最適輸送と自然言語処理

横井 祥 (東北大学)

2022-03-14, 言語処理学会 第28回年次大会 チュートリアル

増補改訂版

自己紹介

- 横井 祥
  - http://www.cl.ecei.tohoku.ac.jp/~yokoi/

🕒 略歴

- **B**: 京都大学 工学部 情報学科 (鹿島 久嗣 先生) 機械学習
- **M**, **D**, **現**: 東北大学 情報科学研究科 (乾 健太郎 先生) 自然言語処理
- 共同研究・アドバイザ業など気軽にお声掛けください

📝 最近の研究の興味

- 言語の “意味” が 埋込空間の “形状” にどのように反映されるのか
  - 長さ [EMNLP’20, ICLR’21], 混ざり具合 [EMNLP’20, EMNLP’21], 集積 [TACL’21], 🚚 **輸送** 🚚 [EMNLP’20], …

謝辞

- **言語処理学会年次大会の委員の皆さん**に感謝します.
- 当日**質問・コメントをくださった皆さん**に感謝します.
  いただいた QA の多くをこのスライドに反映しました.
- スライドに書かれた内容は**広義同僚の皆さん**からの建設的な助言や楽しい議論に負うところ大です．深く感謝します.
  - 佐藤竜馬さん (京都大)
  - 包含さん (京都大)
  - 最適輸送ゼミの皆さん：とくに 宇田智紀さん (東北大)・木村悠紀さん・谷地村敏明さん (京都大)・童祺俊さん (ALBERT)
  - 東北大学 乾研究室・鈴木研究室の皆さん：とくに 小林颯介さん・栗林樹生さん・鈴木潤さん・乾健太郎さん
  - 京都大学 下平研究室の皆さん：とくに 山際宏明さん・下平英寿さん
- **JST ACT-X 数理・情報 領域** からの研究支援に感謝します.

チュートリアルの進め方

スライド

- **Ver. 1**: 事前配布版
  - 学会参加者への事前配布版
- **Ver. 2**: 当日利用版
  - 事前配布版の感想を踏まえてコンテンツを増強しました
- **Ver. 3**: 一般公開版

このスライド

  - チュートリアル当日の質疑応答を踏まえ増補改訂しました
    https://speakerdeck.com/eumesy/optimal-transport-for-natural-language-processing

QA

- **随時気軽に質問・コメント・野次を投稿してください**
  - Slack チャンネルをできるだけリアルタイムでチェックします
    - #s42-tutorial4-最適輸送と自然言語処理
    - ラフなコメント，雑談，スタンプコミュニケーションも大歓迎です
  - 拾えそうなコメント・質問は可能な範囲で拾いながら話します
  - 拾えなかったコメントも質問タイムにできるだけ回答します
- **QAタイムを設けます**
  - 真ん中あたりで休憩兼質問タイムを取ります
    - (90分を超えるトークは聞き手がむちゃくちゃ疲れるので…)
    - スタンプが集まっている質問を中心に
    - もちろん挙手質問も歓迎です
  - 最後にも質問タイムをとります

インタラクション

- 突然アンケートをとったり反応を求めたりすると思うので皆さんぜひ気軽に反応してください

- 1コマですが充実した時間にできればと思います

基礎編

もっとも標準的な形式の最適輸送問題の気持ちを知る
「最適輸送わかってしまった……」になる

最適輸送を知る はじめの一歩

10分でわかる最適輸送
荷物の配置替えコストという気持ちを知る

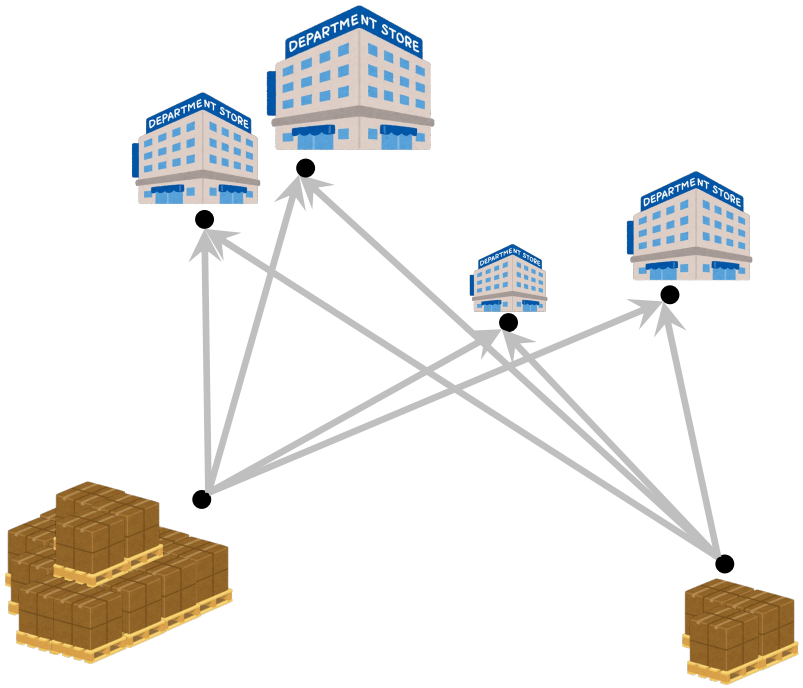
目次

復習に便利なようにリンク付き目次を置いておきます．
いま全体像を把握しなくても大丈夫です．

今日のテーマ：最適輸送

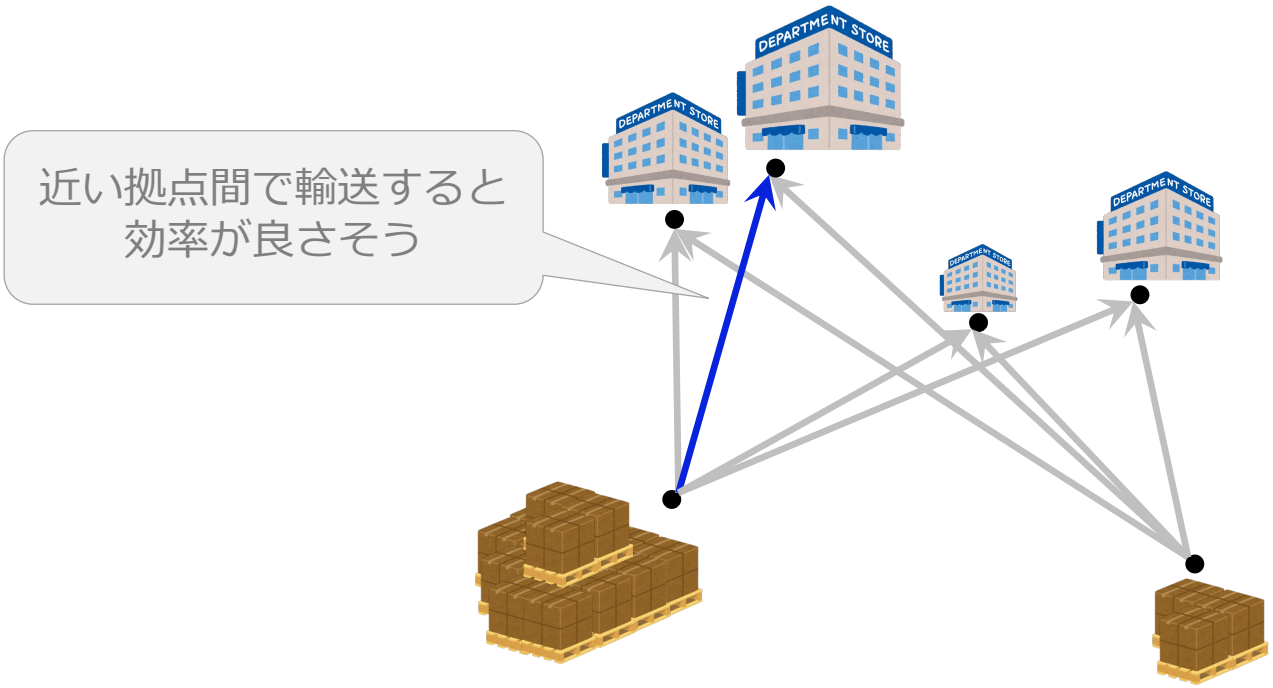
突然ですが

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？



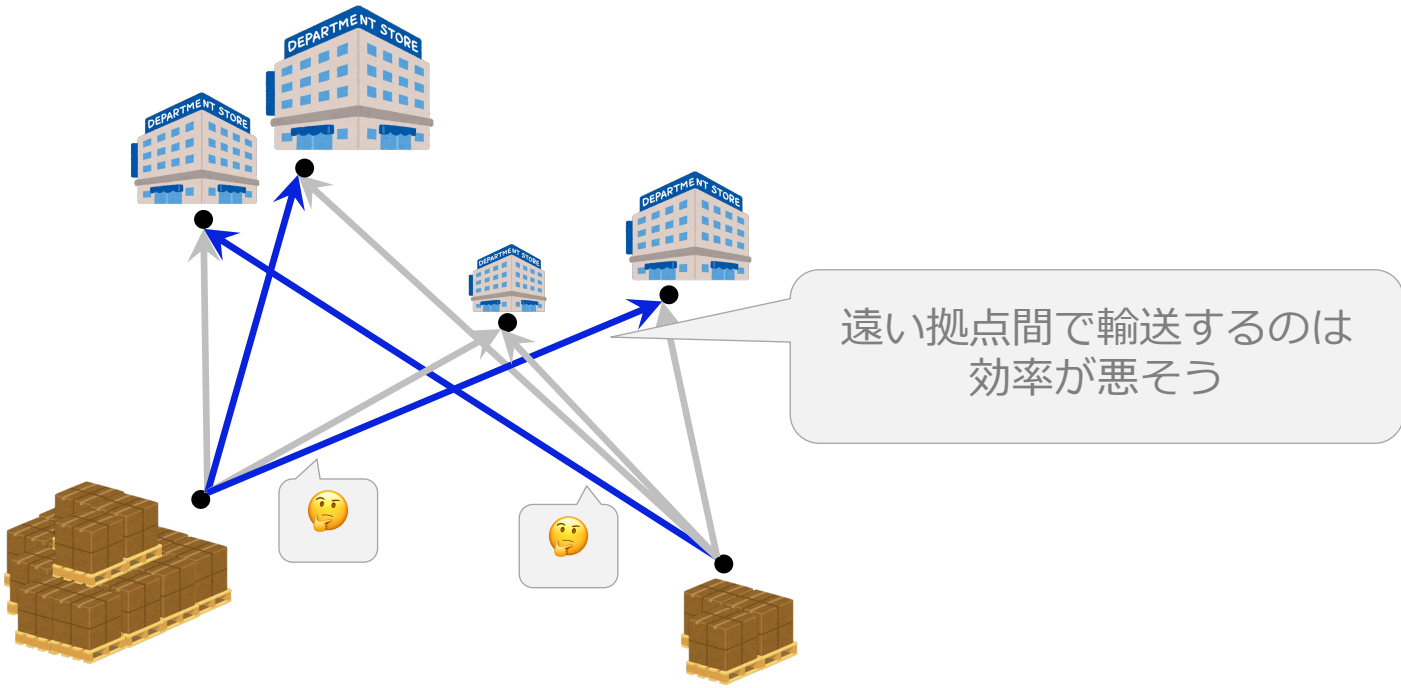
今日のテーマ：最適輸送

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？



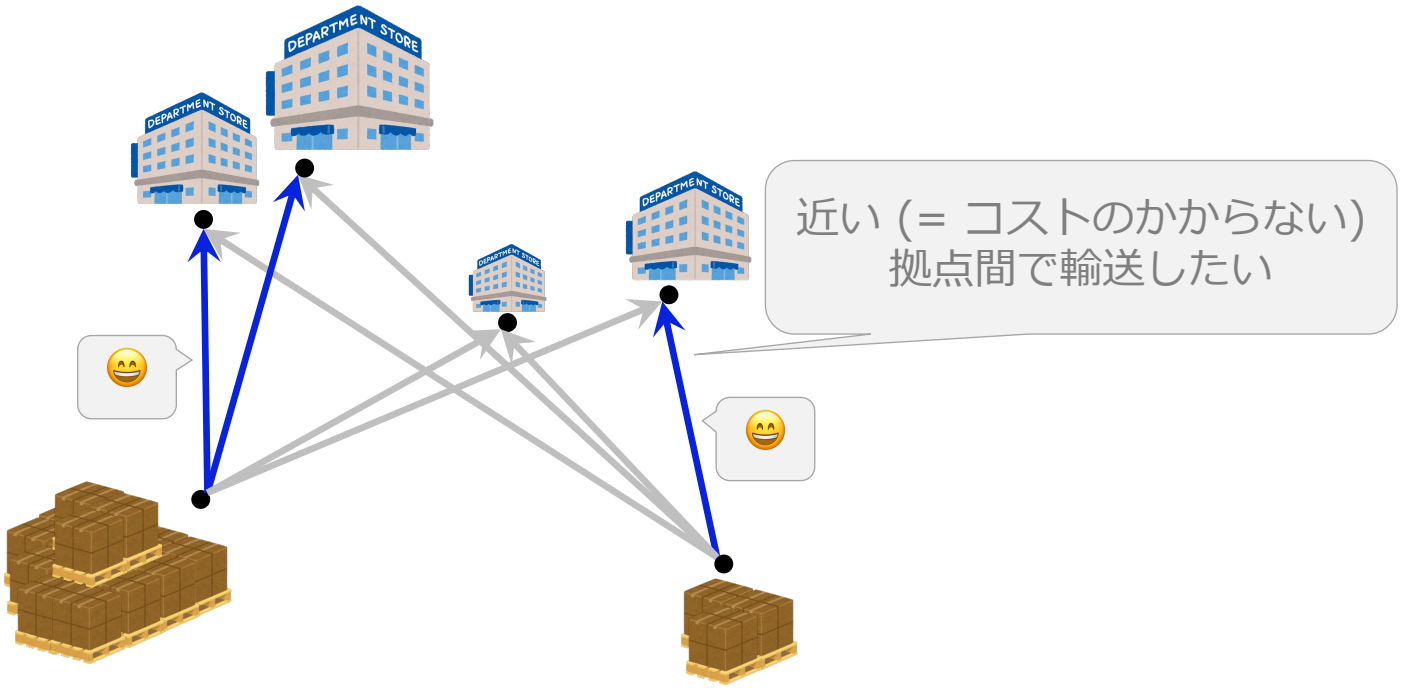
今日のテーマ：最適輸送

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？



今日のテーマ：最適輸送

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？

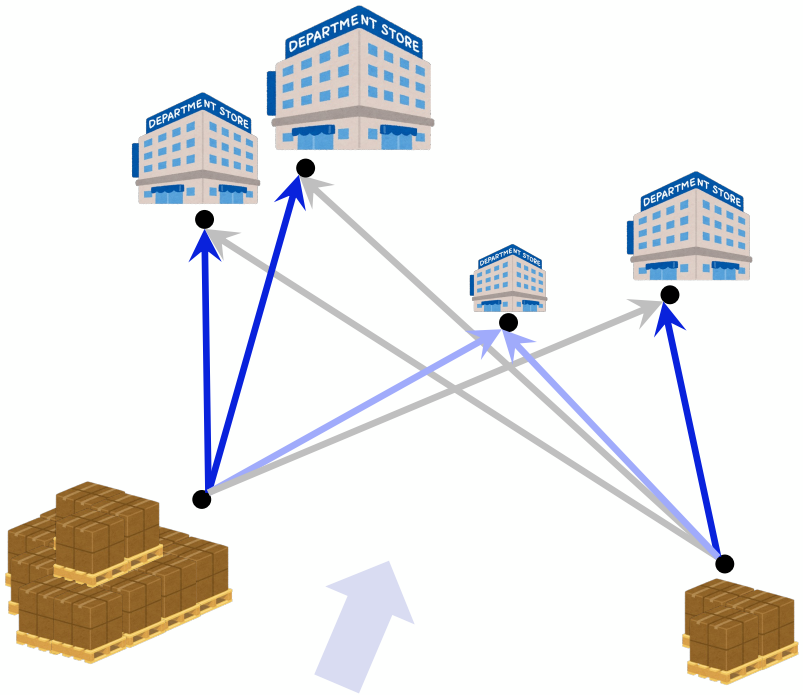


今日のテーマ：最適輸送

まとめスライドだけ
見返せば手早く
復習できます

まとめ

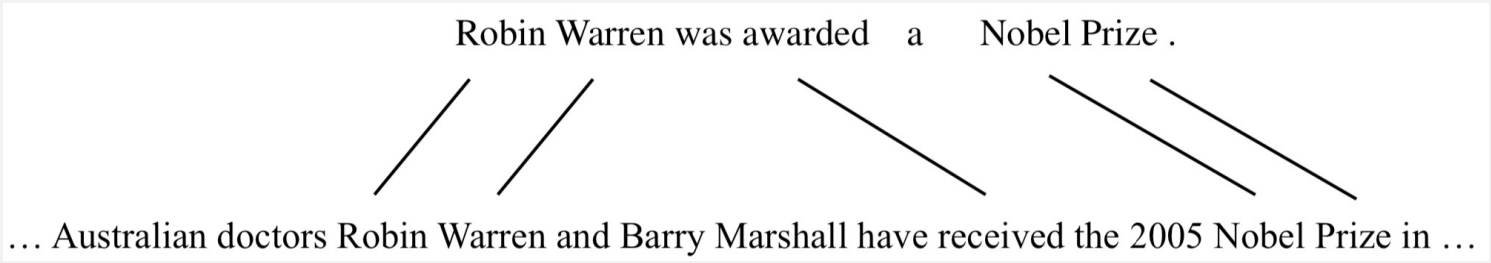
- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？



- **最適輸送** = 最良の輸送プラン と総輸送コストを求める道具

またまた突然ですが
テキストの類似度測定

- ふたつの**テキスト**の**類似度**の測定
  - NLP で極めて頻繁に必要とされるサブルーチン
  - テキスト生成：生成文 ↔ リファレンス文
    - 損失関数，自動評価尺度
  - 関連テキスト検索：文 ↔ 文，文書 ↔ 文書，クエリ ↔ 文書
    - retrieve-and-edit 言語モデル，類似性に基づくモデル解釈，etc.
- 基本指針：**要素単語の重なりの度合い**を柔らかく測る



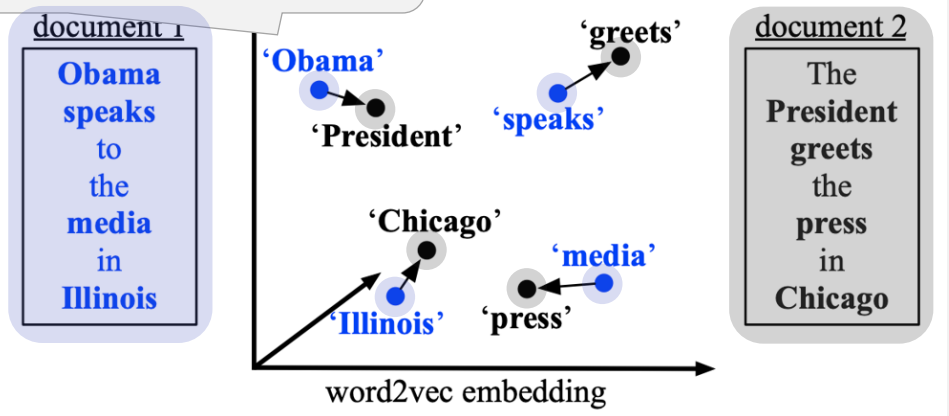
Sultan+, DLS$@$CU: Sentence Similarity from Word Alignment and Semantic Vector Composition (SemEval 2015)

Word Mover's Distance [Kusner+'15]

文類似度 ↔ 最適輸送コスト

- 単語ベクトル空間を地図だと思ってみる
  - 文1：単語埋込空間の工場の位置
  - 文2：単語埋込空間のデパートの位置

意味が近い単語同士 (工場とデパート) は輸送コストが低い



Kusner+, From Word Embeddings To Document Distances (ICML 2015)

- 意味の近さの問題が輸送コストの大きさの問題に置き換わる
  - 文1, 文2: テキストの意味が似ている ↔ 低コストで輸送しきれる

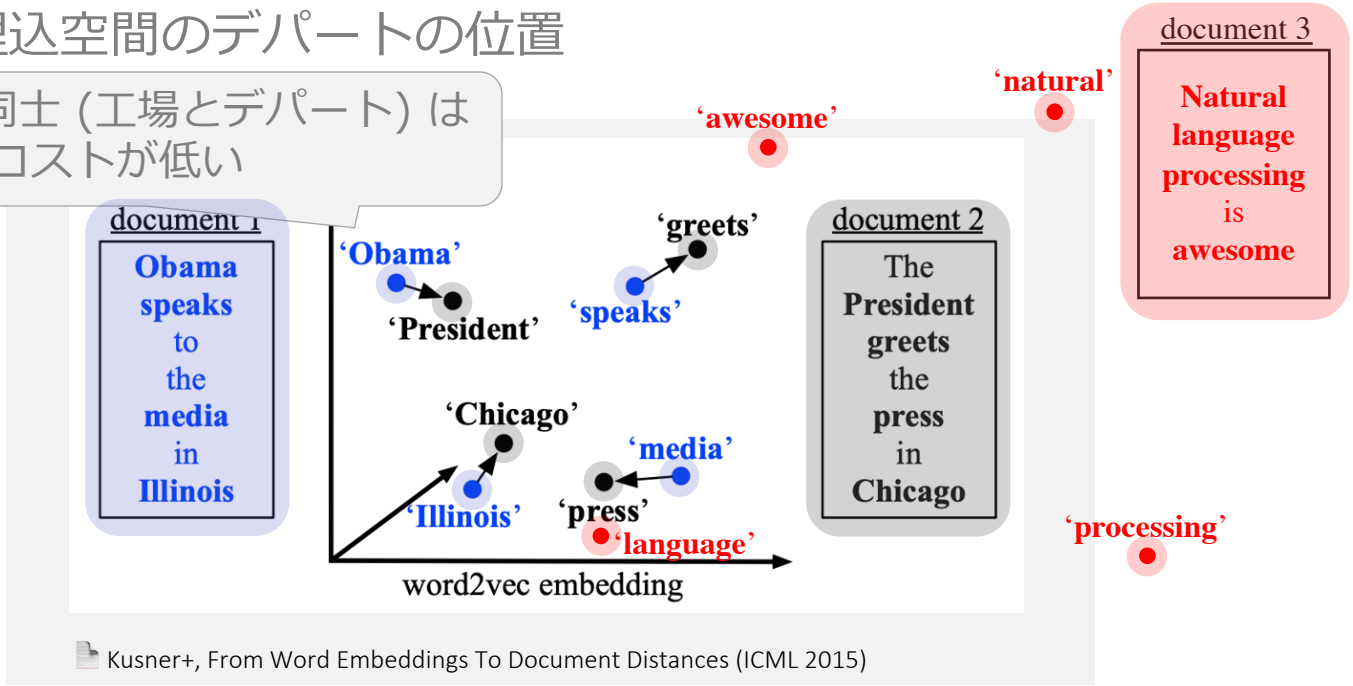
Word Mover's Distance [Kusner+'15]

文類似度 ↔ 最適輸送コスト

まとめ

- 単語ベクトル空間を地図だと思ってみる
  - 文1：単語埋込空間の工場の位置
  - 文2：単語埋込空間のデパートの位置

意味が近い単語同士 (工場とデパート) は輸送コストが低い



Kusner+, From Word Embeddings To Document Distances (ICML 2015)

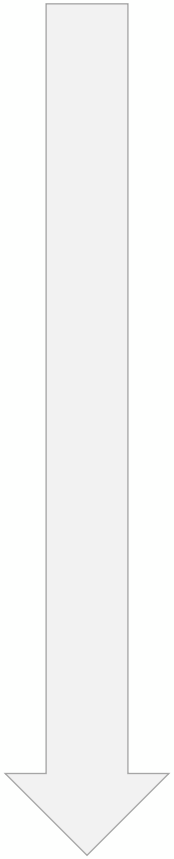
- 意味の近さの問題が輸送コストの大きさの問題に置き換わる
  - 文1, 文2: テキストの意味が似ている ↔ 低コストで輸送しきれる
  - 文1, 文3: テキストの意味が似ていない ↔ 高い輸送コストがかかる

今日のテーマ：最適輸送

- お気持ちは以上です
  - 10分で最適輸送をわかってしまった……
- この一見単純な道具が広大な世界に繋がります
  - このあと，もっと正確な定式化・自然言語処理での面白い利用方法・豊かな拡張について触れていきます

前半の目標：
最適輸送わかった…… になる

- 最適輸送の**視覚的・物理的な気持ち**を知る 済
  - NLPの例題 Word Mover's Distance の気持ちがわかる
- 最適輸送の**入出力**を理解する
  - **ライブラリ**の使い方がわかる
  - NLPの例題 Word Mover's Distance を動かせる，改造できる
- 最適輸送の**形式的な定義**を理解する
  - 論文や教科書の**数式**が読める



- 同じ概念に関して少しずつ抽象度と厳密性を上げていきます
  - 途中で置いていかれても次のセクションですぐ戻って来れるので安心してください.

なぜ今最適輸送なのか

目次

最適輸送は自然言語処理の人にとって勉強する価値が高い

- 最適輸送
  - **“近さ” “遠さ” を考えられる空間** で

    荷物全体 (点群) を移し換えるコストを計算する 道具

  - 副次効果として **アラインメント情報** が得られる

最適輸送は自然言語処理の人にとって勉強する価値が高い

- 最適輸送は自然言語処理とすごく相性が良い
  - **“近さ” “遠さ” を考えられる空間** で

    埋込ベース，ニューラルネットベースの各種手法
    (＝対象が自然に距離空間に入っている状態) との相性が良い

    荷物全体 (点群) を移し換えるコストを計算する 道具

  - 副次効果として **アラインメント情報** が得られる

最適輸送は自然言語処理の人にとって勉強する価値が高い

- 最適輸送は自然言語処理とすごく相性が良い
  - **“近さ” “遠さ” を考えられる空間** で

埋込ベース，ニューラルネットベースの各種手法
(＝対象が自然に距離空間に入っている状態) との相性が良い

荷物全体 (点群) を移し換えるコストを計算する 道具

言語的対象は (たいてい) 何かの集まり
文＝単語列，文書＝文の列，コーパス＝文集合，…

対象間の類似度や距離の計算は
自然言語処理で頻出

  - 副次効果として **アラインメント情報** が得られる

最適輸送は自然言語処理の人にとって勉強する価値が高い

まとめ

- 最適輸送は自然言語処理とすごく相性が良い
  - **“近さ” “遠さ” を考えられる空間** で

埋込ベース，ニューラルネットベースの各種手法
(＝対象が自然に距離空間に入っている状態) との相性が良い

荷物全体 (点群) を移し換えるコストを計算する 道具

言語的対象は (たいてい) 何かの集まり
文＝単語列，文書＝文の列，コーパス＝文集合，…

対象間の類似度や距離の計算は
自然言語処理で頻出

  - 副次効果として **アラインメント情報** が得られる

自然言語処理でしばしば要請される
例：文と文の関係を単語と単語の関係に帰着させたい

高い解釈性；
輸送コスト (最適値) だけではなく輸送プラン (最適解) がわかる

最適輸送を勉強するタイミングは今

Python インタフェース

勢いのある最適輸送の研究者たちが
ばりばりメンテナンスしてくれている

- **利用しやすく安心して使えるライブラリ (ソルバ) の整備**
  - Python Optimal Transport (POT) (2020) GitHub
  - Optimal Transport Tools (OTT) with JAX (2022) GitHub

Flamary+, POT: Python Optimal Transport (JMLR 2021) Cuturi+, Optimal Transport Tools (OTT): A JAX Toolbox for all things Wasserstein (arXiv 2022)
Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019) 佐藤, 最適輸送の理論とアルゴリズム (講談社 2022)

最適輸送を勉強するタイミングは今

- **利用しやすく安心して使えるライブラリ (ソルバ) の整備**
  - Python Optimal Transport (POT) (2020) GitHub
  - Optimal Transport Tools (OTT) with JAX (2022) GitHub
- **計算技法・理論に関する勉強しやすい教科書・資料の整備**
  - 英：Peyré&Cuturi, Computational Optimal Transport (2019)
  - 和：佐藤，最適輸送の理論とアルゴリズム (2022, 予定)

機械学習プロフェッショナルシリーズ新作
2022年12月刊行予定とのこと

arXiv にも最新版が
上がっています

Flamary+, POT: Python Optimal Transport (JMLR 2021) Cuturi+, Optimal Transport Tools (OTT): A JAX Toolbox for all things Wasserstein (arXiv 2022)
Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019) 佐藤, 最適輸送の理論とアルゴリズム (講談社 2022)

最適輸送を勉強するタイミングは今

まとめ

- **利用しやすく安心して使えるライブラリ (ソルバ) の整備**
  - Python Optimal Transport (POT) (2020) GitHub
  - Optimal Transport Tools (OTT) with JAX (2022) GitHub
- **計算技法・理論に関する勉強しやすい教科書・資料の整備**
  - 英：Peyré&Cuturi, Computational Optimal Transport (2019)
  - 和：佐藤，最適輸送の理論とアルゴリズム (2022, 予定)
- **最適輸送を利用している自然言語処理の研究も急増**
  - ちょっと勉強すれば多くの論文の気持ちがわかるようになる
- **最適輸送が適した問題はまだ大量に残っている**
  - 対象がベクトル集合で表現できる，アラインメントが必要，確率分布間の距離を測りたい，etc.

