KDDI Research

転移学習の基礎からコーディングまで
～様々なデータに適応するための機械学習の方法論～

名古屋大学 松井孝太

大阪大学 Zhi Li

KDDI総合研究所 米川慧、黒川茂莉

目次

■ **転移学習概略**
（名古屋大学 松井、KDDI総合研究所 黒川：20分）

■ **転移学習の手法**
（KDDI総合研究所 黒川、大阪大学 Li、KDDI総合研究所 米川：45分）

■ **まとめ・発展的な話題**
（名古屋大学 松井、KDDI総合研究所 黒川：10分）

転移学習とは？

Inductive Transfer : 10 Years Later (NIPS'05 Workshop)

帰納的転移または**転移学習**とは, 新しいタスクに対する有効な仮説を効率的に見つけ出すために, 一つ以上の別のタスクで学習された知識を保持・ 適用する問題

→ 人間の「過去に経験した問題解決から得た知識を現在の問題に利用」する能力のアナロジー

- 過去の知識の積み上げができる
- 問題の間の類似構造を自然に把握できる
- 獲得した知識を繰り返し更新・適用することができる

これらの機能を機械学習モデルで実現したい

関連（類似）タスク
知識・経験・能力の蓄積

関連タスクの
知識・経験・能力
を新たなタスクに利用

新規タスク
新たな知識・経験・能力の学習

転移学習の機械学習の問題としての定式化（1/2）

KDDI Research

■ 通常の機械学習の定式化（期待リスク最小化）：

$$\min_{h \in \mathcal{H}} R(h) := \mathbb{E}_{(X,Y) \sim P_{\mathcal{X} \times \mathcal{Y}}} [\ell(h(X), Y)]$$

- データ生成分布　$(X, Y) \sim_{i.i.d} P_{\mathcal{X} \times \mathcal{Y}}$
- 仮説（予測モデル）　$h : \mathcal{X} \to \mathcal{Y}$
- 損失関数　$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$

$P_{\mathcal{X} \times \mathcal{Y}}$ が未知のため, 実際には観測データ $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$ から定まる**経験リスク**を最小化して仮説を学習

経験リスク最小化（empirical risk minimization, ERM）

$$\hat{h} = \operatorname*{argmin}_{h \in \mathcal{H}} \hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(\boldsymbol{x}_i), y_i)$$

転移学習の機械学習の問題としての定式化（2/2）

■ 期待リスク最小化による転移学習の定式化

前提：十分なラベル付きデータがないため目標ドメインのみで学習の実行が困難

元ドメイン
既知タスクのドメイン

$(\mathcal{X}_S \times \mathcal{Y}_S, P_{\mathcal{X}_S \times \mathcal{Y}_S})$

$\mathcal{D}_n^S$　データ集合

F^S　特徴量

$\hat{h}_S$　学習済みモデル

目標ドメイン
新規タスクのドメイン

$(\mathcal{X}_T \times \mathcal{Y}_T, P_{\mathcal{X}_T \times \mathcal{Y}_T})$

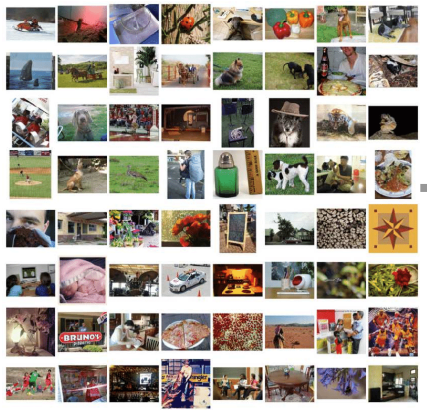
目的：期待リスク最小のモデルの学習

$$h_T^* = \underset{h \in \mathcal{H}_T}{\operatorname{argmin}} \, \mathbb{E}_{P_{\mathcal{X}_T \times \mathcal{Y}_T}} [\ell(h(X), Y)]$$

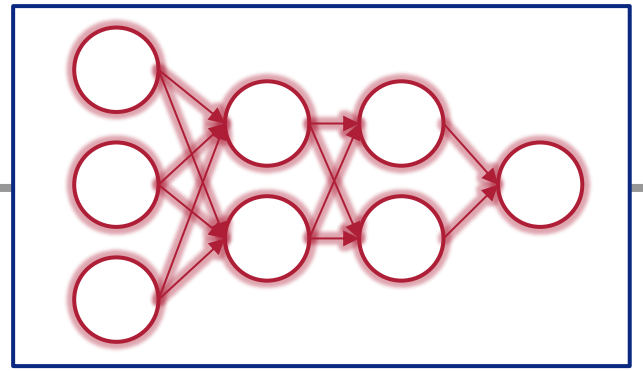
元ドメインの「知識」を「転移」して目標ドメインのモデルを効率的に学習

転移学習の重要性：通常の機械学習モデルの限界（画像認識の例）

入力：画像



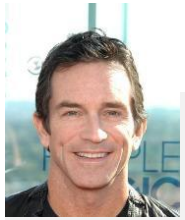
画像認識モデル



初見の画像を認識（汎化）

■ モデルの学習時と適用時でデータの質が異なる

- 例：**屋外の顔画像**で訓練したモデルを**屋内の顔認識**に用いる

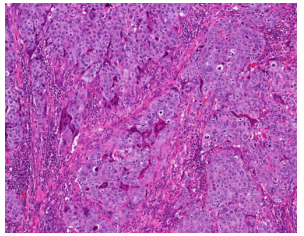


影などはなく
はっきりとした画像

顔の半分が陰っており
左右でコントラストが異なる画像

■ データが少なく、スクラッチからモデルを学習するのが困難

- 例：**病理画像**はデータの専門性が高く十分な枚数を用意できない



転移学習の重要性：通常の機械学習モデルの限界（理論的な説明）

■ 通常の機械学習の理論的正当化：**大数の法則**

$(X_i, Y_i) \sim_{i.i.d} P_{\mathcal{X}\times\mathcal{Y}}$ のとき，任意の $\varepsilon > 0$ に対して以下が成り立つ

$$\lim_{n\to\infty} \Pr_{\mathcal{D}_n}\left(|\hat{R}(h) - R(h)| > \varepsilon\right) = 0$$

データが独立同一に分布 $P_{\mathcal{X}\times\mathcal{Y}}$ から得られるとき，データ数を十分大きく取れば経験リスクと期待リスクの差は確率的に0に収束

➡ たくさんデータを集めれば精度の高いモデルが学習できる！

■ しかし複数のドメインを考える問題ではデータの独立同一性が成り立たない

➡ 転移学習の様々な方法で精度の高いモデルを学習する！

転移学習への注目（1/2）機械学習コミュニティでの注目度

NeurIPS2020, ICLR2021, ICML2021（機械学習分野のトップ国際会議）で発表された転移学習関連論文の集計

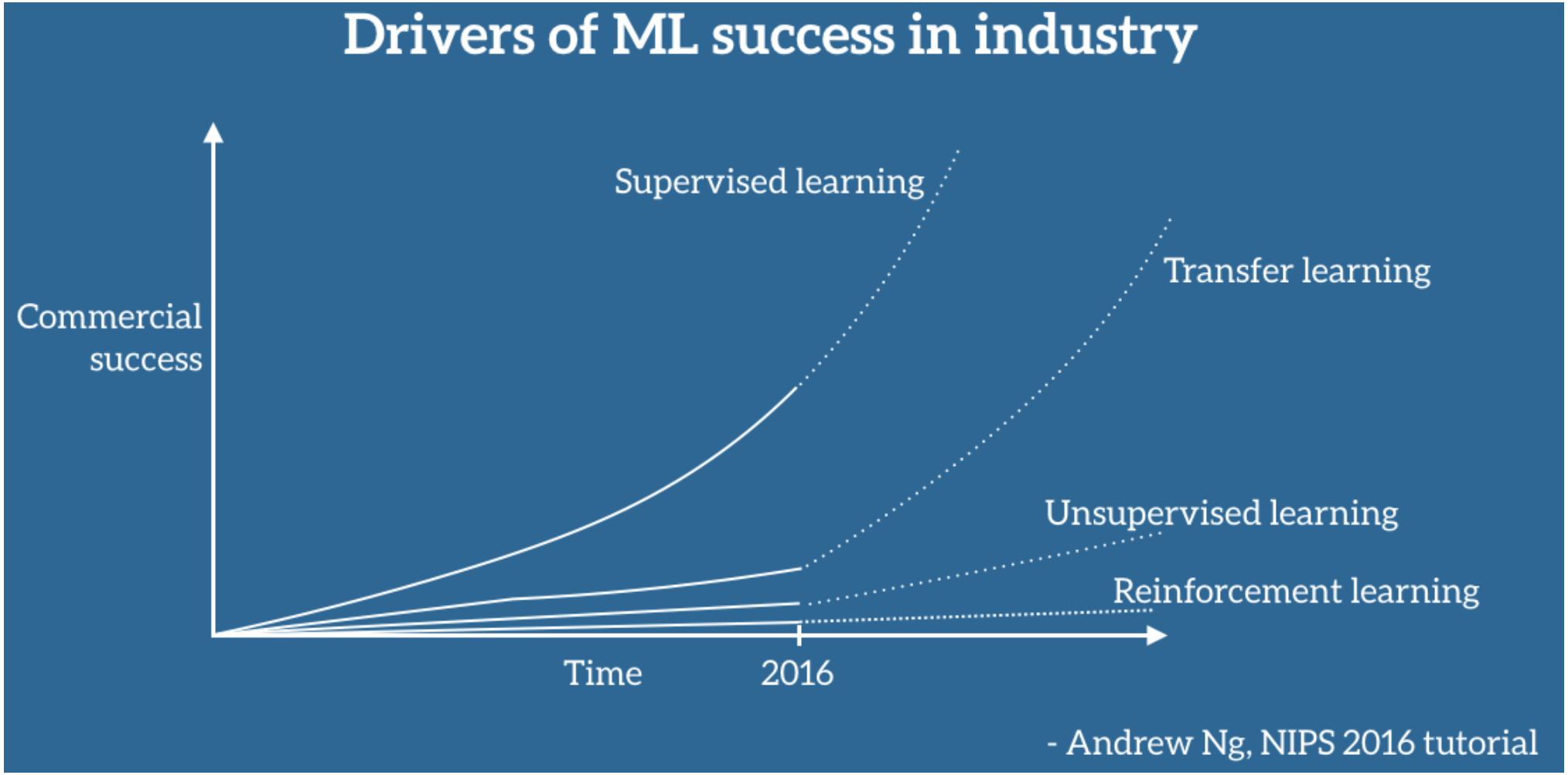
- 転移学習論文数 / 全採択論文数
  - NeurIPS2020：150 / 1898 (8%)
  - ICLR2021：96 / 860 (11%)
  - ICML2021：85/1184 (7%)

全体の10%前後が転移学習の関連論文（高い注目度）

- 主要キーワード
  - meta learning
  - few-shot learning
  - continual / lifelong learning
  - domain / distribution shift
  - domain adaptation
  - domain generalization
  - out of distribution
  - (knowledge) distillation
  - disentanglement

