何でも微分する

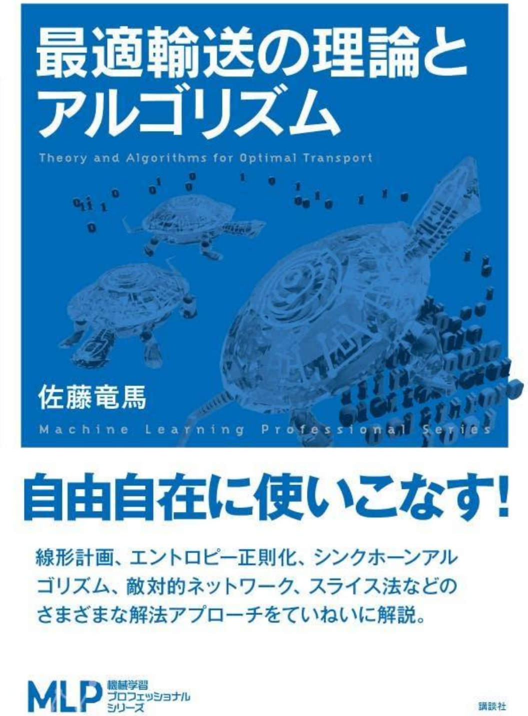
Differentiate Everything

佐藤竜馬

KYOTO UNIVERSITY

自己紹介

- 名前：佐藤　竜馬（さとう　りょうま）
- 京都大学 博士課程 3 年生



好評発売中！

様々な操作を微分し連続的に最適化する方法を学ぶ

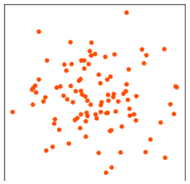
- 様々な離散的な操作や最適化を微分する方法を学びます。
  - 最適輸送
  - ソート、ランキング
  - 最短経路問題　　　など
- これらが微分できるようになると
  - 輸送コストが最小となる配置を求める
  - 真のラベルが top-K に入る確率を最大化する

を勾配法ベースの連続最適化により解くことができます。

最適輸送は重み付き点群を比較するツール

- 最適輸送：重み付き点群を輸送コストを基に比較するツール

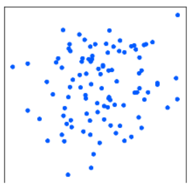
- 入力：



$a \in \mathbb{R}_+^n$: 点群 A の各点の重み

$X \in \mathbb{R}^{n \times d}$: 点群 A の各点の位置

例：ソースデータの集合
（一つの点が 1 データ）



$b \in \mathbb{R}_+^m$: 点群 B の各点の重み

$Y \in \mathbb{R}^{m \times d}$: 点群 B の各点の位置

例：ターゲットデータの集合
（一つの点が 1 データ）

- 出力：

例：ソースとターゲットの乖離度

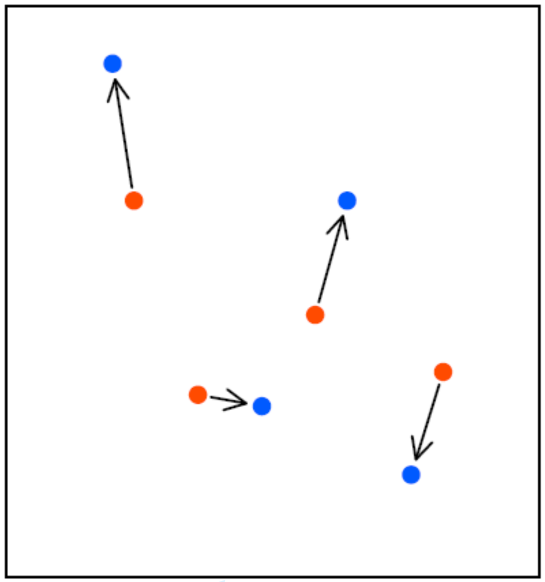
$d(a, X, b, Y) \in \mathbb{R}$: 点群 A と点群 B の距離（スカラー）