Flamary+, POT: Python Optimal Transport (JMLR 2021) Cuturi+, Optimal Transport Tools (OTT): A JAX Toolbox for all things Wasserstein (arXiv 2022)
Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019) 佐藤, 最適輸送の理論とアルゴリズム (講談社 2022)

最適輸送を知る もう少し丁寧に

最適輸送問題の入出力を形式的に把握する

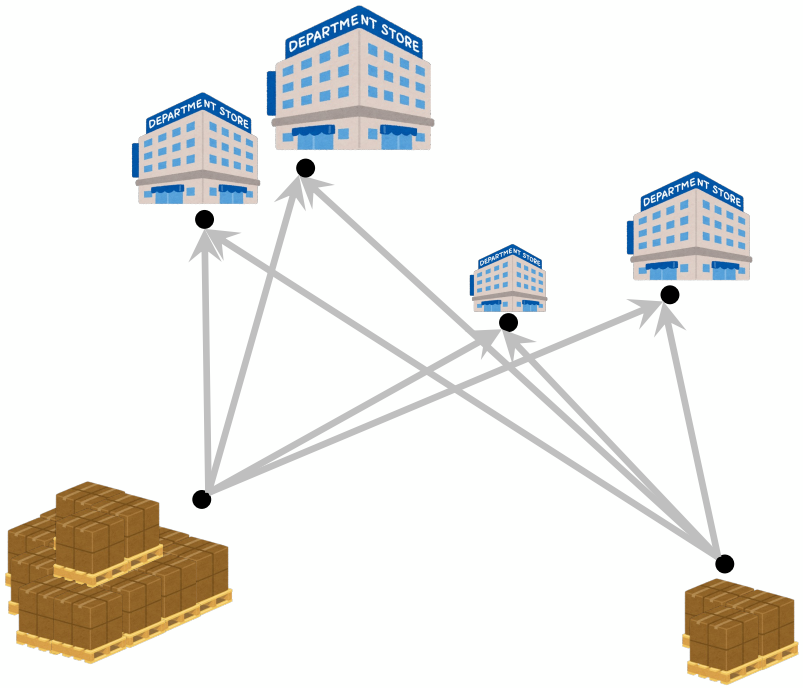
目次

- **基礎編**：もっとも標準的な形式の最適輸送問題を知る
  - 【最適輸送を知る】1：荷物の配置換えという気持ちを知る
    - ゴール：最適輸送の話題が出てきても怖くなくなる
  - なぜ今最適輸送なのか
  - 【最適輸送を知る】2：入出力を形式的に理解する　イマココ
    - ゴール：最適輸送を自分の研究・開発プロジェクトで使えるようになる
  - 入力をカスタマイズしながら使う
  - 【最適輸送を知る】3：線形計画問題としての定式化を理解する
    - ゴール：最適輸送を道具として用いている論文の式を読めるようになる
  - 用語に関する落穂拾い
    - ワッサーシュタイン距離，Earth Mover's Distance
  - 前半のまとめ
- **応用編**：自動微分可能な最適輸送 / 最適輸送の変種 / 全体まとめ

最適輸送問題の入出力を特産品の例で理解

どんな情報があれば問題を解き始められる？ 最終的にどんな情報が手に入る？

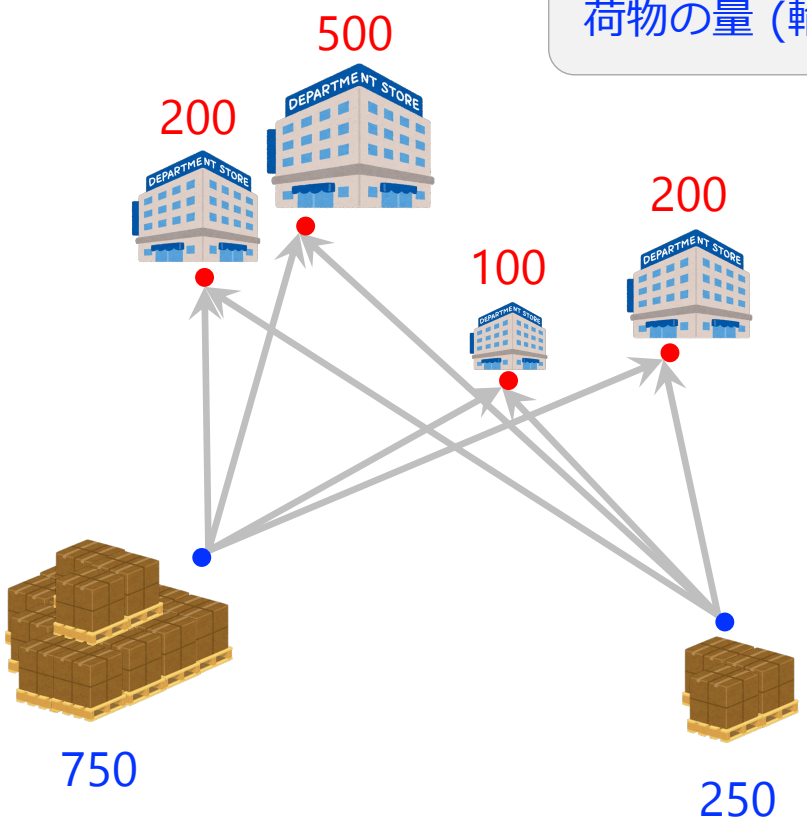
- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？



最適輸送問題の入出力を特産品の例で理解

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？

必要な情報 (1)
荷物の量 (輸送元) / 荷物の量 (輸送先)



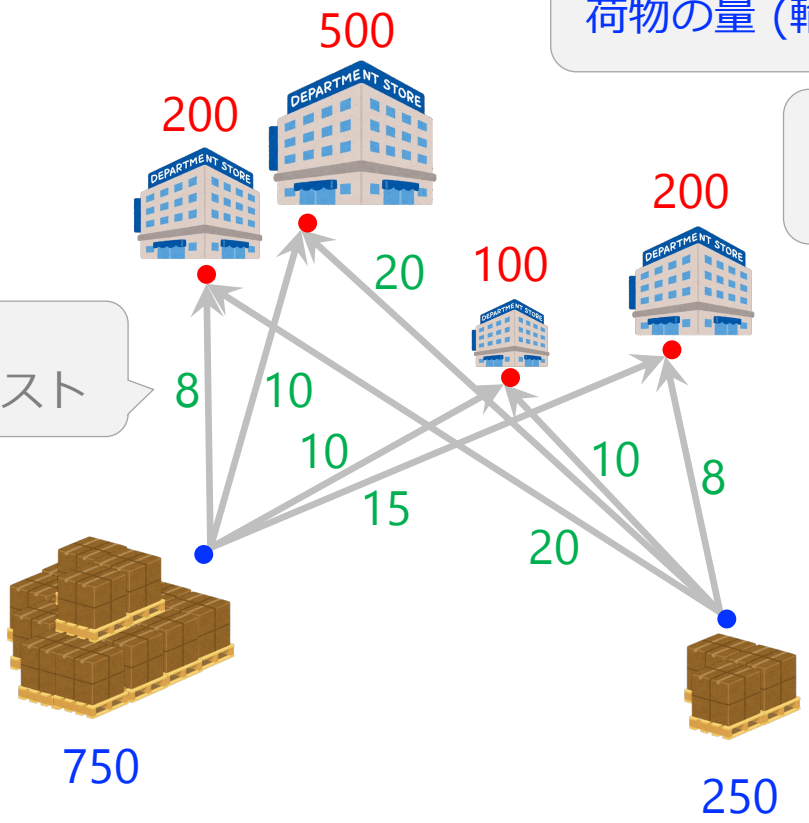
最適輸送問題の入出力を特産品の例で理解

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？

必要な情報 (1)
荷物の量 (輸送元) / 荷物の量 (輸送先)

必要な情報 (2)
各拠点ペア間の輸送コスト

特産品ひとつを
このルートで運ぶコスト



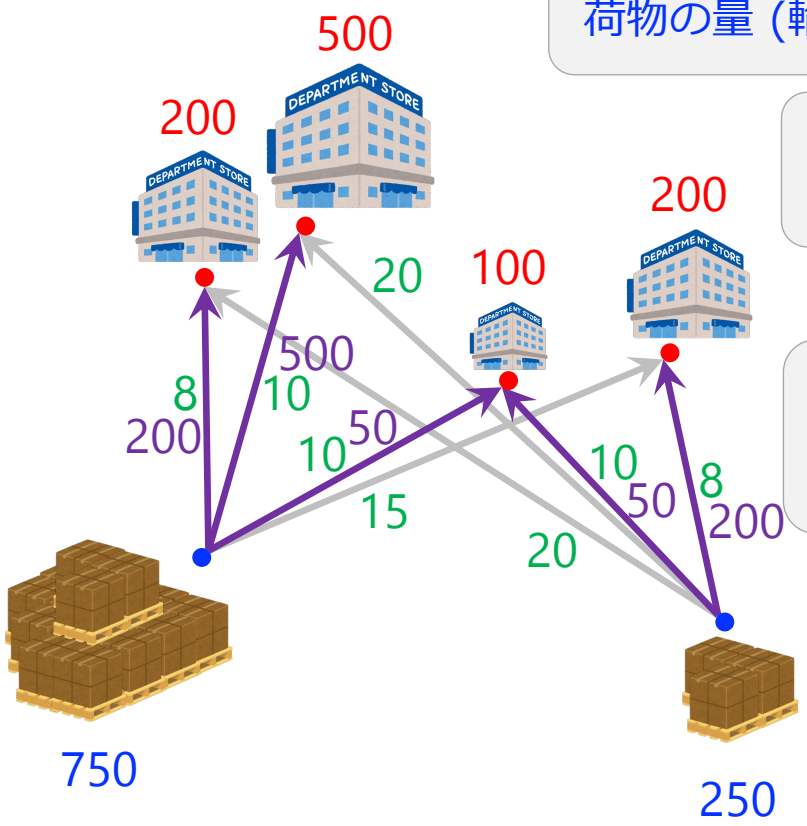
最適輸送問題の入出力を特産品の例で理解

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？

必要な情報 (1)
荷物の量 (輸送元) / 荷物の量 (輸送先)

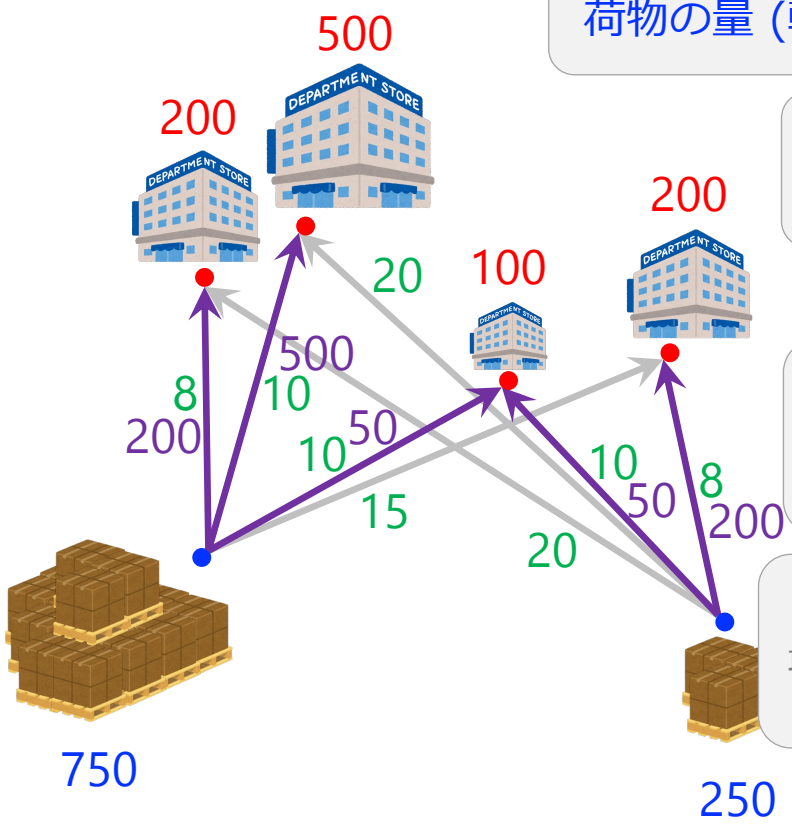
必要な情報 (2)
各拠点ペア間の輸送コスト

わかること (1)
最適な輸送計画
＝各拠点ペア間の輸送量



最適輸送問題の入出力を特産品の例で理解

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？



必要な情報 (1)
荷物の量 (輸送元) / 荷物の量 (輸送先)

必要な情報 (2)
各拠点ペア間の輸送コスト

わかること (1)
最適な輸送計画
＝各拠点ペア間の輸送量

わかること (2)
最適な輸送計画での総コスト
8×200 + 10×500 + … = 9200

最適輸送問題の入出力を特産品の例で理解

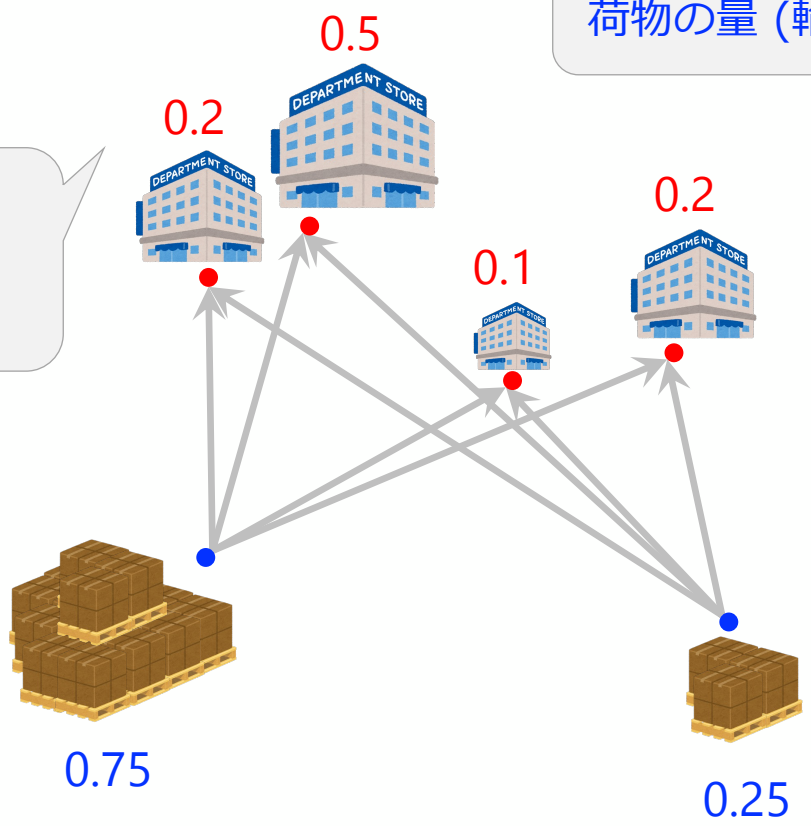
確率分布同士の距離に見えるように

数学・情報科学としての最適輸送問題は
確率分布の距離として定式化されるのが一般的

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？

必要な情報 (1)
荷物の量 (輸送元) / 荷物の量 (輸送先)

荷物の量の**合計が1**
になるように正規化
※ 情報は落ちない



最適輸送問題の入出力を特産品の例で理解

確率分布同士の距離に見えるように

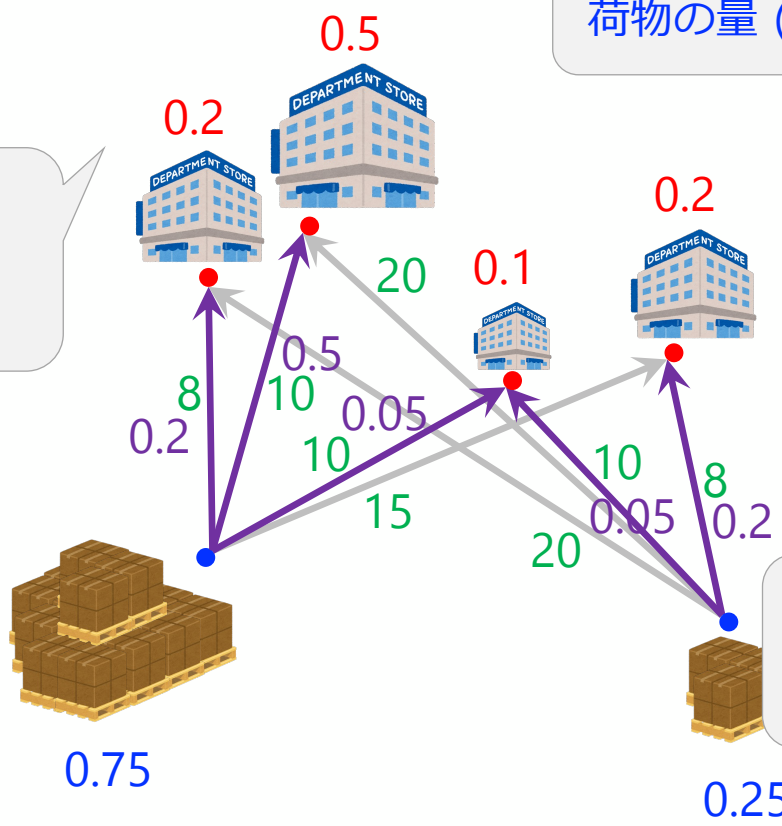
数学・情報科学としての最適輸送問題は
確率分布の距離として定式化されるのが一般的

- 工場で作った特産品を市内のデパートに届けたい.
  もっとも効率的な届け方は？

必要な情報 (1)
荷物の量 (輸送元) / 荷物の量 (輸送先)

必要な情報 (2)
各拠点ペア間の輸送コスト

荷物の量の**合計が1**
になるように正規化
※ 情報は落ちない



わかること (1)
最適な輸送計画
＝各拠点ペア間の輸送量

わかること (2)
最適な輸送計画での総コスト
8×0.2 + 10×0.5 + … = 9.2

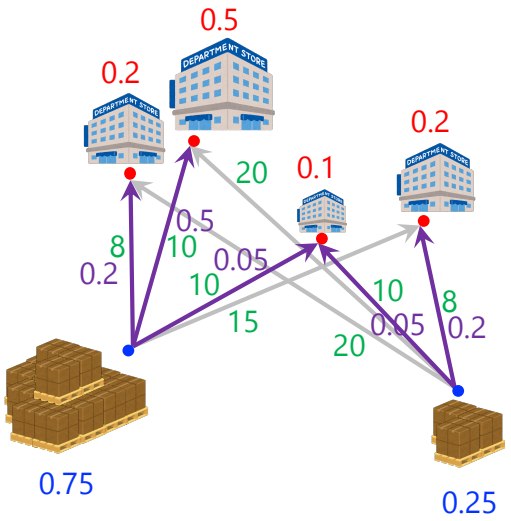
最適輸送問題の入出力

まとめ

- **入力**
  - 輸送元側の荷物の量分布
  - 輸送先側の荷物の量分布
  - 各 (輸送元, 輸送先) の輸送コスト
- **出力**
  - 最適な輸送コストを達成するような 輸送計画
    - 各 (輸送元, 輸送先) 間の輸送量
    - =ソフトな**アラインメント**
  - このときの 総輸送コスト
    - Σ 輸送コスト × 輸送量

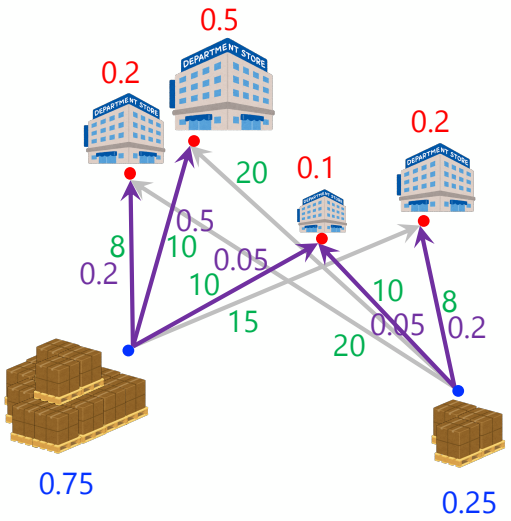
argmin

min



最適輸送問題の入出力

- **入力**
  - 輸送元側の荷物の量分布
  - 輸送先側の荷物の量分布
  - 各 (輸送元, 輸送先) の輸送コスト
- **出力**
  - 最適な輸送コストを達成するような 輸送計画
    - 各 (輸送元, 輸送先) 間の輸送量
    - =ソフトな**アラインメント**
  - このときの 総輸送コスト
    - Σ 輸送コスト × 輸送量
- → ここまで分かればライブラリを使うのは簡単 (次頁)



argmin

min

最適輸送のライブラリを用いる

POT (Python Optimal Transport) の例

まとめ

- 入力
  - `a: numpy array, (n,)`
  - `b: numpy array, (m,)`
  - `C: numpy array, (n,m)`

`a[i]`: i 番目の工場から出荷される特産品の量
※ 工場全体で合計1になるように正規化

`b[j]`: j 番目のデパートが入荷する特産品の量
※ デパート全体で合計1になるように正規化

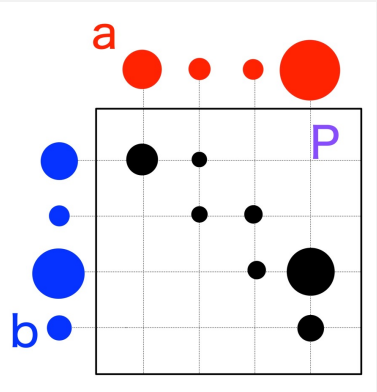
`C[i,j]`: i 番目の工場と j 番目のデパートの間の輸送コスト

- ソルバを利用して最適輸送問題を解く (関数を呼ぶだけ)
  - `P = ot.emd2(a,b,C): numpy ndarray, (n,m)`

最適な**輸送計画** (＝アラインメント)
`P[i,j]`: i 番目の工場と j 番目のデパートの間の輸送量

  - `cost = ot.emd(a,b,C): float`

最適な輸送計画での総**輸送コスト**

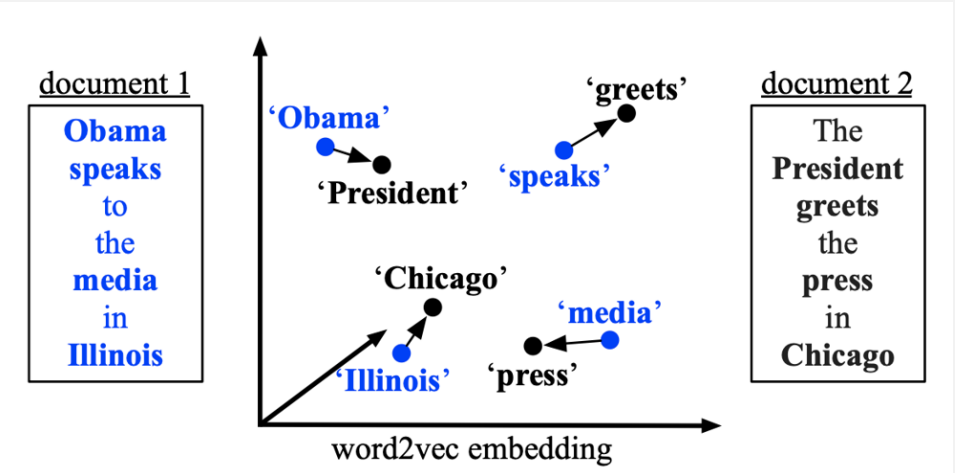


Peyré&Cuturi, Computational Optimal Transport, Fig. 2.6, 改変

Word Mover's Distance [Kusner+'15] (再)

最適輸送の問題としての入出力の確認

- 単語ベクトル空間を地図だと思ってみる
  - 文1：単語埋込空間の工場の位置
  - 文2：単語埋込空間のデパートの位置



Kusner+, From Word Embeddings To Document Distances (ICML 2015)

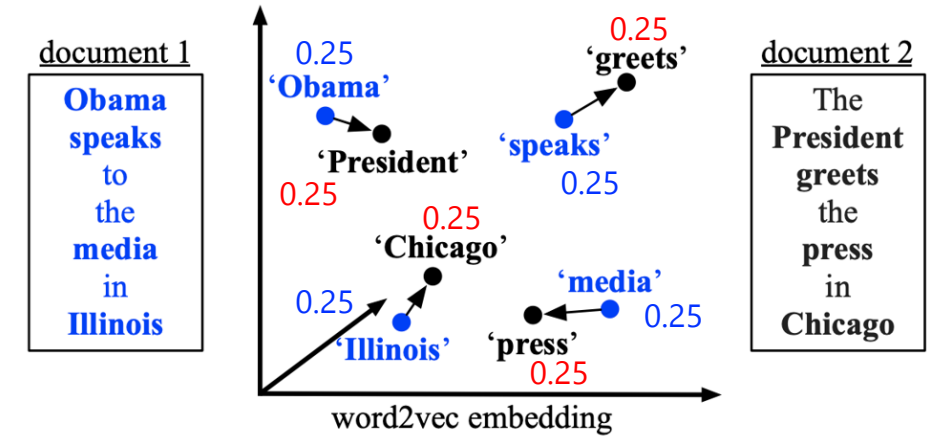
- 意味の近さの問題が輸送コストの大きさの問題に置き換わる
  - テキストの意味が似ている　↔ 低コストで輸送しきれる
  - テキストの意味が似ていない ↔ 高い輸送コストがかかる

Word Mover's Distance [Kusner+'15] (再)

最適輸送の問題としての入出力の確認

- 単語ベクトル空間を地図だと思ってみる
  - 文1：単語埋込空間の工場の位置
  - 文2：単語埋込空間のデパートの位置

必要な情報 (1)
荷物の量 (輸送元) / 荷物の量 (輸送先)
→ **各単語に等分**



Kusner+, From Word Embeddings To Document Distances (ICML 2015)

- 意味の近さの問題が輸送コストの大きさの問題に置き換わる
  - テキストの意味が似ている　↔ 低コストで輸送しきれる
  - テキストの意味が似ていない ↔ 高い輸送コストがかかる

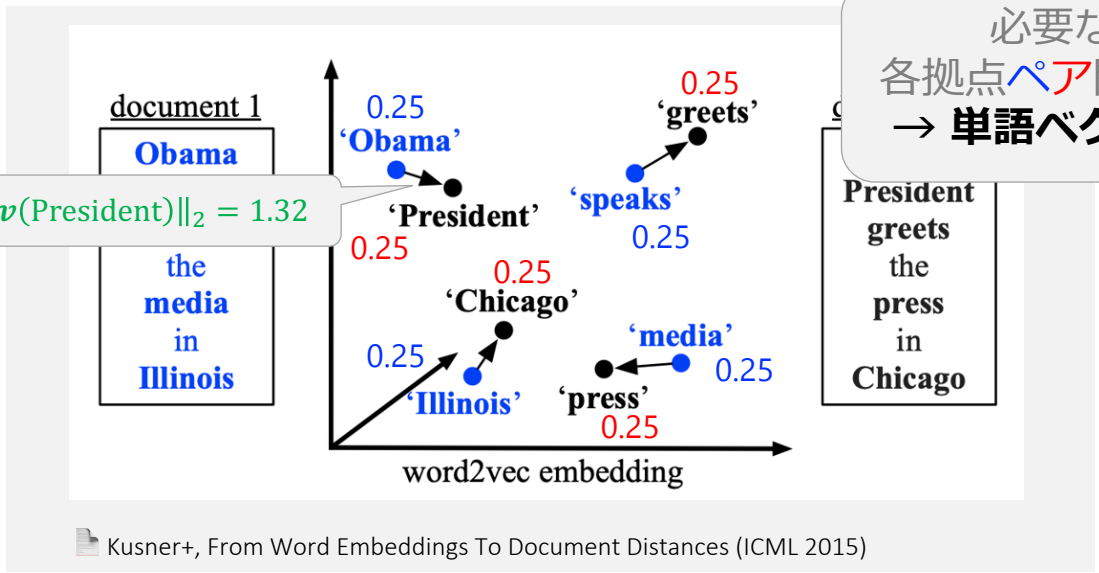
Word Mover's Distance [Kusner+'15] (再)

最適輸送の問題としての入出力の確認

- 単語ベクトル空間を地図だと思ってみる
  - 文1：単語埋込空間の工場の位置
  - 文2：単語埋込空間のデパートの位置

必要な情報 (1)
荷物の量 (輸送元) / 荷物の量 (輸送先)
→ **各単語に等分**

必要な情報 (2)
各拠点ペア間の輸送コスト
→ **単語ベクトル間の距離**



Kusner+, From Word Embeddings To Document Distances (ICML 2015)

