分

Levenshtein 距離：漸化式のまとめ

文字列 S と T の Levenshtein 距離 $d(S, T)$ は次を満たす

S か T が空のとき

$$d(S, T) = \max\{|S|, |T|\}$$

S も T も空ではないとき

S と T の最後の文字が同じ場合

$$d(S, T) = \min\{d(S', T'), d(S, T') + 1, d(S', T) + 1\}$$

S と T の最後の文字が異なる場合

$$d(S, T) = \min\{d(S', T') + 1, d(S, T') + 1, d(S', T) + 1\}$$

- ▶ $|S| = S$ の文字数, $|T| = T$ の文字数
- ▶ $S' = S$ の最後の文字を取り除いた文字列
- ▶ $T' = T$ の最後の文字を取り除いた文字列

漸化式を利用した計算

$$d(\text{kitten}, \text{sitting}) = \min\{d(\text{kitte}, \text{sittin}) + 1, \\ d(\text{kitten}, \text{sittin}) + 1, \\ d(\text{kitte}, \text{sitting}) + 1\}$$

漸化式を利用した計算

$$\begin{aligned}d(\text{kitten}, \text{sitting}) &= \min\{d(\text{kitte}, \text{sittin}) + 1, \\ &\quad d(\text{kitten}, \text{sittin}) + 1, \\ &\quad d(\text{kitte}, \text{sitting}) + 1\} \\ &= \min\{\min\{d(\text{kitt}, \text{sitti}) + 2, \\ &\quad d(\text{kitte}, \text{sitti}) + 2, \\ &\quad d(\text{kitt}, \text{sittin}) + 2\}, \\ &\quad \min\{d(\text{kitte}, \text{sitti}) + 1, \\ &\quad d(\text{kitten}, \text{sitti}) + 2, \\ &\quad d(\text{kitte}, \text{sittin}) + 2\}, \\ &\quad \min\{d(\text{kitt}, \text{sittin}) + 2, \\ &\quad d(\text{kitt}, \text{sitting}) + 2, \\ &\quad d(\text{kitte}, \text{sittin}) + 2\}\}\end{aligned}$$

漸化式を利用した計算

$$\begin{aligned}
 d(\text{kitten}, \text{sitting}) &= \min\{d(\text{kitte}, \text{sittin}) + 1, \\
 &\quad d(\text{kitten}, \text{sittin}) + 1, \\
 &\quad d(\text{kitte}, \text{sitting}) + 1\} \\
 &= \min\{\min\{d(\text{kitt}, \text{sitti}) + 2, \\
 &\quad d(\text{kitte}, \text{sitti}) + 2, \\
 &\quad d(\text{kitt}, \text{sittin}) + 2\}, \\
 &\quad \min\{d(\text{kitte}, \text{sitti}) + 1, \\
 &\quad d(\text{kitten}, \text{sitti}) + 2, \\
 &\quad d(\text{kitte}, \text{sittin}) + 2\}, \\
 &\quad \min\{d(\text{kitt}, \text{sittin}) + 2, \\
 &\quad d(\text{kitt}, \text{sitting}) + 2, \\
 &\quad d(\text{kitte}, \text{sittin}) + 2\}\} \\
 &= \dots
 \end{aligned}$$

- ▶ 式がどんどん膨れ上がっていく
 \rightsquigarrow 計算量の爆発
- ▶ 同じ計算が何度も現れる

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
s							
i							
t							
t							
i							
n							
g							

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s							
i							
t							
t							
i							
n							
g							

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1						
i	2						
t	3						
t	4						
i	5						
n	6						
g	7						

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1					
i	2						
t	3						
t	4						
i	5						
n	6						
g	7						

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2				
i	2						
t	3						
t	4						
i	5						
n	6						
g	7						

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3			
i	2						
t	3						
t	4						
i	5						
n	6						
g	7						

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4		
i	2						
t	3						
t	4						
i	5						
n	6						
g	7						

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	
i	2						
t	3						
t	4						
i	5						
n	6						
g	7						

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2						
t	3						
t	4						
i	5						
n	6						
g	7						

Levenstein 距離の計算は $O(mn)$ 時間で行える

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3						
t	4						
i	5						
n	6						
g	7						

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

表を利用した再計算の抑制 — 動的計画法と呼ばれる手法

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

Levenstein 距離の計算は $O(mn)$ 時間でできる

($m = S$ の文字数, $n = T$ の文字数)