$P(a, X, b, Y) \in \mathbb{R}^{n \times \mathrm{m}}$: 点群 A と点群 B の割り当て

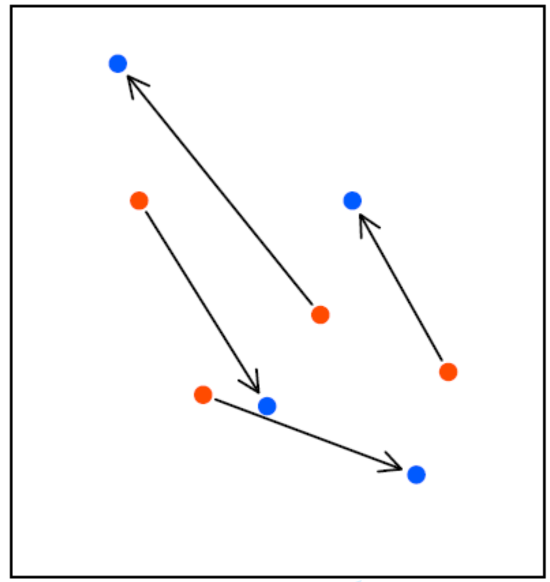
最適輸送は最適な輸送における移動コストを測る

- 最適輸送のイメージ図

  点群 A と点群 B の距離（違いの大きさ）を測りたい。



最適な輸送
このときの移動コストで距離を測る



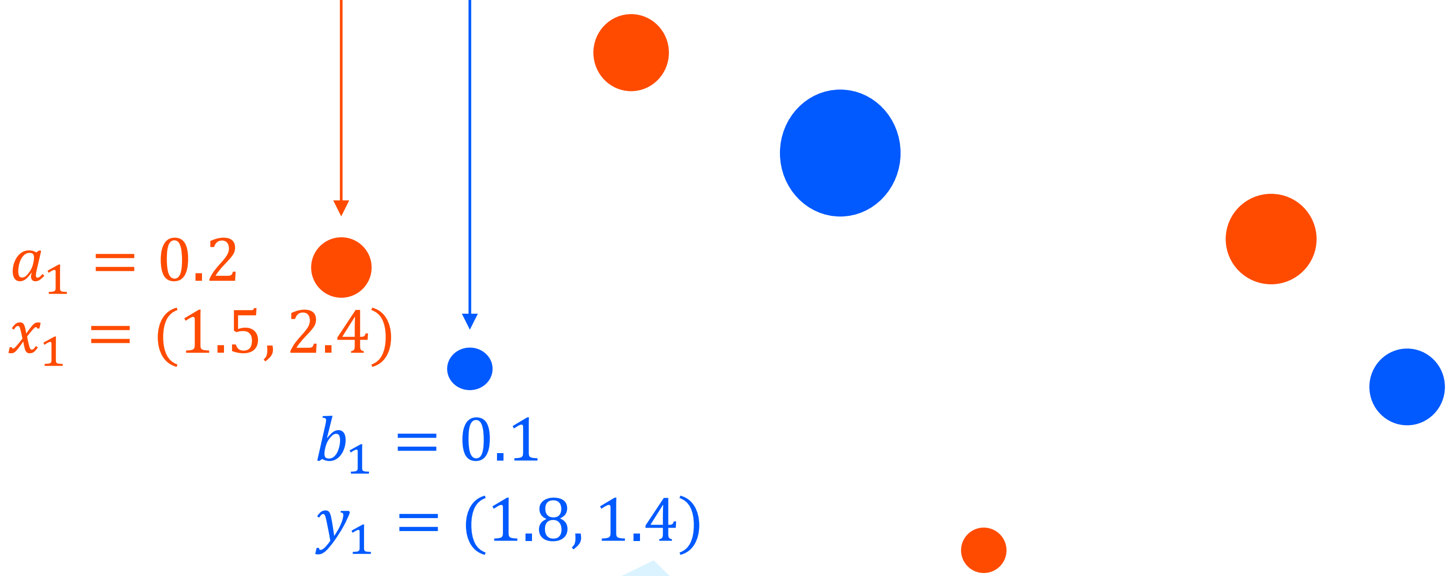
最適でない輸送

最適輸送の入力例

- 入力例：

点の大きさ（重み）

$a = (0.2, 0.3, 0.4, 0.1)$　　$b = (0.1, 0.6, 0.3)$

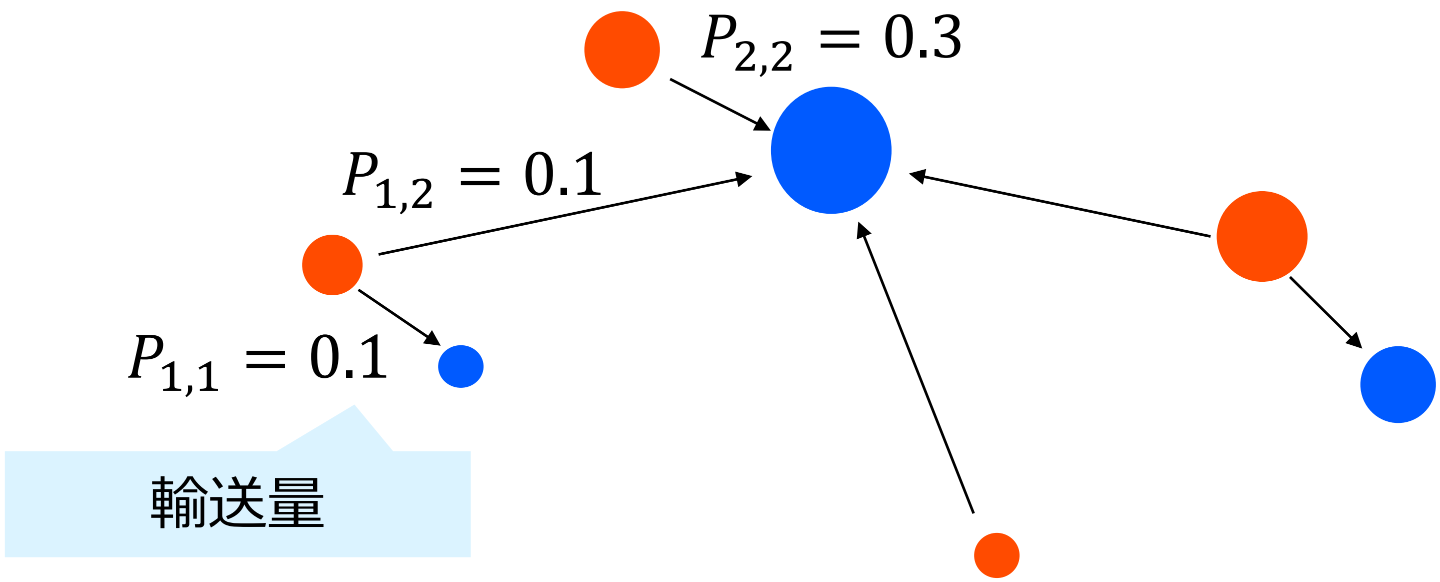


位置

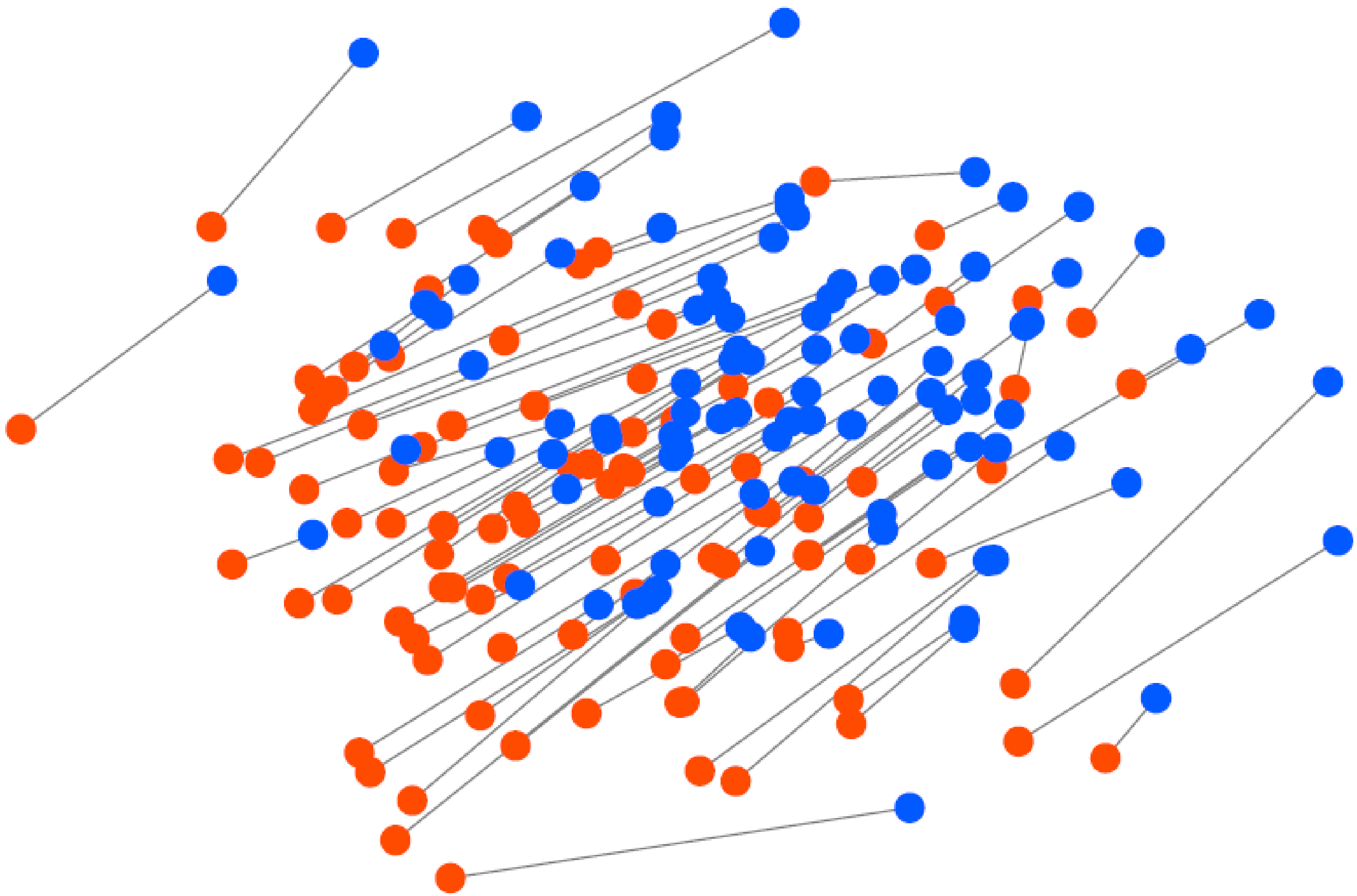
最適輸送の出力例

- 出力例：

  総移動コスト：$d(a, X, b, Y) = 0.83$



もう少し大きい点群の例



線形計画としての定式化

- 最適輸送を最適化問題として定式化する

$$\underset{P \in \mathbb{R}^{n \times m}}{\text{minimize}} \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} C_{ij}$$