- 意味の近さの問題が輸送コストの大きさの問題に置き換わる
  - テキストの意味が似ている　↔ 低コストで輸送しきれる
  - テキストの意味が似ていない ↔ 高い輸送コストがかかる

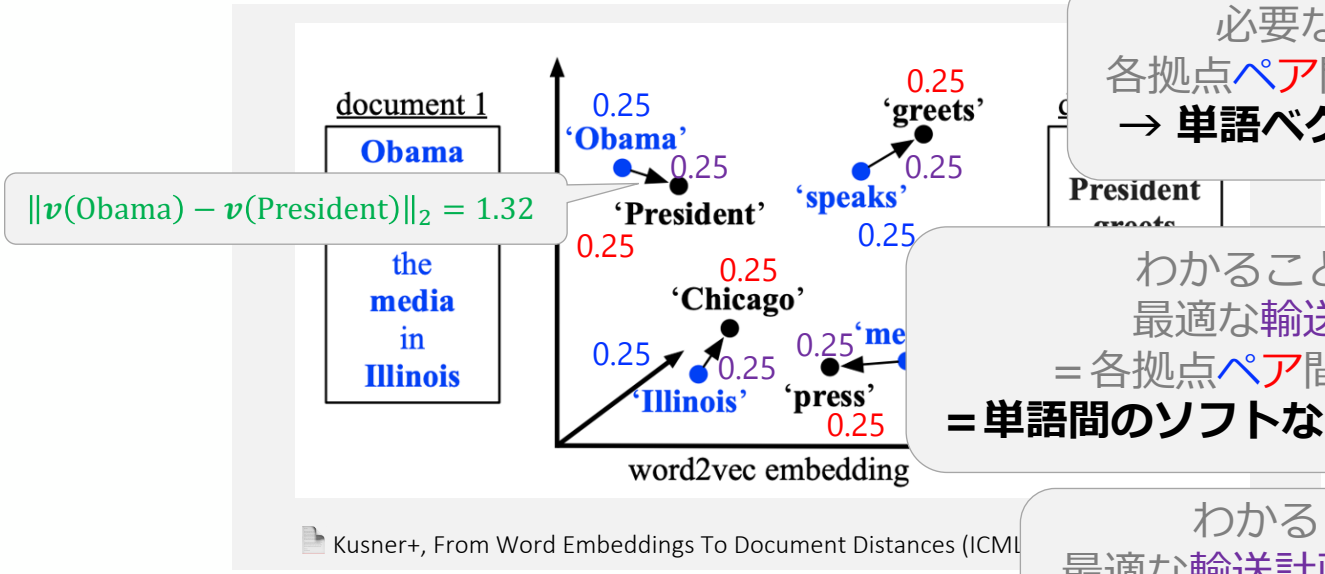
Word Mover's Distance [Kusner+'15] (再)

最適輸送の問題としての入出力の確認

- 単語ベクトル空間を地図だと思ってみる
  - 文1：単語埋込空間の工場の位置
  - 文2：単語埋込空間のデパートの位置

必要な情報 (1)
荷物の量 (輸送元) / 荷物の量 (輸送先)
→ **各単語に等分**

必要な情報 (2)
各拠点ペア間の輸送コスト
→ **単語ベクトル間の距離**



わかること (1)
最適な輸送計画
＝各拠点ペア間の輸送量
＝単語間のソフトなアラインメント

わかること (2)
最適な輸送計画での総コスト
$1.32 \times 0.25 + \cdots$
＝テキスト間の非類似度

- 意味の近さの問題が輸送コストの大きさの問
  - テキストの意味が似ている　↔ 低コストで輸送しきれる
  - テキストの意味が似ていない ↔ 高い輸送コストがかかる

Word Mover's Distance [Kusner+'15]

…を POT (Python Optimal Transport) で求める

- 入力
  - sent1 = ['Obama', 'speak', 'media', 'Illinois']
  - sent2 = ['President', 'greets', 'press', 'Chicago']
  - a = np.ones(len(sent1)) / len(sent1)
  - b = np.ones(len(sent2)) / len(sent2)
  - vecs1 = np.array([vec(word) for word in sent1])
  - vecs2 = np.array([vec(word) for word in sent2])
  - C = sklearn.metrics.pairwise_distances(vecs1, vecs2, metric='euclidean')

適当な単語ベクトルを用意しておく

一様な荷物の量分布 (1)

一様な荷物の量分布 (2)

輸送コスト (L2距離)

- ソルバを利用して最適輸送問題を解く
  - P = ot.emd2(a,b,C)
  - WMD = ot.emd(a,b,C)

輸送計画行列

輸送コスト

Word Mover’s Distance 再考

入力（荷物の量の比率，輸送コスト）は使う人次第
カスタマイズしながら使ってみる

目次

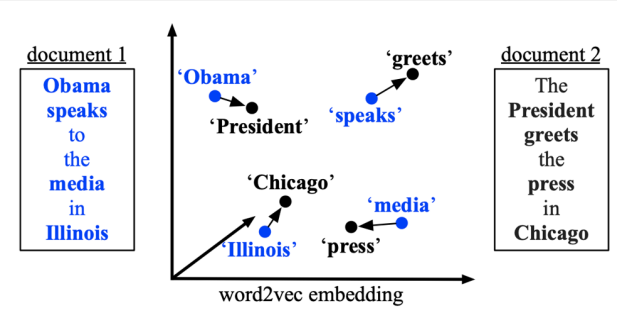
- **基礎編**：もっとも標準的な形式の最適輸送問題を知る
  - 【最適輸送を知る】1：荷物の配置換えという気持ちを知る
    - ゴール：最適輸送の話題が出てきても怖くなくなる
  - なぜ今最適輸送なのか
  - 【最適輸送を知る】2：入出力を形式的に理解する
    - ゴール：最適輸送を自分の研究・開発プロジェクトで使えるようになる
  - 入力をカスタマイズしながら使う　イマココ
  - 【最適輸送を知る】3：線形計画問題としての定式化を理解する
    - ゴール：最適輸送を道具として用いている論文の式を読めるようになる
  - 用語に関する落穂拾い
    - ワッサーシュタイン距離，Earth Mover’s Distance
  - 前半のまとめ
- **応用編**：自動微分可能な最適輸送 / 最適輸送の変種 / 全体まとめ

改めて：Word Mover's Distance の入出力

- **元の問題：文の非類似度を測る**
  - 入力
    - 文 $s = (w_1, \ldots, w_n)$，対応する単語ベクトル $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$
    - 文 $s' = (w'_1, \ldots, w'_m)$，対応する単語ベクトル $(\boldsymbol{w}'_1, \ldots, \boldsymbol{w}'_m)$
  - 出力
    - 類似度 $\mathrm{sim}(s, s')$

Kusner+, From Word Embeddings To Document Distances (ICML 2015)

改めて：Word Mover's Distance の入出力



Kusner+, From Word Embeddings To Document Distances

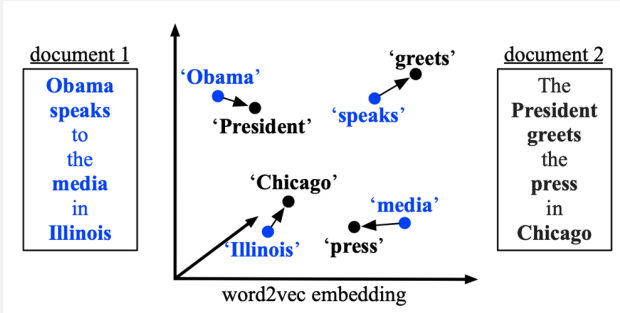
- **元の問題：文の非類似度を測る**
  - 入力
    - 文 $s = (w_1, \ldots, w_n)$，対応する単語ベクトル $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$
    - 文 $s' = (w'_1, \ldots, w'_m)$，対応する単語ベクトル $(\boldsymbol{w}'_1, \ldots, \boldsymbol{w}'_m)$
  - 出力
    - 類似度 $\mathrm{sim}(s, s')$
- **最適輸送問題への読み替え** [Kusner+'15]
  - 入力
    - 確率分布：$\boldsymbol{a} = (1/n, \ldots, 1/n) \in [0,1]^n$
    - 確率分布：$\boldsymbol{b} = (1/m, \ldots, 1/m) \in [0,1]^m$
    - 輸送コスト：$\boldsymbol{C}_{ij} = \|\boldsymbol{w}_i - \boldsymbol{w}'_j\|_2$
  - 出力
    - 最適輸送コスト（非類似度）$\mathrm{OT}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C})$

各単語の重みは均等

単語の非類似度をユークリッド距離で測る

Kusner+, From Word Embeddings To Document Distances (ICML 2015)

改めて：Word Mover's Distance の入出力



Kusner+, From Word Embeddings To Document Distances

- **元の問題：文の非類似度を測る**
  - 入力
    - 文 $s = (w_1, \ldots, w_n)$，対応する単語ベクトル $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$
    - 文 $s' = (w'_1, \ldots, w'_m)$，対応する単語ベクトル $(\boldsymbol{w}'_1, \ldots, \boldsymbol{w}'_m)$
  - 出力
    - 類似度 $\mathrm{sim}(s, s')$
- **最適輸送問題への読み替え** [Kusner+'15]
  - 入力
    - 確率分布：$\boldsymbol{a} = (1/n, \ldots, 1/n) \in [0,1]^n$
    - 確率分布：$\boldsymbol{b} = (1/m, \ldots, 1/m) \in [0,1]^m$
    - 輸送コスト：$\boldsymbol{C}_{ij} = \|\boldsymbol{w}_i - \boldsymbol{w}'_j\|_2$
  - 出力
    - 最適輸送コスト（非類似度）$\mathrm{OT}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C})$

各単語の重みは均等

単語の非類似度をユークリッド距離で測る

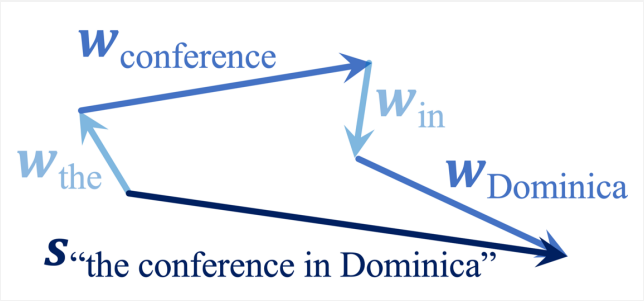
入力はこれで良い？

＝言語の問題を最適輸送の言葉に翻訳する方法はこれが最適？

Kusner+, From Word Embeddings To Document Distances (ICML 2015)

重みも輸送コストも自由に決めて良い

- **Word Mover's Distance** [Kusner+'15
  - 確率分布：$\boldsymbol{a} = (1/n, \ldots, 1/n)$
  - 確率分布：$\boldsymbol{b} = (1/m, \ldots, 1/m)$
  - 輸送コスト：$\boldsymbol{C}_{ij} = \|\boldsymbol{w}_i - \boldsymbol{w}'_j\|_2$ ：w_i と w'_j (



Yokoi+, Word Rotator's Distance (EMNLP 2020)

単語ベクトルの長さには
単語の重要度が近似的に埋め込まれている
[Schakel&Wilson'15], [大山+'22]

- **変更例：Word Rotator's Distance** [Yokoi+'20]
  - 確率分布：$\boldsymbol{a} \propto (\|\boldsymbol{w}_1\|_2, \ldots, \|\boldsymbol{w}_n\|_2)$
  - 確率分布：$\boldsymbol{b} \propto (\|\boldsymbol{w}'_1\|_2, \ldots, \|\boldsymbol{w}'_m\|_2)$
  - 輸送コスト：$\boldsymbol{C}_{ij} = 1 - \cos(\boldsymbol{w}_i, \boldsymbol{w}'_j)$ ：w_i と w'_i の非類似度

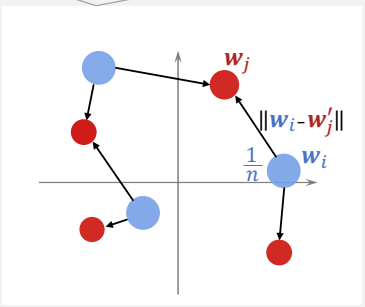
なす角 (※) が単語の非類似度として有用
※ 単語ベクトルの長さを無視した距離

Kusner+, From Word Embeddings To Document Distances (ICML 2015)
Yokoi+, Word Rotator's Distance (EMNLP 2020)
Schakel&Wilson, Measuring Word Significance using Distributed Representations of Words (arXiv 2015)
大山+, 単語ベクトルの長さは意味の強さを表す (NLP 2022)

重みも輸送コストも自由に決めて良い

数学・数理科学の目線からすると
おそらく埋込空間の点群 (単語ベクトル集合) 間の
最適輸送コストとしてもっとも自然な選択

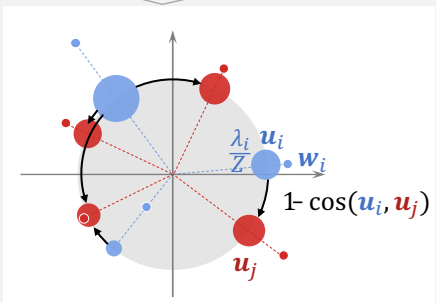
- **Word Mover's Distance** [Kusner+'15]
  - 確率分布：$\boldsymbol{a} = (1/n, \ldots, 1/n)$
  - 確率分布：$\boldsymbol{b} = (1/m, \ldots, 1/m)$
  - 輸送コスト：$\boldsymbol{C}_{ij} = \|\boldsymbol{w}_i - \boldsymbol{w}'_j\|_2$



Yokoi+, Word Rotator's Distance

自然言語処理目線で分布・コストを選ぶと
「埋込空間上の超球上で輸送」の方が自然

- **例：Word Rotator's Distance** [Yokoi+'20]
  - 確率分布：$\boldsymbol{a} \propto (\|\boldsymbol{w}_1\|_2, \ldots, \|\boldsymbol{w}_n\|_2)$
  - 確率分布：$\boldsymbol{b} \propto (\|\boldsymbol{w}'_1\|_2, \ldots, \|\boldsymbol{w}'_m\|_2)$
  - 輸送コスト：$\boldsymbol{C}_{ij} = 1 - \cos(\boldsymbol{w}_i, \boldsymbol{w}'_j)$

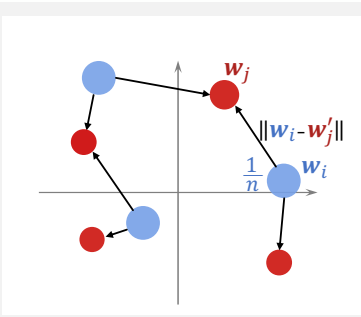


Yokoi+, Word Rotator's Distance

Kusner+, From Word Embeddings To Document Distances (ICML 2015)
Yokoi+, Word Rotator's Distance (EMNLP 2020)

重みも輸送コストも自由に決めて良い

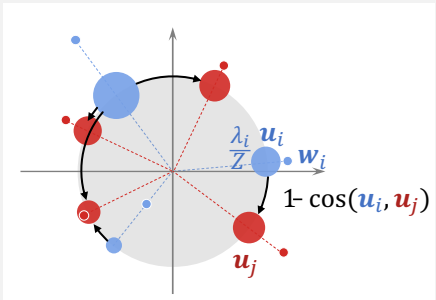
- **Word Mover's Distance** [Kusner+'15]
  - 確率分布：$\boldsymbol{a} = (1/n, \ldots, 1/n)$
  - 確率分布：$\boldsymbol{b} = (1/m, \ldots, 1/m)$
  - 輸送コスト：$\boldsymbol{C}_{ij} = \|\boldsymbol{w}_i - \boldsymbol{w}'_j\|_2$



Yokoi+, Word Rotator's Distance

文類似度タスクで10ポイント程度性能が改善 [Yokoi+'20]
k-NN文書分類でも安定した性能改善 [Smirnov&Yamshchikov'22]

- **例：Word Rotator's Distance** [Yokoi+'20]
  - 確率分布：$\boldsymbol{a} \propto (\|\boldsymbol{w}_1\|_2, \ldots, \|\boldsymbol{w}_n\|_2)$
  - 確率分布：$\boldsymbol{b} \propto (\|\boldsymbol{w}'_1\|_2, \ldots, \|\boldsymbol{w}'_m\|_2)$
  - 輸送コスト：$\boldsymbol{C}_{ij} = 1 - \cos(\boldsymbol{w}_i, \boldsymbol{w}'_j)$



Yokoi+, Word Rotator's Distance

Kusner+, From Word Embeddings To Document Distances (ICML 2015)
Yokoi+, Word Rotator's Distance (EMNLP 2020)
Smirnov&Yamshchikov, Moving Other Way: Exploring Word Mover Distance Extensions (arXiv 2022)

重みも輸送コストも自由に決めて良い

まとめ

- **Take-home message**
  - 解きたい問題の気持ちになって重みと輸送コスト (入力) を決める
  - ドメイン知識を使う
- **テキスト (単語ベクトル集合) 間の類似度**
  - 重み
    - ノルムでの重み付けは有用 [Yokoi+'20]
    - TF-IDF, SIF [Arora+'17] など，加法構成で用いられる重み付けも有用
    - ストップワード除去も {0,1} の重み付けに見える [Kusner+'15]
    - 重複させても良い
  - 輸送コスト
    - 単語ベクトル (含 BERT) を使う場合，L2 よりもコサインが安定しそう
    - なお本家 WMD も 実験では 正規化された単語ベクトルを使っている (!) [Sato+'21, Sato'22]
      - ※ 事前に単語ベクトルを正規化した上での L2 コスト $c_E(\bar{w}, \bar{w}') = \|\bar{w} - \bar{w}'\|_2 \geq 0$ は コサインコスト $c_{\cos}(\bar{w}, \bar{w}') = 1 - \cos(\bar{w}, \bar{w}') \geq 0$ と単調：$c_E(\bar{w}, \bar{w}') = \sqrt{2c_{\cos}(\bar{w}, \bar{w}')}$

Yokoi+, Word Rotator's Distance (EMNLP 2020) Arora+'17, <SIF> A Simple but Tough-to-Beat Baseline for Sentence Embeddings (ICLR 2017)
Kusner+, From Word Embeddings To Document Distances (ICML 2015) Sato+, Re-evaluating Word Mover's Distance (arXiv 2021)
personal communication with Ryoma Sato-san (2022)

最適輸送を知る さらに丁寧に

線形計画としての定式化を理解する

式が少し登場します．
このあたりの話がわかると，論文が読みやすくなり，またこのあとの
セクションで説明する様々な概念を理解しやすくなります．
置いていかれても大丈夫です．今後も視覚的な説明を常に付記します．

目次

- **基礎編**：もっとも標準的な形式の最適輸送問題を知る
  - 【最適輸送を知る】1：荷物の配置換えという気持ちを知る
    - ゴール：最適輸送の話題が出てきても怖くなくなる
  - なぜ今最適輸送なのか
  - 【最適輸送を知る】2：入出力を形式的に理解する
    - ゴール：最適輸送を自分の研究・開発プロジェクトで使えるようになる
  - 入力をカスタマイズしながら使う
  - 【最適輸送を知る】3：線形計画問題としての定式化を理解する
    - ゴール：最適輸送を道具として用いている論文の式を読めるようになる
  - 用語に関する落穂拾い
    - ワッサーシュタイン距離，Earth Mover's Distance
  - 前半のまとめ
- **応用編**：自動微分可能な最適輸送 / 最適輸送の変種 / 全体まとめ

イマココ

最適輸送の定式化

論文によく登場する式を読めるようになる

- こういうの

$$\mathbf{U}(\mathbf{a},\mathbf{b}) \overset{\text{def.}}{=} \left\{ \mathbf{P} \in \mathbb{R}_+^{n \times m} \ : \ \mathbf{P}\mathbb{1}_m = \mathbf{a} \quad \text{and} \quad \mathbf{P}^{\mathrm{T}}\mathbb{1}_n = \mathbf{b} \right\}, \tag{2.10}$$

$$\mathrm{L}_{\mathbf{C}}(\mathbf{a},\mathbf{b}) \overset{\text{def.}}{=} \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle \overset{\text{def.}}{=} \sum_{i,j} \mathbf{C}_{i,j}\mathbf{P}_{i,j}. \tag{2.11}$$

Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019)

最適輸送の定式化

論文によく登場する式を読めるようになる

- こういうの

$$\mathbf{U}(\mathbf{a}, \mathbf{b}) \stackrel{\text{def.}}{=} \left\{ \mathbf{P} \in \mathbb{R}_+^{n \times m} \;:\; \mathbf{P}\mathbb{1}_m = \mathbf{a} \quad \text{and} \quad \mathbf{P}^{\mathrm{T}}\mathbb{1}_n = \mathbf{b} \right\}, \tag{2.10}$$

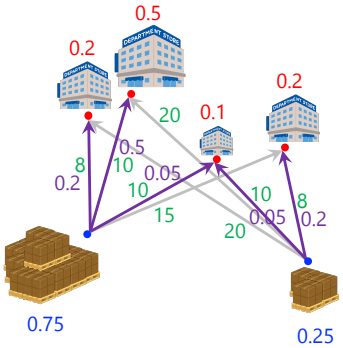
$$\mathrm{L}_{\mathbf{C}}(\mathbf{a}, \mathbf{b}) \stackrel{\text{def.}}{=} \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a}, \mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle \stackrel{\text{def.}}{=} \sum_{i,j} \mathbf{C}_{i,j} \mathbf{P}_{i,j}. \tag{2.11}$$

Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019)

- 登場人物は3人だけ
  - ベクトル $\boldsymbol{a}$
  - ベクトル $\boldsymbol{b}$
  - 行列 $\boldsymbol{C}$

最適輸送の定式化

論文によく登場する式を読めるようになる



- こういうの

$$\mathbf{U}(\mathbf{a},\mathbf{b}) \overset{\text{def.}}{=} \left\{ \mathbf{P} \in \mathbb{R}_+^{n\times m} \;:\; \mathbf{P}\mathbb{1}_m = \mathbf{a} \quad \text{and} \quad \mathbf{P}^{\mathrm{T}}\mathbb{1}_n = \mathbf{b} \right\}, \tag{2.10}$$

Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019)

- 登場人物は3人だけ
  - ベクトル $\boldsymbol{a} = (a_1, \dots, a_n) \in [0,1]^n,\ \sum_i a_i = 1$
  - ベクトル $\boldsymbol{b} = (b_1, \dots, b_m) \in [0,1]^m,\ \sum_j b_j = 1$
  - 行列 $\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n\times m}$

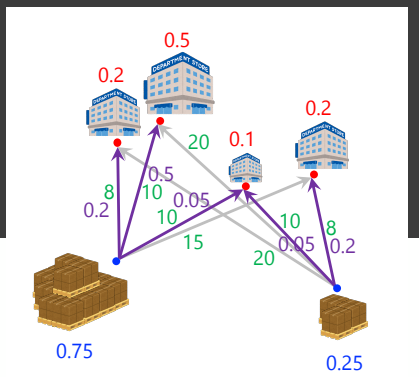
$a_i \in \mathbb{R}_{\geq 0}$: i 番目の工場から出荷される特産品の量の分布

$b_j \in \mathbb{R}_{\geq 0}$: j 番目のデパートが入荷する特産品の量の分布

$C_{ij} \in \mathbb{R}_{\geq 0}$: i 番目の工場と j 番目のデパートの間の輸送コスト

最適輸送の定式化

論文によく登場する式を読めるようになる



- 出力

「輸送計画 $\boldsymbol{P} \in \mathbb{R}_{\geq 0}^{n \times m}$ を色々検討してみましょう」の意
$\boldsymbol{P}_{ij} \in \mathbb{R}_{\geq 0}$: i 番目の工場と j 番目のデパートの間の輸送量

$$\mathbf{U}(\mathbf{a}, \mathbf{b}) \stackrel{\text{def.}}{=} \left\{ \mathbf{P} \in \mathbb{R}_{+}^{n \times m} \ : \ \mathbf{P}\mathbb{1}_m = \mathbf{a} \quad \text{and} \quad \mathbf{P}^{\mathrm{T}}\mathbb{1}_n = \mathbf{b} \right\}, \tag{2.10}$$

「荷物が余ったり足りなくなったりしないように」の意

$$\mathrm{L}_{\mathbf{C}}(\mathbf{a}, \mathbf{b}) \stackrel{\text{def.}}{=} \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a}, \mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle \stackrel{\text{def.}}{=} \sum_{i,j} \mathbf{C}_{i,j} \mathbf{P}_{i,j}. \tag{2.11}$$

Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019)

- 入力
  - ベクトル $\boldsymbol{a} = (a_1, \ldots, a_n) \in [0,1]^n, \ \sum_i a_i = 1$
  - ベクトル $\boldsymbol{b} = (b_1, \ldots, b_m) \in [0,1]^m, \ \sum_i b_j = 1$
  - 行列 $\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n \times m}$

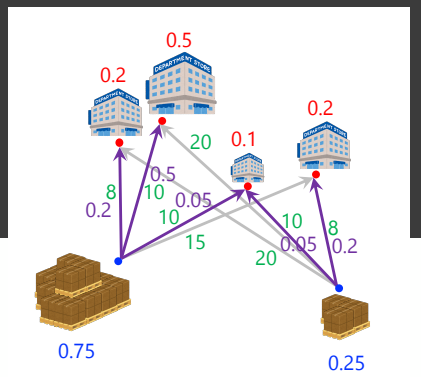
$a_i \in \mathbb{R}_{\geq 0}$: i 番目の工場から出荷される特産品の量の分布

$b_j \in \mathbb{R}_{\geq 0}$: j 番目のデパートが入荷する特産品の量の分布

$\boldsymbol{C}_{ij} \in \mathbb{R}_{\geq 0}$: i 番目の工場と j 番目のデパートの間の輸送コスト

最適輸送の定式化（線形計画）

論文によく登場する式を読めるようになる



- 出力

「輸送計画 $\boldsymbol{P} \in \mathbb{R}_{\geq 0}^{n \times m}$ を色々検討してみましょう」の意
$\boldsymbol{P}_{ij} \in \mathbb{R}_{\geq 0}$：$i$ 番目の工場と j 番目のデパートの間の輸送量

$$\mathbf{U}(\mathbf{a}, \mathbf{b}) \stackrel{\text{def.}}{=} \left\{ \mathbf{P} \in \mathbb{R}_{+}^{n \times m} \ : \ \mathbf{P}\mathbb{1}_m = \mathbf{a} \ \text{ and } \ \mathbf{P}^{\mathrm{T}}\mathbb{1}_n = \mathbf{b} \right\}, \qquad (2.10)$$

「荷物が余ったり足りなくなったりしないように」の意

$$\mathrm{L}_{\mathbf{C}}(\mathbf{a}, \mathbf{b}) \stackrel{\text{def.}}{=} \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a}, \mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle \stackrel{\text{def.}}{=} \sum_{i,j} \mathbf{C}_{i,j} \mathbf{P}_{i,j}. \qquad (2.11)$$

最適輸送コスト

総輸送コスト「$\sum_{\text{拠点ペア}_{ij}}$ 輸送コスト $\boldsymbol{C}_{ij}$ × 輸送量 $\boldsymbol{P}_{ij}$」を最小化

Peyré&Cuturi, Computational Optim

- 入力
  - ベクトル $\boldsymbol{a} = (a_1, \ldots, a_n) \in [0,1]^n, \ \sum_i a_i = 1$
  - ベクトル $\boldsymbol{b} = (b_1, \ldots, b_m) \in [0,1]^m, \ \sum_i b_j = 1$
  - 行列 $\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n \times m}$

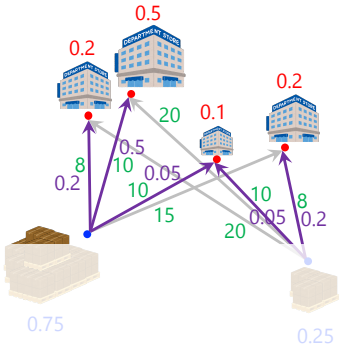
$a_i \in \mathbb{R}_{\geq 0}$：$i$ 番目の工場から出荷される特産品の量の分布

$b_j \in \mathbb{R}_{\geq 0}$：$j$ 番目のデパートが入荷する特産品の量の分布

$\boldsymbol{C}_{ij} \in \mathbb{R}_{\geq 0}$：$i$ 番目の工場と j 番目のデパートの間の輸送コスト

最適輸送の定式化（線形計画）

論文によく登場する式を読めるようになる



- 出力

「輸送計画 $P \in \mathbb{R}_{\geq 0}^{n \times m}$ を色々検討してみましょう」の意

$P_{ij} \in \mathbb{R}_{\geq 0}$：$i$ 番目の工場と j 番目のデパートの間の輸送量

$$\mathbf{U}(\mathbf{a}, \mathbf{b}) \stackrel{\text{def.}}{=} \left\{ \mathbf{P} \in \mathbb{R}_{+}^{n \times m} \;:\; \mathbf{P}\mathbb{1}_m = \mathbf{a} \;\text{ and }\; \mathbf{P}^{\mathrm{T}}\mathbb{1}_n = \mathbf{b} \right\}, \qquad (2.10)$$