目次

① 距離の定義

② 距離の応用

分類

検索

配置

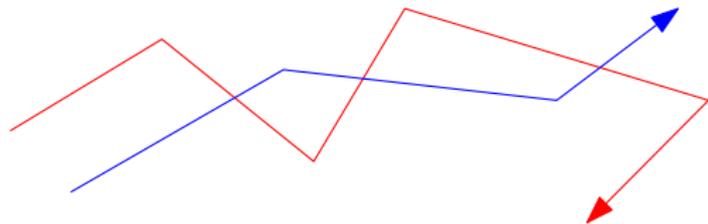
③ 距離の計算

文字列間の Levenshtein 距離

曲線間の Frechét 距離

曲線間の Frechét 距離

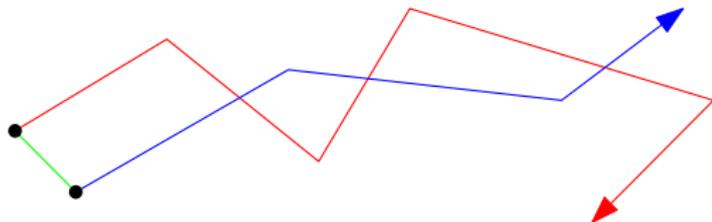
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

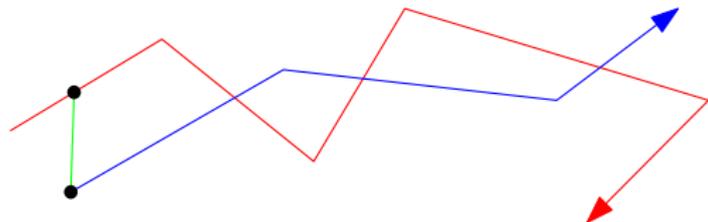
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

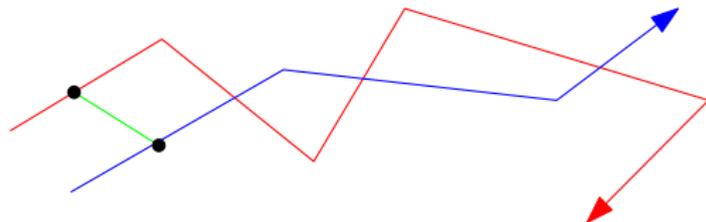
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

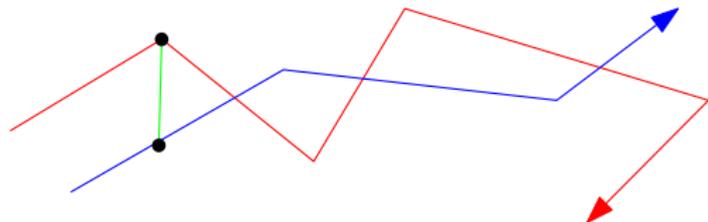
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

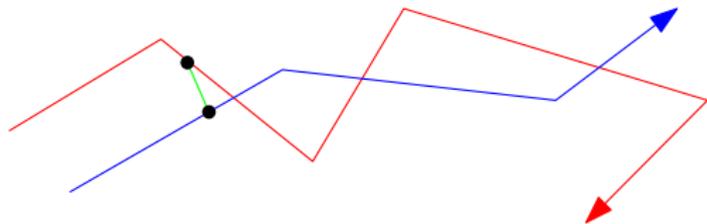
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

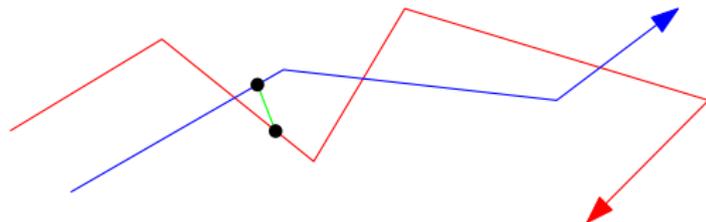
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

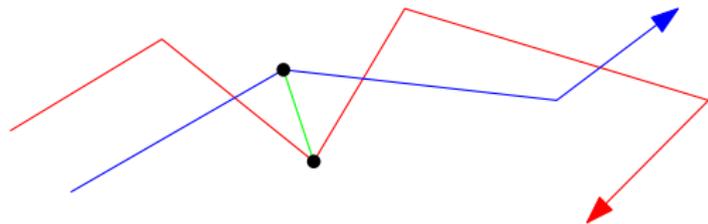
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