$C \in \mathbb{R}^{n \times \mathrm{m}}$ は移動コストを並べた行列
例： $C_{ij} = \|x_i - y_j\|_2^2$

移動コストを最小化する移動方法 P を求める

$$\text{s.t.} \quad P_{ij} \geq 0 \; \forall i, j$$

輸送量は非負

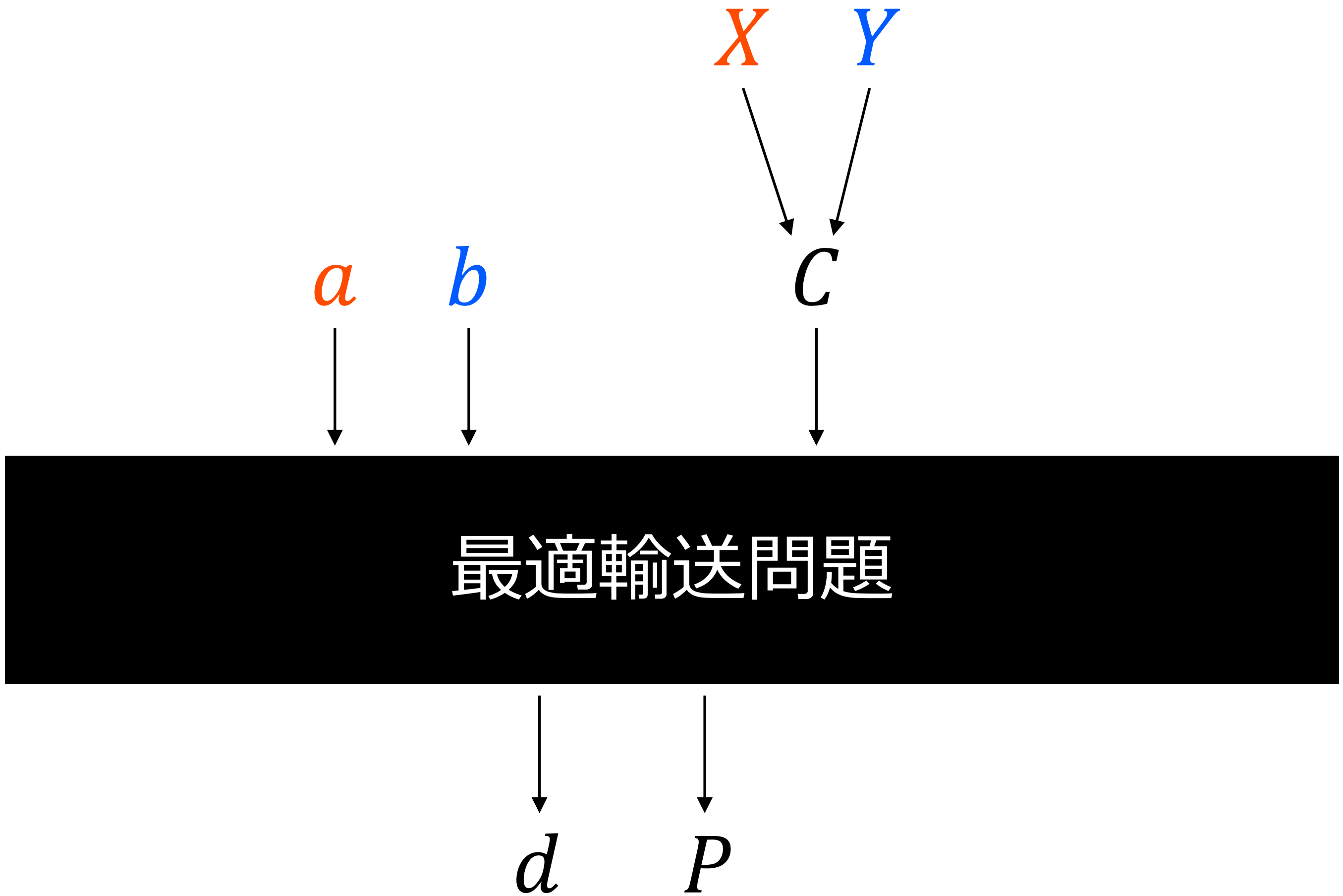
$$\sum_{j=1}^{m} P_{ij} = a_i \; \forall i$$

点群 A の点 i から出ていく量の合計は a_i

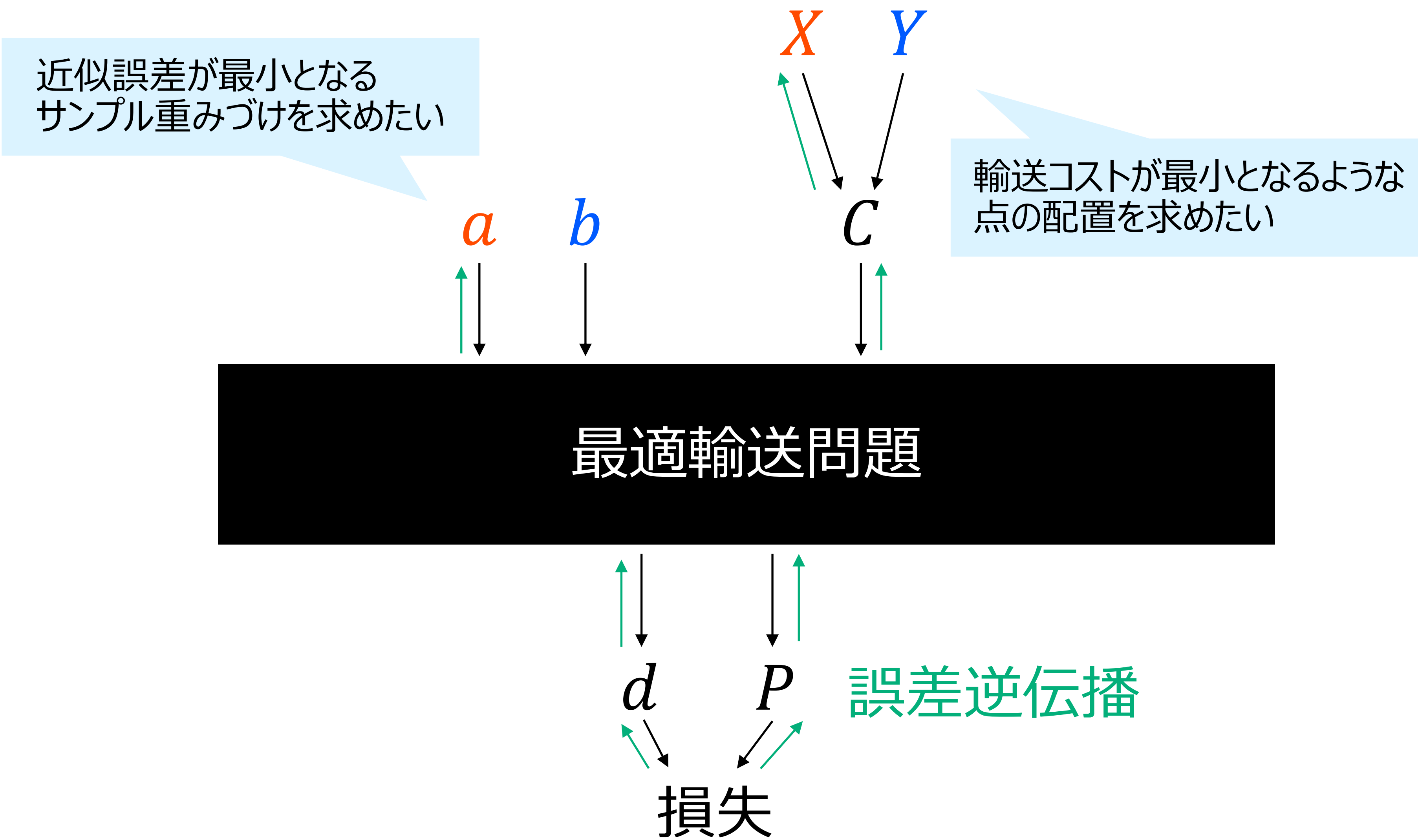
$$\sum_{i=1}^{n} P_{ij} = b_j \; \forall j$$