転移学習への注目（2/2）

■ Andrew Ng先生の講演でも次の機械学習のドライバーとみなされている



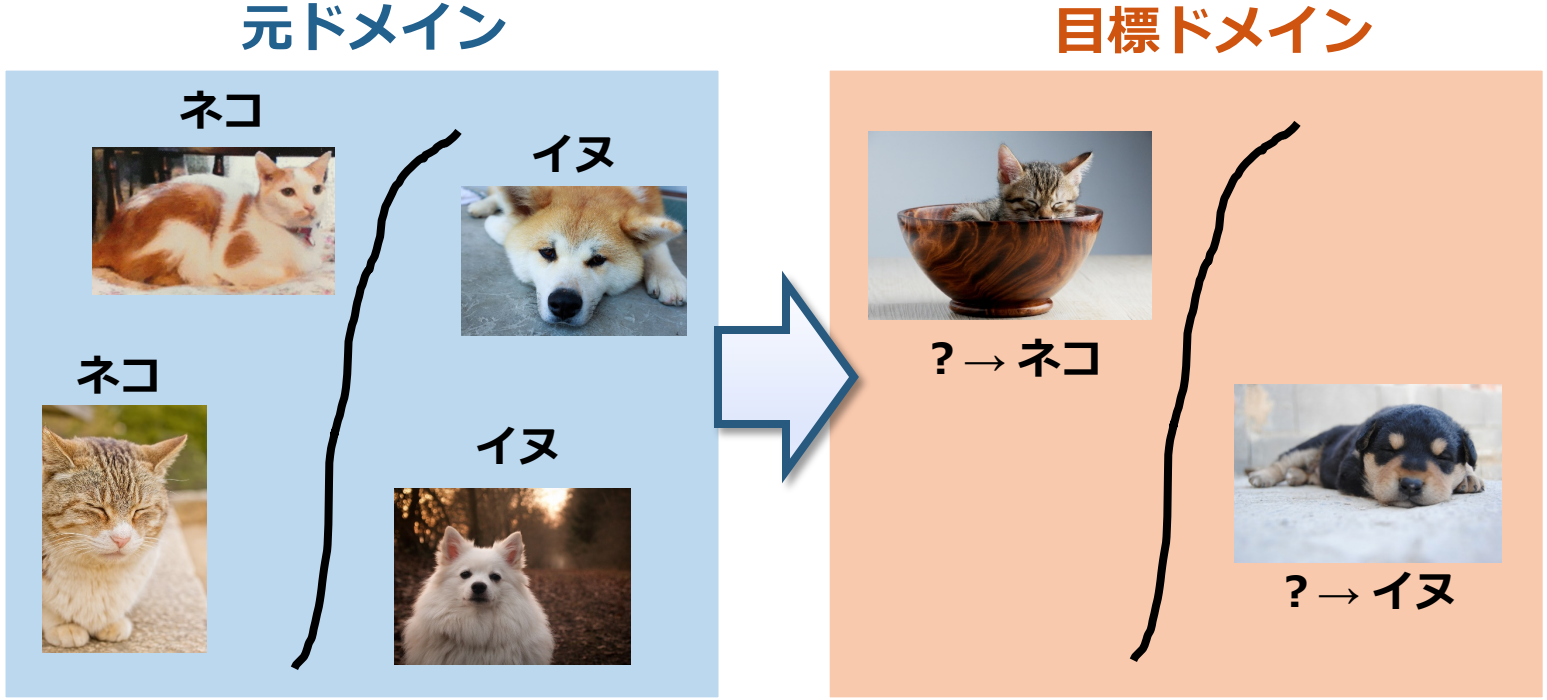
出典：https://ruder.io/transfer-learning/index.html

想定する状況（通常の機械学習）

■ 元ドメインと目標ドメインで標本空間も分布も同じ

標本空間 $x_S \times y_S = x_T \times y_T$

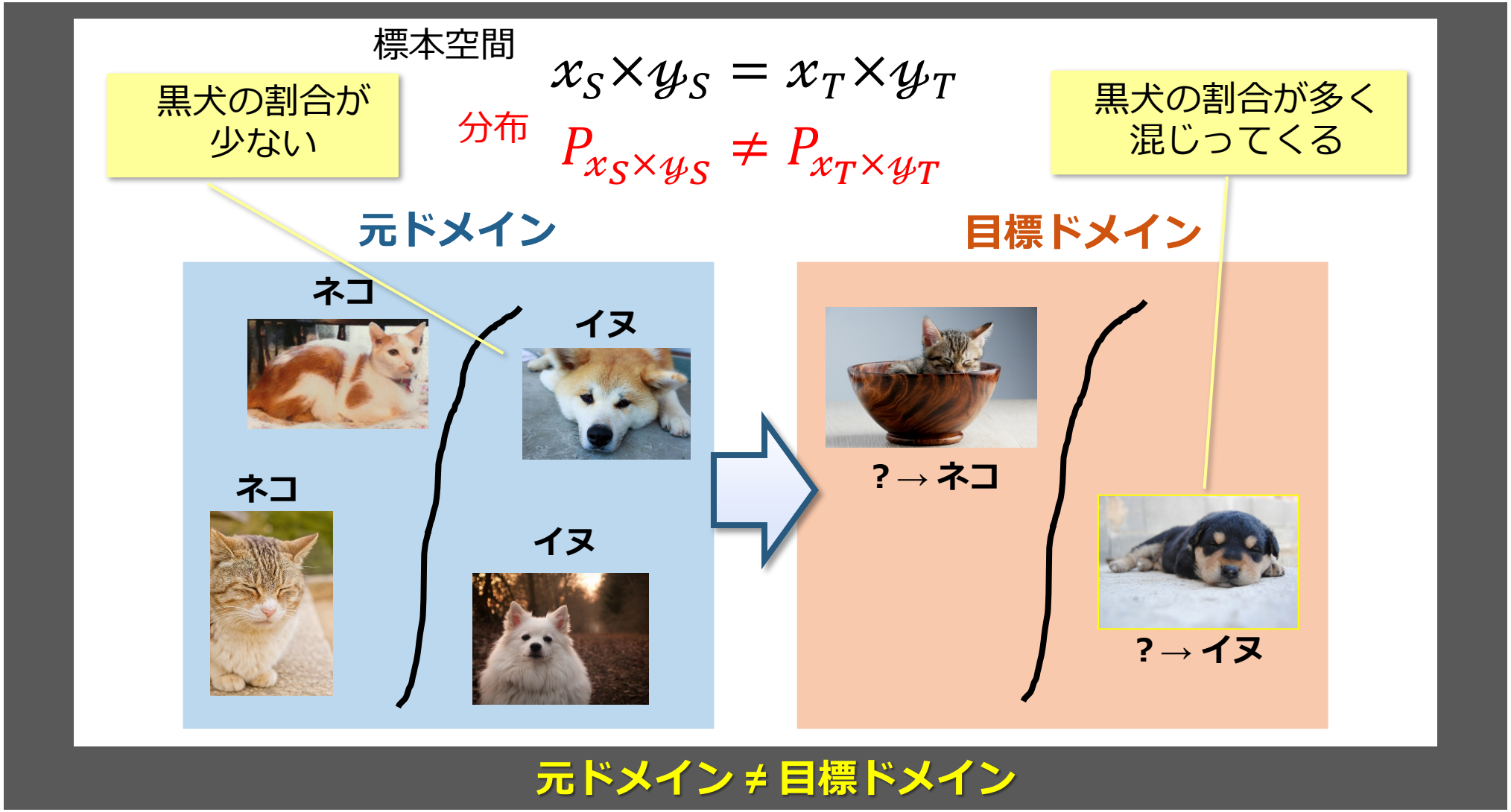
分布 $P_{x_S \times y_S} = P_{x_T \times y_T}$