「荷物が余ったり足りなくなったりしないように」の意

$$\mathrm{L_C}(\mathbf{a}, \mathbf{b}) \qquad (2.11)$$

最適輸送コス

C_{ij} × 輸送量 P_{ij}」を最小化

Peyré&Cuturi, Computational

まずは入力だけわかれば，お気持ちはOK
「あとは最適輸送というブラックボックスを使って
最適な輸送プランと輸送コストを求めるのね」

- 入力
  - ベクトル $\boldsymbol{a} = (a_1, \ldots, a_n) \in [0,1]^n,\ \sum_i a_i = 1$
  - ベクトル $\boldsymbol{b} = (b_1, \ldots, b_m) \in [0,1]^m,\ \sum_i b_j = 1$
  - 行列 $\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n \times m}$

$a_i \in \mathbb{R}_{\geq 0}$：$i$ 番目の工場から出荷される特産品の量の分布

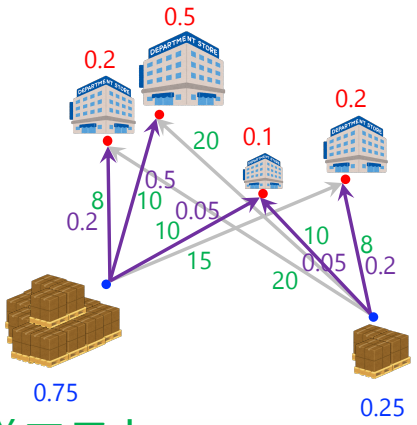
$b_j \in \mathbb{R}_{\geq 0}$：$j$ 番目のデパートが入荷する特産品の量の分布

$C_{ij} \in \mathbb{R}_{\geq 0}$：$i$ 番目の工場と j 番目のデパートの間の輸送コスト

最適輸送問題の定式化（線形計画問題）

論文に出てくる式の読み方

まとめ



- **入力**
  - 確率分布：$\boldsymbol{a} = (a_1, \ldots, a_n) \in [0,1]^n,\ \sum_i a_i = 1$
  - 確率分布：$\boldsymbol{b} = (b_1, \ldots, b_m) \in [0,1]^m,\ \sum_i b_j = 1$
  - 輸送コスト：$\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n \times m}$
    - $\boldsymbol{C}_{ij} \in \mathbb{R}_{\geq 0}$：$a_i$ に対応する拠点と b_j に対応する拠点の間の輸送コスト
- **制約**
  - $\boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b}) \coloneqq \{\boldsymbol{P} \in \mathbb{R}_{\geq 0}^{n \times m} \mid \boldsymbol{P}\mathbf{1} = \boldsymbol{a}, \boldsymbol{P}^\top \mathbf{1} = \boldsymbol{b}\}$
    - 「荷物が余ったり足りなくなったりしない」…とちゃんと書いてくれている
- **出力**
  - 輸送計画：$\boldsymbol{P}^* \coloneqq \underset{\boldsymbol{T} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})}{\operatorname{argmin}} \sum_{ij} \boldsymbol{C}_{ij} \boldsymbol{P}_{ij} \in \mathbb{R}_{\geq 0}^{n \times m}$
    - $\boldsymbol{P}^*_{ij} \in \mathbb{R}_{\geq 0}$：$a_i$ に対応する拠点と b_j に対応する拠点の間の輸送量
    - ソフトなアラインメント
  - 最適輸送コスト：$\mathrm{OT}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}) \coloneqq \min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_{ij} \boldsymbol{C}_{ij} \boldsymbol{P}_{ij} \in \mathbb{R}_{\geq 0}$

最適輸送問題の定式化（線形計画問題）

まとめ

+ 表記揺れの補足

入力が確率測度であることを強調してこう書く場合も：
$\alpha = \sum_i a_i \delta[\boldsymbol{x}_i]$, $\beta = \sum_j b_i \delta[\boldsymbol{y}_i]$, $c(\boldsymbol{x}_i, \boldsymbol{y}_j) = \cdots$

$a_i \delta[\boldsymbol{x}_i]$ の読み方：位置 $\boldsymbol{x}_i$ に荷物が a_i だけ置かれている．$a_i \delta_{\boldsymbol{x}_i}$ も同様．
位置 $\boldsymbol{x}_i, \boldsymbol{y}_j$ とコスト関数 $c(\cdot,\cdot)$ がわかれば各 $\boldsymbol{C}_{ij}$ も定まるので結局同じこと．

- **入力**
  - 確率分布：$\boldsymbol{a} = (a_1, \dots, a_n) \in [0,1]^n, \ \sum_i a_i = 1$
  - 確率分布：$\boldsymbol{b} = (b_1, \dots, b_m) \in [0,1]^m, \ \sum_i b_j = 1$
  - 輸送コスト：$\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n \times m}$

「$\boldsymbol{P}$」の気持ち：輸送計画行列は行方向/列方向の周辺分布がそれぞれ $\boldsymbol{a}, \boldsymbol{b}$ である同時分布．
$\boldsymbol{P}$ のかわりに $\boldsymbol{T}$ (輸送計画行列; Transportation plan) を 用いることも．
また $\boldsymbol{U}$ の代わりに $\mathcal{C}$ や Π を用いることも．$\mathcal{C}$ はカップリングの意．

- **制約**
  - $\boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b}) \coloneqq \{\boldsymbol{P} \in \mathbb{R}_{\geq 0}^{n \times m} \mid \boldsymbol{P}\mathbf{1} = \boldsymbol{a}, \boldsymbol{P}^\top \mathbf{1} = \boldsymbol{b}\}$

行列同士の内積 (Frobenius inner product) であることを強調して
$\langle \boldsymbol{C}, \boldsymbol{P} \rangle \coloneqq \sum_{ij} \boldsymbol{C}_{ij} \boldsymbol{P}_{ij}$ という記号が使われることもしばしば．

- **出力**
  - 輸送計画：$\boldsymbol{P}^* \coloneqq \underset{\boldsymbol{T} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})}{\operatorname{argmin}} \sum_{ij} \boldsymbol{C}_{ij} \boldsymbol{P}_{ij} \ \in \mathbb{R}_{\geq 0}^{n \times m}$
  - 最適輸送コスト：$\mathrm{OT}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}) \coloneqq \min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_{ij} \boldsymbol{C}_{ij} \boldsymbol{P}_{ij} \ \in \mathbb{R}_{\geq 0}$

用語に関する落穂拾い

ワッサーシュタイン距離，Earth Mover’s Distance

目次

- **基礎編**：もっとも標準的な形式の最適輸送問題を知る
  - 【最適輸送を知る】1：荷物の配置換えという気持ちを知る
    - ゴール：最適輸送の話題が出てきても怖くなくなる
  - なぜ今最適輸送なのか
  - 【最適輸送を知る】2：入出力を形式的に理解する
    - ゴール：最適輸送を自分の研究・開発プロジェクトで使えるようになる
  - 入力をカスタマイズしながら使う
  - 【最適輸送を知る】3：線形計画問題としての定式化を理解する
    - ゴール：最適輸送を道具として用いている論文の式を読めるようになる
  - 用語に関する落穂拾い　イマココ
    - ワッサーシュタイン距離，Earth Mover’s Distance
  - 前半のまとめ
- **応用編**：自動微分可能な最適輸送 / 最適輸送の変種 / 全体まとめ

ワッサーシュタイン距離

- **ワッサーシュタイン距離**
  - 最適輸送コストの特殊ケース
  - コスト $\boldsymbol{C}_{ij} = c(\boldsymbol{x}_i, \boldsymbol{y}_j)$ が 点間の距離 となっているとき，

    最適輸送コストは 確率分布 (点群) 間の距離 を定める

$$W(\boldsymbol{a}, \boldsymbol{b}; \boldsymbol{C}) \coloneqq \min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_i \sum_j c(\boldsymbol{x}_i, \boldsymbol{y}_j)\, \boldsymbol{P}_{ij}$$

ワッサーシュタイン距離

一般に最適輸送の計算に用いるコストは
距離でなくて良い

- $\forall x, y.\ c(x, y) = 0 \Leftrightarrow x = y$
- $\forall x, y.\ c(x, y) = c(y, x)$
- $\forall x, y, z.\ c(x, y) + c(y, z) \geq c(x, z)$

- **ワッサーシュタイン距離**
  - 最適輸送コストの特殊ケース
  - コスト $C_{ij} = c(x_i, y_j)$ が 点間の距離 となっているとき，

    最適輸送コストは 確率分布 (点群) 間の距離 を定める

$$W(a, b; C) \coloneqq \min_{P \in U(a,b)} \sum_i \sum_j c(x_i, y_j) P_{ij}$$

ワッサーシュタイン距離

一般に最適輸送の計算に用いるコストは
距離でなくて良い

- $\forall x, y.\ c(x, y) = 0 \Leftrightarrow x = y$
- $\forall x, y.\ c(x, y) = c(y, x)$
- $\forall x, y, z.\ c(x, y) + c(y, z) \geq c(x, z)$

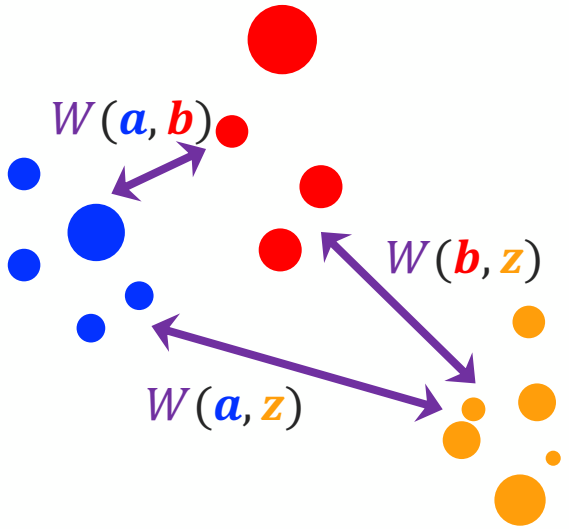
- **ワッサーシュタイン距離**
  - 最適輸送コストの特殊ケース
  - コスト $C_{ij} = c(x_i, y_j)$ が 点間の距離 となっているとき，

    最適輸送コストは 確率分布 (点群) 間の距離 を定める

$$W(a, b; C) := \min_{P \in U(a,b)} \sum_i \sum_j c(x_i, y_j) P_{ij}$$

- $\forall a, b.\ W(a, b) = 0 \Leftrightarrow a = b$
- $\forall a, b.\ W(a, b) = W(b, a)$
- $\forall a, b, z.\ W(a, b) + W(b, z) \geq W(a, z)$

最適輸送コストを確率分布間の**距離**尺度として
用いることができる



p-ワッサーシュタイン距離

まとめ

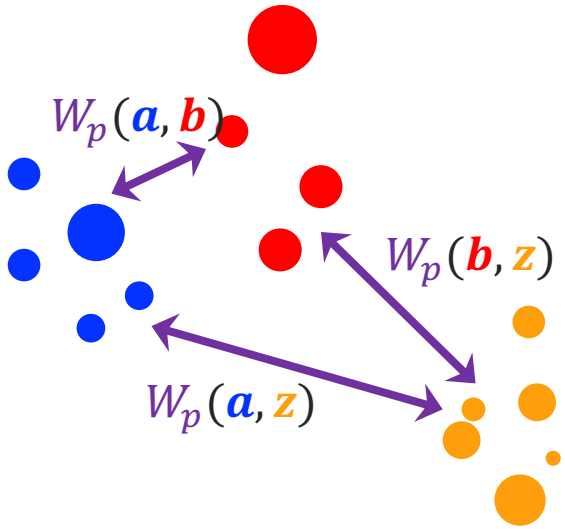
- **p-ワッサーシュタイン距離**
  - 最適輸送コストの特殊ケース
  - コスト $\boldsymbol{C}_{ij} = c(\boldsymbol{x}_i, \boldsymbol{y}_j)$ が 点間の距離 となっているとき，

    以下は 確率分布 (点群) 間の距離 を定める

ここと

ここに p がつく，よくある形式

$$W_p(\boldsymbol{a}, \boldsymbol{b}; \boldsymbol{C}) \coloneqq \left(\min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_i \sum_j c(\boldsymbol{x}_i, \boldsymbol{y}_j)^p \boldsymbol{P}_{ij} \right)^{1/p}$$



p-ワッサーシュタイン距離

- **p-ワッサーシュタイン距離**
  - 最適輸送コストの特殊ケース
  - コスト $\boldsymbol{C}_{ij} = c(\boldsymbol{x}_i, \boldsymbol{y}_j)$ が 点間の距離 となっているとき，

    以下は 確率分布 (点群) 間の距離 を定める

$$W_p(\boldsymbol{a}, \boldsymbol{b}; \boldsymbol{C}) \coloneqq \left(\min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_i \sum_j c(\boldsymbol{x}_i, \boldsymbol{y}_j)^p \boldsymbol{P}_{ij} \right)^{1/p}$$

  - 先ほどの p なしバージョンは **1-ワッサーシュタイン距離**
    - Word Mover's Distance は，コストとして距離そのもの (ユークリッド距離) を用い，さらに $p = 1$ とした量なので，1-ワッサーシュタイン距離
  - 1-ワッサーシュタイン距離や 2-ワッサーシュタイン距離がよく用いられる

Earth mover’s distance

- **Earth mover’s distance**
  - おそらく未定義語.
  - 画像処理で1-ワッサーシュタイン距離の呼称として使われはじめ [Rubner+’97; ’98; ’00]，情報科学を中心に広まっている.
    - 命名は Jorge Stolfi の模様 [Rubner+’97 (§3, 謝辞)][Rubner+’00 (§1)]
  - いまは，一般の最適輸送コストないしワッサーシュタイン距離の意味で使われているように見える.
    - 荷物の配置変更という最適輸送のイメージを喚起しやすいから?
    - 最適輸送コスト (optimal transport cost) やワッサーシュタイン距離 (Wasserstein distance) という名前が “怖い” ので緩和用?

Rubner+, The Earth Mover's Distance, Multi-Dimensional Scaling, and Color-Based Image Retrieval (Image Understanding Workshop 1997)
Rubner+, A metric for distributions with applications to image databases (ICCV 1998)
Rubner+, The Earth Mover's Distance as a Metric for Image Retrieval (Int. J. Comput. Vis. 2000)

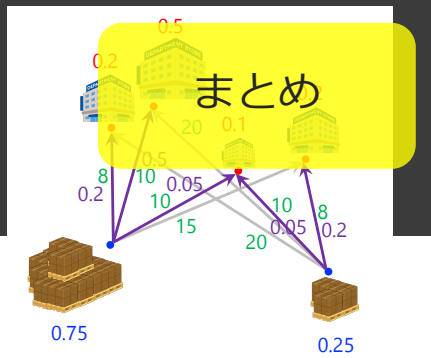
前半のまとめ・後半の目標

目次

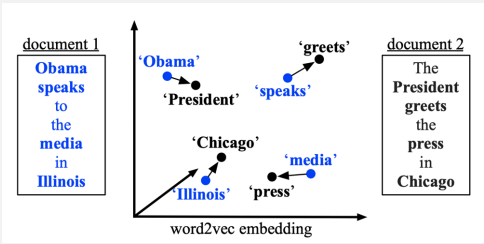
- **基礎編**：もっとも標準的な形式の最適輸送問題を知る
  - 【最適輸送を知る】1：荷物の配置換えという気持ちを知る
    - ゴール：最適輸送の話題が出てきても怖くなくなる
  - なぜ今最適輸送なのか
  - 【最適輸送を知る】2：入出力を形式的に理解する
    - ゴール：最適輸送を自分の研究・開発プロジェクトで使えるようになる
  - 入力をカスタマイズしながら使う
  - 【最適輸送を知る】3：線形計画問題としての定式化を理解する
    - ゴール：最適輸送を道具として用いている論文の式を読めるようになる
  - 用語に関する落穂拾い
    - ワッサーシュタイン距離，Earth Mover's Distance
  - 前半のまとめ　イマココ
- **応用編**：自動微分可能な最適輸送 / 最適輸送の変種 / 全体まとめ

前半のまとめ

目標：もっとも標準的な形式の最適輸送問題を知る
「最適輸送わかってしまった…」になる



- 最適輸送の**視覚的・物理的な気持ち**を知る
  - 荷物を最低コストで移し替えるプランと総コストを求める手法
  - WMD：単語ベクトル集合を移し替えるコスト ＝ テキストの非類似度
- 最適輸送の**入出力**を理解する
  - 入力：**a** (荷物の量分布1)，**b** (荷物の量分布2)，**C** (輸送コスト行列)
  - 出力：**P** (輸送計画行列)，総輸送コスト ($\Sigma_{\text{道}}$ 輸送コスト × 輸送量)
  - POT 等のライブラリに投げれば解いてくれる
  - 解きたい問題に合わせて入力 (分布，輸送コスト) を考える
- 最適輸送の**形式的な定義**を理解する
  - 線形計画で定式化される
  - ワッサーシュタイン距離は最適輸送コストの特殊ケース



Kusner+, From Word Embeddings To Document Distances

- **最適輸送はアラインメントしながら点群の違いを測る道具**

後半の目標：

手元のすべての問題を最適輸送で解きたくなる

まとめ

- **自動微分可能な最適輸送コスト**という選択肢を知る
  - 最適輸送コストが下がる方向に入力を更新できる
- **最適輸送の変種**を知る
  - 違う空間に存在する点群をマッチングする
  - 荷物の過不足を許す
  - 構造を考慮する

応用編1
自動微分可能な最適輸送コスト

目次

ここから後半戦です．
復習に便利なようにリンク付き目次を置いておきます．
いま全体像を把握しなくても大丈夫です．

自動微分可能な最適輸送コスト

POT の ot.emd() 等

- **最適輸送問題を線形計画問題として定式化し厳密解法で解く**ことの **デメリット**
  - n, m (たとえば単語数) が 10^3, 10^4 を超えたあたりから**計算が重い**
    - cf. 文同士 (単語数 n ~ 10^2) の類似度であればソルバを使えば一瞬
  - ニューラルネットに組み込んで**自動微分・逆伝播ができない**
    - KL ダイバージェンス (交差エントロピー) のように損失に使いたい

自動微分可能な最適輸送コスト

POT の ot.emd() 等

- **最適輸送問題を線形計画問題として定式化し厳密解法で解く**ことの **デメリット**
  - n, m (たとえば単語数) が 10^3, 10^4 を超えたあたりから**計算が重い**
    - cf. 文同士 (単語数 n ~ 10^2) の類似度であればソルバを使えば一瞬
  - ニューラルネットに組み込んで**自動微分・逆伝播ができない**
    - KL ダイバージェンス (交差エントロピー) のように損失に使いたい

- → 反復解法に帰着させた**最適輸送コストの亜種**
  - **✓計算が軽く**, **✓自動微分可能**
    1. **エントロピー正則化つき最適輸送コスト**
    2. **シンクホーン・ダイバージェンス**

    } 目的関数を変える

    3. 近接点法に基づく計算 (**IPOT**)

    } 解きかただけ変える

具体的な目的関数や解き方は一旦置いておいて…

もし自動微分可能な最適輸送コストが手に入ったら何ができる？

目次

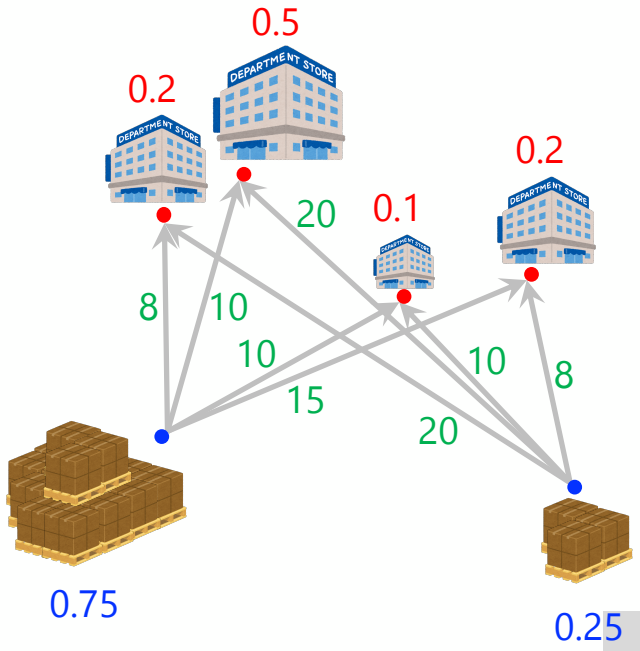
自動微分可能な最適輸送コストの嬉しさ

- 自動微分可能な最適輸送コストの嬉しさ：
  最適輸送コストが小さくなるように**入力**を更新できる

自動微分可能な最適輸送コストの嬉しさ

最適輸送問題の入力は
荷物の量分布 (**重み** × 2) と
輸送コスト行列

- 自動微分可能な最適輸送コストの嬉しさ：
  最適輸送コストが小さくなるように**入力**を更新できる
  - = 最適輸送コストが小さくなるように**重み**を更新できる
  - = 最適輸送コストが小さくなるように**輸送コスト**を更新できる

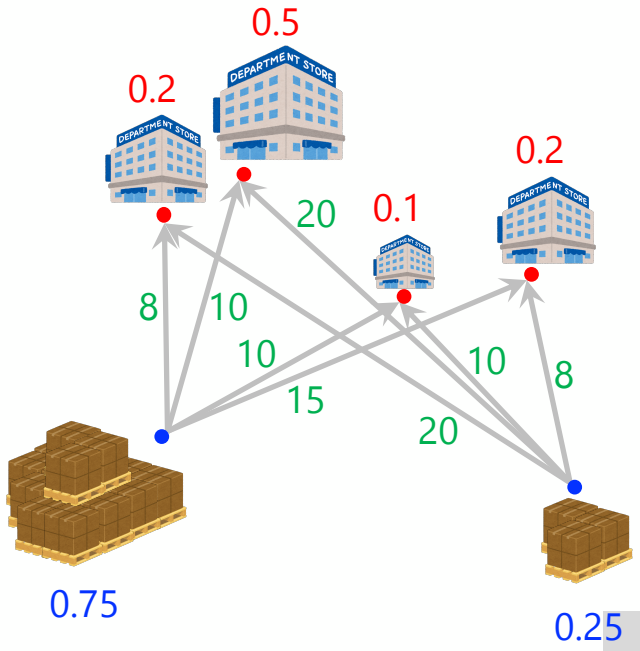


自動微分可能な最適輸送コストの嬉しさ

最適輸送問題の入力は
荷物の量分布 (**重み** × 2) と
輸送コスト行列

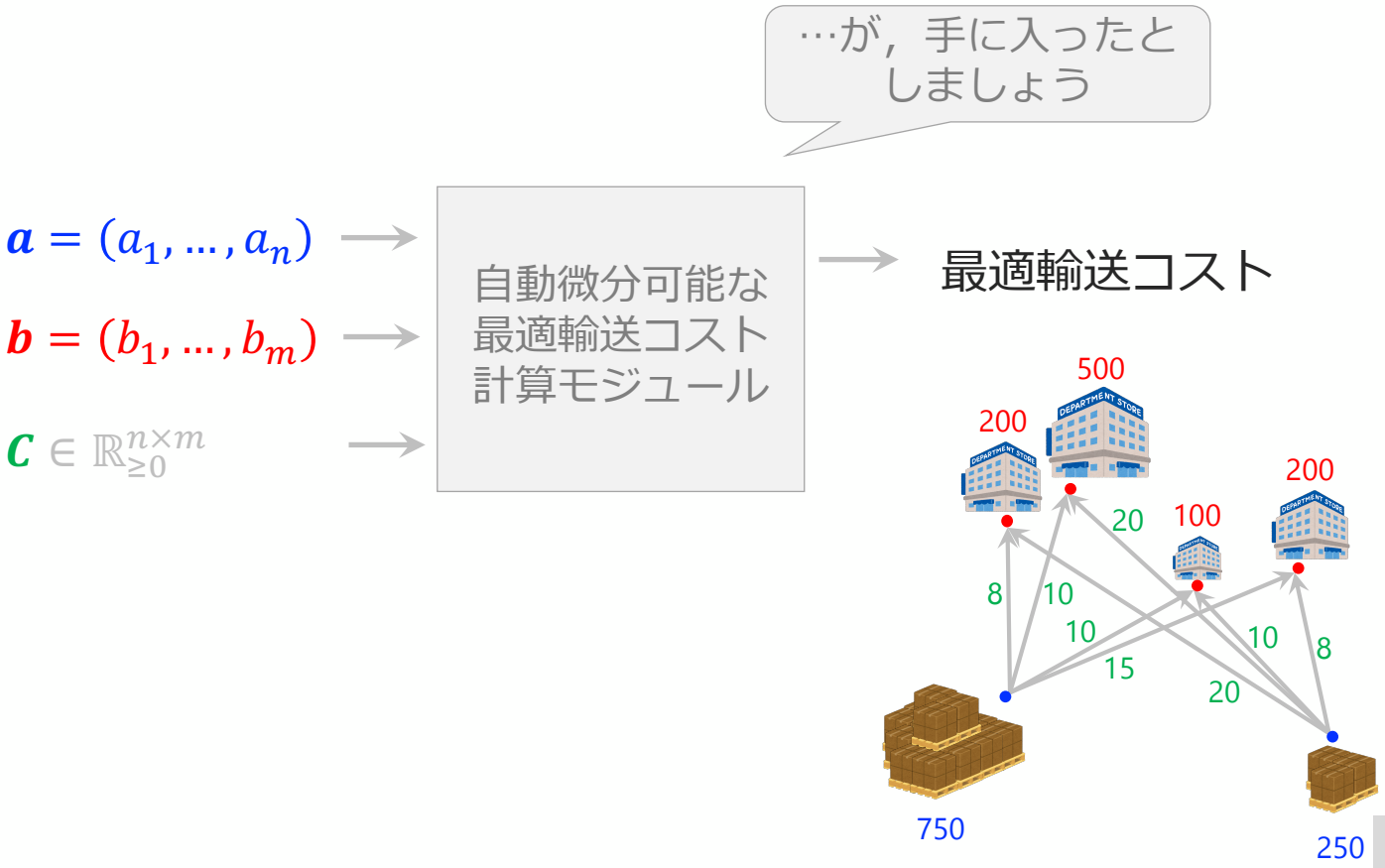
- 自動微分可能な最適輸送コストの嬉しさ：
  最適輸送コストが小さくなるように**入力**を更新できる
  - = 最適輸送コストが小さくなるように**重み**を更新できる
  - = 最適輸送コストが小さくなるように**輸送コスト**を更新できる