点群 B の点 j から出ていく量の合計は b_i

a, b, X, Y を入れると d, P が出てくる



入力についての勾配を求めて最適化をしたい



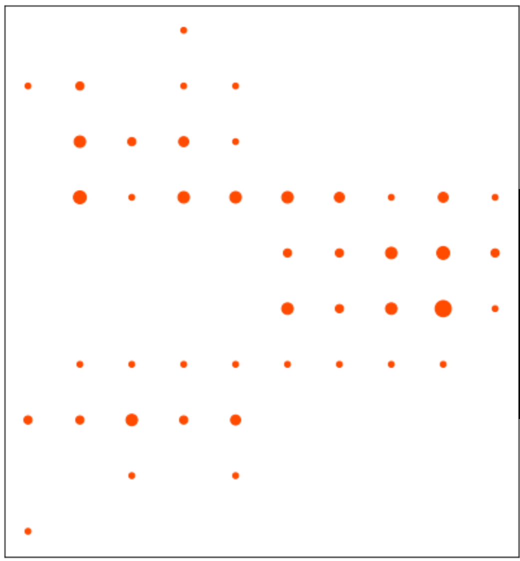
入力についての勾配を求めて最適化をしたい

近似誤差が最小となる
サンプル重みづけを求めたい

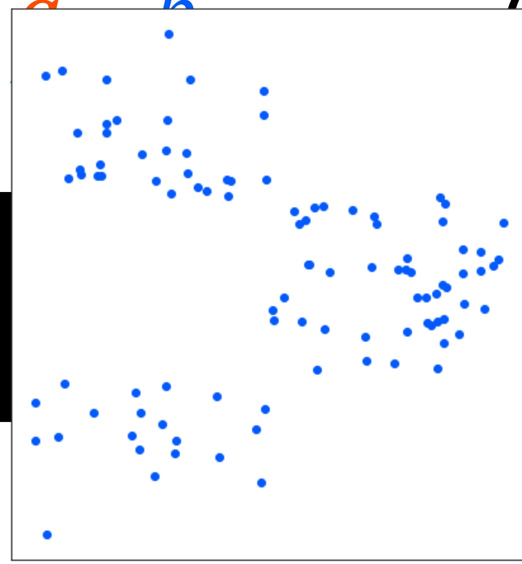
X Y

輸送コストが最小となるような
点の配置を求めたい

C



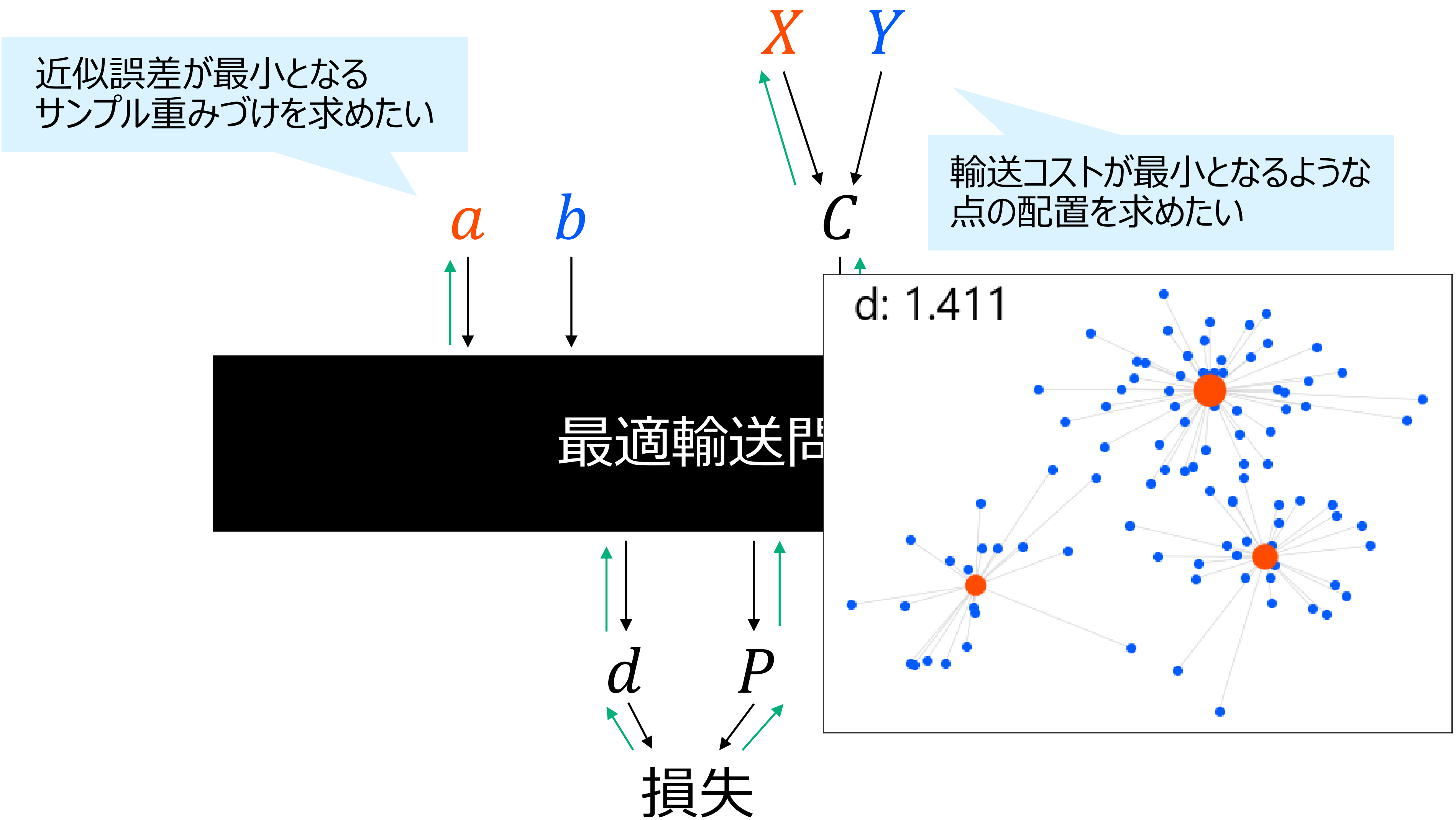
[redacted]