元ドメイン＝目標ドメイン

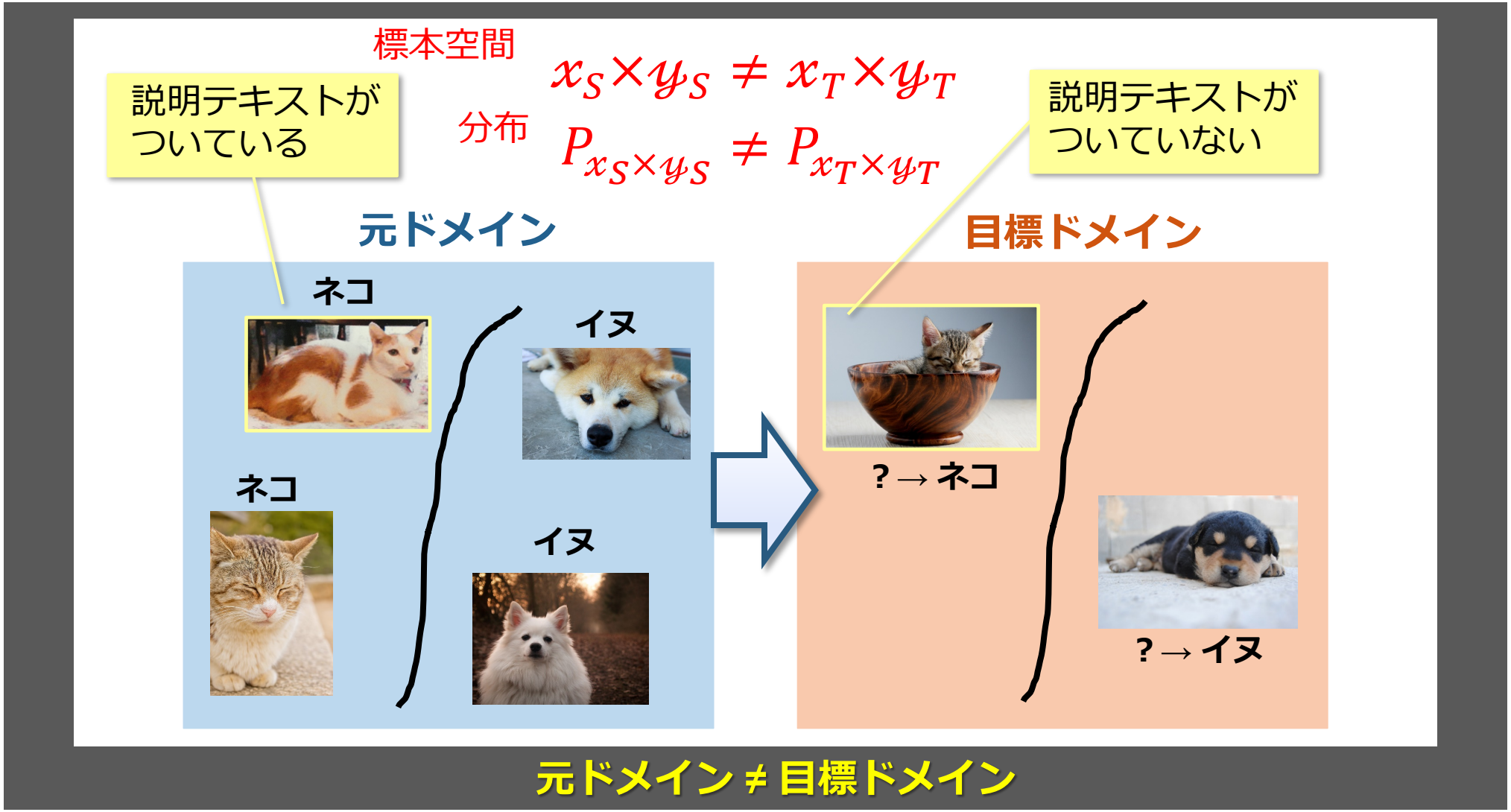
想定する状況（同種転移）

■ 標本空間は同じだが、分布は異なる



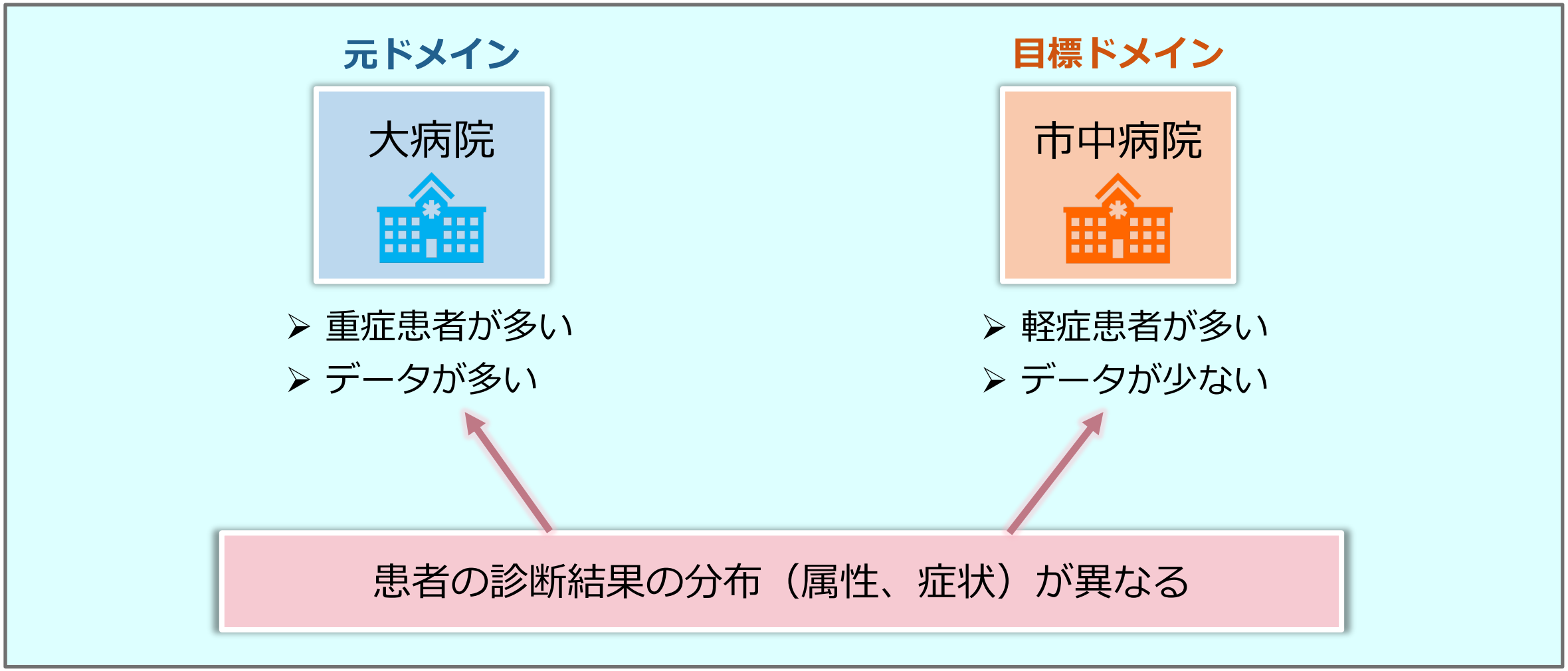
想定する状況（異種転移）

■ 標本空間も分布も異なる



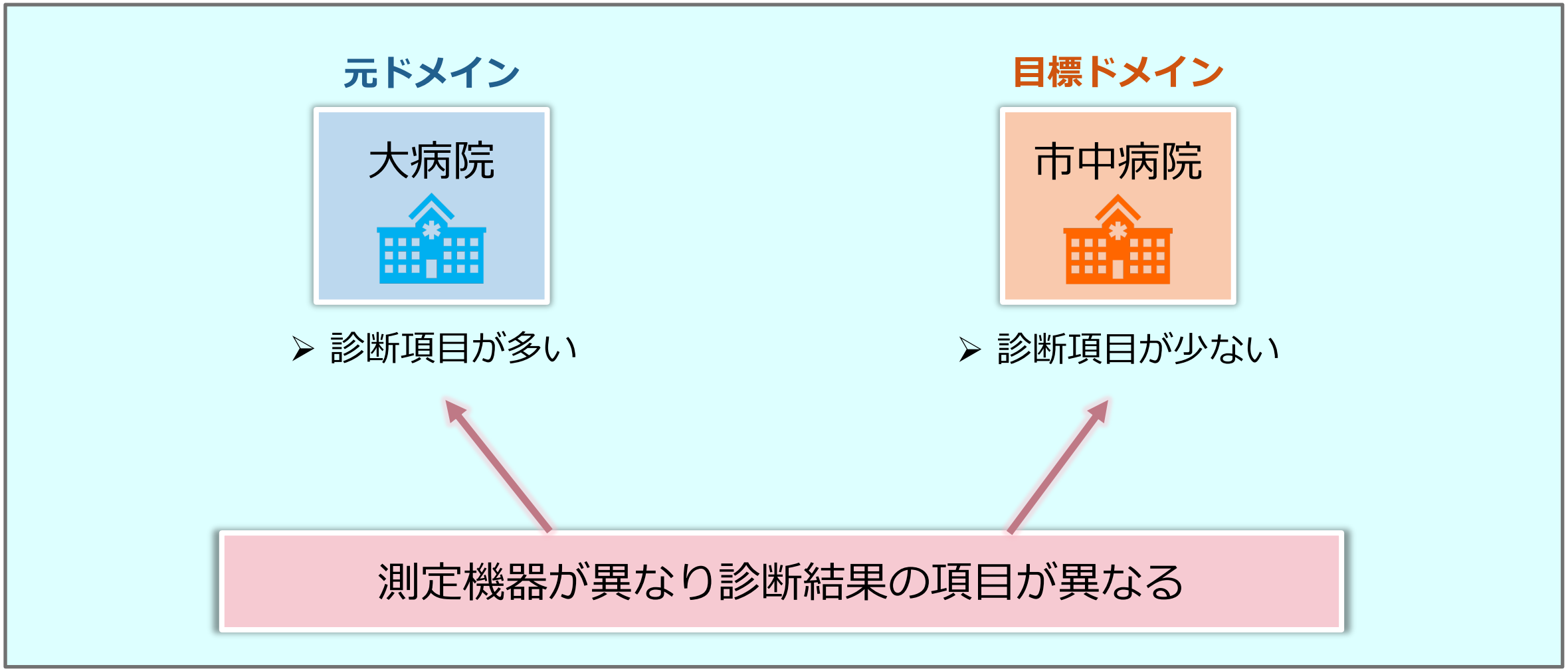
想定する状況（同種転移＝病院の場合）

＜タスク＞ ある疾患の予後（治療を受けた後の経過）を予測



想定する状況（異種転移＝病院の場合）

＜タスク＞ ある疾患の予後（治療を受けた後の経過）を予測



想定する状況（同種転移＝マーケティングの場合）

＜タスク＞ ある商品の購買を予測

元ドメイン

企業1

➢ 若い女性向けのサイトを運営

目標ドメイン

企業2

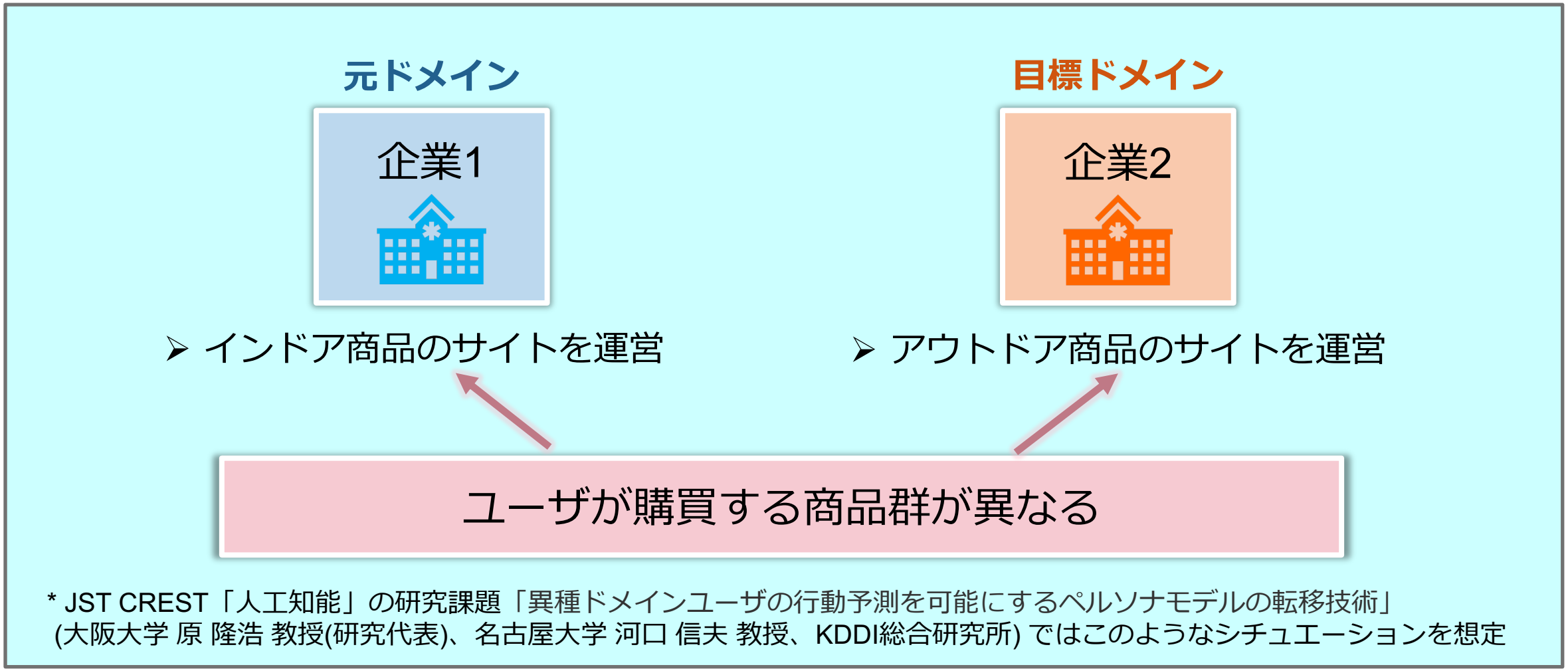
➢ シニア男性向けのサイトを運営

ユーザの属性が異なり、商品群は同じだが売れ行きが異なる

想定する状況（異種転移＝マーケティングの場合）

KDDI Research

＜タスク＞ ある商品の購買を予測



* JST CREST「人工知能」の研究課題「異種ドメインユーザの行動予測を可能にするペルソナモデルの転移技術」
(大阪大学 原 隆浩 教授(研究代表)、名古屋大学 河口 信夫 教授、KDDI総合研究所) ではこのようなシチュエーションを想定

転移学習の基本問題

When to transfer : どんな場合に転移するか

▶ 2つのドメインが似ていると転移がうまく行く

▶ 似ていないと転移がうまく行かない（負の転移）

What to transfer: 何を転移するか

- データそのものを転移
- モデル／パラメータを転移
- 共通特徴を抽出

How to transfer: いかにして転移するか

- 事例転移
- パラメータ転移
- 特徴転移

次で解説

転移学習の手法

構成

■同種転移

- ①事例転移（KDDI総合研究所 黒川）
- パラメータ転移（大阪大学 Li）
  - ②事前学習モデルの利用
  - ③メタ学習
- ④特徴転移（KDDI総合研究所 米川）

■異種転移（KDDI総合研究所 米川）

- 特徴転移の異種拡張

①事例転移：重要度重み付き学習～定式化

KDDI Research

■ 転移の仮定（When）：共変量シフト

$P_S(X) \neq P_T(X)$：　入力データの分布が変わる
$P_S(Y|X) = P_T(Y|X)$：入出力関係は変わらない

■ 転移の対象（What）：元ドメインのデータ集合をそのままを使う

元ドメイン

$(\mathcal{X}_S \times \mathcal{Y}_S, P_{\mathcal{X}_S \times \mathcal{Y}_S})$

$\mathcal{D}_n^S$　データ集合

目標ドメイン

$(\mathcal{X}_T \times \mathcal{Y}_T, P_{\mathcal{X}_T \times \mathcal{Y}_T})$

目的：期待リスク最小のモデルの学習

$$h_T^* = \mathrm{argmin}\mathbb{E}_{P_{x_T \times y_T}} l(h(X), Y)$$

元ドメインの「**データ**」を転移して目標ドメインのモデルを学習
目標ドメインの「**分布**」に従って期待リスクを評価することが必要

①事例転移：重要度重み付き学習～手法

■ 重点サンプリング

- **ある確率分布 $P'(X)$ での期待値 $\mathbb{E}_{P'(x)}A\,(X)$ を別のお手軽な確率分布 $P(X)$ での期待値に置き換える方法**

$$\mathbb{E}_{P'(x)}A(X) = \sum_x A(x)p(x) = \sum_x A(x)\frac{p'(x)}{p(x)}p(x)$$
$$= \mathbb{E}_{P(x)}\left[A(X)\frac{p'(x)}{p(x)}\right]$$