- …つまり？



自動微分可能な最適輸送コストの嬉しさ
(1) 最適輸送コストが下がる方向に**重み**を更新できる

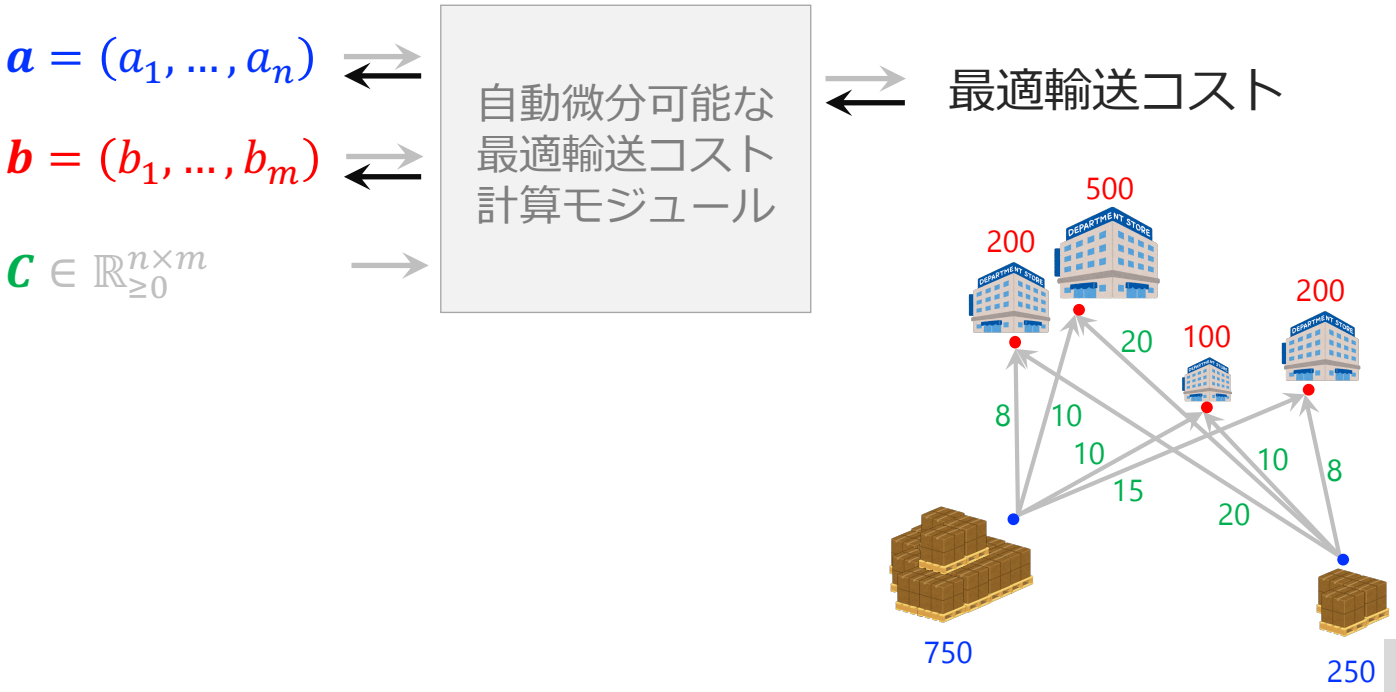
- forward → 微分可能なモジュールを使った最適輸送コストの計算



自動微分可能な最適輸送コストの嬉しさ

(1) 最適輸送コストが下がる方向に重みを更新できる

- forward → 微分可能なモジュールを使った最適輸送コストの計算
- ← backward 最適輸送コストが小さくなるように重みを更新できる

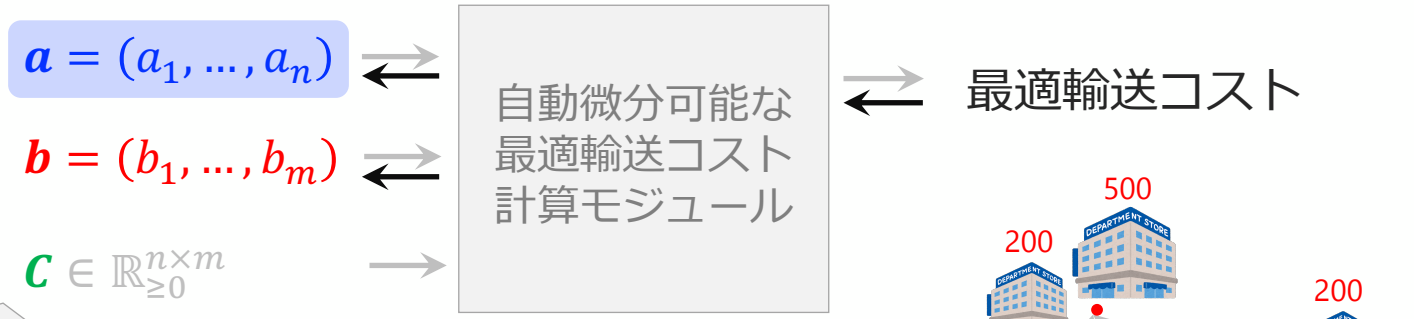


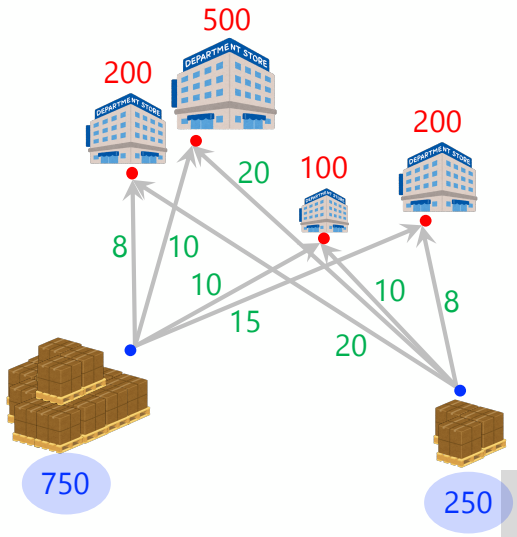
自動微分可能な最適輸送コストの嬉しさ

まとめ

(1) 最適輸送コストが下がる方向に重みを更新できる

- forward → 微分可能なモジュールを使った最適輸送コストの計算
- ← backward 最適輸送コストが小さくなるように重みを更新できる





たとえば,
各デパートが必要とする特産品量や
工場・デパートの位置を固定したまま
輸送コストが下がる方向に
工場毎の生産量バランスを更新できる

自動微分可能な最適輸送コストの嬉しさ

(2) 最適輸送コストが下がる方向に**コスト行列**を更新できる

もうひとつの入力

- forward → 微分可能なモジュールを使った最適輸送コストの計算

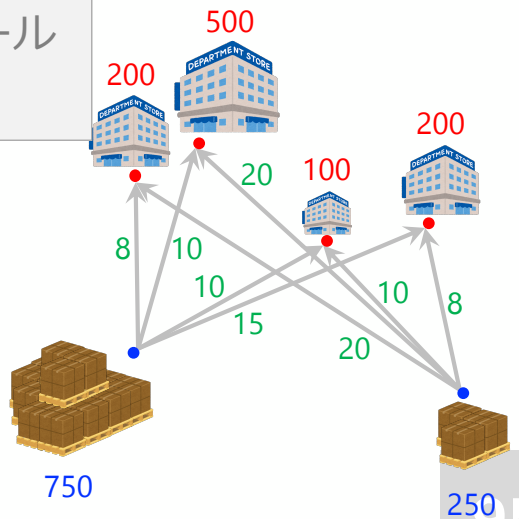
…が，手に入ったとしましょう

$\boldsymbol{a} = (a_1, \ldots, a_n)$ →

$\boldsymbol{b} = (b_1, \ldots, b_m)$ →

$\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n \times m}$ →

自動微分可能な最適輸送コスト計算モジュール → 最適輸送コスト



自動微分可能な最適輸送コストの嬉しさ
(2) 最適輸送コストが下がる方向にコスト行列を更新できる

- forward → 微分可能なモジュールを使った最適輸送コストの計算

$\boldsymbol{a} = (a_1, \ldots, a_n)$ →

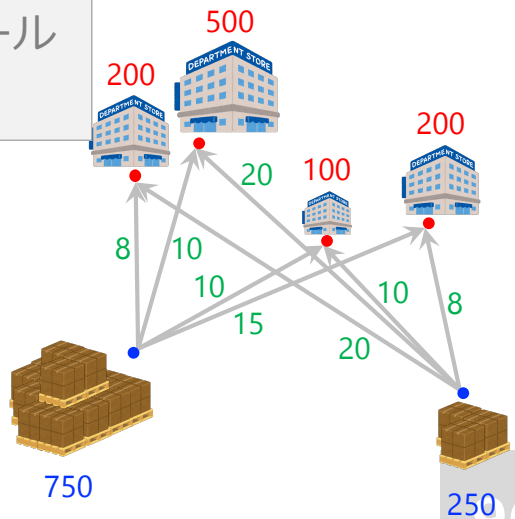
$\boldsymbol{b} = (b_1, \ldots, b_m)$ →

$\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n \times m}$ →

自動微分可能な
最適輸送コスト
計算モジュール

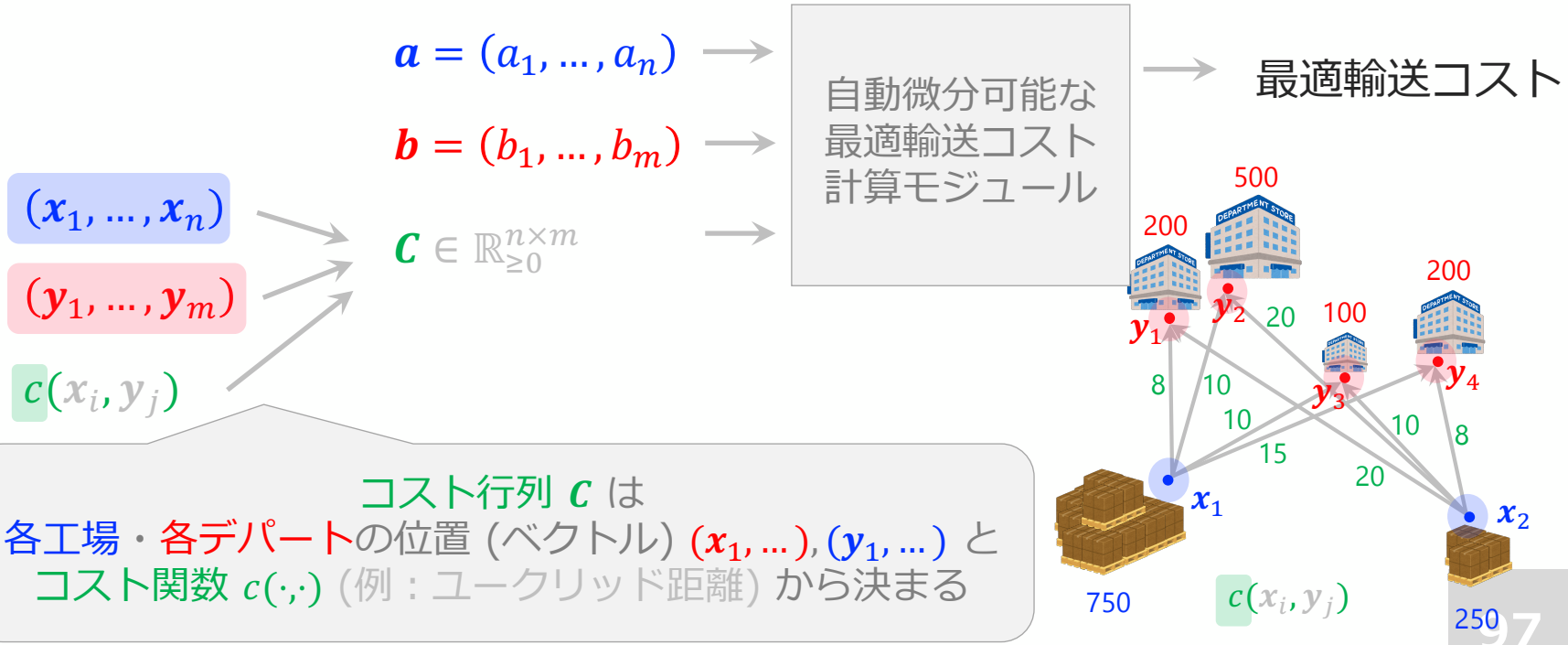
→ 最適輸送コスト

ところで大抵の場合，我々ユーザはコスト行列 $\boldsymbol{C}$ を
プリミティブに持っているわけではなく……



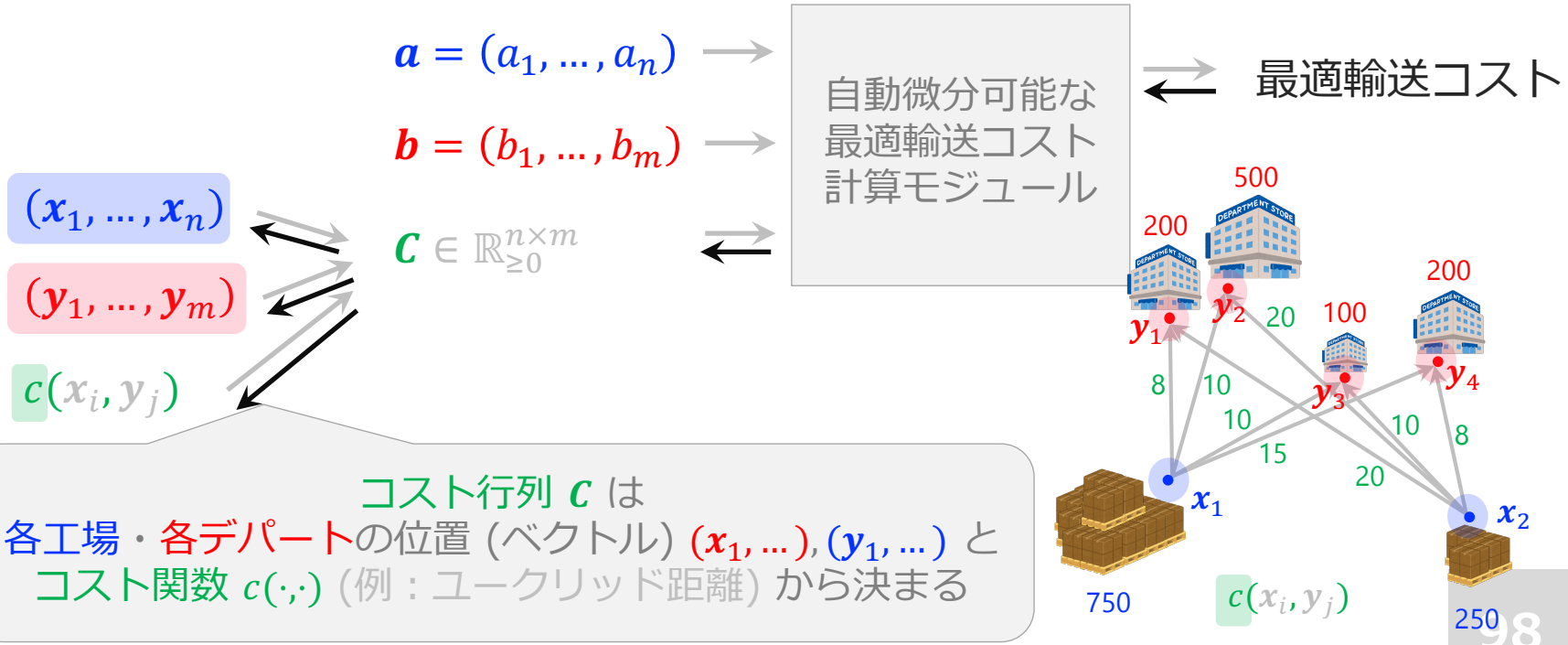
自動微分可能な最適輸送コストの嬉しさ
(2) 最適輸送コストが下がる方向に位置やコスト関数を更新できる

- forward → 微分可能なモジュールを使った最適輸送コストの計算



自動微分可能な最適輸送コストの嬉しさ
(2) 最適輸送コストが下がる方向に位置やコスト関数を更新できる

- forward → 微分可能なモジュールを使った最適輸送コストの計算
- ← backward 最適輸送コストが小さくなるように (コスト行列…を決める) 位置やコスト関数を更新できる



自動微分可能な最適輸送コストの嬉しさ

まとめ

(2) 最適輸送コストが下がる方向に位置やコスト関数を更新できる

- forward → 微分可能なモジュールを使った最適輸送コストの計算
- ← backward 最適輸送コストが小さくなるように (コスト行列…を決める) 位置やコスト関数を更新できる

たとえば，
各工場の生産高 $\boldsymbol{a}$ と 各デパートの需要 $\boldsymbol{b}$ を固定したまま，輸送コストが下がる方向に各工場の位置 $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ を更新できる

$\boldsymbol{a} = (a_1, \ldots, a_n)$ →

$\boldsymbol{b} = (b_1, \ldots, b_m)$ →

$\boldsymbol{C} \in \mathbb{R}^{n \times m}_{\geq 0}$ ⇄

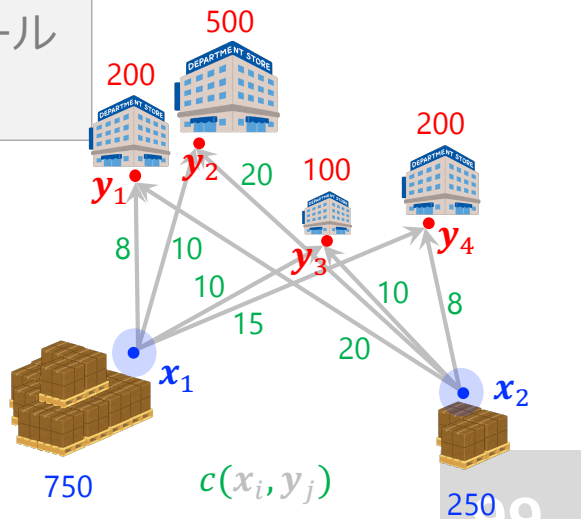
自動微分可能な最適輸送コスト計算モジュール

⇄ 最適輸送コスト

$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ ⇄

$(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$

$c(x_i, y_j)$

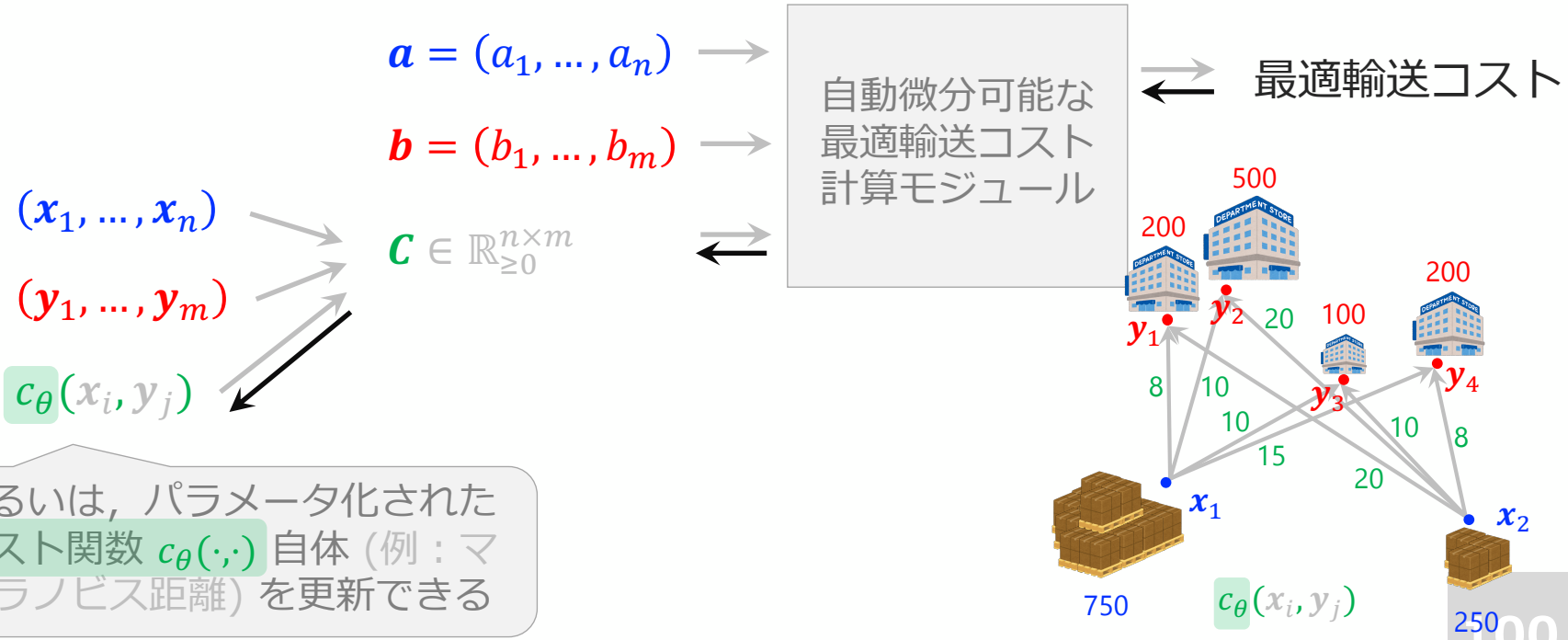


自動微分可能な最適輸送コストの嬉しさ

まとめ

(2) 最適輸送コストが下がる方向に位置や**コスト関数**を更新できる

- forward → 微分可能なモジュールを使った最適輸送コストの計算
- ← backward 最適輸送コストが小さくなるように (コスト行列…を決める) 位置や**コスト関数**を更新できる

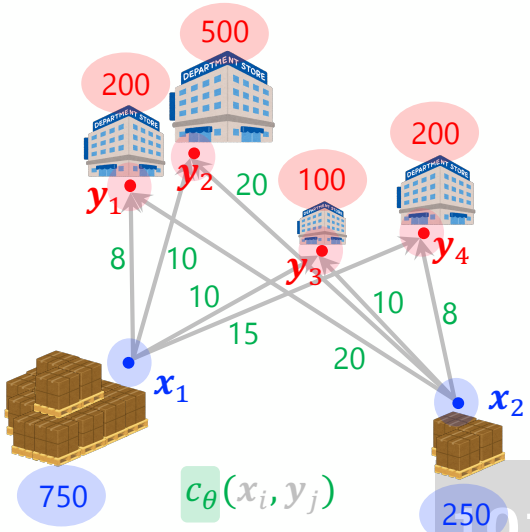


あるいは，パラメータ化されたコスト関数 $c_\theta(\cdot,\cdot)$ 自体 (例：マハラノビス距離) を更新できる

自動微分可能な最適輸送コストの嬉しさ

まとめ

- 自動微分可能な最適輸送コストの嬉しさ：
  最適輸送コストが小さくなるように**入力**を更新できる
  - (1) 最適輸送コストが小さくなるように**重み**を更新できる
  - (2) 最適輸送コストが小さくなるように輸送コスト行列
    …を決めるための**位置**や**コスト関数**を更新できる



引き続き，具体的な目的関数や解き方は一旦置いておいて…

自動微分可能な最適輸送コストの NLP での利用例

目次

自動微分可能な最適輸送コストの利用例

- 自動微分可能な最適輸送コストの NLP での利用例を3つ紹介
  1. **テキスト生成モデルの損失**に最適輸送コストを用いる [Chen+'19]
  2. 文書分類データを用いて**単語ベクトルを育てる** [Huang+'16]
  3. "単語ベクトル" ならぬ **"単語分布" を学習する** [Frogner+'19]

Chen+, Improving Sequence-to-Sequence Learning via Optimal Transport (ICLR 2019)
Huang+, Supervised Word Mover's Distance (NIPS 2016)
Frogner+, Learning Embeddings into Entropic Wasserstein Spaces (ICLR 2019)

自動微分可能な最適輸送コストの利用例 (1)
テキスト生成の損失に用いる [Chen+'19]

テキストを吐く seq2seq モデルや言語モデルが手元にあるとしましょう

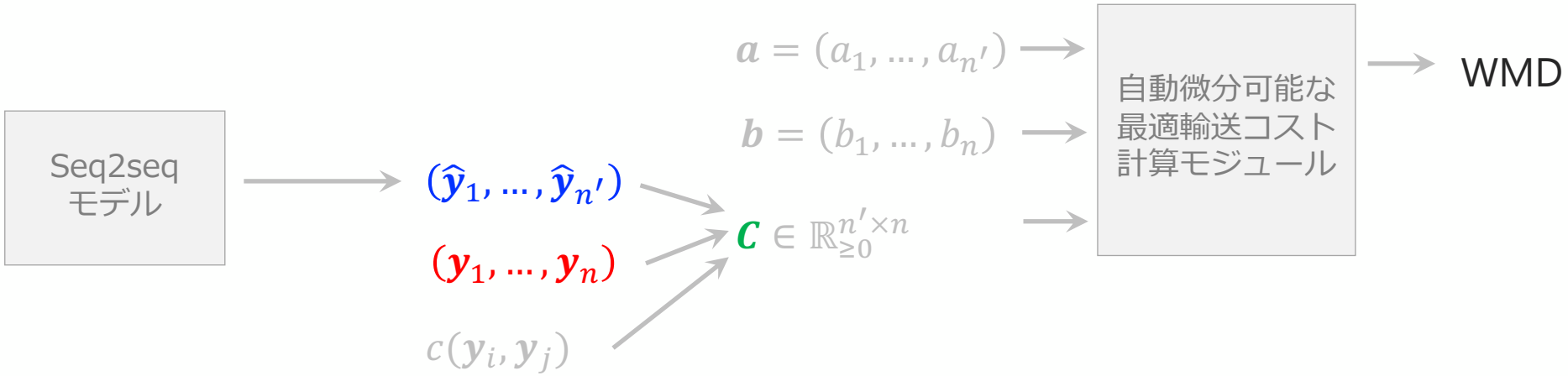
Seq2seq モデル → $(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_{n'})$

Chen+, Improving Sequence-to-Sequence Learning via Optimal Transport (ICLR 2019)

自動微分可能な最適輸送コストの利用例 (1)
テキスト生成の損失に用いる [Chen+'19]

- forward → **生成文** (単語ベクトル集合) と**リファレンス文** (単語ベクトル集合) の間の違い (WMD；損失) を計算

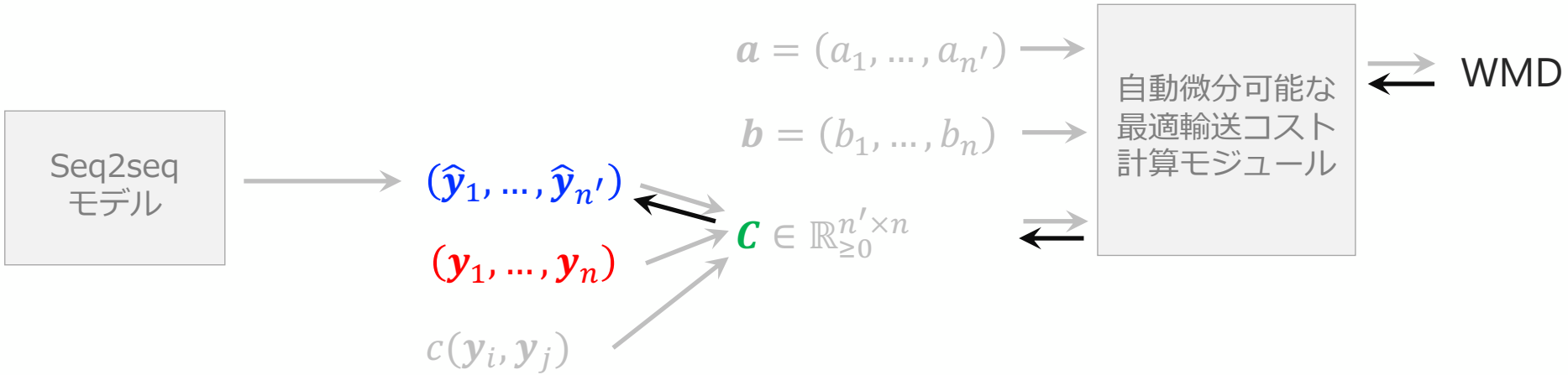
クロスエントロピーを計算するように



Chen+, Improving Sequence-to-Sequence Learning via Optimal Transport (ICLR 2019)

自動微分可能な最適輸送コストの利用例 (1)
テキスト生成の損失に用いる [Chen+'19]

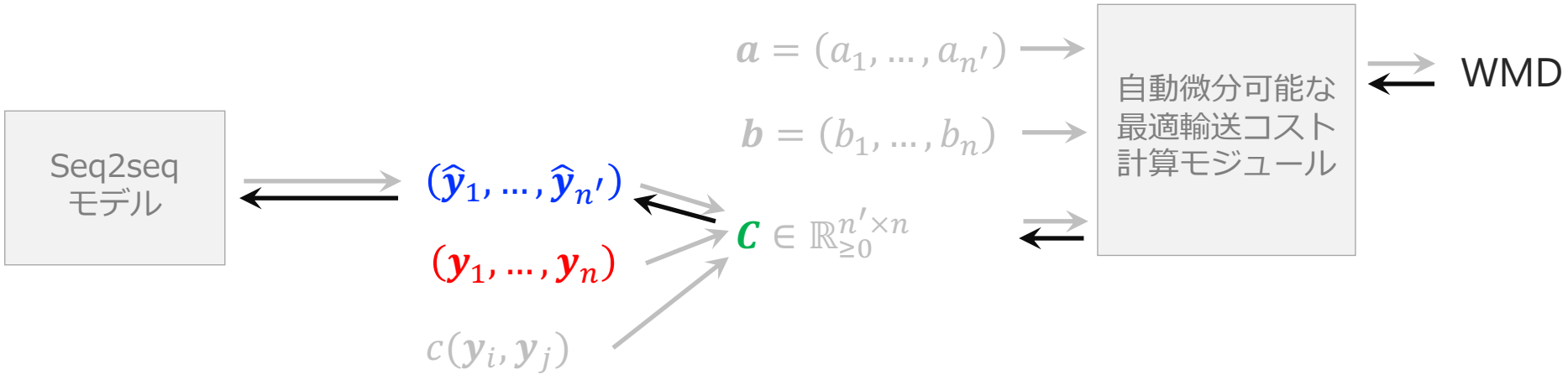
- forward → **生成文** (単語ベクトル集合) と**リファレンス文** (単語ベクトル集合) の間の違い (WMD；損失) を計算
- backward ← WMD を損失として **生成文** (単語ベクトル集合) を更新



Chen+, Improving Sequence-to-Sequence Learning via Optimal Transport (ICLR 2019)

自動微分可能な最適輸送コストの利用例 (1)
テキスト生成の損失に用いる [Chen+'19]

- forward → **生成文** (単語ベクトル集合) と**リファレンス文** (単語ベクトル集合) の間の違い (WMD；損失) を計算
- ← backward WMD を損失として **生成文** (単語ベクトル集合) を作る seq2seq モデル を更新



- → 機械翻訳モデルに適用して BLEU が微向上

Chen+, Improving Sequence-to-Sequence Learning via Optimal Transport (ICLR 2019)