d P 誤差逆伝播

損失

入力についての勾配を求めて最適化をしたい



正則化を追加して滑らかにする

- 悲報：最適輸送は微分できない　朗報：ちょっと変えればできる

$$\underset{P \in \mathbb{R}^{n \times m}}{\text{minimize}} \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} C_{ij} + \varepsilon \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} \left(\log P_{ij} - 1\right)$$

$$\text{s.t.} \quad P_{ij} \geq 0$$

$$\sum_{j=1}^{m} P_{ij} = a_i$$

$$\sum_{i=1}^{n} P_{ij} = b_j$$

問題を滑らかにするための
エントロピー正則化項
一様に近い輸送を優遇する
$\varepsilon \in \mathbb{R}$ はハイパーパラメータ

正則化を追加して滑らかにする

- シンクホーンアルゴリズム：正則化付き最適輸送を解く

```
K = np.exp(- C / eps)
u = np.ones(n)
for i in range(100):
    v = b / (K.T @ u)
    u = a / (K @ v)
P = u.reshape(n, 1) * K * v.reshape(1, m)
d = (C * P).sum()
```

超シンプル！

導出は『最適輸送の理論とアルゴリズム』第三章や『最適輸送の解き方』p.198- を参照してください。

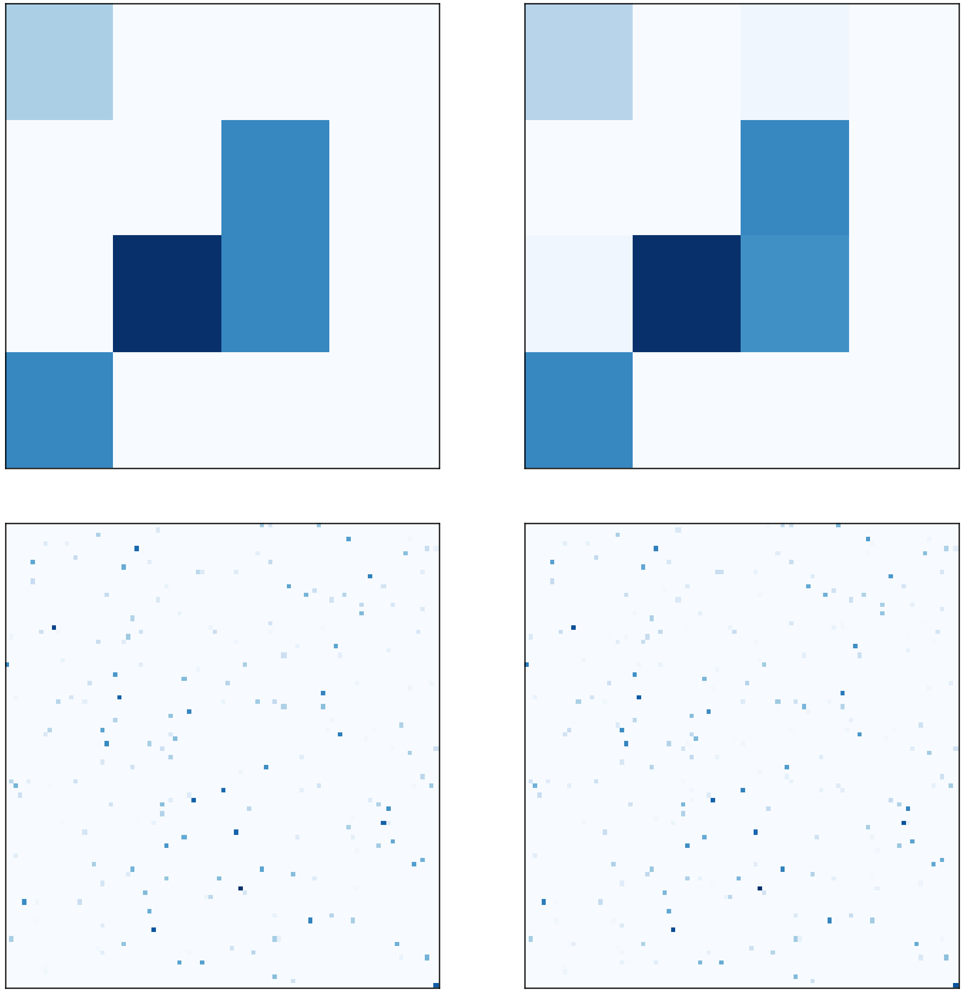
https://speakerdeck.com/joisino/zui-shi-shu-song-nojie-kifang?slide=198

線形計画解とシンクホーン解はほぼ同じ

線形計画解　　シンクホーン解

n = m = 4

n = m = 100



行列 $P \in \mathbb{R}^{n \times m}$ の図示

ほぼ同じ
→ 以降同一視する

ソースコード

https://colab.research.google.com/drive/1RrQhsS52B-Q8ZvBeo57vKVjAARI2SMwM?usp=sharing

再掲：シンクホーンアルゴリズム

- シンクホーンアルゴリズム：正則化付き最適輸送を解く

```
K = np.exp(- C / eps)
u = np.ones(n)
for i in range(100):
    v = b / (K.T @ u)
    u = a / (K @ v)
P = u.reshape(n, 1) * K * v.reshape(1, m)
d = (C * P).sum()
```

超シンプル！

シンクホーンアルゴリズムは自動微分できる

- 四則計算と exp だけからなるので自動微分が可能

```
a.requires_grad = True
K = torch.exp(- C / eps)
u = torch.ones(n)
for i in range(100):
    v = b / (K.T @ u)
    u = a / (K @ v)
P = u.reshape(n, 1) * K * v.reshape(1, m)
d = (C * P).sum()
d.backward()
print(a.grad)
```