密度比

$A\,(X)$ … 期待値を取りたい関数
$p'(x), p(x)$ … 確率分布 $P'(X), P(X)$ に対応する確率密度

- **結論：密度比で重み付けすると別の確率分布で期待値を評価できる**

■ 上記を事例転移の文脈で読み替え

確率分布 $P'(X), P(X)$ → 目標ドメイン、元ドメインの確率分布 $P_{x_T \times y_T}$, $P_{x_S \times y_S}$

期待値を取りたい関数$A\,(X)$ → リスク関数 $l(h(X), Y)$ ※任意の口ス関数

密度比 $p'(x)/p(x)$ → 目標ドメイン/元ドメインの密度比 $p_T(x)/p_S(x)$

①事例転移：重要度重み付き学習～手法

■ 密度比の推定

● 代表的な手法：KLIEP [Sugiyama+ 08]

- 密度比を所定の基底関数の線形モデルと仮定

$$\hat{r}(x) = \sum_l \alpha_l \varphi_l(x)$$

- 目標ドメインの推定密度は以下

$$\hat{p}_T(x) = \hat{r}(x) p_S(x)$$

- 基底関数の重みαを真の密度とのカルバックライブラーダイバージェンスを最小化するように推定

$$KL[p_T(x)||\hat{p}_T] = \sum_x p_T(x) \log \frac{p_T(x)}{p_S(x)} - \sum_x p_T(x) \log \hat{r}(x)$$

定数　　標本平均で近似して最適化

● その他の手法

- uLSIF [Kanamori+ 09]：制約なしの最小二乗法で直接、密度比を推定
- RuLSIF [Yamada+ 13]：推定を安定させるために相対密度比を採用し、uLSIFと同様の方法で推定

※密度比推定は分布が変化するケース一般に使える方法（外れ値や異常値の検出にも使える）

①事例転移：重要度重み付き学習～コード

KDDI Research

■ ADAPT (Awesome Domain Adaptation Python Toolbox) ライブラリ*を利用

- pipでインストールできます。
- numpy scipy tensorflow(>=2.0) scikit-learn cvxoptに依存しています。

■ 準備

- **ライブラリインポート**

```
from adapt.instance_based import KLIEP
import tensorflow as tf

from tensorflow.keras import Sequential
from tensorflow.keras.layers import Input, Dense, Reshape
from tensorflow.keras.optimizers import Adam

from sklearn.datasets import make_moons
from sklearn.metrics import accuracy_score
```

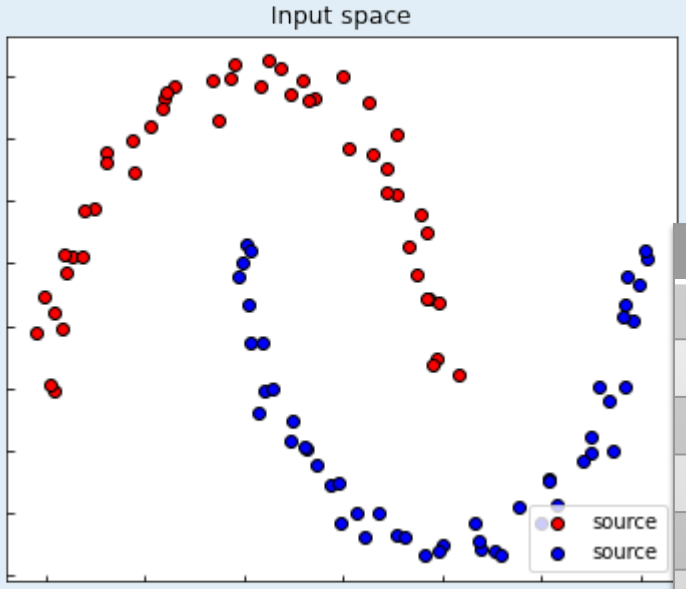
* https://github.com/adapt-python/adapt

①事例転移：重要度重み付き学習～実例

■ 実験データ

- Two Moonsデータセットを利用

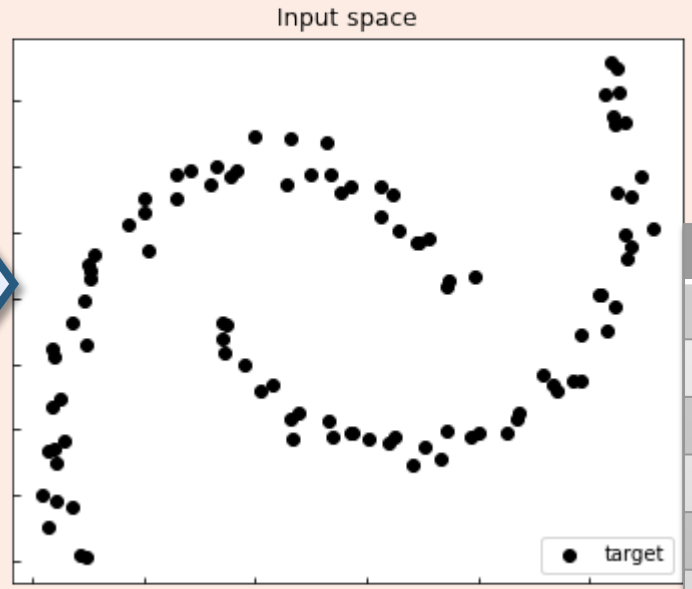
元ドメイン



X1	X2	Y
-0.574753	0.993981	0
1.246100	-0.207939	1
0.473573	0.093938	0
-0.466737	0.147854	1
0.498300	-0.506125	1
…		

- 2次元のデータ
- 上弦部分：ラベル0（赤）
- 下弦部分：ラベル1（青）

目標ドメイン



X1	X2
-0.994741	0.573436
1.183124	0.442970
0.363157	0.318140
-0.478133	-0.105323
0.684604	-0.189167
…	

- 元データを原点中心に30度回転
- 教師（ラベル）無し
  （入出力関係は同じと仮定）

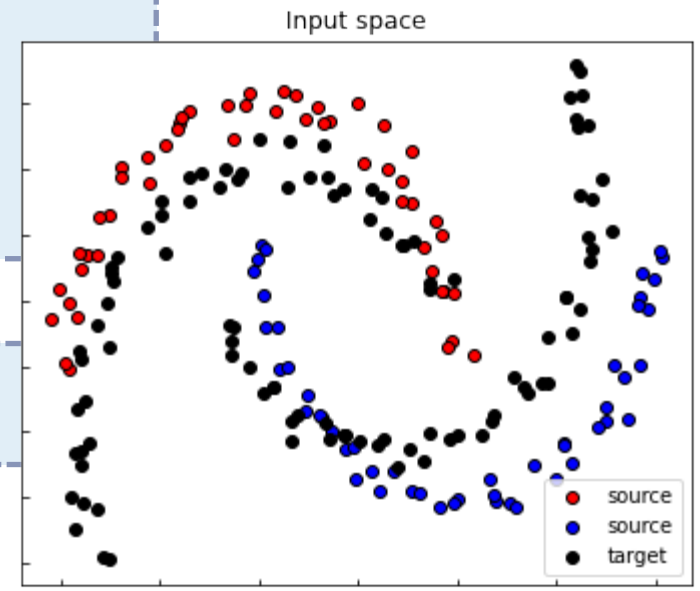
* Two Moons: https://adapt-python.github.io/adapt/examples/Two_moons.html

参考）データ生成の実行コード

```
def make_moons_da(n_samples=100, rotation=30, noise=0.05, random_state=0):
    Xs, ys = make_moons(n_samples=n_samples,
                        noise=noise,
                        random_state=random_state)
    Xs[:, 0] -= 0.5
    theta = np.radians(-rotation)
    cos_theta, sin_theta = np.cos(theta), np.sin(theta)
    rot_matrix = np.array(
        ((cos_theta, -sin_theta),
         (sin_theta, cos_theta))
    )
    Xt = Xs.dot(rot_matrix)
    yt = ys
    return Xs, ys, Xt, yt
```

```
Xs, ys, Xt, yt = make_moons_da()
```

得られるデータ



* Two Moons: https://adapt-python.github.io/adapt/examples/Two_moons.html

参考）モデル作成の実行コード

■ **任意のモデルを作成**

```
def make_model():
    model = Sequential()
    model.add(Dense(10, activation="relu", input_shape=(2,)))
    model.add(Dense(10, activation="relu"))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss="bce", optimizer=Adam(0.001))
    return model
```

■ **モデルを引数として転移モデル（KLIEP）を作成**

基底関数（ここではrbfカーネル）**のパラメータを3種設定**

```
kliep = KLIEP(make_model(), sigmas=[0.1, 1, 10], random_state=0)
```

■ **モデルを学習（fit）**

密度比推定アルゴリズムを変えたい場合はKLIEPクラスを改変

```
fit_params = dict(epochs=800, batch_size=34, verbose=0)
kliep.fit(Xs, ys, Xt, **fit_params)
```

目標ドメインのデータを密度比推定用に渡す

元ドメインのデータを学習データとして渡す

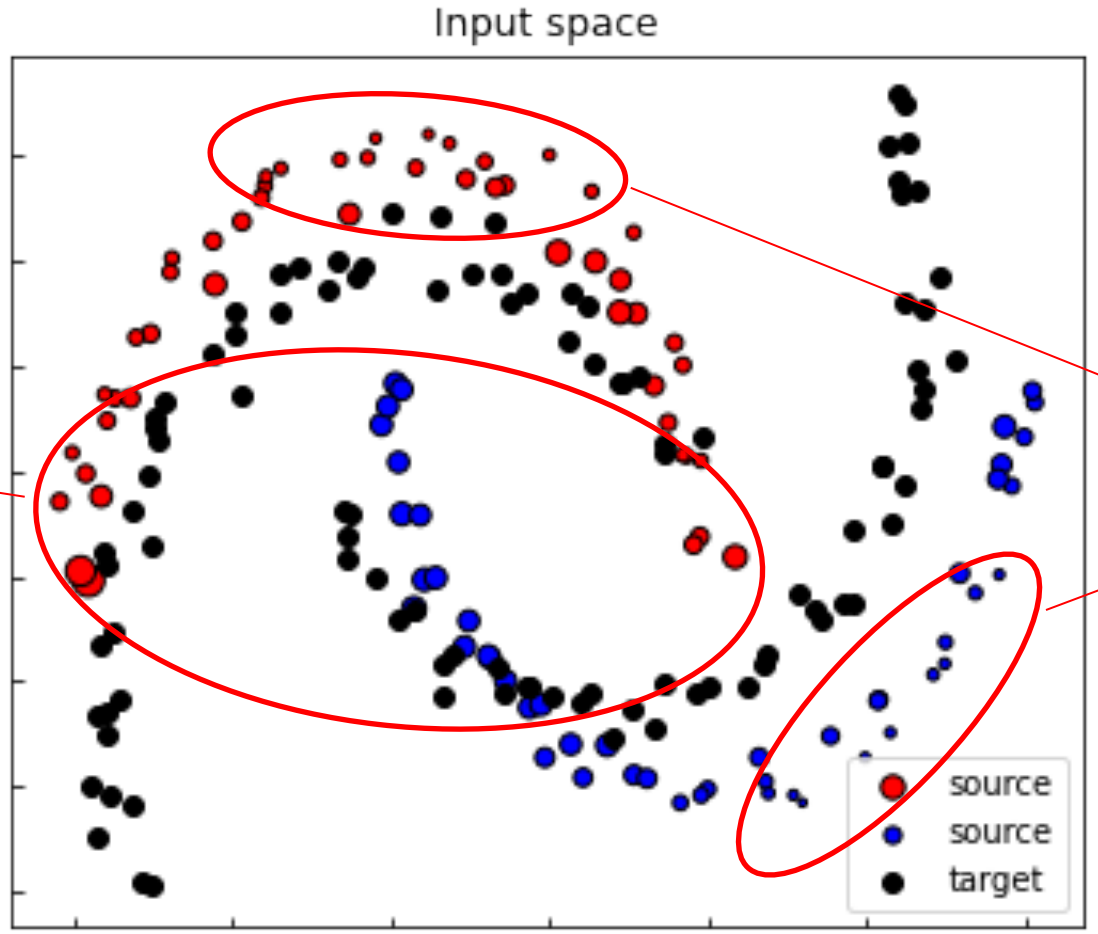
■ **モデルを評価**

```
yt_pred = kliep.predict(Xt)
acc = accuracy_score(yt, yt_pred>0.5)
```