自動微分可能な最適輸送コストの利用例 (1)
テキスト生成の損失に用いる [Chen+'19]

※気になる人向けの詳細

- forward → **生成文** (単語ベクトル集合) と**リファレンス文** (単語ベクトル集合) の間の違い (WMD；損失) を計算

ル集合) を作る seq2seq モデルを更新

モデルから各単語を決定的にサンプルすると損失を戻せない
→ 予測分布に単語埋込行列をかけ直し**各時刻 t で予測されそうな平均的単語ベクトル**を計算

各重みは uniform
※ Alg. 1 から読み取れる

$\boldsymbol{a} = (a_1, \ldots, a_{n'})$ →

$\boldsymbol{b} = (b_1, \ldots, b_n)$ →

自動微分可能な最適輸送コスト計算モジュール ⇄ WMD

Seq2seq モデル ⇄ $(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_{n'})$

$(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$

$c(\boldsymbol{y}_i, \boldsymbol{y}_j)$

$\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n' \times n}$

$\boldsymbol{y}_t$ を作る際もモデル内の埋込行列を利用

近接勾配法に基づく反復計算 (IPOT) を用いる [Chen+'19] 後述

輸送コストはコサイン
※ ソースに書いてある

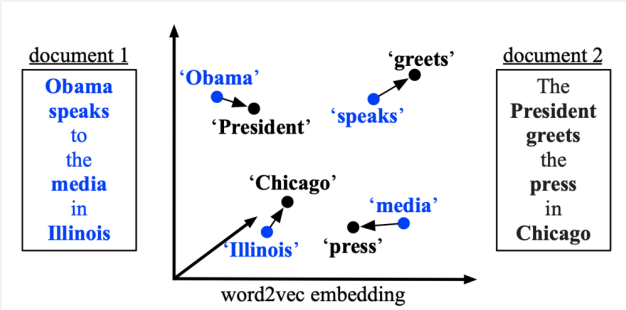
Chen+, Improving Sequence-to-Sequence Learning via Optimal Transport (ICLR 2019)
Xie+, A Fast Proximal Point Method for Computing Exact Wasserstein Distance (UAI 2019)

自動微分可能な最適輸送コストの利用例 (2)
WMDのための単語重みと距離を探す [Huang+'16]

- **復習**：Word Mover's Distance [Kusner+'15]
  - 確率分布：$\boldsymbol{s} = (1/n, \ldots, 1/n)$
  - 確率分布：$\boldsymbol{s}' = (1/m, \ldots, 1/m)$
  - 輸送コスト：$\boldsymbol{c}_{ij} = \|\boldsymbol{w}_i - \boldsymbol{w}'_j\|_2$

各単語の重みは均等

単語間の輸送コストにはL2距離を利用



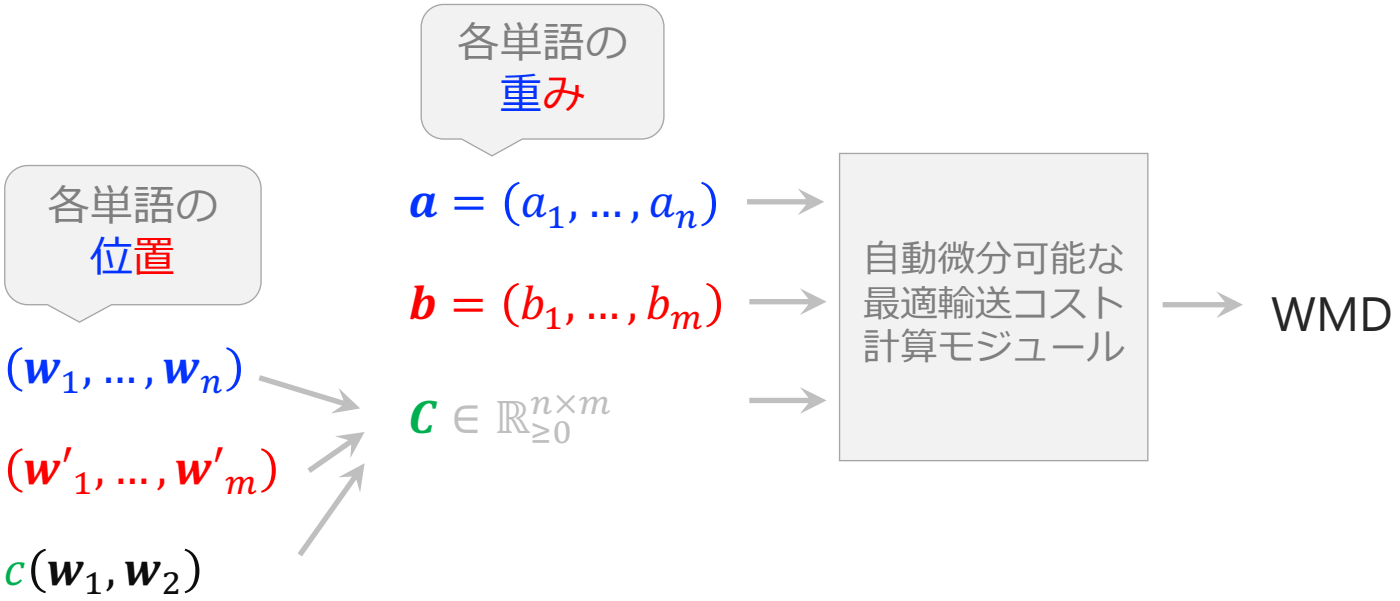
Kusner+, From Word Embeddings To Document Distances (ICML 2015)

- **モチベーション**：WMD に利用する各単語重みや輸送コスト (単語ベクトル間距離) を教師ありで学習できない…？

Huang+, Supervised Word Mover's Distance (NIPS 2016)

自動微分可能な最適輸送コストの利用例 (2)
WMDのための単語重みと距離を探す [Huang+'16]

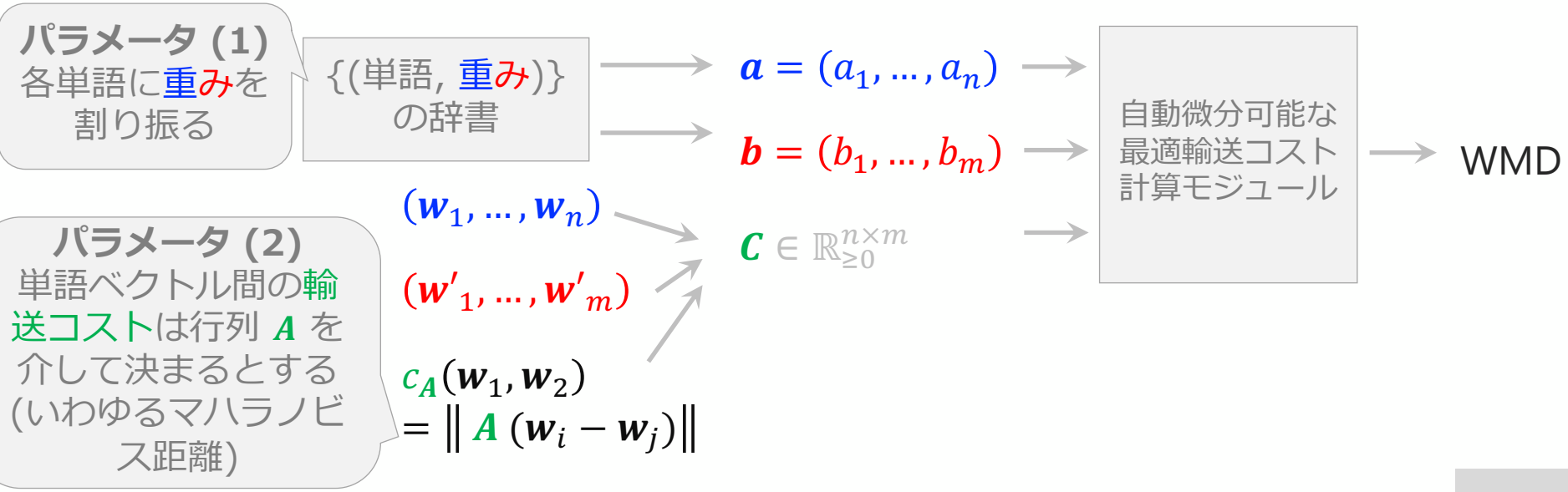
- forward → 文書間の WMD を自動微分可能なモジュールで計算



Huang+, Supervised Word Mover's Distance (NIPS 2016)

自動微分可能な最適輸送コストの利用例 (2)
WMDのための単語重みと距離を探す [Huang+'16]

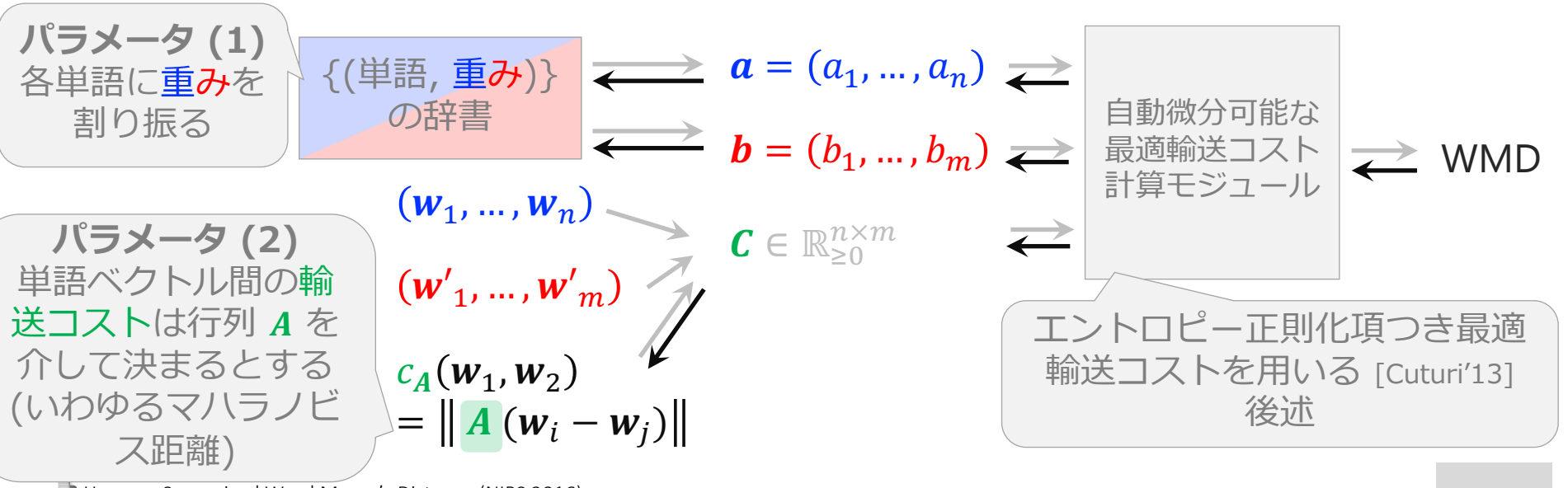
- forward → 文書間の WMD を自動微分可能なモジュールで計算
  重みや距離をパラメトライズしておく



Huang+, Supervised Word Mover's Distance (NIPS 2016)

自動微分可能な最適輸送コストの利用例 (2)
WMDのための単語重みと距離を探す [Huang+'16]

- forward → 文書間の WMD を自動微分可能なモジュールで計算
  重みや距離をパラメトライズしておく
- ← backward 同クラスの文書間の WMD が小さくなるよう
  (負例ペアの WMD が大きくなるよう) パラメータ更新

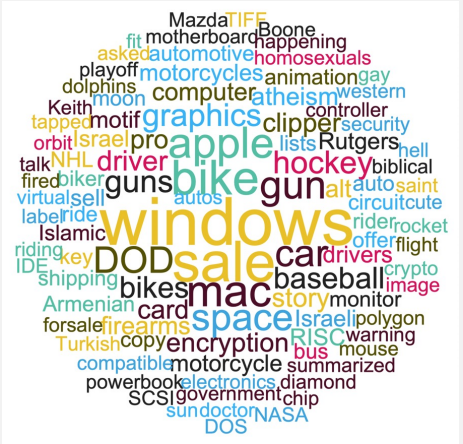


Huang+, Supervised Word Mover's Distance (NIPS 2016)
Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)

自動微分可能な最適輸送コストの利用例 (2)
WMDのための単語重みと距離を探す [Huang+'16]

- forward → 文書間の WMD を自動微分可能なモジュールで計算
  重みや距離をパラメトライズしておく
- ← backward 同クラスの文書間の WMD が小さくなるよう
  (負例ペアの WMD が大きくなるよう) パラメータ更新

- ✓ (教師なしだった) WMD よりk-NN文書分類の性能が向上
- ✓ 当該文書分類タスクに効く
  単語重みが自動で学習される

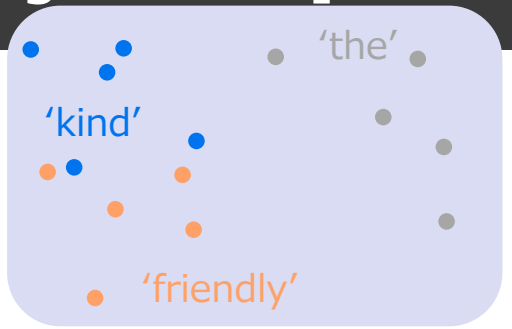


Huang+'16

Huang+, Supervised Word Mover's Distance (NIPS 2016)

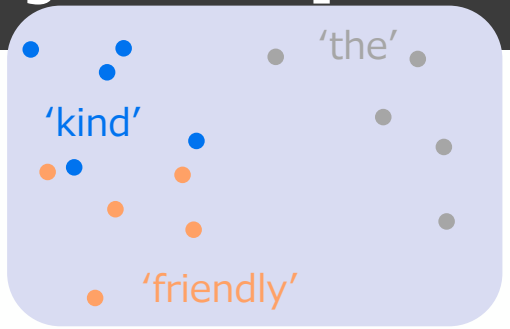
自動微分可能な最適輸送コストの利用例 (3)
単語の表現学習の損失に用いる [Frogner+'19]

- 各単語の表現として**ベクトル集合**を学習する
  - 各ベクトルの位置だけパラメトライズ



Frogner+, Learning Embeddings into Entropic Wasserstein Spaces (ICLR 2019)

自動微分可能な最適輸送コストの利用例 (3)
単語の表現学習の損失に用いる [Frogner+'19]



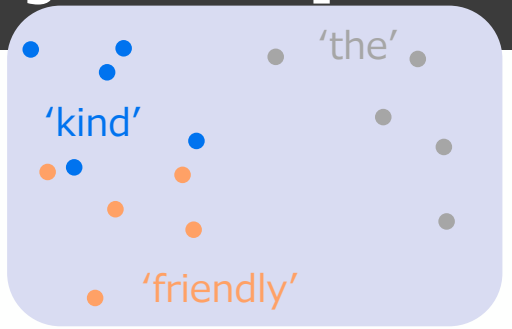
- 各単語の表現として**ベクトル集合**を学習する
  - 各ベクトルの位置だけパラメトライズ
- 共起単語ペアの最適輸送コストを下げる方向で表現学習

	単語表現	共起単語ペア
cf. word2vec	ベクトル	内積↑
Frogner+'19	**ベクトル集合**	**最適輸送コスト↓**

エントロピー正則化項つき最適輸送コストを用いる [Cuturi'13] 後述

Frogner+, Learning Embeddings into Entropic Wasserstein Spaces (ICLR 2019)

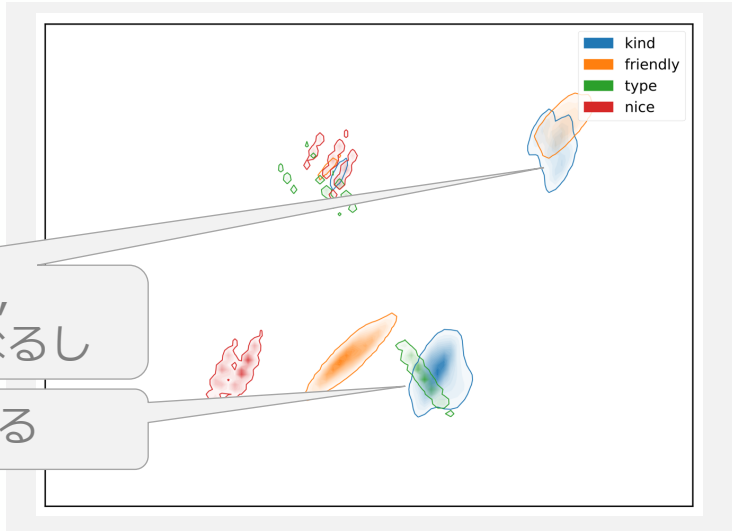
自動微分可能な最適輸送コストの利用例 (3)
単語の表現学習の損失に用いる [Frogner+'19]



- 各単語の表現として**ベクトル集合**を学習する
  - 各ベクトルの位置だけパラメトライズ
- 共起単語ペアの最適輸送コストを下げる方向で表現学習

	単語表現	共起単語ペア
cf. word2vec	ベクトル	内積↑
Frogner+'19	**ベクトル集合**	**最適輸送コスト↓**

- → 多義性が自然に学習される



Frogner+, Learning Embeddings into Entropic Wasserstein Spaces (ICLR 2019)

具体的な計算方法 × 3

自動微分可能な最適輸送コストを使いたい場合の具体的な選択肢

目次

1. エントロピー正則化付き最適輸送 [Cuturi'13]

注：元の最適化問題とは別の問題を解く

- **エントロピー正則化**を入れて**目的関数を変更**
  - $\mathrm{OT}_{\varepsilon}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}) \coloneqq \min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_{ij} \boldsymbol{P}_{ij} \boldsymbol{C}_{ij} - \varepsilon H(\boldsymbol{P})$

元の最適輸送問題の目的関数

正則化項

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)

1. エントロピー正則化付き最適輸送 [Cuturi'13]

注：元の最適化問題とは別の問題を解く

- **エントロピー正則化**を入れて**目的関数を変更**
  - $\mathrm{OT}_\varepsilon(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}) \coloneqq \min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_{ij} \boldsymbol{P}_{ij} \boldsymbol{C}_{ij} - \varepsilon H(\boldsymbol{P})$

元の最適輸送問題の目的関数

正則化項

$H(\boldsymbol{P}) \coloneqq -\sum_{ij} \boldsymbol{P}_{ij} (\log \boldsymbol{P}_{ij} - 1)$

“輸送計画 $\boldsymbol{P}$ が一様なほど嬉しい”

輸送計画をぼわっとさせたい

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)

1. エントロピー正則化付き最適輸送 [Cuturi'13]

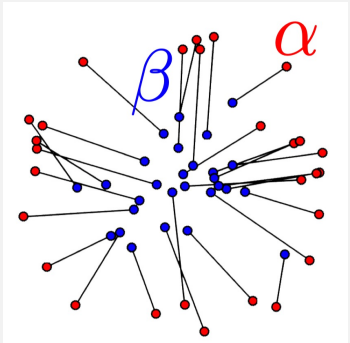
注：元の最適化問題とは別の問題を解く

- **エントロピー正則化**を入れて**目的関数を変更**
  - $\mathrm{OT}_\varepsilon(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}) \coloneqq \min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_{ij} \boldsymbol{P}_{ij} \boldsymbol{C}_{ij} - \varepsilon H(\boldsymbol{P})$

$H(\boldsymbol{P}) \coloneqq -\sum_{ij} \boldsymbol{P}_{ij} (\log \boldsymbol{P}_{ij} - 1)$

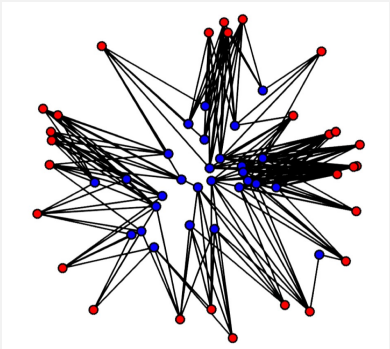
"輸送計画 $\boldsymbol{P}$ が一様なほど嬉しい"
輸送計画をぼわっとさせたい

元の最適輸送問題の最適解



Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019) Figure 4.3 より

エントロピー正則化つきの最適輸送問題の最適解
(ぼわっと輸送される)



Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019) Figure 4.3 より

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)

1. エントロピー正則化付き最適輸送 [Cuturi'13]

- **エントロピー正則化**を入れて**目的関数を変更**
  - $\mathrm{OT}_{\varepsilon}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}) \coloneqq \min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_{ij} \boldsymbol{P}_{ij} \boldsymbol{C}_{ij} - \varepsilon H(\boldsymbol{P})$

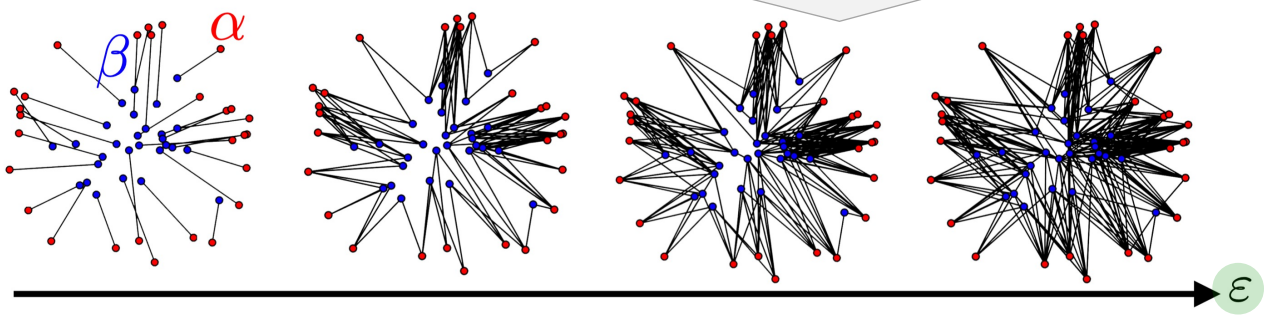
$H(\boldsymbol{P}) \coloneqq -\sum_{ij} \boldsymbol{P}_{ij} (\log \boldsymbol{P}_{ij} - 1)$

"輸送計画 $\boldsymbol{P}$ が一様なほど嬉しい"
輸送計画をぼわっとさせたい

- パラメータ $\varepsilon > 0$ で「ぼわっと送る」度が決まる

元の最適輸送問題の最適解

エントロピー正則化つきの最適輸送問題の最適解
(εを大きくすると「ぼわっと送る」度が高まる)



Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019), Figure 4.3

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)

1. エントロピー正則化付き最適輸送 [Cuturi'13]

- **エントロピー正則化**を入れて**目的関数を変更**
  - $\mathrm{OT}_\varepsilon(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}) \coloneqq \min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_{ij} \boldsymbol{P}_{ij} \boldsymbol{C}_{ij} - \varepsilon H(\boldsymbol{P})$

最適化問題として“良い”形
(目的関数が強凸，解空間が凸)

(双対問題の)

- → 最適化が**反復計算に帰着** (**シンクホーンアルゴリズム**)
  - $\boldsymbol{K}_{ij} \leftarrow \exp(-\boldsymbol{C}_{ij}/\varepsilon)$
  - $\boldsymbol{u} \leftarrow (1, \ldots, 1)^\mathsf{T}$
  - $\boldsymbol{v} \leftarrow \boldsymbol{b}/\boldsymbol{K}^\mathsf{T}\boldsymbol{u}$
  - $\boldsymbol{u} \leftarrow \boldsymbol{a}/\boldsymbol{K}\boldsymbol{v}$

  } 適当回数反復

  - $\boldsymbol{P} \leftarrow \boldsymbol{u}^\mathsf{T}\boldsymbol{K}\boldsymbol{v}$

ひたすら掛け算・割り算なので全体として自動微分可能

詳細な解説・Python コードなど：
佐藤, 最適輸送入門, IBIS 2021
https://www.slideshare.net/joisino/ss-251328369

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)

1. エントロピー正則化付き最適輸送 [Cuturi'13]
あまり使われない別名：シンクホーン距離

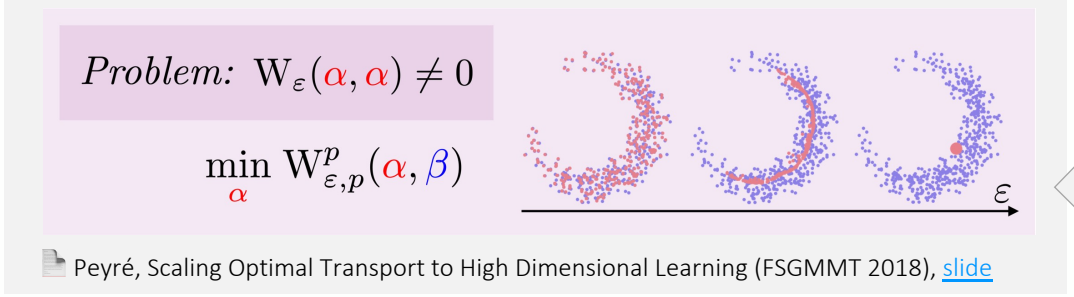
- 用語に関する落穂拾い：シンクホーンアルゴリズム
  - シンクホーンアルゴリズムは [Cuturi'13] の発明ではない
    - 数理最適化や数値計算の文脈で古くから扱われていた問題設定・解法
    - [Peyré&Cuturi'19] の Remark 4.5 に歴史的経緯が書かれている
  - 「シンクホーンアルゴリズム」という名前は，同アルゴリズムの収束について議論されている [Sinkhorn'64] から
    - [Sinkhorn&Knopp'67] を参照して「Sinkhorn–Knopp アルゴリズム」と呼ばれることも
  - [Cuturi'13] はものすごく時代にあったタイミングでシンクホーンアルゴリズムを最適輸送と結びつけた
    - (1) 確率分布分布間の距離という機械学習の重要な道具として (2) GPU による並列計算 (3) 深層学習の時代のニーズにあった "微分可能" な仕組み

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)
Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019)
Sinkhorn, A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices (Ann. Math. Statist. 1964)
Sinkhorn&Knopp, Concerning nonnegative matrices and doubly stochastic matrices (Pac. J. Appl. Math. 1967)

NIPS'13 段階では意識されていなかったっぽい

1. エントロピー正則化付き最適輸送 [Cuturi'13]

- **ご利益**
  - 😀 計算コスト：最適化が反復的な行列計算に帰着 (**シンクホーン**)
  - 😀 → 自動微分可能に
- **注**
  - 😩 求まるコストは最適輸送コストの **近似** の **近似**
    - 目的関数は最適輸送コストではない (エントロピー正則化)
    - …その別の目的関数の解に辿り着く前に計算を打ち切る (有限反復)
  - 😩 この "最適輸送コスト" を下げる ≠ 分布を一致させる
    - モチベーション (思い出し)：テキスト生成モデルの損失として Word Mover's Distance を使いたい = 生成文とリファレンス文を**一致**させたい
    - が，「OT_ε を小さくする」と「単語ベクトル集合が概ね一致する」は**異なる**



Peyré, Scaling Optimal Transport to High Dimensional Learning (FSGMMT 2018), slide

赤分布と青分布の輸送コストの最小化を目的関数として赤分布を動かす
特に ε が大きい場合，青分布とは大きく異なる解が得られてしまう

2. シンクホーン・ダイバージェンス
[Ramdas+'17][Genevay+'18]

- **エントロピー正則化つき最適輸送コスト** [Cuturi'13] **を組み合わせる**
  - $\overline{\mathrm{OT}}_\varepsilon(\boldsymbol{a}, \boldsymbol{b}) := \mathrm{OT}_\varepsilon(\boldsymbol{a}, \boldsymbol{b}) - \frac{1}{2}\mathrm{OT}_\varepsilon(\boldsymbol{a}, \boldsymbol{a}) - \frac{1}{2}\mathrm{OT}_\varepsilon(\boldsymbol{b}, \boldsymbol{b})$
    - 分布 (before) 同士のエントロピー正則化つき最適輸送コスト
    - 分布 (after) 同士のエントロピー正則化つき最適輸送コスト
- **ご利益**
  - エントロピー正則化つき最適輸送コストのメリットを維持
    - 😀 ベクトル・行列による反復解法 → 😀 自動微分可能
  - 😀 自分自身との最適輸送コストが0
    - 損失関数に使いやすい；小さくすればリファレンス文が出るように**なる**
    - $\overline{\mathrm{OT}}_\varepsilon(\boldsymbol{a}, \boldsymbol{a}) = 0$, $\overline{\mathrm{OT}}_\varepsilon(\boldsymbol{a}, \boldsymbol{a}_n) \to 0 \Leftrightarrow \boldsymbol{a}_n \to \boldsymbol{a}$ [Feydy+'19, 一定の条件の下で]
- **注**
  - 😩 元の最適輸送コストとは異なるコストが求まる

Ramdas+, On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests (Entropy 2017)
Genevay+, Learning Generative Models with Sinkhorn Divergences (AISTATS 2018)
Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)
Feydy+, Interpolating between Optimal Transport and MMD using Sinkhorn Divergences (AISTATS 2019)