シンクホーンアルゴリズムは自動微分できる

- 他のニューラルネットワークと組み合わせてもオーケー

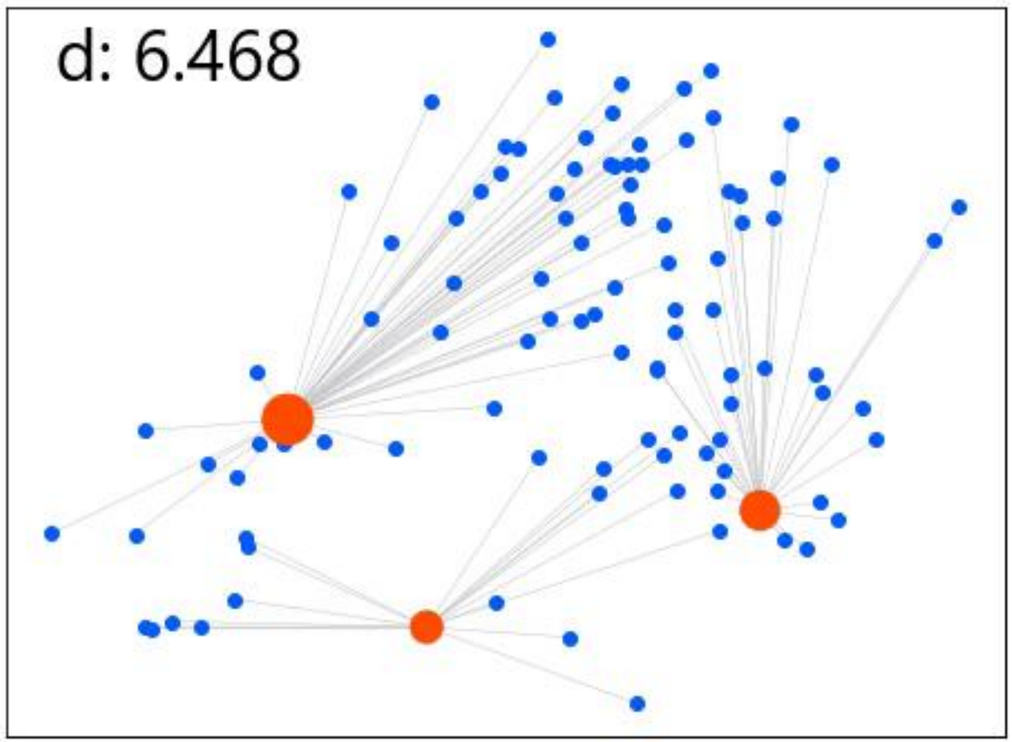
```
C = net1(z)
K = torch.exp(- C / eps)
u = torch.ones(n)
for i in range(100):
    v = b / (K.T @ u)
    u = a / (K @ v)
P = u.reshape(n, 1) * K * v.reshape(1, m)
d = (C * P).sum()
loss = net2(P, d)
loss.backward()
```

何かしらのニューラルネットワーク

自動微分を使って配置を最適化する例

- 数値例：点群 A を点群 B に近づける

パラメータは位置 X　　Adam で最適化

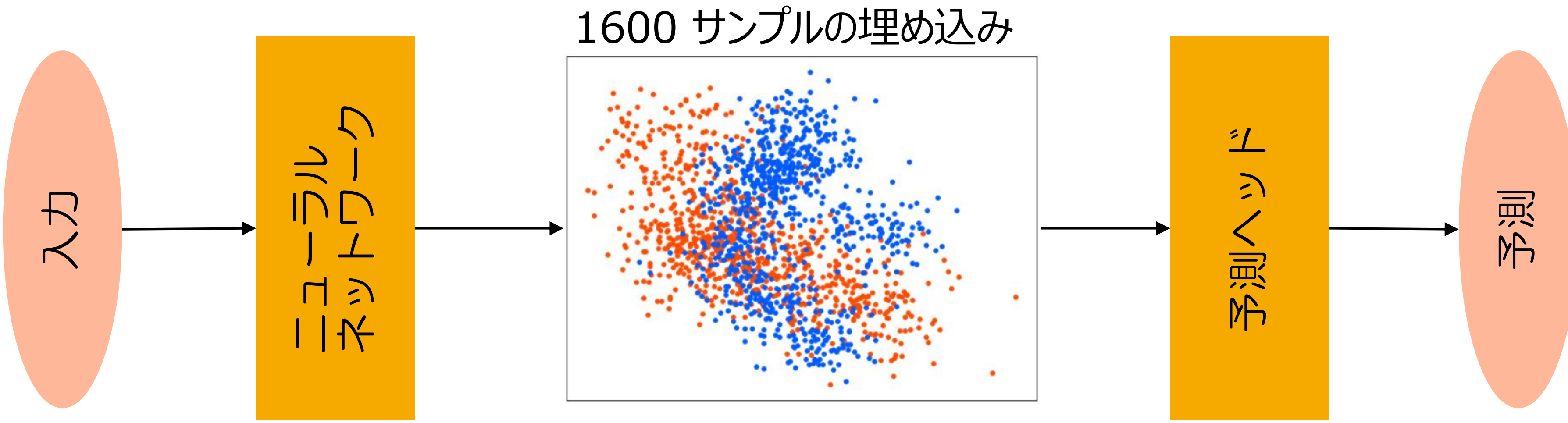


動画

ソースコード

https://drive.google.com/file/d/19XNtttaSr-Kc8yfv1VKRz0O8dUpcxSZM/view?usp=sharing
https://colab.research.google.com/drive/1u8lu0I7GwzR48BQqoGqOp2A_7mTHxzrk?usp=sharing

応用例：転移学習



赤：シミュレーションデータについての埋め込み
青：本番環境データについての埋め込み

- 予測誤差 ＋ 赤と青の最適輸送コストを最小化
- ・シミュレーションデータと本番環境の差を中間層で吸収する

ランキング問題を考える

- ランキング問題

- 入力：

  $x \in \mathbb{R}^n$: 配列

  出力：

  $r \in \mathbb{N}^n$: ランク（$r_i = k \Leftrightarrow x_i$ は k 番目に大きい）

- 入力例：

  $x = (6.2, 1.4, 1.5, 3.9, 2.2)$

  出力例：

  $r = (1, 5, 4, 2, 3)$

分類問題では正解率を最大化したい

- 分類問題において本当にやりたいことは正解率の最大化。
  二値分類問題においてクラス 1 のデータの予測確率が
  (0.6, 0.4) だろうが (0.99, 0.01) だろうが正解なら十分。
- 正解率を最適化するのが難しいので、クロスエントロピーを使うことが多い。
- しかし、クロスエントロピーは (0.99, 0.01) を優遇する。
  もう正解できているデータの損失を無駄に下げるために、
  際どいデータが不正解に転じることがある。

正解率や top-K 正解率を直接最大化したい

例：豹、鳥、犬、猫、猿の五クラス分類

ニューラルネットワーク

$\mathrm{logit} = (6.2, 1.4, 1.5, 3.9, 2.2)$

誤差逆伝播
（をやりたい）

ランキング

教師ラベル

$r = (1, 5, 4, 2, 3)$　　$y = 4$

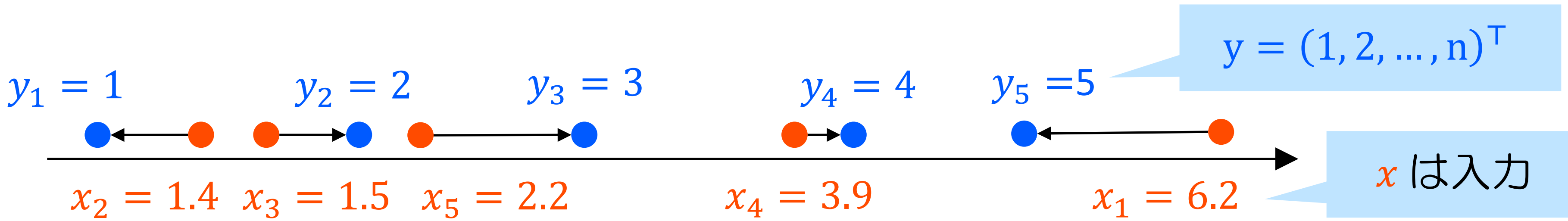
「猫」の予測順位は 2 位
1 位にして正解率を上げるには？

$r_y = 2$

正解率や top-K 正解率
を直接最適化したい

ランキング問題は最適輸送の特殊例

- ランキング問題は $d = 1$ 次元の最適輸送問題の特殊例



$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$r = P \begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 5 \\ 4 \\ 2 \\ 3 \end{pmatrix}$$