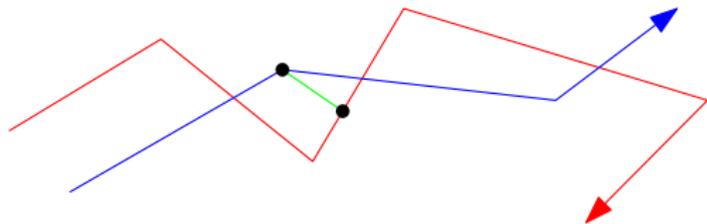
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

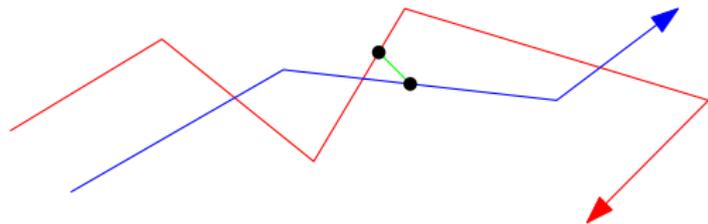
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

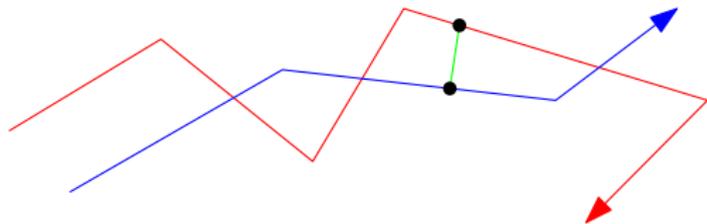
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

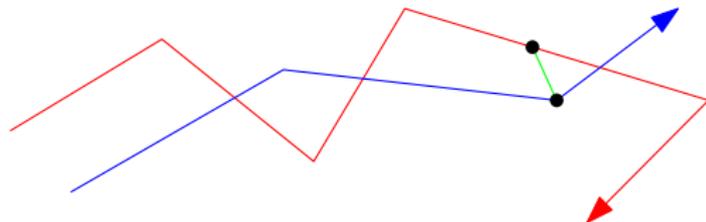
曲線上を単調に (逆戻りせずに) 歩く



Frechet 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離

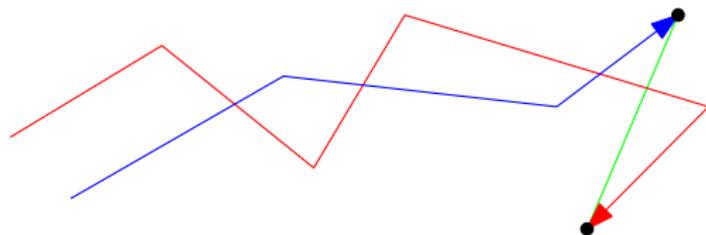
曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