①事例転移：重要度重み付き学習～結果

■ 密度比推定の結果

- 密度比が大きい点を大きく、小さい点を小さく表示



目標ドメインとずれが
小さい真ん中から左側
の重みを強く評価

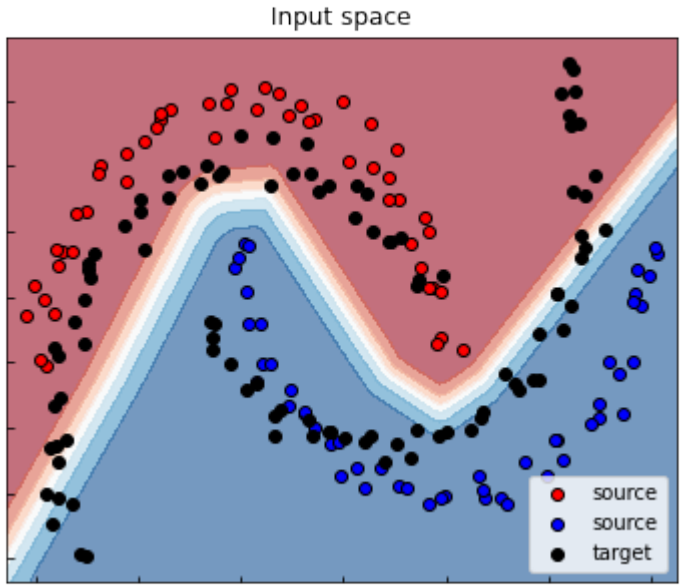
目標ドメインとずれが
大きい上下の重みを
小さく評価

※値の大小を強調するため密度比はMAX-MINスケーリングして表示

①事例転移：重要度重み付き学習～結果

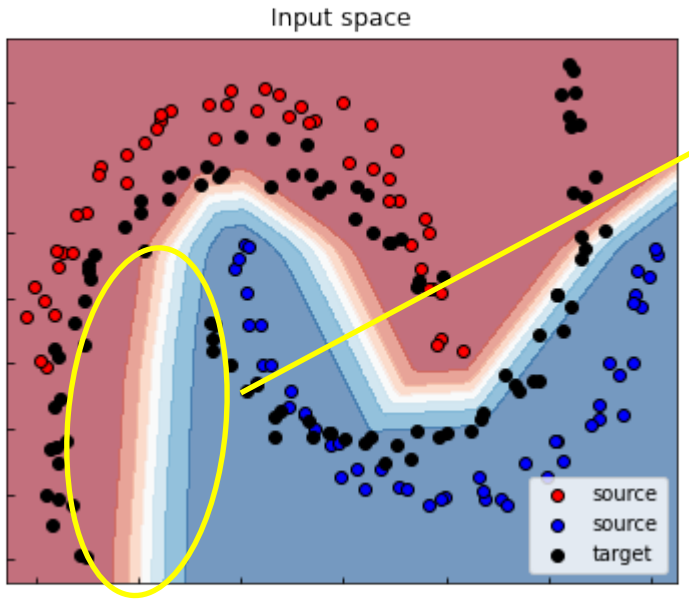
■ 密度比で重み付けした結果の決定境界

重み付けなし
（元ドメインのデータを一様重みで学習）



acc: 0.82

重み付けあり
（目標ドメインのデータを参照し重み付きで学習）



左下部分の
決定境界が改善

acc: 0.87

①事例転移：その他の手法

■ アンサンブル手法の拡張

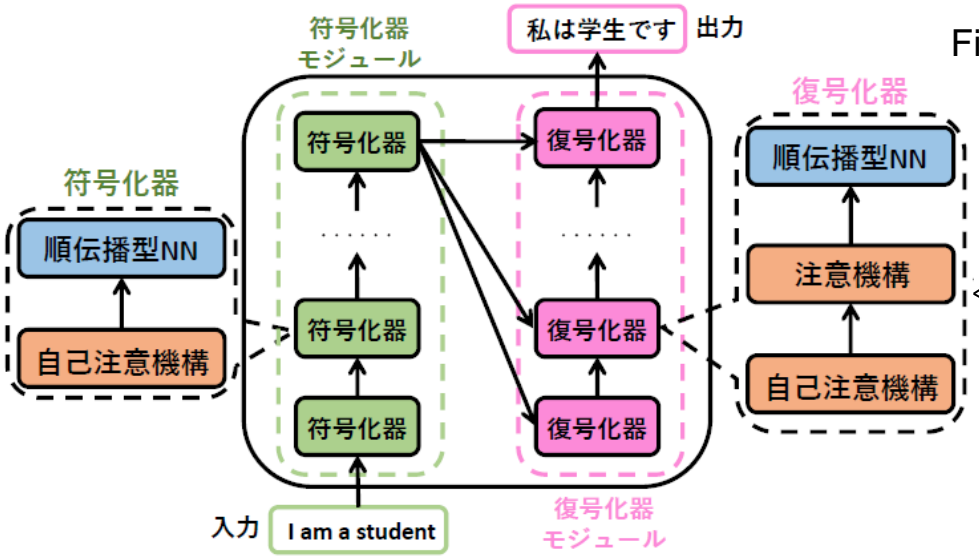
- **TrAdaBoost** [Dai+ 07]：モデルのアンサンブル手法のひとつであるAdaBoostを拡張。両ドメインのデータを用いて各モデルを学習し、各モデルの誤差評価を目標ドメインのデータに基づいて行いデータの重みを変化

■ 事例選択の手法

- **HEGS** [Shi+ 10]：両ドメインのデータを特徴空間上でクラスタリングし、各ドメイン由来のデータができるだけ均等なクラスタを選択

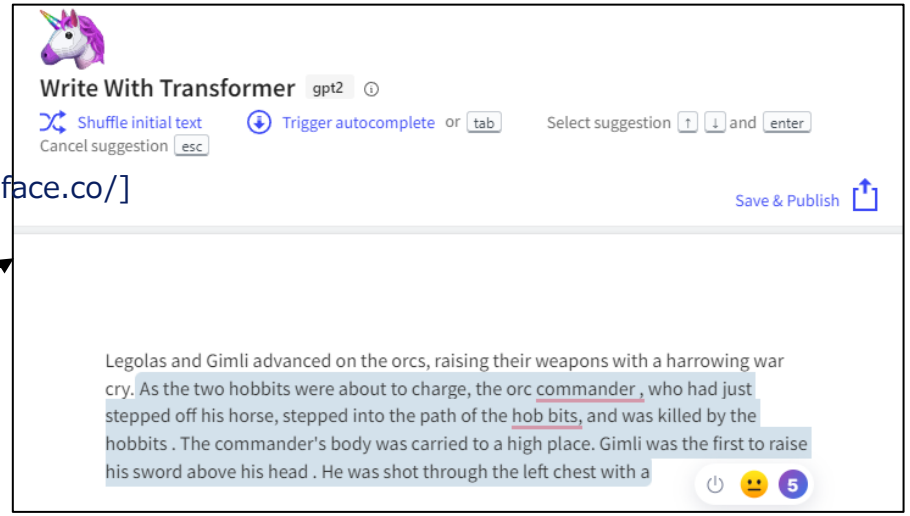
②事前学習モデルの利用

■ 事前学習とファインチューニング
- 例：　トランスフォーマー



Fine-Tuning

GPT-2による文章生成
[https://transformer.huggingface.co/]



Fine-Tuning

DALL-Eによる文章からの画像生成
[https://openai.com/]

- 巨大な事前学習済みモデルをファインチューンして利用
- スクラッチ学習に膨大なコストがかかるモデルでも相対的に低コストで使える

②事前学習モデルの利用

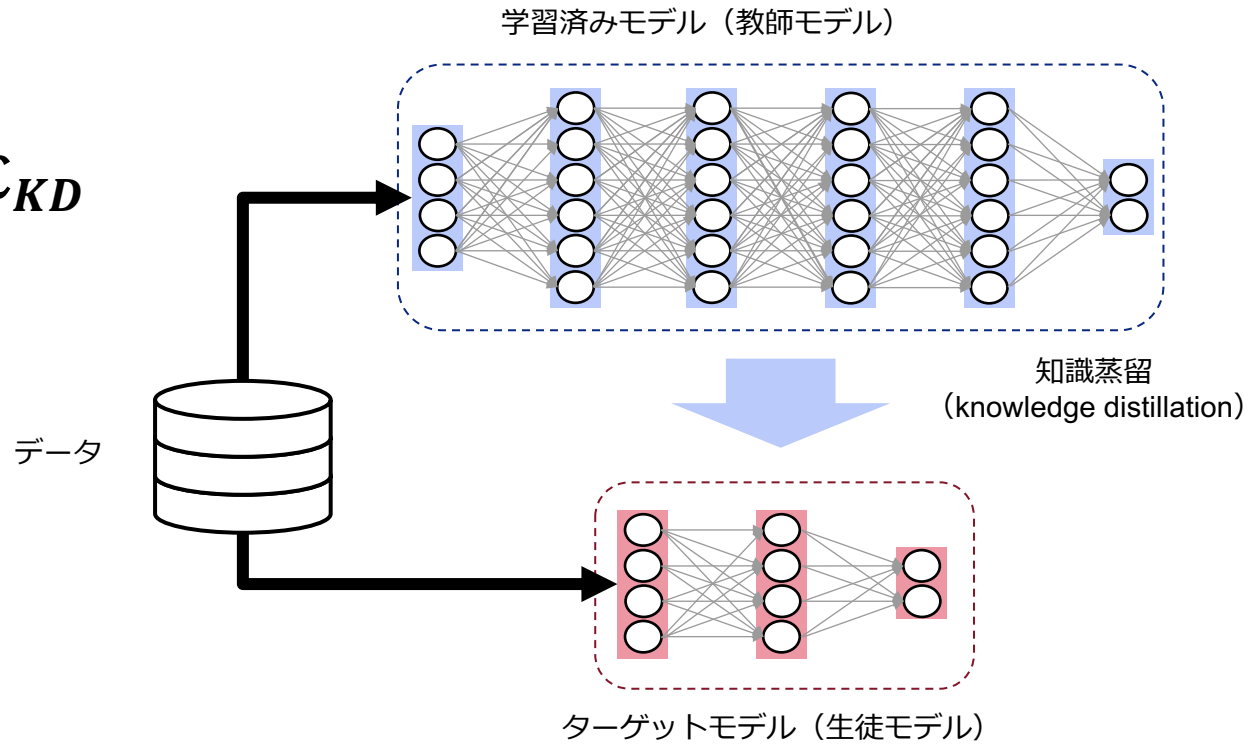
■ **知識蒸留:** 学習済みの巨大モデルから知識を軽量モデルに転移する手法
- 転移する知識：学習済みモデルのパラメータ

■ ターゲットモデルの学習
- ロス関数：

$$\mathcal{L} = \mathcal{L}_{task} + \boldsymbol{\alpha} * \boldsymbol{\mathcal{L}_{KD}}$$

  - $\mathcal{L}_{task}$： 目標タスクのロス
  - α：ファクター
  - $\mathcal{L}_{KD}$：蒸留ロス

■ メリット
- 計算コストを減らす
- 過学習を抑える



②事前学習モデルの利用～手法

■ 応答ベース [Hinton+ 15]

- 教師モデルの予測結果を直接模倣するように生徒モデルを学習
- 利用条件：
  - 教師あり学習のみ
  - 生徒と教師の学習タスクが一致
- 既存研究：ソフトターゲット[Hinton+ 15] 、Deep Mutual Learning [Zhang+ 17].