最も小さいものは 1 に、二番に小さいものは 2 に … と輸送するのが最適

順列行列をランクに変換

→ シンクホーンアルゴリズムで計算すればランクも微分できる

正解率や top-K 正解率を直接最大化できる

ニューラルネットワーク

logit = $(6.2, 1.4, 1.5, 3.9, 2.2)$

誤差逆伝播

シンクホーンによるランク計算

$r = (1.01, 4.84, 4.13, 2.02, 3.04)$ $y = 4$

教師ラベル

「猫」の予測順位は 2.02 位 $r_y = 2.02$

正解率や top-K 正解率
を直接最適化できる

ビームサーチなど、様々な過程全体を微分可能にできる

- 同様の考えから、ランキング・ソートなどを end-to-end 学習パイプラインの中に組み込むことができる。

- 言語モデルの訓練においてビームサーチを微分する [1]
訓練時は teacher forcing して、テスト時はビームサーチをすることが多いが、これだと乖離が生じる。
ビームサーチの top-K をシンクホーンで計算し、ビームサーチの過程全体を微分可能にする。これを使って訓練する。

[1] Xie et al. Differentiable Top-k with Optimal Transport. NeurIPS 2020.
[2] Goyal et al. A continuous relaxation of beam search for end-to-end training of neural sequence models. AAAI 2018.

ビームサーチなど、様々な過程全体を微分可能にできる

- シンクホーンアルゴリズムはブレグマン法の特殊例である [1]
- ブレグマン法は制約あり凸計画問題のアルゴリズム。
  制約なしの解からはじめて、制約に射影していく。
- シンクホーンアルゴリズムは、$\mathrm{P}1 = a$ と $P^\top 1 = b$ に交互に
  射影していくことに対応する。

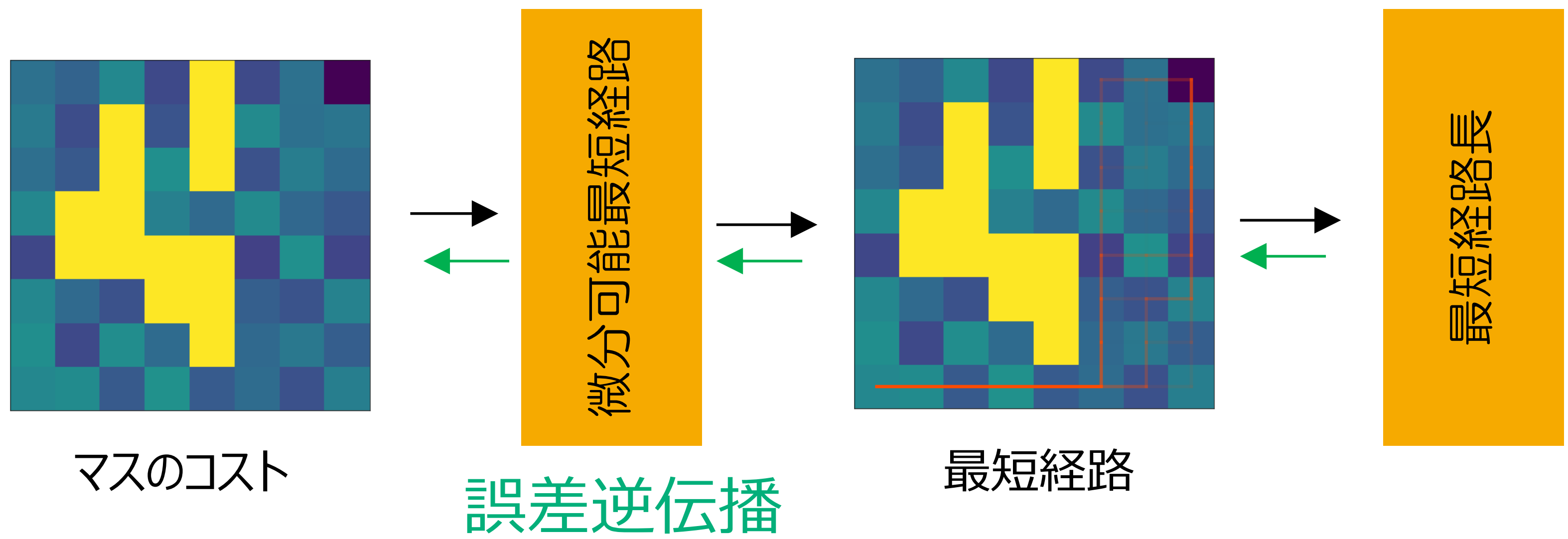
```
for i in range(100):
    v = b / (K.T @ u)
    u = a / (K @ v)
```

- 一般の線形計画もブレグマン法により、
  シンクホーンアルゴリズムと同様の簡単な反復アルゴリズムで
  解くことができ、これにより同様に微分ができる。

[1] Benamou et al. Iterative Bregman Projections for Regularized Transportation Problems. 2015.

最短経路問題の数値例

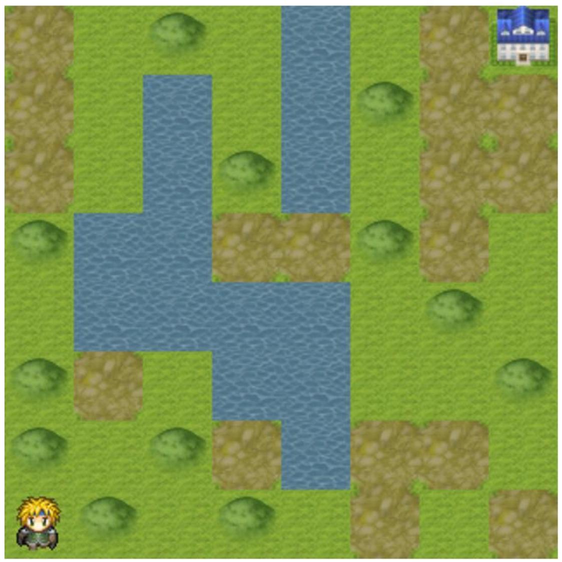
- 例1: 最短経路を長くして邪魔をする（※最短経路問題は線形計画）
  パラメータ：マスのコスト（総和は一定）　Adam で最適化