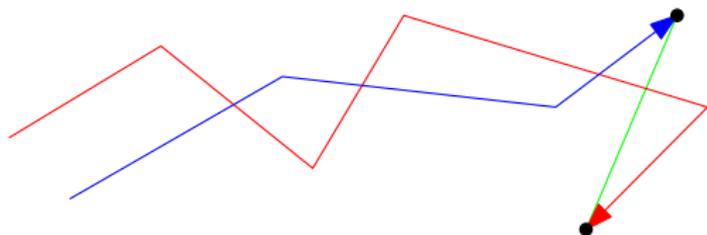
曲線間の Frechét 距離

曲線上を単調に (逆戻りせずに) 歩く



Frechét 距離 = すべての歩き方における
道中における 2 点の最大ユークリッド距離の最小値

曲線間の Frechét 距離の計算法

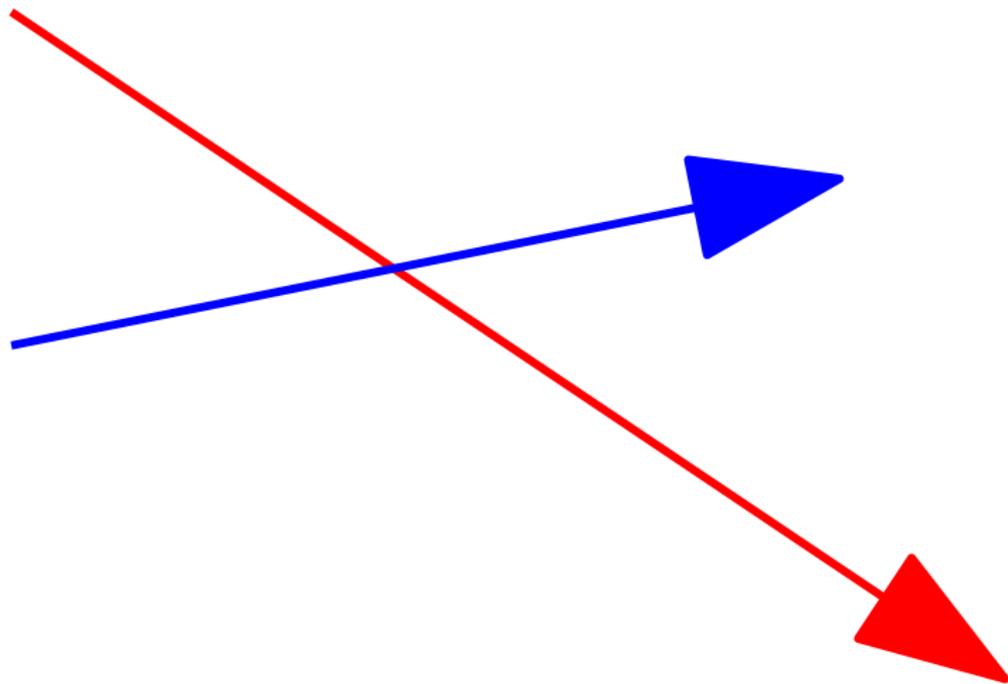


Frechet 距離をどのように計算すればよいのか？

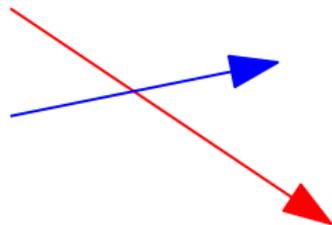
「歩き方」が無限にあるので、「すべての場合を尽くす」ことが不可

まずは、「Frechet 距離がある値 ε 以下か？」という質問に
正しく答えられるようにする

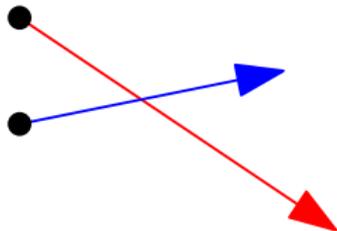
Free space diagram : 簡単な例



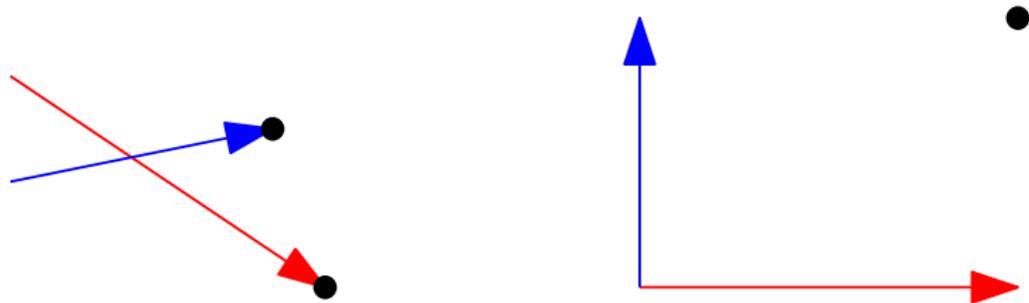
Free space diagram : 簡単な例



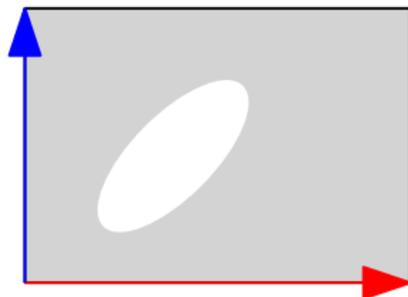
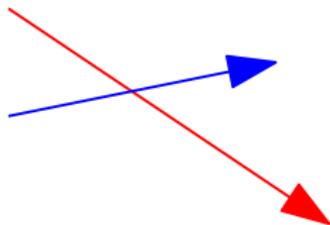
Free space diagram : 簡単な例