■ 特徴ベース

- 生徒モデルの中間層を教師モデルにマッピング
- 既存研究：FitNet [Romero+ 14].

■ 関係ベース

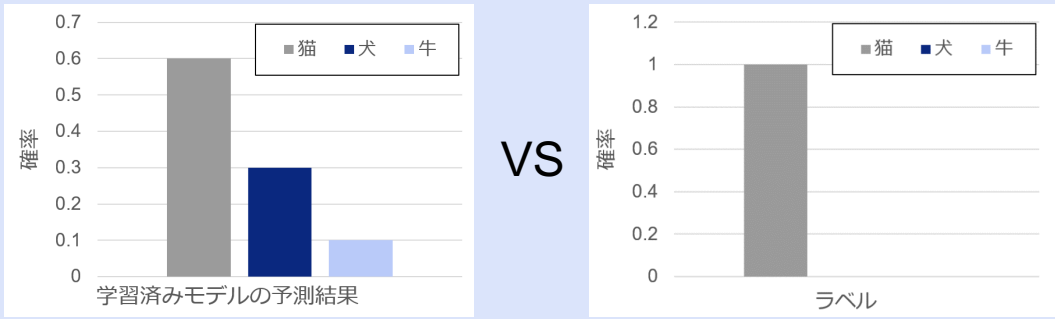
- 教師が学習された異なる層の間の関係や、異なる事例の間の関係を知識として転移
- 既存研究：[Yim+ 17].

②事前学習モデルの利用～手法

■ ソフトターゲット [Hinton+ 15]

- 仮説：学習済みモデルの予測結果にラベルがない情報を持っている。

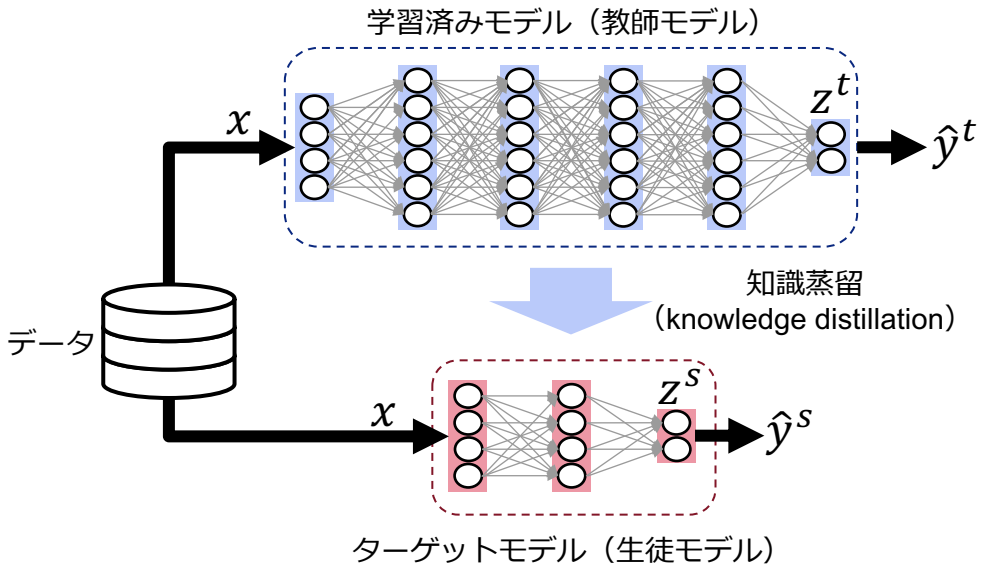
  - 例えば、
    - 猫は牛よりも犬に似ている。
    - 足し算は微分より引き算に似ている。



- ソフトターゲット

$$\hat{y}_i(\tau) = softmax(z_i, \tau) = \frac{\exp(z_i/\tau)}{\sum_{j=1}^{C}\exp(z_j/\tau)}$$

- z_i：クラス i に対する中間層の出力
- $\hat{y}_i$：クラス i に対するモデルの出力
- τ：温度パラメータ

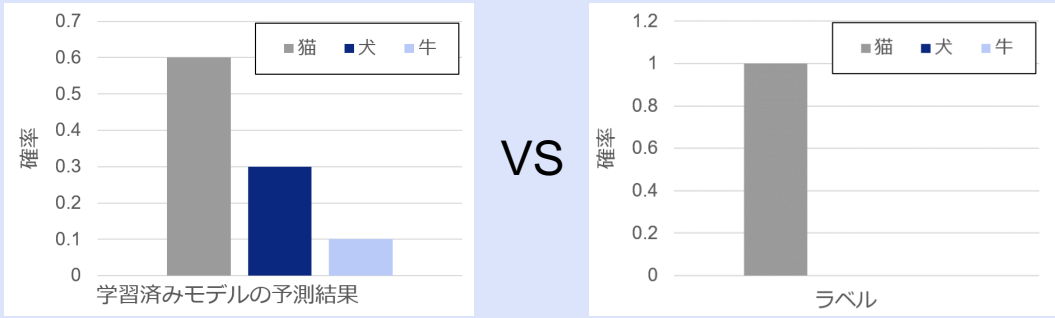


②事前学習モデルの利用～手法

■ ソフトターゲット [Hinton+ 15]

- 仮説：学習済みモデルの予測結果は、ラベルにはない情報を持っている。

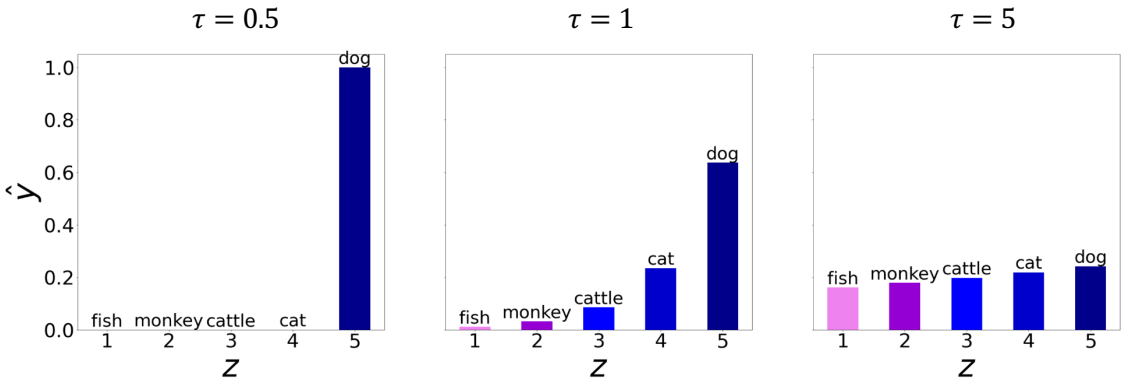
  - 例えば、
    - 猫は牛よりも犬に似ている。
    - 足し算は微分より引き算に似ている。



- ソフトターゲット

$$\hat{y}_i(\tau) = softmax(z_i, \tau) = \frac{\exp(z_i/\tau)}{\sum_{j=1}^{C} \exp(z_j/\tau)}$$

- z_i：クラス i に対する中間層の出力
- $\hat{y}_i$：クラス i に対するモデルの出力
- τ：温度パラメータ



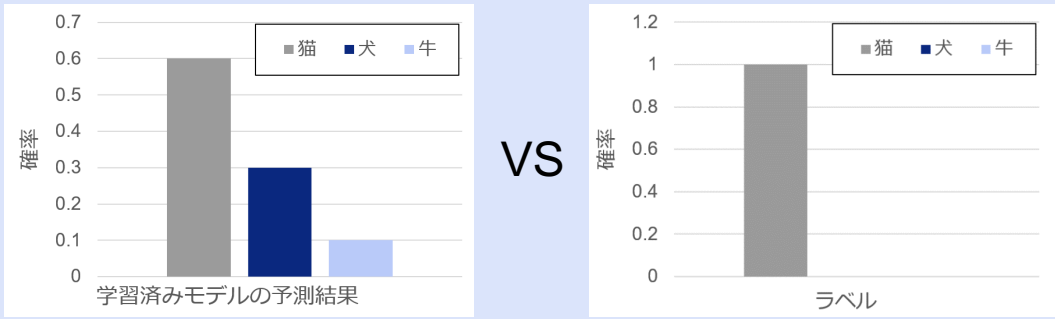
温度パラメータを上昇すると $\hat{y}_i$ の分布が柔らかくなる

②事前学習モデルの利用～手法

■ ソフトターゲット [Hinton+ 15]

- 仮説：学習済みモデルの予測結果は、ラベルにはない情報を持っている。

  - 例えば、
    - 猫は牛よりも犬に似ている。
    - 足し算は微分より引き算に似ている。



- 生徒モデルのロス関数：

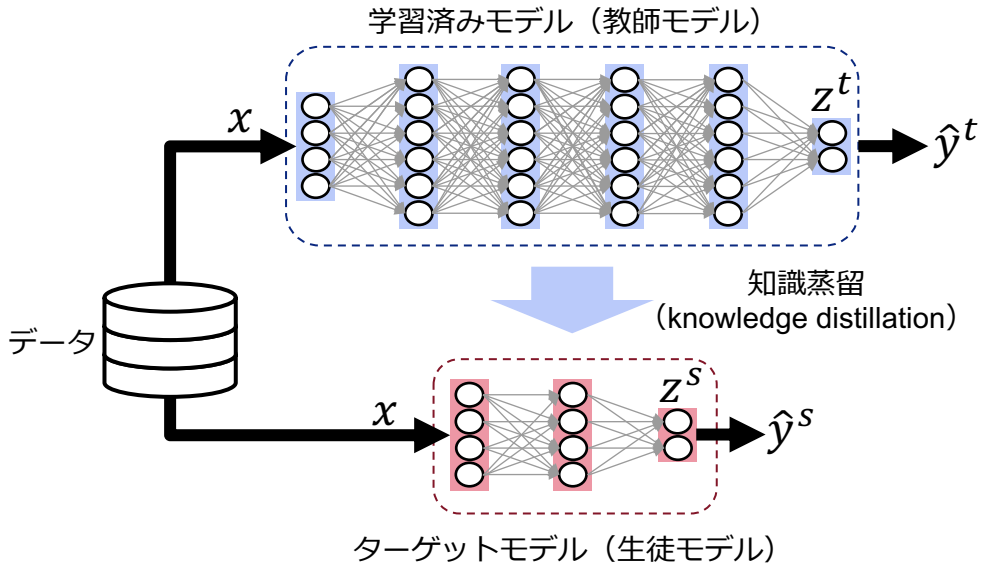
$$\mathcal{L} = \alpha * \sum_{i \in C} -\hat{y_i}^t(\tau) \log \hat{y_i}^s(\tau) + (1-\alpha) * \sum_{i \in C} -y \log \hat{y_i}^s(1)$$