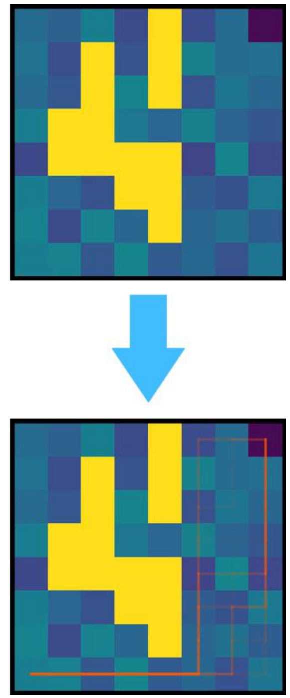
最短経路問題の数値例

- 例1: 最短経路を長くして邪魔をする （※最短経路問題は線形計画）
  パラメータ：マスのコスト（総和は一定）　Adam で最適化

可視化



コスト
（パラメータ）



最短経路

動画

コード

この例はマスコストを生パラメータとしているが、
生成モデルでマップ生成してモデルまで逆伝播なども可能

https://drive.google.com/file/d/1_eijS6R83nTcBOMzUM1QoR74Uk4S0qvw/view?usp=sharing
https://colab.research.google.com/drive/1yB_tcEA2OppiyaInzM1GKmGAIDKw6VNL?usp=sharing

最短経路を最適化する問題を勾配法で解くことができる

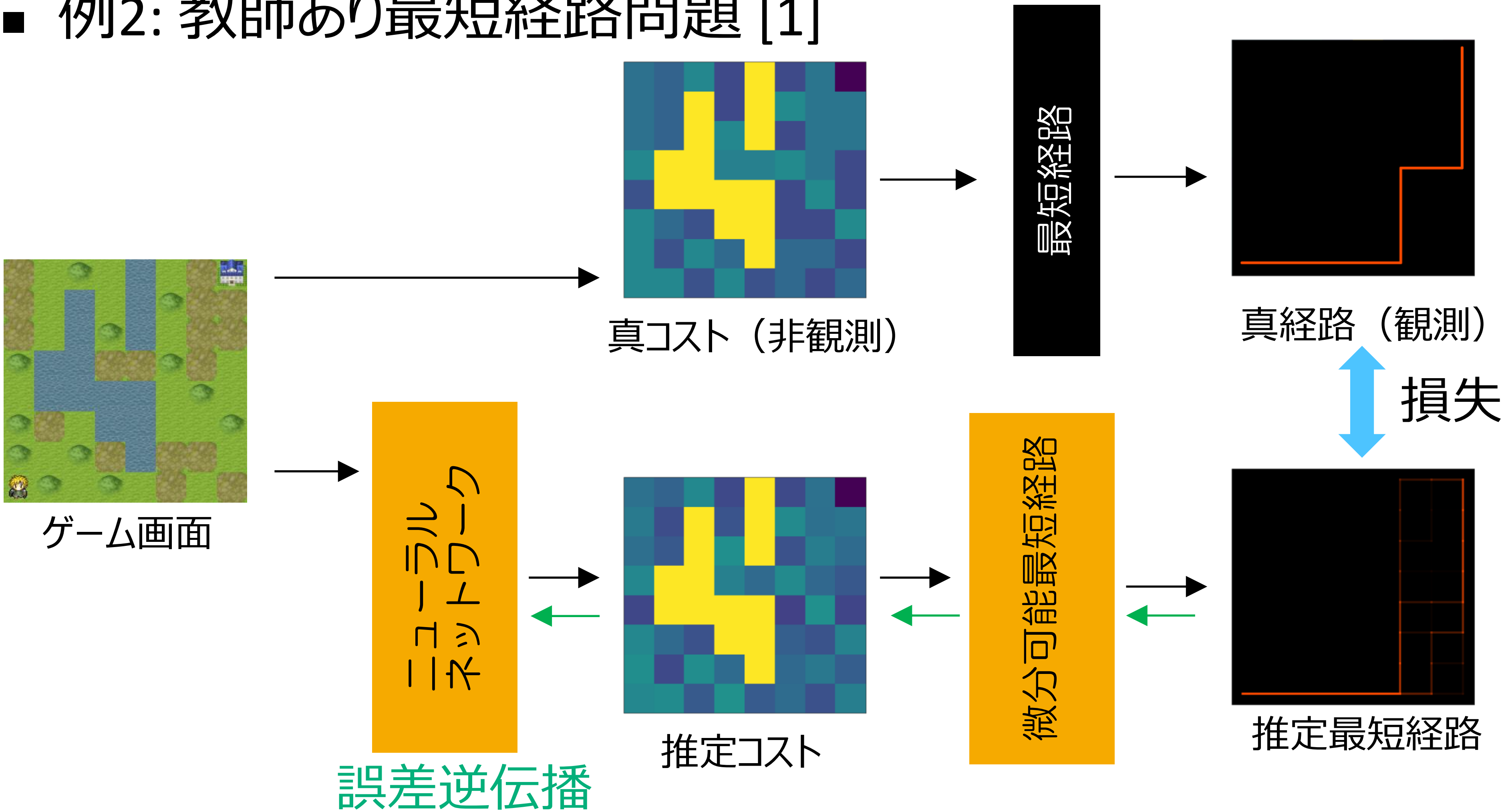
- 例1: 最短経路を長くして邪魔をする（※最短経路問題は線形計画）
パラメータ：マスのコスト（総和は一定）　Adam で最適化

- 観察1:最短経路などの組合せ的な問題も行列積を用いた
反復法により解が求まる。

- 観察2:最短経路を最適化するという 2 レベルの最適化問題も
Adam などの勾配法ベースの連続最適化で解ける。

最短経路問題のその他の問題例

- 例2: 教師あり最短経路問題 [1]



[1] Vlastelica et al. Differentiation of Blackbox Combinatorial Solvers. ICLR 2020.

離散的な操作を微分可能にすることができる

- 最適輸送、ランキング、最短経路問題などの微分可能版を考えることができる。
- シンクホーンアルゴリズムやブレグマン法で計算できる。
- 「モデルの予測の順位」「モデルの出力を基にしたビームサーチの結果」「モデルの出力を最短経路問題に入力した結果」などの量を直接最適化することができる。

Take Home Message

離散的な操作を微分可能にしてニューラルネットワークの end-to-end 最適化パイプラインに組み込むことができる

参考文献

- Xie et al. Differentiable Top-k with Optimal Transport. NeurIPS 2020.
- Goyal et al. A continuous relaxation of beam search for end-to-end training of neural sequence models. AAAI 2018.
- Benamou et al. Iterative Bregman Projections for Regularized Transportation Problems. 2015.
- Vlastelica et al. Differentiation of Blackbox Combinatorial Solvers. ICLR 2020.
- Cuturi et al. Differentiable Ranks and Sorting using Optimal Transport. NeurIPS 2019.
- Blondel et al. Fast Differentiable Sorting and Ranking. ICML 2020.
- Berthet et al. Learning with Differentiable Perturbed Optimizers. NeurIPS 2020.
- Weed. An explicit analysis of the entropic penalty in linear programming. COLT 2018.
- 『最適輸送の解き方』 https://speakerdeck.com/joisino/zui-shi-shu-song-nojie-kifang
- 佐藤竜馬『最適輸送の理論とアルゴリズム』