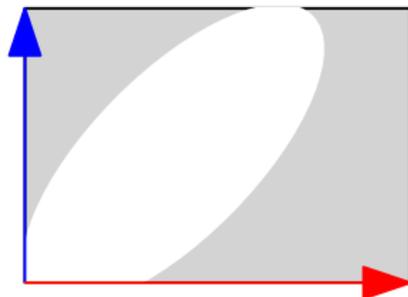
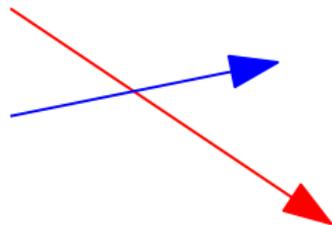
Free space diagram : 簡単な例



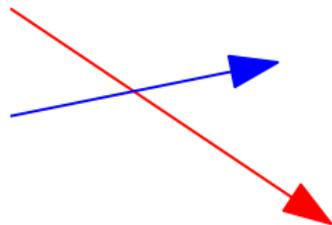
Free space diagram : 簡単な例



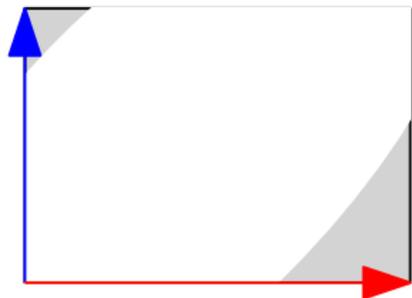
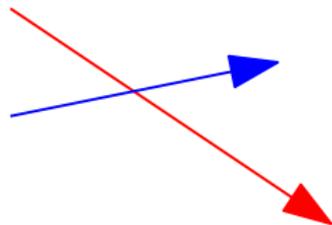
Free space diagram : 簡単な例