蒸留損失

教師モデルと生徒モデルのソフトターゲット間の損失

予測損失

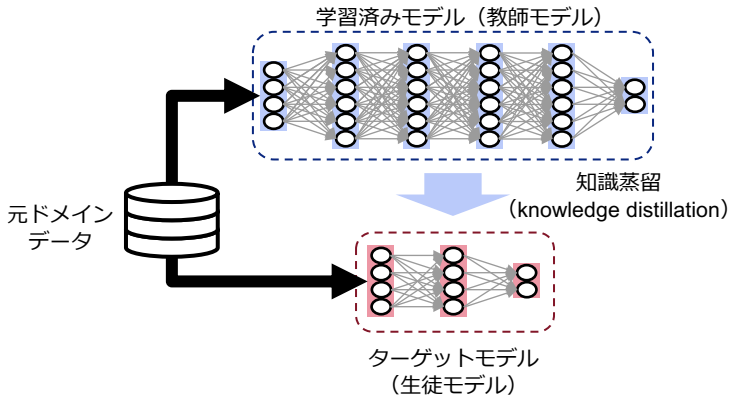
生徒モデルがラベルへの予測損失



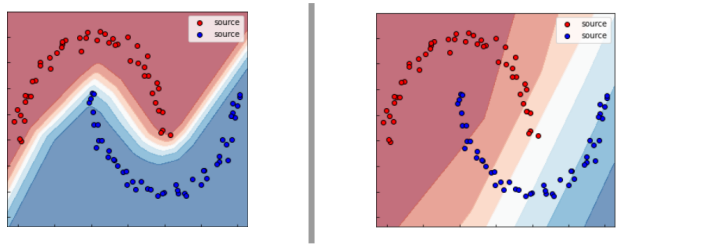
②事前学習モデルの利用～結果

■ 例：**ソフトターゲット**をTwo Moonsの二値分類タスクへの応用

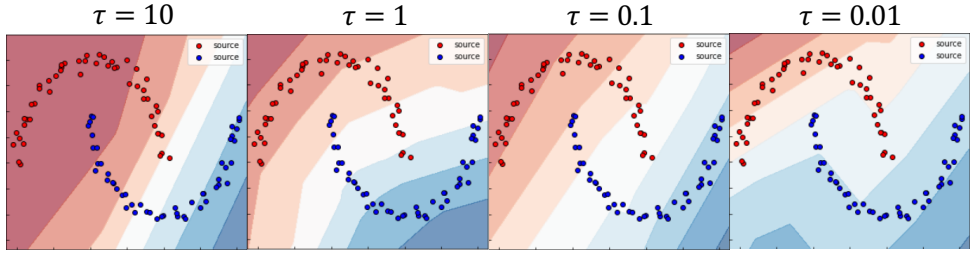
● 実験１：元ドメイン内の転移学習



モデルの
決定境界

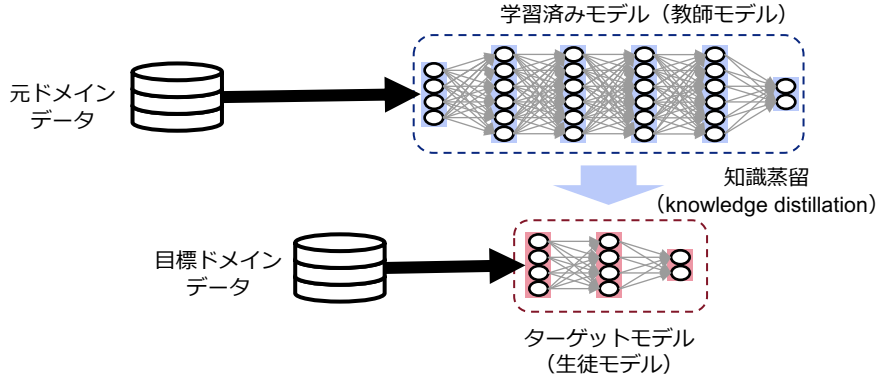


教師モデル　　生徒モデル（転移学習なし）

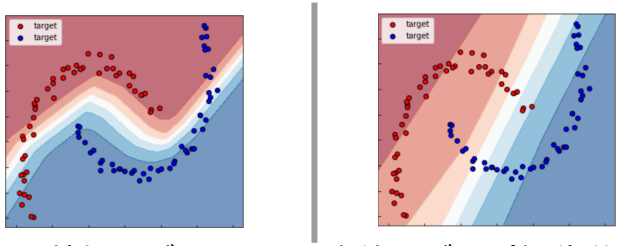


生徒モデル（転移学習あり）

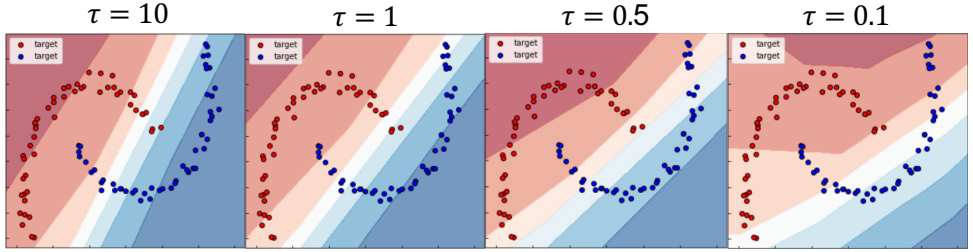
● 実験2：ドメイン間の転移学習



モデルの
決定境界



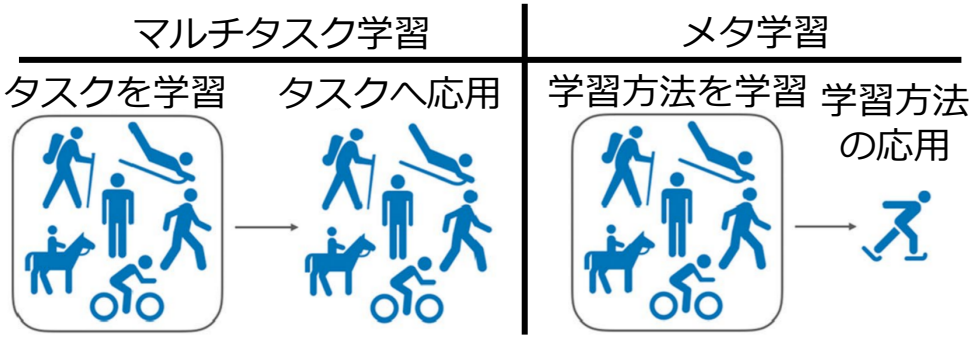
教師モデル　　生徒モデル（転移学習なし）



生徒モデル（転移学習あり）

③メタ学習

■ メタ学習
- タスクごとの学習データが少ないfew-shot learningという問題によく用いられる。
- 特定の目標ドメインに限定せず，モデルの汎化性能に着目



マルチタスク学習とメタ学習の差異 [https://meta-world.github.io/]

学習方法を事前知識として転移

- メタ学習の定式化

$$\omega \coloneqq \underset{\omega}{\operatorname{argmin}} \underbrace{\mathrm{E}_{T\sim p(T)}[\mathcal{L}_T}_{\text{outer-level}}(\underbrace{g_\omega(T,\mathcal{L}_T)}_{\text{inner-level}})]$$

学習方法を事前知識として学ぶ　事前知識を用いてタスクを学習

  - ω：学習器（モデル）のパラメータ
  - T：学習タスク T ， $\mathcal{L}_T$：タスク T への損失
  - g：最適化器（例えば，勾配降下法）

③メタ学習～手法

■ メトリックベース

- データから事前知識を抽出
- 応用：
  - 教師あり学習の分類問題

■ モデルベース

- 専用モジュールで事前知識を学習
- 応用：
  - 教師あり学習、強化学習

■ 最適化ベース

- 事前知識を含むモデルのパラメータ初期値を学習
- 応用：
  - 教師あり学習、強化学習

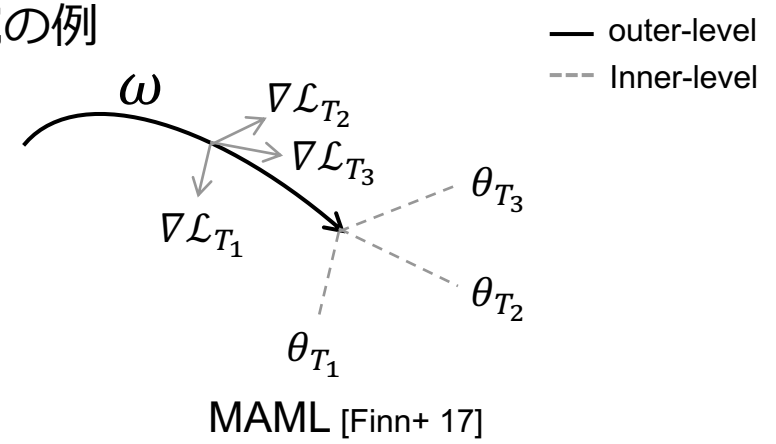
③メタ学習～手法

■ メトリックベース
- データから事前知識を抽出
- 応用：
  - 教師あり学習の分類問題

■ モデルベース
- 専用モジュールで事前知識を学習
- 応用：
  - 教師あり学習、強化学習

■ 最適化ベース
- 事前知識を含むモデルのパラメータ初期値を学習
- 応用：
  - 教師あり学習、強化学習

既存研究の例



MAML [Finn+ 17]

ω：モデルのパラメータ初期値**（タスク間共有する）**
θ_T：タスク T の学習済みモデルのパラメータ

MAMLの学習目標：

- タスクを学習（Inner-level）：

$$\theta_{T_i} = \omega - \alpha\nabla_{\omega}\mathcal{L}_{T_i}(f_{\omega})$$

- 事前知識を学習（Outer-level）

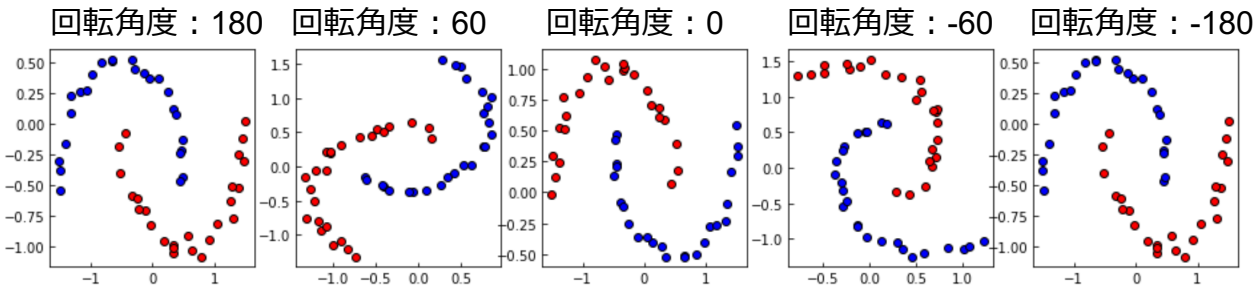
$$\omega := \underset{\omega}{\operatorname{argmin}}\, \mathrm{E}_{T_i\sim p(T)}[\mathcal{L}_{T_i}(f_{\theta_{T_i}})]$$

f_{ω}：学習器

③メタ学習～コーディング

■ 例：**MAML**をTwo Moonsの二値分類タスクへの応用

- タスク群 $p(T)$ ：回転角度が-180度から180度までのダブルムーンデータ.
  - 例：



MAMLの**学習**アルゴリズム：

入力：タスク群 $p(T)$ ，学習率 αとβ，パラメータの初期値ω

1. 収束まで繰り返し：
2. タスクバッチをサンプリング　$T \sim p(T)$
3. **for** タスク T_i **in** T:
4. T_i からデータをサンプリング　$D = \{x, y\}$
5. データ D を用いてタスク T_i への損失を計算 $\mathcal{L}_{T_i}(f_\omega, D)$
6. タスクを学習　$\theta_{\mathrm{T}_i} = \omega - \alpha \nabla_\omega \mathcal{L}_{\mathrm{T}_i}(f_\omega)$
7. T_i からデータをサンプリング　$D' = \{x, y\}$
8. **end for**
9. パラメータを更新　$\omega = \omega - \beta \nabla_\omega \sum_{T_i \in T} (\mathcal{L}_{T_i}(f_{\theta_{\mathrm{T}_i}}, D'))$