2. 近接勾配法に基づく計算 (IPOT) [Xie+'19]

- 元の最適輸送問題のまま，最適化問題を上手に解く
  - 徐々にボケない解に近づけていく
    - $\boldsymbol{P}^{(0)} = \mathbf{1}\mathbf{1}^{\top}$　初期 “解” … まずは一様にアラインメントさせる
    - $\boldsymbol{P}^{(t+1)} = \underset{\boldsymbol{P}\in\boldsymbol{U}(\boldsymbol{a},\boldsymbol{b})}{\operatorname{argmin}} \sum_{ij} \boldsymbol{P}_{ij}\boldsymbol{C}_{ij} + \varepsilon D(\boldsymbol{P}^{(t)}, \boldsymbol{P})$ … 解は少しずつ変えていく
- **ご利益**
  - 😀 簡単な反復アルゴリズムに帰着
  - 😀 → 自動微分可能
  - 😀 元の最適輸送問題を解いている
- 注
  - 😩 二重ループを真面目にやると重い

Algorithm 1 IPOT($\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{C}$)

Input: Probabilities $\{\boldsymbol{\mu}, \boldsymbol{\nu}\}$ on support points $\{x_i\}_{i=1}^m$, $\{y_j\}_{j=1}^n$, cost matrix $\boldsymbol{C} = [\|x_i - y_j\|]$

$\boldsymbol{b} \leftarrow \frac{1}{m}\mathbf{1}_m$

$G_{ij} \leftarrow e^{-\frac{C_{ij}}{\beta}}$

$\boldsymbol{\Gamma}^{(1)} \leftarrow \mathbf{1}\mathbf{1}^T$

for $t = 1, 2, 3, \ldots$ **do**

  $\boldsymbol{Q} \leftarrow \boldsymbol{G} \odot \boldsymbol{\Gamma}^{(t)}$

  for $l = 1, 2, 3, \ldots, L$ **do**　// Usually set $L = 1$

    $\boldsymbol{a} \leftarrow \frac{\boldsymbol{\mu}}{\boldsymbol{Q}\boldsymbol{b}}, \boldsymbol{b} \leftarrow \frac{\boldsymbol{\nu}}{\boldsymbol{Q}^T\boldsymbol{a}}$

  end for

  $\boldsymbol{\Gamma}^{(t+1)} \leftarrow \mathrm{diag}(\boldsymbol{a})\boldsymbol{Q}\mathrm{diag}(\boldsymbol{b})$

end for

Xie+'19, A Fast Proximal Point Method for Computing Exact Wasserstein Distance (UAI 2019)

「経験的に 内側のループは1回まわせば十分 だよ」とのこと

Xie+, A Fast Proximal Point Method for Computing Exact Wasserstein Distance (UAI 2019)

自動微分可能な最適輸送コストの選択肢

まとめ

	自動微分可能	コストを0にする ⇔ 分布が近づく	計算の軽さ
最適輸送コストの厳密解法	✗	✓	✗ $n \sim 10^3$ 程度でつらくなる
エントロピー正則化つき最適輸送 [Cuturi'13]	✓	✗	✓
シンクホーン・ダイバージェンス [Ramdas+'17] [Genevay+'18]	✓	✓	✓ [Cuturi+'13]の2〜3倍程度
近接勾配法に基づく計算 (IPOT) [Xie+'19]	✓	✓	✓ 二重ループ

目的関数を変える

解き方を変える

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)
Ramdas+, On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests (Entropy 2017)
Genevay+, Learning Generative Models with Sinkhorn Divergences (AISTATS 2018)
Xie+, A Fast Proximal Point Method for Computing Exact Wasserstein Distance (UAI 2019)

……で，どの自動微分可能な最適輸送コストを使えば良い？

- **損失に使いたい → シンクホーン・ダイバージェンス** [Ramdas+'17] [Genevay+'18] or **近接勾配法に基づく計算 (IPOT)** [Xie+'19]
  - 😀 バイアスがない (コスト→0 にすれば予測とリファレンスが一致)
  - Cuturi 等の "主流派" は シンクホーン・ダイバージェンス 推しに見える
  - NLP では (おそらく [Chen+'19] の影響で) IPOT がよく利用されている
- **エントロピー正則化つき最適輸送** [Cuturi'13] **の使い道？**
  - 損失に使う場合も含め気軽に採用すれば良さそう
    - 最適輸送コストが何らかの言語現象の "正確な" モデルになっているかは不明．むしろ正則化が経験的に好ましい効果を生む可能性も．
    - ε が小さい場合 (真の最適輸送コストに近い設定) での数値計算を安定させるための各種改良が出てくる．たとえば POT 内のマニュアル 参照．
  - 注意機構とシンクホーンアルゴリズムを結びつける研究も [Sander+'22]

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)
Ramdas+, On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests (Entropy 2017)
Genevay+, Learning Generative Models with Sinkhorn Divergences (AISTATS 2018)
Xie+, A Fast Proximal Point Method for Computing Exact Wasserstein Distance (UAI 2019)
Chen+, Improving Sequence-to-Sequence Learning via Optimal Transport (ICLR 2019)
Sander+, Sinkformers: Transformers with Doubly Stochastic Attention (AISTATS 2022)

自動微分可能な最適輸送コスト
まとめ

自動微分可能な最適輸送コスト

まとめ

POT の ot.emd() など

- **線形計画問題として定式化＋厳密解法**
  - 😩 **計算コスト**：n, m (たとえば単語数) が 10^3, 10^4 あたりから重い
  - 😩 **自動微分**：最適輸送を既存 NN ライブラリと混ぜるのが困難
- **→ 反復計算に帰着させた自動微分可能な最適輸送コスト**
  - 選択肢
    1. **エントロピー正則化つき最適輸送コスト** [Cuturi'13] ※ バイアス有
    2. **シンクホーン・ダイバージェンス** [Ramdas+'17][Genevay+'18]

    } 目的関数を変える

    3. 近接勾配法に基づく計算 (**IPOT**) [Xie+'19]

    } 解きかたを変える
  - 😀 **計算コスト**：並列化やGPU利用が容易
  - 😀 **自動微分**：最適輸送コストが下がる方向に入力 (重み，位置，コスト関数) を更新できる

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)
Ramdas+, On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests (Entropy 2017)
Genevay+, Learning Generative Models with Sinkhorn Divergences (AISTATS 2018)
Xie+'19, A Fast Proximal Point Method for Computing Exact Wasserstein Distance (UAI 2019)

応用編2
最適輸送の拡張

最適輸送の拡張

- 触れる話題
  - **Gromov–Wasserstein 距離**
    - ふたつの分布が違う空間にあってもアラインメントはできる.
  - **不均衡最適輸送**
    - カップリングの制約を外す.
  - **構造を考慮した最適輸送**
    - 扱う対象が持っている構造を持っている構造を考慮する.

Gromov–Wasserstein

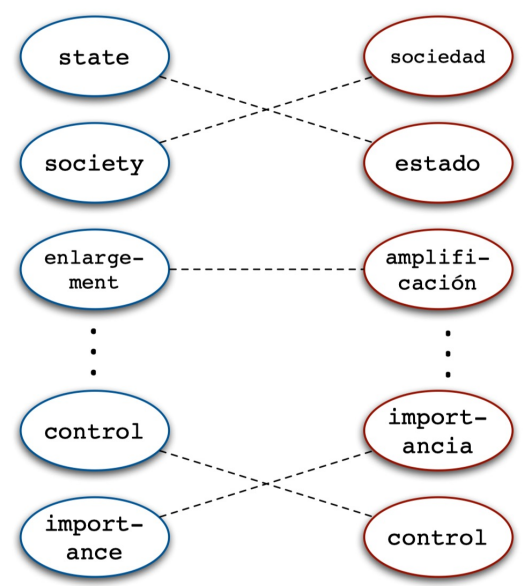
別の空間にある分布同士をマッチングする

目次

イマココ

モチベーション：
違う空間同士でアラインメントしたい

- **対訳辞書構築**
  - 任意の言語対での辞書構築はコスト大
  - 単言語コーパスならそれなりに存在 → 単言語の単語ベクトルは手に入る
  - アラインメントが目的…… 最適輸送か……？



Haghighi+, Learning Bilingual Lexicons from Monolingual Corpora (ACL 2008)

  - 問題
    - 工場 (英単語ベクトル) → デパート (スペイン語単語ベクトル) の**輸送コストがわからない**，別の空間なので測りようがない
    - ただし工場同士 (英単語ベクトル同士)，デパート同士 (スペイン語単語ベクトル同士) の距離はわかる

Gromov–Wasserstein 距離の気持ち：
違う空間同士でアラインメントしたい

2つの分布が別の空間に
居るときを考える

- **入力**
  - 確率分布：$\boldsymbol{a} = (a_1, \dots, a_n) \in [0,1]^n,\ \sum_i a_i = 1$
  - 確率分布：$\boldsymbol{b} = (b_1, \dots, b_m) \in [0,1]^m,\ \sum_i b_j = 1$
  - ~~輸送コスト：$\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n \times m}$~~

もはや「a 同士の距離」
「b 同士の距離」しかわからない

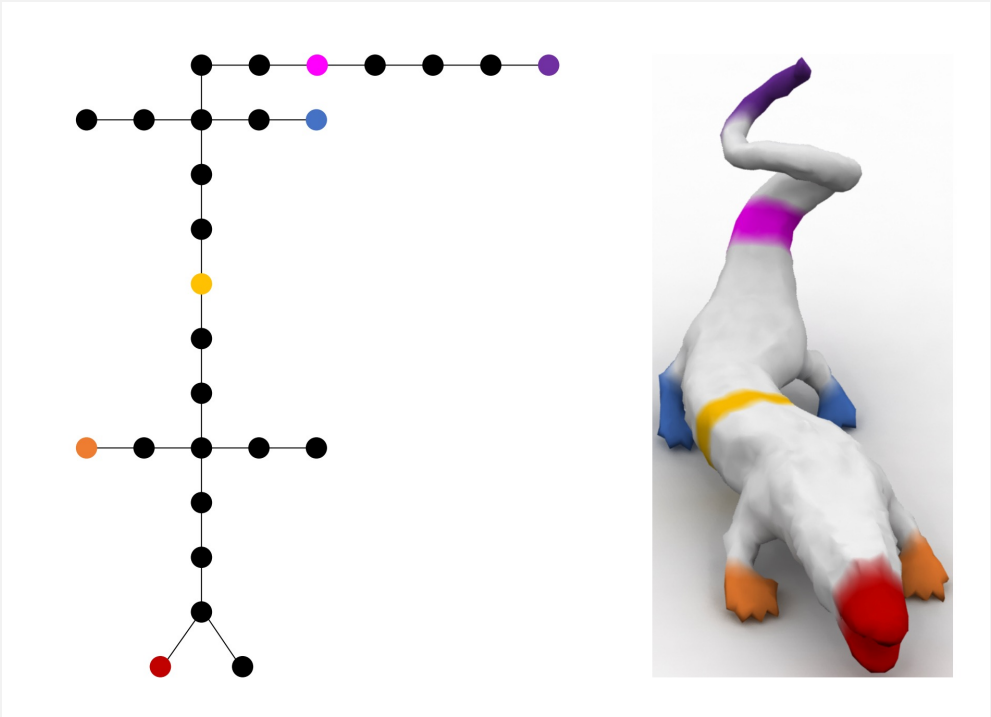
- **制約**
  - $\boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b}) := \{\boldsymbol{P} \in \mathbb{R}_{\geq 0}^{n \times m} \mid \boldsymbol{P}\mathbf{1} = \boldsymbol{a}, \boldsymbol{P}^\top \mathbf{1} = \boldsymbol{b}\}$
    - 荷物の輸送漏れや輸送不足はNG
- **出力**
  - 輸送計画：$\boldsymbol{T}^* := \underset{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})}{\operatorname{argmin}} \sum_{ij} \boldsymbol{C}_{ij} \boldsymbol{P}_{ij} \in \mathbb{R}_{\geq 0}^{n \times m}$

それでもいい感じに
アラインメントしてほしい…！

    - =ソフトなアラインメント
  - 最適輸送コスト：$\mathrm{OT}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}) := \min_{\boldsymbol{T} = \boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_{ij} \boldsymbol{C}_{ij} \boldsymbol{P}_{ij} \in \mathbb{R}_{\geq 0}$

Gromov–Wasserstein 距離

- 異なる空間に置かれた点群のマッチングがしたい



Solomon+, Entropic Metric Alignment for Correspondence Problems (SIGGRAPH 2016)

- → できます

Gromov–Wasserstein 距離

まとめ

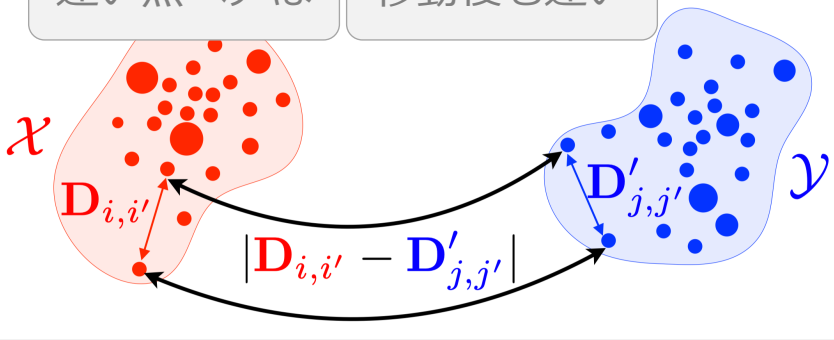
- **Gromov–Wasserstein 距離**
  - 入力
    - 分布情報 × 2：$\boldsymbol{a} \in \mathbb{R}_+^n$, $\boldsymbol{b} \in \mathbb{R}_+^m$
    - コスト行列 × 2：$\boldsymbol{D} \in \mathbb{R}_+^{n \times n}$, $\boldsymbol{D'} \in \mathbb{R}_+^{m \times m}$
  - 最適化

工場同士，デパート同士の距離しかわからない

近い点ペアは　移動後も近い

$$\mathrm{GW}((\mathbf{a},\mathbf{D}),(\mathbf{b},\mathbf{D}'))^2 \stackrel{\text{def.}}{=} \min_{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})} \sum_{i,j,i',j'} |\mathbf{D}_{i,i'} - \mathbf{D}'_{j,j'}|^2 \mathbf{P}_{i,j}\mathbf{P}_{i',j'}$$

遠い点ペアは　移動後も遠い



Peyré&Cuturi, Computational Optimal Transport, Figure 10.8

Gromov–Wasserstein 距離

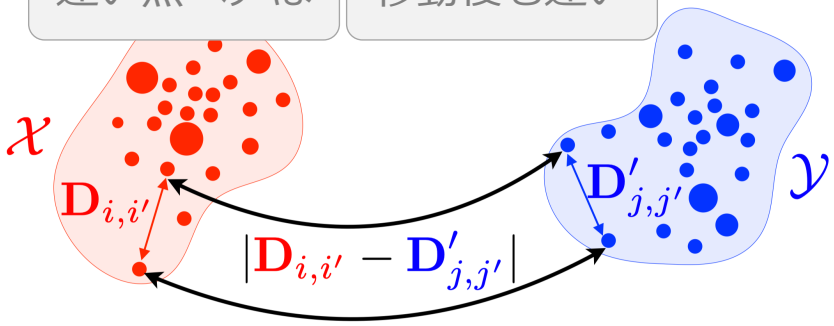
- **Gromov–Wasserstein 距離**
  - 入力
    - 分布情報 × 2：$\boldsymbol{a} \in \mathbb{R}_+^n$, $\boldsymbol{b} \in \mathbb{R}_+^m$
    - コスト行列 × 2：$\boldsymbol{D} \in \mathbb{R}_+^{n \times n}$, $\boldsymbol{D'} \in \mathbb{R}_+^{m \times m}$
  - 最適化

工場同士，デパート同士の距離しかわからない

近い点ペアは　移動後も近い

$$\mathrm{GW}((\mathbf{a},\mathbf{D}),(\mathbf{b},\mathbf{D}'))^2 \stackrel{\text{def.}}{=} \min_{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})} \sum_{i,j,i',j'} |\mathbf{D}_{i,i'} - \mathbf{D}'_{j,j'}|^2 \mathbf{P}_{i,j}\mathbf{P}_{i',j'}$$

遠い点ペアは　移動後も遠い



Peyré&Cuturi, Computational Optimal Transport, Figure 10.8

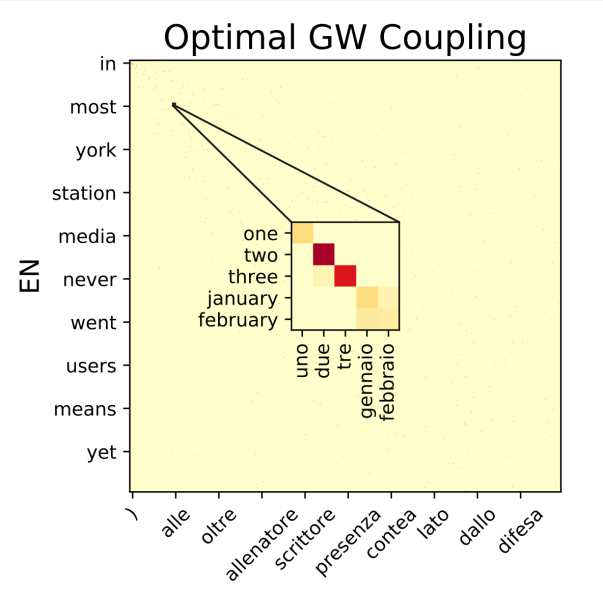
  - 参考：G–W をさらに一般化 [Alvarez-Melis+, 2019]

Alvarez-Melis+, Towards Optimal Transport with Global Invariances (AISTATS 2019)

対訳辞書構築
[Alvarez-Melis&Jaakkola'18]

計算方法：エントロピー正則化 → シンクホーンアルゴリズム [Peyré+'16]

- 言語＝単語埋め込み空間 (単語ベクトル集合) だと思って 2つの言語 (単語ベクトル集合) を G–W 距離でマッチング
- → 結構できてしまう



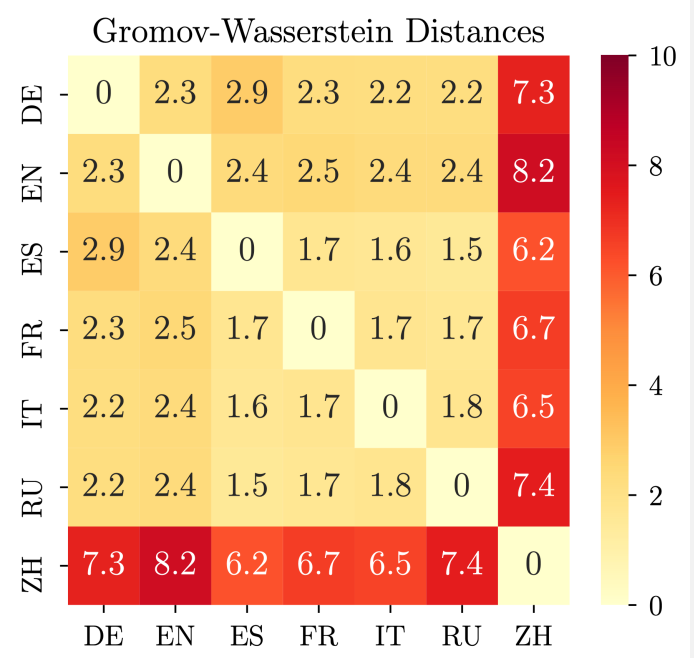
Alvarez-Melis&Jaakkola, Gromov-Wasserstein Alignment of Word Embedding Spaces (EMNLP 2018)

Alvarez-Melis&Jaakkola, Gromov-Wasserstein Alignment of Word Embedding Spaces (EMNLP 2018)
Peyré+, Gromov-Wasserstein Averaging of Kernel and Distance Matrices (ICML 2016)

対訳辞書構築
[Alvarez-Melis&Jaakkola'18]

- 言語＝単語埋め込み空間 (単語ベクトル集合) だと思って
  2つの言語 (単語ベクトル集合) を G–W 距離でマッチング

- → 副次効果：“言語間の距離” が
  見える
  - スライド前半で触れた最適輸送の使い方
    - 輸送コスト (最適値) ← 求めたいもの
    - 輸送プラン (最適解) ← 副次的にわかる
  - この論文での最適輸送の使い方
    - 言語間距離 (最適値) ← 副次的にわかる
    - アラインメント (最適解) ← 求めたいもの



Alvarez-Melis&Jaakkola, EMNLP 2018

Alvarez-Melis&Jaakkola, Gromov-Wasserstein Alignment of Word Embedding Spaces (EMNLP 2018)

不均衡最適輸送

分布の大きさが異なる場合に対応する

目次

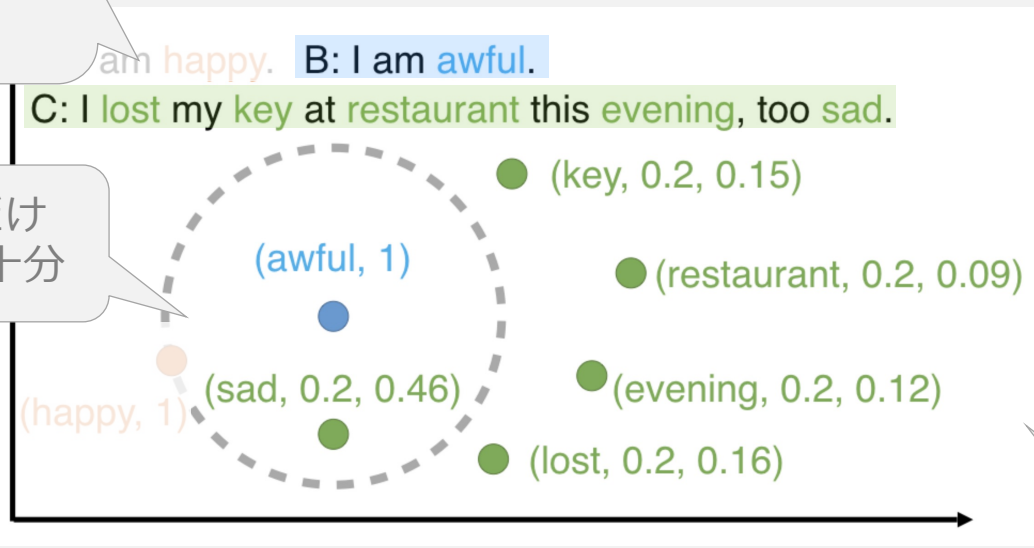
イマココ

モチベーション:
一部だけマッチングさせたい

- **長さの大きく異なる文を比較したい** [Wang+'20]

文B と 文C がともに否定的な意味であることを計算したければ

"awful" と "sad" だけマッチングできれば十分



Wang+, Robust Document Distance with Wasserstein-Fisher-Rao metric (ACML 2020), Figure 1, 改変

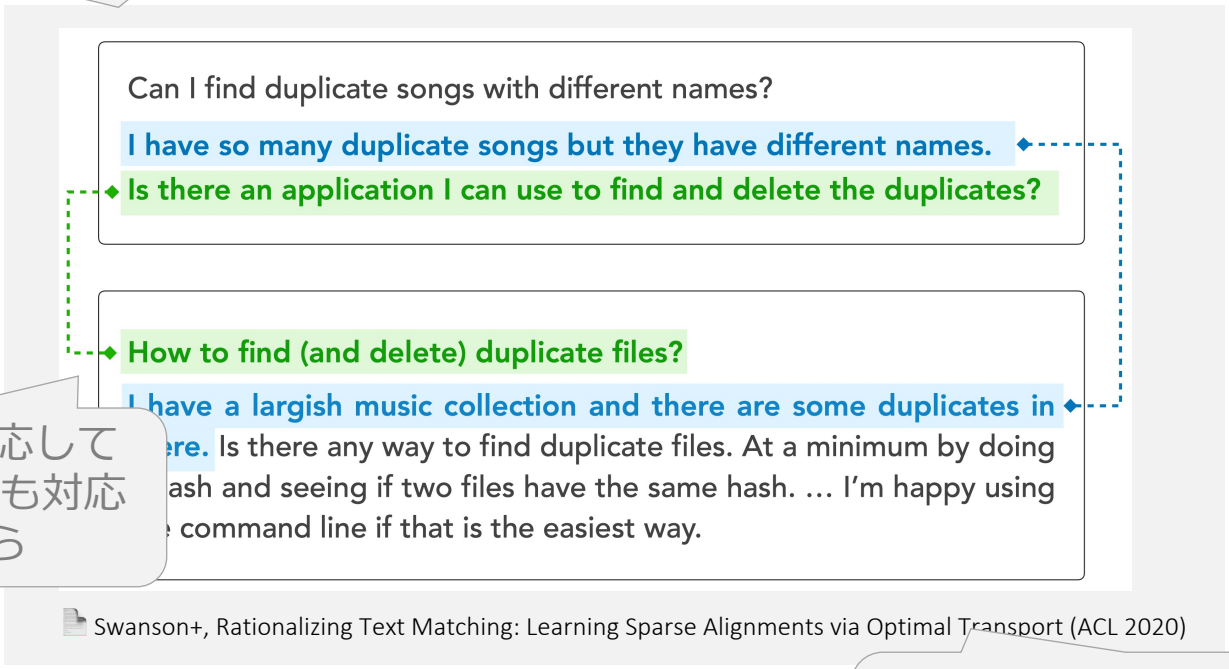
文C 内のほかの単語たち ("key", "restaurant", …) は無視したい
「運びませんでした，おしまい」にしたい

モチベーション:
一部だけマッチングさせたい

モチベーションとなる例をもうひとつ

- **関連文書を当てたい** [Swanson+'20]

これら2つの質問投稿が関連した内容だとわかるのは



Swanson+, Rationalizing Text Matching: Learning Sparse Alignments via Optimal Transport (ACL 2020)

緑の文同士 が対応していて 青の文同士 も対応しているから

他の文たちに対応する文が相手方の質問投稿に存在していなくても構わない

不均衡最適輸送の気持ち：
一部だけマッチングさせたい

荷物の量が入出力で異なる場合も扱いたい
※ 確率測度という条件を外したい

- 入力
  - 確率分布：$\boldsymbol{a} = (a_1, \dots, a_n) \in [0,1]^n, \ \sum_i a_i = 1$
  - 確率分布：$\boldsymbol{b} = (b_1, \dots, b_m) \in [0,1]^m, \ \sum_j b_j = 1$
  - 輸送コスト：$\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n \times m}$
    - $\boldsymbol{C}_{ij} \in \mathbb{R}_{\geq 0}$：$a_i$ に対応する拠点と b_j に対応する拠点の間の輸送コスト
- 制約
  - $\boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b}) := \{\boldsymbol{P} \in \mathbb{R}_{\geq 0}^{n \times m} \mid \boldsymbol{P}\mathbf{1} = \boldsymbol{a}, \boldsymbol{P}^\top \mathbf{1} = \boldsymbol{b}\}$
    - 荷物の輸送漏れや輸送不足はNG

過不足なくカップリングという条件を外したい
足りなくても溢れても構わない

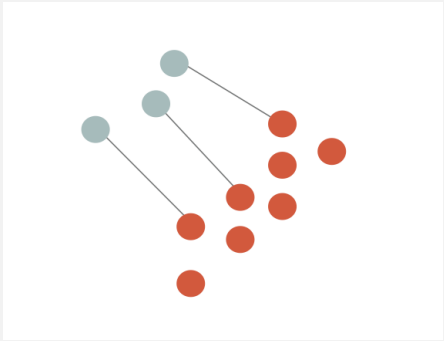
- 出力
  - 輸送計画：$\boldsymbol{T} = \boldsymbol{P}^* := \underset{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})}{\operatorname{argmin}} \sum_{ij} \boldsymbol{C}_{ij} \boldsymbol{P}_{ij} \in \mathbb{R}_{\geq 0}^{n \times m}$
    - =ソフトなアラインメント
  - 最適輸送コスト：$\mathrm{OT}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{C}) := \min_{\boldsymbol{P} \in \boldsymbol{U}(\boldsymbol{a}, \boldsymbol{b})} \sum_{ij} \boldsymbol{C}_{ij} \boldsymbol{P}_{ij} \in \mathbb{R}_{\geq 0}$

不均衡最適輸送

- 色々な定式化・計算方法が提案されている
- ここでは**2つの典型的なアプローチ** + 統一的な解釈を紹介
  1. **Optimal Partial Transport**：荷物を一部だけ輸送する
  2. **(エントロピー正則化つき) 不均衡最適輸送**：過不足にペナルティ
  3. 外側にブラックホールを作る
  4. 統一的な解釈

不均衡最適輸送の定式化 その1
Optimal Partial Transport [Caffarelli&McCann'10][Figalli'10]