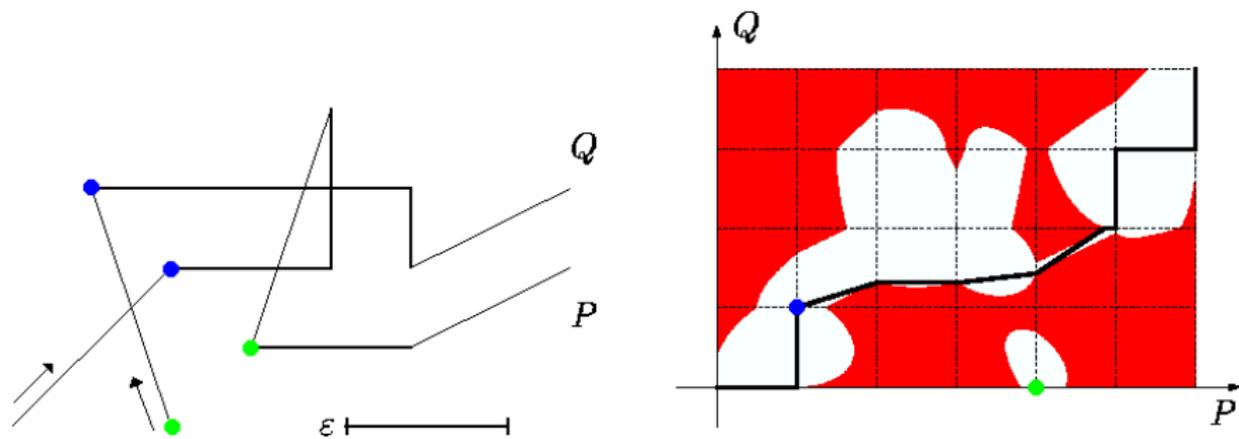
Free space diagram : 簡単な例



Free space diagram : 簡単な例



Free space diagram : 全体像



<http://www.cim.mcgill.ca/~stephane/cs507/Project.html>

単調な経路が存在するかどうかは、動的計画法で確認できる

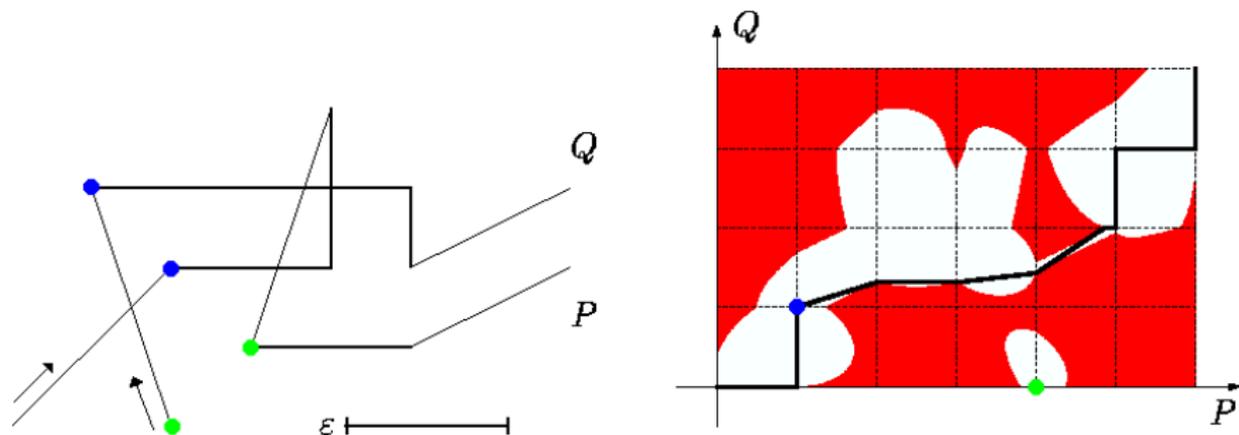
Alt & Godau ('95)

2 曲線間の Frechét 距離が ϵ 以下であるかどうかの判定は

$O(n^2)$ 時間でできる

($n = 2$ つの曲線を構成する線分の数)

Free space diagram : 全体像



<http://www.cim.mcgill.ca/~stephane/cs507/Project.html>

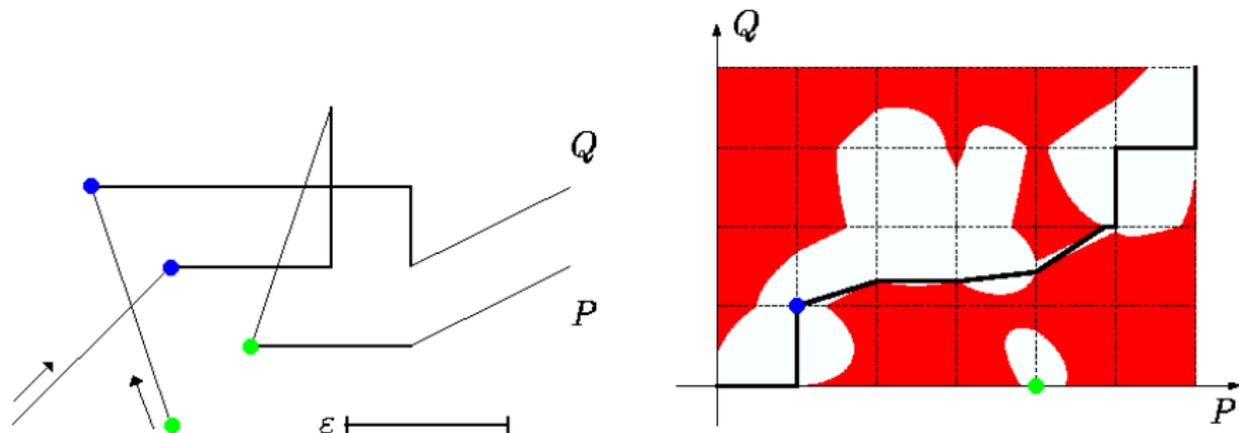
前頁の判定アルゴリズムに parametric search を組み合わせて次を得る

Alt & Godau ('95)

2 曲線間の Frechét 距離の計算は $O(n^2 \log n)$ 時間で行える

($n = 2$ つの曲線を構成する線分の数)

Free space diagram : 全体像 — 最近の結果



<http://www.cim.mcgill.ca/~stephane/cs507/Project.html>

Buchin, Buchin, Meulemans, Mulzer ('14)

2 曲線間の Frechét 距離の計算は $O(n^2 \log^2 \log n / \log n)$ 時間で行える
 ($n = 2$ つの曲線を構成する線分の数)

この 2 コマの目標

「距離」にまつわる話

- ▶ 距離とは何か？ (定義)
- ▶ 距離をどう使うのか？ (応用)
- ▶ 距離をどう計算するのか？ (計算)