- 考え方：**動かさない荷物があっても良い**
  - 工場で作ったものを全部運ばなくても良い
  - デパートの需要を全部満たさなくても良い



Bonneel&Coeurjolly, SPOT: Sliced Partial Optimal Transport (SIGGRAPH 2019) ……の，著者による解説記事 より

- **合計どれだけの荷物を動かすかをあらかじめ決めておく**
  - 例
    - 工場全体の生産量： 2.5
    - デパート全体の需要： 1.5
    - **動かす荷物の量 (パラメータ)**：1.2 ≤ min(2.5, 1.5)

Caffarelli&McCann, Free boundaries in optimal transport and Monge-Ampère obstacle problems (Ann. Math. 2010)
Figalli, The Optimal Partial Transport Problem (Arch. Ration. Mech. Anal. 2010)

不均衡最適輸送の定式化 その1
Optimal Partial Transport [Caffarelli&McCann'10][Figalli'10]

まとめ

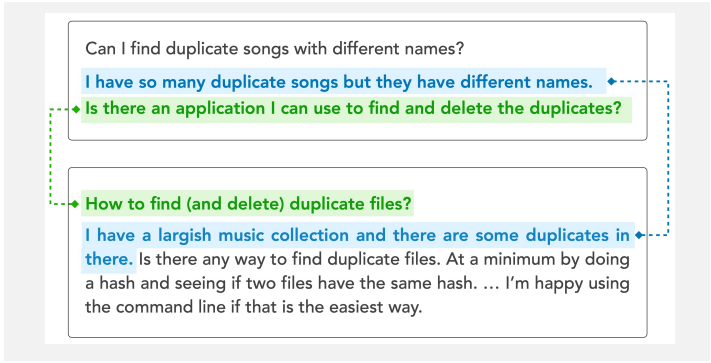
- 考え方：**動かさない荷物があっても良い**
- **合計どれだけの荷物を動かすかをあらかじめ決めておく**
- → アルゴリズム：標準的な最適輸送問題に帰着できる [Chapel+'20]
  1. 「空の荷物を運び出す**ダミー工場**」や「荷物を受け取ったことにして捨てる**ダミーデパート**」を用意
  2. 標準的な (balanced な) 設定で最適輸送問題を解く
  3. **ダミー工場**や**ダミーデパート**が関わる輸送は無視する (輸送計画行列のダミー工場行とダミーデパート列を捨てる)

  – ※ 自分でやるのも簡単，POT にも 実装あり

Caffarelli&McCann, Free boundaries in optimal transport and Monge-Ampère obstacle problems (Ann. Math. 2010)
Figalli, The Optimal Partial Transport Problem (Arch. Ration. Mech. Anal. 2010)
Chapel+, Partial Optimal Transport with Applications on Positive-Unlabeled Learning (NeurIPS 2020)

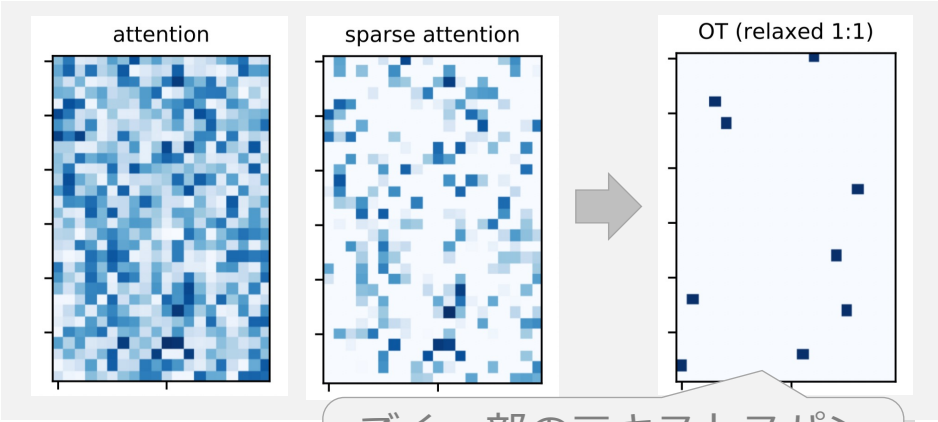
不均衡最適輸送の定式化 その1
Optimal Partial Transport の NLP 利用 [Swanson+'20]

- 2文書が入力される分類問題を注意機構つきの NN で解く
- アイデア：注意機構による
  マッチングはスパースで良いはず



おおよそ

- → 注意機構を Optimal Partial Transport (+ Sinkhorn) 化
  - 一部だけ輸送すれば OK
  - ✔ 性能は既存法とコンパラ
  - ✔ 超スパースな (解釈性の高い)
    アテンション行列が得られる



ごく一部のテキストスパンのみを輸送

Swanson+, Rationalizing Text Matching: Learning Sparse Alignments via Optimal Transport (ACL 2020)

不均衡最適輸送の定式化 その2
(エントロピー正則化つき) 不均衡最適輸送

- 考え方：過不足の量に応じてペナルティ
  - ある工場では 0.3 の製品を作る
    - 0.2 < 0.3 だけ出荷しても良い (余った 0.1 分ペナルティ)
    - 0.5 > 0.3 だけ出荷しても良い (足りない 0.2 分ペナルティ)
  - あるデパートでは 0.4 の製品が必要
    - 0.3 < 0.4 だけ受け取っても良い (足りない 0.1 分ペナルティ)
    - 0.8 > 0.4 だけ受け取っても良い (もらいすぎの 0.4 分ペナルティ)

不均衡最適輸送の定式化 その2
(エントロピー正則化つき) 不均衡最適輸送

- カップリング制約を外す代わりに過不足にペナルティ

$$\min_{\mathbf{P}\in\mathbb{R}_+^{n\times m}} \langle \mathbf{C}, \mathbf{P}\rangle + \tau_1 \mathbf{D}_\varphi(\mathbf{P}\mathbb{1}_m|\mathbf{a}) + \tau_2 \mathbf{D}_\varphi(\mathbf{P}^\top \mathbb{1}_m|\mathbf{b})$$

&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019), Eq. (10.8)

同時確率でなくてもOK

荷物の過不足に応じてペナルティ

cf. $\mathbf{U}(\mathbf{a},\mathbf{b}) \overset{\text{def.}}{=} \left\{\mathbf{P}\in\mathbb{R}_+^{n\times m} \;:\; \mathbf{P}\mathbb{1}_m = \mathbf{a} \quad \text{and} \quad \mathbf{P}^{\mathrm{T}}\mathbb{1}_n = \mathbf{b}\right\}$

不均衡最適輸送の定式化 その2
(エントロピー正則化つき) 不均衡最適輸送

- カップリング制約を外す代わりに過不足にペナルティ

$$\min_{\mathbf{P}\in\mathbb{R}_+^{n\times m}} \langle \mathbf{C}, \mathbf{P}\rangle + \tau_1 \mathbf{D}_\varphi(\mathbf{P}\mathbb{1}_m|\mathbf{a}) + \tau_2 \mathbf{D}_\varphi(\mathbf{P}^\top \mathbb{1}_m|\mathbf{b})$$

Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019), Eq. (10.8)

- 特殊ケースには名前がついている
  - {Wasserstein–Fisher–Rao, Hellinger–Kantorovich} distance
    - $c(x,y) = -\log\cos(\min(d(x,y)/\kappa, \pi/2))$
    - $\mathcal{D}_\varphi = \mathrm{KL}$
  - {Gauss–Hellinger, Gaussian–Hellinger–Kantorovich} distance
    - $c(x,y) = \|x-y\|^2$
    - $\mathcal{D}_\varphi = \mathrm{KL}$

不均衡最適輸送の定式化 その2
(エントロピー正則化つき) 不均衡最適輸送

- カップリング制約を外す代わりに過不足にペナルティ

$$\min_{\mathbf{P}\in\mathbb{R}_+^{n\times m}} \langle \mathbf{C}, \mathbf{P}\rangle + \tau_1 \mathbf{D}_\varphi(\mathbf{P}\mathbb{1}_m|\mathbf{a}) + \tau_2 \mathbf{D}_\varphi(\mathbf{P}^\top \mathbb{1}_m|\mathbf{b})$$

Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019), Eq. (10.8)

- **解き方**
  - エントロピー正則化項をつければ Sinkhorn アルゴリズムに帰着
    - 「一般化 Sinkhorn アルゴリズム」 [Chizat+'18]
  - → POT の `ot.unbalanced.sinkhorn_knopp_unbalanced` [url]

Chizat+, Scaling algorithms for unbalanced transport problems (Math. Comput. 2018)

不均衡最適輸送の定式化 その2

エントロピー正則化つき不均衡最適輸送 のNLP応用

- kNN による**文書分類**
  - 一般の最適輸送 [Kusner+'15] → **不均衡最適輸送** [Wang+'20]

輸送コスト：−log cos，罰則：KL (Wasserstein–Fisher–Rao)
エントロピー正則化項をつけて一般化 Sinkhorn で解く

- **辞書構築** (変換行列を学習 → 最適輸送でチェック)
  - 一般の最適輸送 [Zhang+'17] → **不均衡最適輸送** [Zhao+'20]

輸送コスト：RCSLS (パラメトライズされた双線形形式)，罰則：KL
エントロピー正則化項をつけて一般化 Sinkhorn で解く

- 機械翻訳の**自動評価**
  - 一般の最適輸送 [Zhao+'19] → **不均衡最適輸送** [Chen+'20]

輸送コスト：cos，罰則：KL
エントロピー正則化項をつけて一般化 Sinkhorn で解く

Kusner+, From Word Embeddings To Document Distances (ICML 2015)
Wang+, Robust Document Distance with Wasserstein-Fisher-Rao metric (ACML 2020)
Zhang+, Earth Mover's Distance Minimization for Unsupervised Bilingual Lexicon Induction (EMNLP 2017)
Zhao+, A Relaxed Matching Procedure for Unsupervised BLI (ACL 2020)
Zhao+, MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance (EMNLP 2019)
Chen+, Evaluating Natural Language Generation via Unbalanced Optimal Transport (IJCAI 2020)

不均衡最適輸送の一般化・統一解釈

- 紹介した定式化・計算方法
  - **Optimal Partial Transport**
    - 荷物を一部だけ輸送する
  - **(エントロピー正則化つき) 不均衡最適輸送**
    - 過不足にペナルティ (とくにペナルティが KL の場合を紹介)
- ほかにも各種定式化・計算方法が提案されている
  - 領域の境界に，**無限の荷物を飲み込め・生成できるブラックホール**を作っておく [Figalli&Gigli'06]
  - etc.

Figalli&Gigli, A new transportation distance between non-negative measures, with applications to gradients flows with Dirichlet boundary conditions (J. Math. Pures Appl. 2006)

不均衡最適輸送の一般化・統一解釈

- 紹介した定式化・計算方法
  - **Optimal Partial Transport**
  - **(エントロピー正則化つき) 不均衡最適輸送**
- ほかにも各種定式化・計算方法が提案されている
  - 領域の境界に，**無限の荷物を飲み込め・生成できるブラックホール**を作っておく [Figalli&Gigli'06]
  - etc.
- → これらを一般化する試み [Liero+'18][Chizat+'18][Sato+'20]
  - たとえば Optimal Partial Transport は過不足のコストが L1 (total variation) で入っているとみなせる

Figalli&Gigli, A new transportation distance between non-negative measures, with applications to gradients flows with Dirichlet boundary conditions (J. Math. Pures Appl. 2006)
Liero+, Optimal Entropy-Transport problems and a new Hellinger–Kantorovich distance between positive measures (Invent. Math. 2018)
Chizat+, Unbalanced Optimal Transport: Dynamic and Kantorovich Formulation (J. Funct. Anal. 2018)
Sato+, Fast Unbalanced Optimal Transport on a Tree (NeurIPS 2020)

構造を考慮した最適輸送

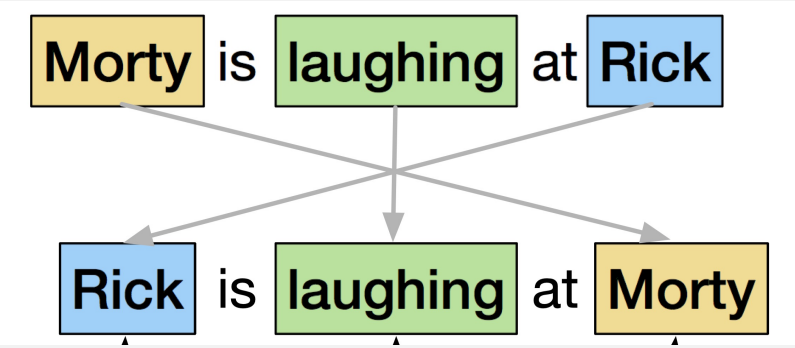
このセクションの話題のみを取り扱った
スライドを別途作りました (2022年9月)
→ 構造を持った言語データと最適輸送

目次

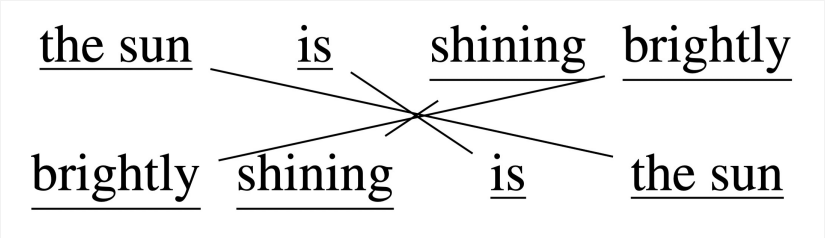
- 基礎編：もっとも標準的な形式の最適輸送問題を知る
- **応用編1**：自動微分可能な最適輸送コスト
  - 自動微分可能な最適輸送コストが手に入れば実現すること
  - 自動微分可能な最適輸送コストの NLP での利用例
  - 自動微分可能な最適輸送コストを使いたい場合の選択肢
  - まとめ
- **応用編2**：最適輸送の変種
  - Gromov–Wasserstein 距離：違う空間に存在する点群をマッチング
  - 不均衡最適輸送：「過不足なくマッチング」という制約を外す
  - 構造を考慮した最適輸送 ：語順や木構造を考慮する イマココ
- 全体まとめ

モチベーション:
テキストは「点群」よりもリッチな構造を持っている

- **たとえば語順**



Liu+, Matching Natural Language Sentences with Hierarchical Sentence Factorization (WWW 2018)



Chow+. WMDo: Fluency-based Word Mover's Distance for Machine Translation Evaluation (workshop 2019)

  - Word Mover's Distance は主語と目的語がひっくり返っていても コストゼロでマッチングさせてしまう

「点群」よりもリッチな構造を考慮したい
その1：語順

- 輸送計画は対角成分ばかり (語順変更しないように) 使ってほしい [Su&Hua'17]

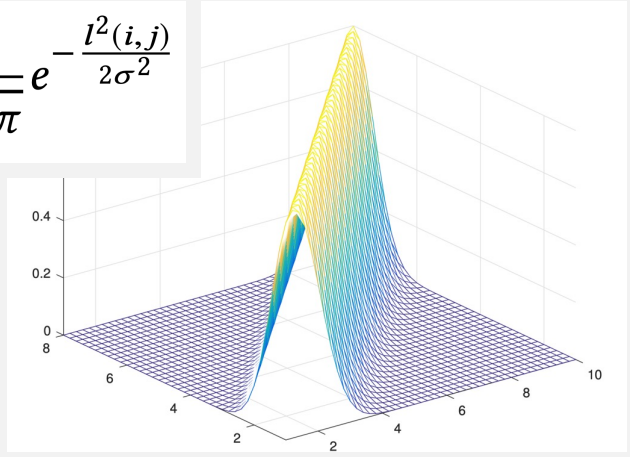
$$\underset{T \in \mathbb{R}_+^{M \times N}}{\text{minimize}} \quad \sum_{i,j} T_{ij} D_{ij} - \lambda_1 I(T) + \lambda_2 KL(T||P)$$

対角成分ばかり使ってほしい その1

$$I(T) = \sum_{i=1}^{M'} \sum_{j=1}^{N'} \frac{T_{ij}}{(\frac{i}{M'} - \frac{j}{N'})^2 + 1}$$

対角成分ばかり使ってほしい その2
輸送行列はこれに近づいてほしい

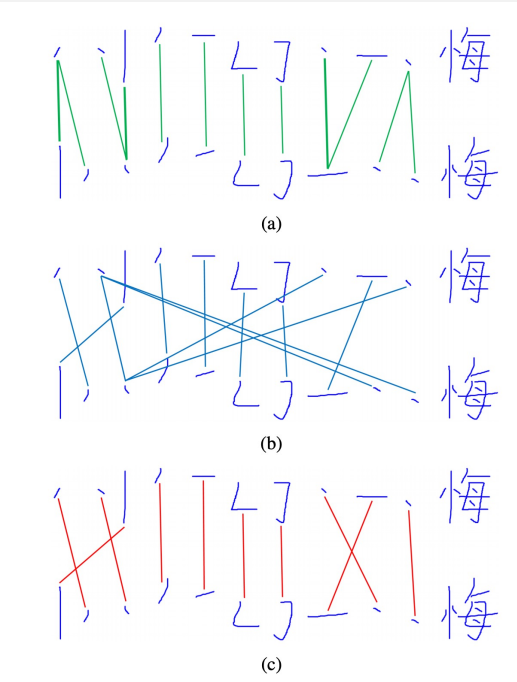
$$P_{ij} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{l^2(i,j)}{2\sigma^2}}$$



Su&Hua, Order-preserving Optimal Transport for Distances between Sequences (CVPR 2017)

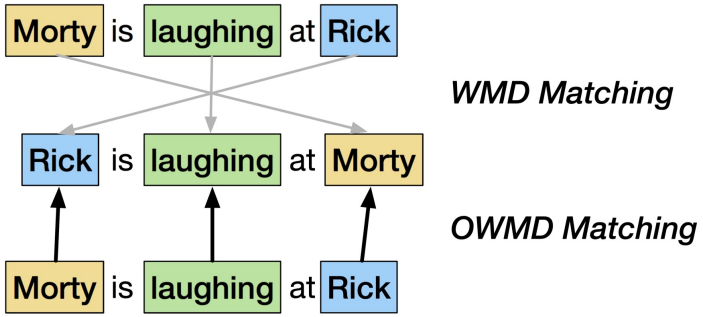
「点群」よりもリッチな構造を考慮したい
その1：語順

- この目的関数を作ったコンピュータヴィジョンでの利用 [Su&Hua'17]
  - 書き順を考慮した漢字の類似性



Su&Hua, Order-preserving Optimal Transport for Distances between Sequences (CVPR 2017)

- 自然言語処理での利用 [Liu+'18]
  - 語順を考慮した文類似度



Liu+, Matching Natural Language Sentences with Hierarchical Sentence Factorization (WWW 2018)

Su&Hua, Order-preserving Optimal Transport for Distances between Sequences (CVPR 2017)
Liu+, Matching Natural Language Sentences with Hierarchical Sentence Factorization (WWW 2018)

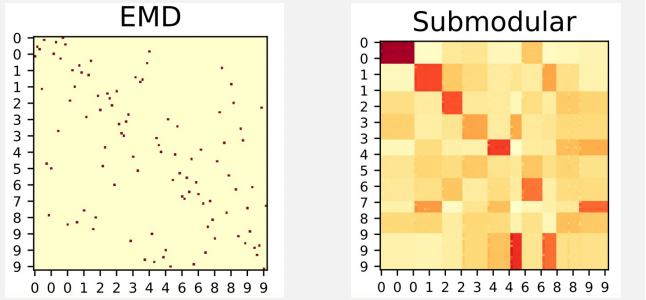
「点群」よりもリッチな構造を考慮したい
その1：語順

- ほかの例
  - [Chow+'19] 最適輸送コストに METEOR の fragment penalty を掛け合わせる
    - 単語の挿入・削除や語順変更によってアラインメントに必要なチャンクがどの程度たくさん必要か……に応じてペナルティ

2チャンク必要な例

4チャンク必要な例

  - [Alvarez-Melis+'18] 輸送元の n-gram ができるだけ輸送先の n-gram となるようにする
    - お互いがグループになっている範囲であれば (n-gram と n-gram の間であれば), 輸送される単語が増えた際に必要なコストはどんどん逓減していって良い (劣モジュラ性)



著者ポスター より

提案法はグループ間での輸送を好む

Chow+. WMDo: Fluency-based Word Mover's Distance for Machine Translation Evaluation (workshop 2019)
Alvarez-Melis+, Structured Optimal Transport (AISTATS 2018)

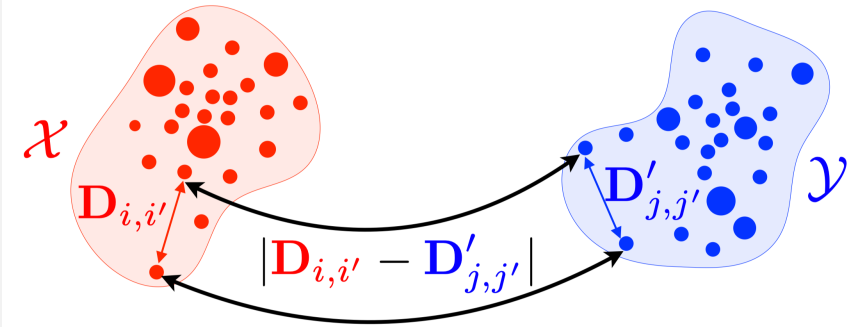
「点群」よりもリッチな構造を考慮したい
その2：木構造

- 復習：**Gromov–Wasserstein 距離**

近い点ペアは　移動後も近い

$$\mathrm{GW}((\mathbf{a},\mathbf{D}),(\mathbf{b},\mathbf{D}'))^2 \overset{\text{def.}}{=} \min_{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})} \sum_{i,j,i',j'} |\mathbf{D}_{i,i'} - \mathbf{D}'_{j,j'}|^2 \mathbf{P}_{i,j}\mathbf{P}_{i',j'}$$

遠い点ペアは　移動後も遠い



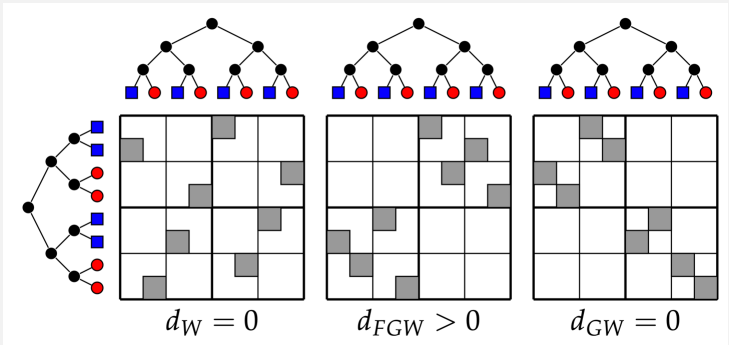
Peyré&Cuturi, Computational Optimal Transport, Figure 10.8

「点群」よりもリッチな構造を考慮したい
その2：木構造

- **Fused Gromov–Wasserstein 距離** [Vayer+'19]
  - **最適輸送** (Wasserstein) と **Gromov–Wasserstein** を半々で使う
  - FGW(a,b) := (1-α) W(a,b) + α GW(a,b)

輸送コストを最小限にとどめてほしい

元の各分布が持っている**木構造**も
輸送前後で**維持**されてほしい



Vayer+, F... Gromov-Wasserstein Distancefor Structured Objects (A... 2020)

Wのみを用いる場合：
同じ色の (近い) ノード同士が
アラインメントされる

GWのみを用いる場合：
木構造が維持される

Vayer+, Fused Gromov-Wasserstein Distancefor Structured Objects (Algorithm 2020)

まとめ

再掲スライドたち

目次

- **前半の目標 ：「最適輸送わかってしまった……」になる**
  - 基礎編：もっとも標準的な形式の最適輸送問題を知る
- **後半の目標：手元のすべての問題で最適輸送を使いたくなる**
  - 応用編1：自動微分可能な最適輸送コスト
  - 応用編2：最適輸送の変種
- 全体まとめ　イマココ

最適輸送は自然言語処理の人にとって勉強する価値が高い

まとめ

- 最適輸送は自然言語処理とすごく相性が良い
  - **“近さ” “遠さ” を考えられる空間** で

埋込ベース，ニューラルネットベースの各種手法
(＝対象が自然に距離空間に入っている状態) との相性が良い

荷物全体 (点群) を移し換えるコストを計算する 道具

言語的対象は (たいてい) 何かの集まり
文＝単語列，文書＝文の列，コーパス＝文集合，…

対象間の類似度や距離の計算は
自然言語処理で頻出

  - 副次効果として **アラインメント情報** が得られる

自然言語処理でしばしば要請される
例：文と文の関係を単語と単語の関係に帰着させたい

高い解釈性；
輸送コスト (最適値) だけではなく輸送プラン (最適解) がわかる

最適輸送を勉強するタイミングは今

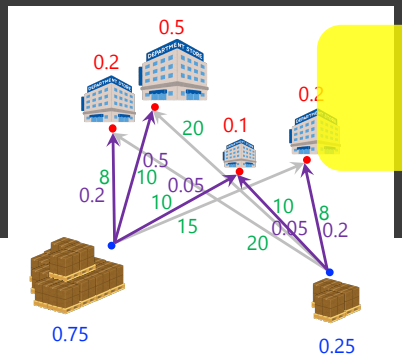
まとめ

- **利用しやすく安心して使えるライブラリ (ソルバ) の整備**
  - Python Optimal Transport (POT) (2020) GitHub
  - Optimal Transport Tools (OTT) with JAX (2022) GitHub
- **計算技法・理論に関する勉強しやすい教科書・資料の整備**
  - 英：Peyré&Cuturi, Computational Optimal Transport (2019)
  - 和：佐藤，最適輸送の理論とアルゴリズム (2022, 予定)
- **最適輸送を利用している自然言語処理の研究も急増**
  - ちょっと勉強すれば多くの論文の気持ちがわかるようになる
- **最適輸送が適した問題はまだ大量に残っている**
  - 対象がベクトル集合で表現できる，アラインメントが必要，確率分布間の距離を測りたい，etc.

Flamary+, POT: Python Optimal Transport (JMLR 2021) Cuturi+, Optimal Transport Tools (OTT): A JAX Toolbox for all things Wasserstein (arXiv 2022)
Peyré&Cuturi, Computational Optimal Transport (Foundations and Trends in Machine Learning 2019) 佐藤, 最適輸送の理論とアルゴリズム (講談社 2022)

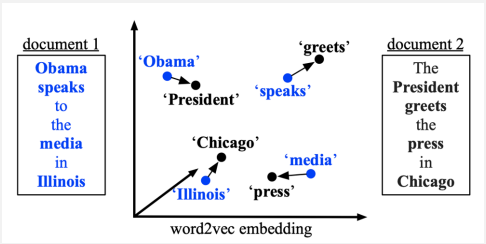
前半のまとめ

目標：最適輸送わかった… になる



まとめ

- 最適輸送の**視覚的・物理的な気持ち**を知る
  - 荷物をもっとも効率よく移し替えるプラン・総コストを求める手法
  - WMD：単語ベクトル集合を移し替えるコスト＝テキストの非類似度
- 最適輸送の**入出力**を理解する
  - 入力：**a** (荷物の量分布1)，**b** (荷物の量分布2)，**C** (輸送コスト行列)
  - 出力：**P** (輸送計画行列)，総輸送コスト ($\Sigma_{\text{道}}$ 輸送コスト × 輸送量)
  - POT 等のライブラリに投げれば解いてくれる
  - 解きたい問題に合わせて入力 (分布，輸送コスト) を考える
- 最適輸送の**形式的な定義**を理解する
  - 線形計画で定式化される
  - ワッサーシュタイン距離は最適輸送コストの特殊ケース



Kusner+, From Word Embeddings To Document Distances

- **最適輸送はアラインメントしながら点群の違いを測る道具**

後半のまとめ

目標：手元のすべての問題を最適輸送で解きたくなる

まとめ

- **反復計算 (シンクホーンアルゴリズム) に帰着させた自動微分可能な最適輸送コスト**を知る
  - 😀 **計算コスト**：並列化やGPU利用が容易
  - 😀 **自動微分**：最適輸送コストが下がる方向に入力 (重み and/or コスト行列) を更新できる
  1. エントロピー正則化つき最適輸送コスト [Cuturi'13] ※ バイアス有
  2. シンクホーン・ダイバージェンス [Ramdas+'17][Genevay+'18]
  3. 近接勾配法に基づく計算 (IPOT) [Xie+'19]
- **最適輸送の変種**を知る
  - Gromov–Wasserstein：違う空間に存在する点群をマッチングする
  - 不均衡最適輸送：荷物の過不足を許す
  - 構造を考慮する：語順，木構造

Cuturi, Sinkhorn Distances: Lightspeed Computation of Optimal Transport (NIPS 2013)
Ramdas+, On Wasserstein Two Sample Testing and Related Families of Nonparametric Tests (Entropy 2017)
Genevay+, Learning Generative Models with Sinkhorn Divergences (AISTATS 2018)
Xie+'19, A Fast Proximal Point Method for Computing Exact Wasserstein Distance (UAI 2019)