MAMLの**推論**アルゴリズム：

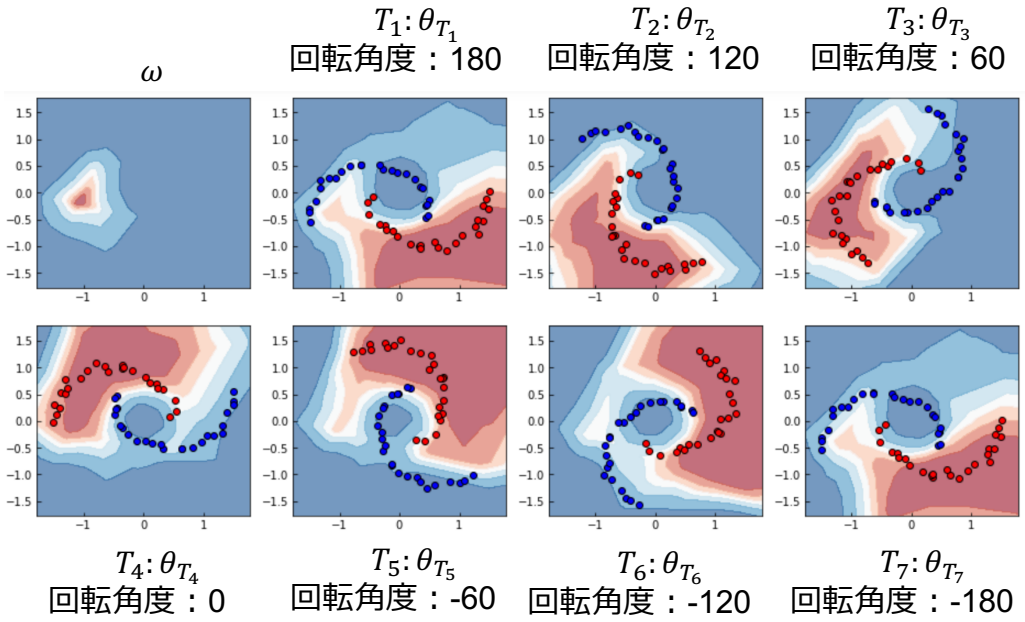
入力：タスク T_i ，学習率 α，パラメータ ω

1. 数回繰り返し：
2. T_i からデータをサンプリング　$D = \{x, y\}$
3. データ D を用いてタスク T_i への損失を計算 $\mathcal{L}_{T_i}(f_\omega, D)$
4. タスクを学習　$\theta_{\mathrm{T}_i} = \omega - \alpha \nabla_\omega \mathcal{L}_{\mathrm{T}_i}(f_\omega)$
5. モデル $f_{\theta_{\mathrm{T}_i}}$で推論結果を計算

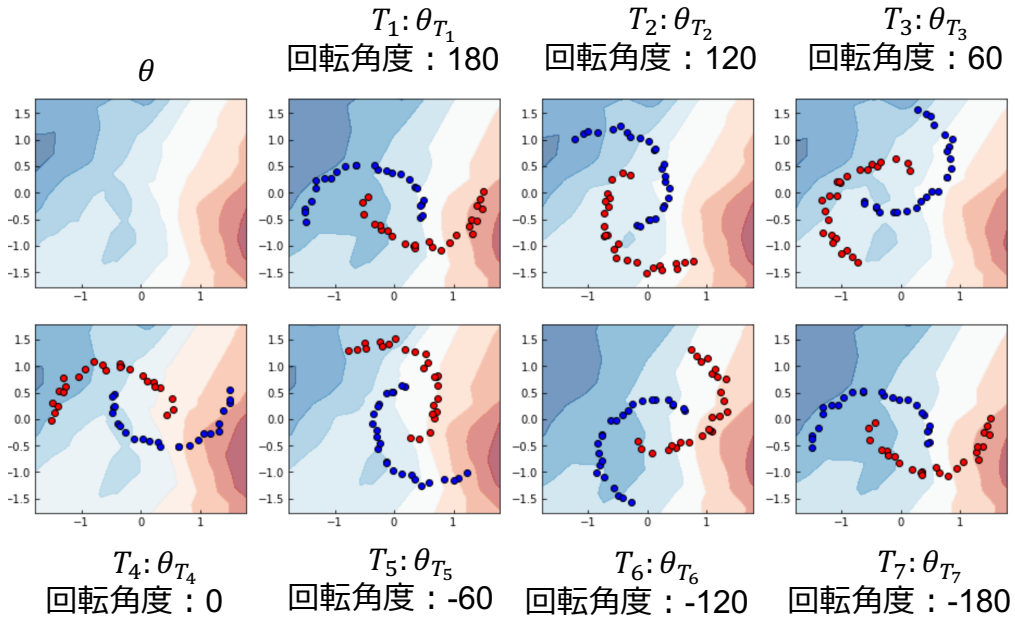
③メタ学習～結果

■ 例：**MAML**をTwo Moonsの二値分類タスクへの応用

- モデルの決定境界



ω：MAMLで学習されたパラメータ初期値。
θ_T： タスク T の学習済みモデルのパラメータ (ωから)



θ：ランダムの初期値。
θ_T： タスク T の学習済みモデルのパラメータ (θから)

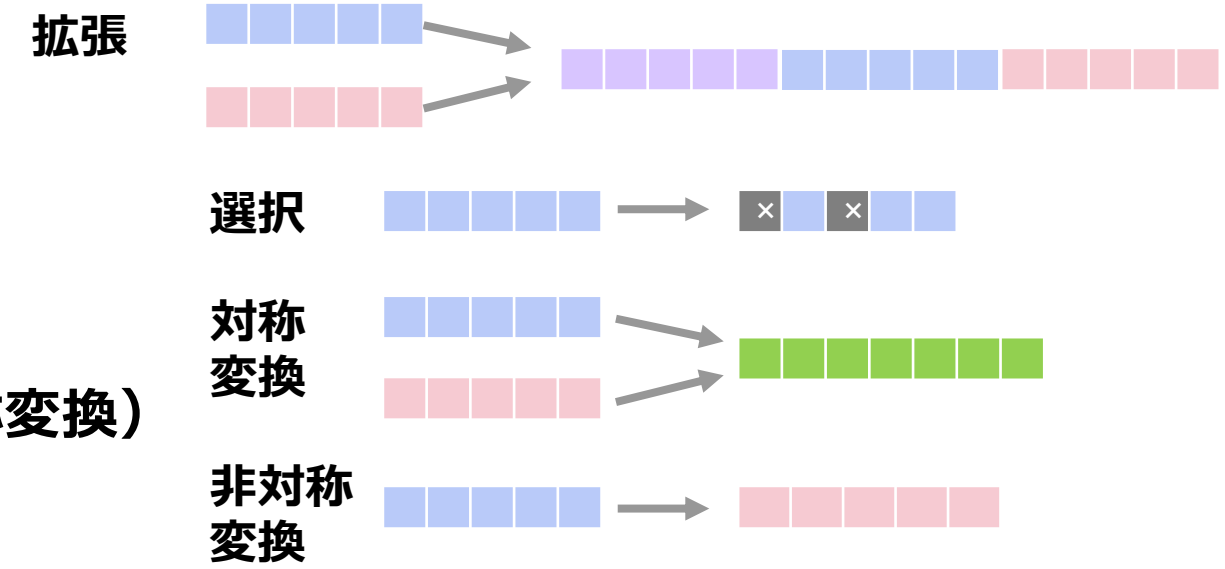
④特徴転移～概要

■ アプローチの特色

- 異なるドメインでも有効となるように特徴空間を変換する

■ 変換

- 特徴次元の拡張
- 特徴次元の選択
- 共通の特徴空間への変換（対称変換）
- 一方の特徴空間から他方への変換（非対称変換）



■ 分類

- 目標ドメインのラベルを用いる（教師ありドメイン適応）
  - $P_S(X) \neq P_T(X),\ P_S(Y|X) \neq P_T(Y|X)$ に対処
- 目標ドメインのラベルを用いない（教師なしドメイン適応）
  - $P_S(X) \neq P_T(X)$ に対処

④特徴転移～手法

■ 教師ありドメイン適応

- **FAM (feature augmentation method)** [Daume III 07]
  - 特徴次元を3倍に拡張し、共通、自ドメイン、他ドメイン(ゼロ埋め) の役割を持たせる
  - ドメイン共通の重みとドメイン固有の重みを学習することで特徴次元をソフトに選択

■ 教師なしドメイン適応

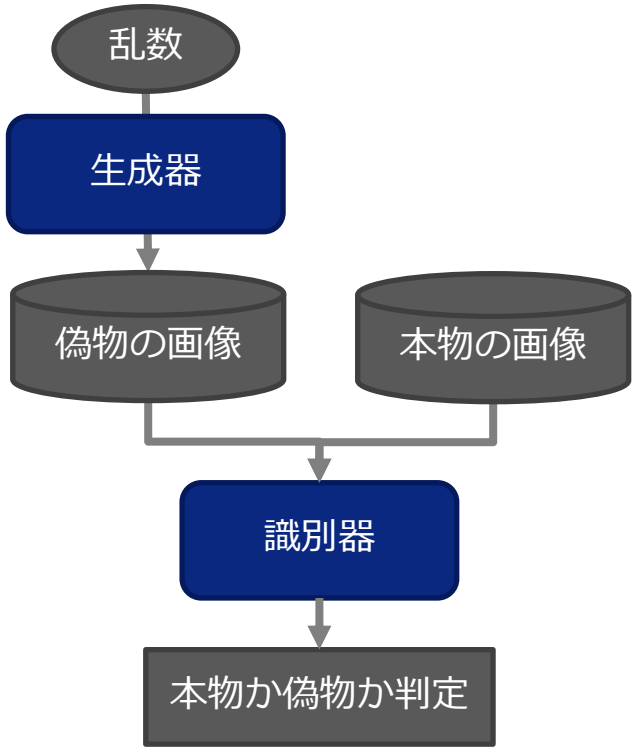
- **CORAL (correlation alignment)** [Sun+ 16]
  - 共分散(二次統計量)が整合するような非対称変換を学習
  - 関連：DNN拡張がDeep CORAL

- **DANN (domain-adversarial neural network)** [Ganin+ 16]
  - 特徴表現のドメイン由来が識別不能となるような共有エンコーダを対称変換として学習
  - 関連：逐次方式としたのがADDA (adversarial discriminative domain adaptation)、ドメイン固有のエンコーダを追加したのがDSN (domain separation networks)
  - ➡以降で詳説

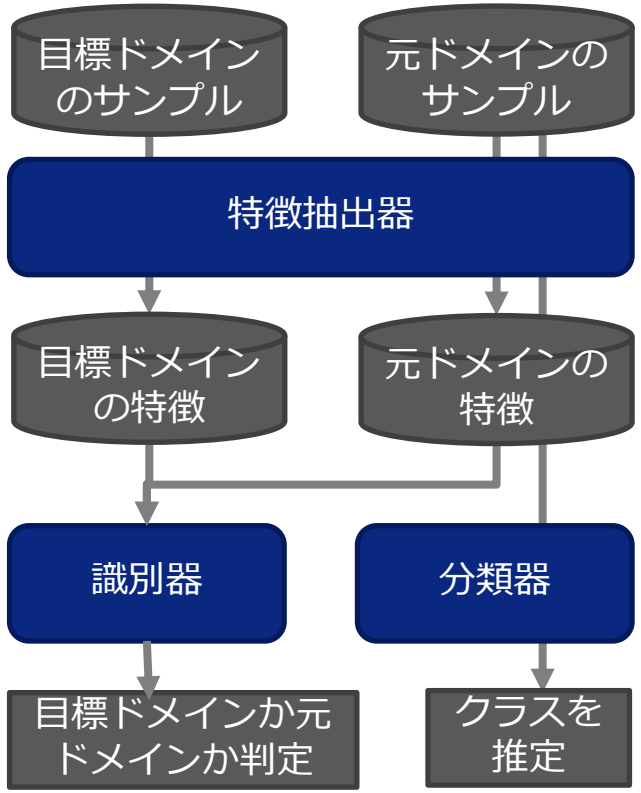
④DANN (domain-adversarial neural network) ～概要

- Generative Adversarial Networks (GAN) の要領
- 目標ドメインのサンプルの特徴表現が元ドメインそっくりになるよう学習
- 元ドメインの教師データで学習した分類器が目標ドメインでも有効に機能

GAN



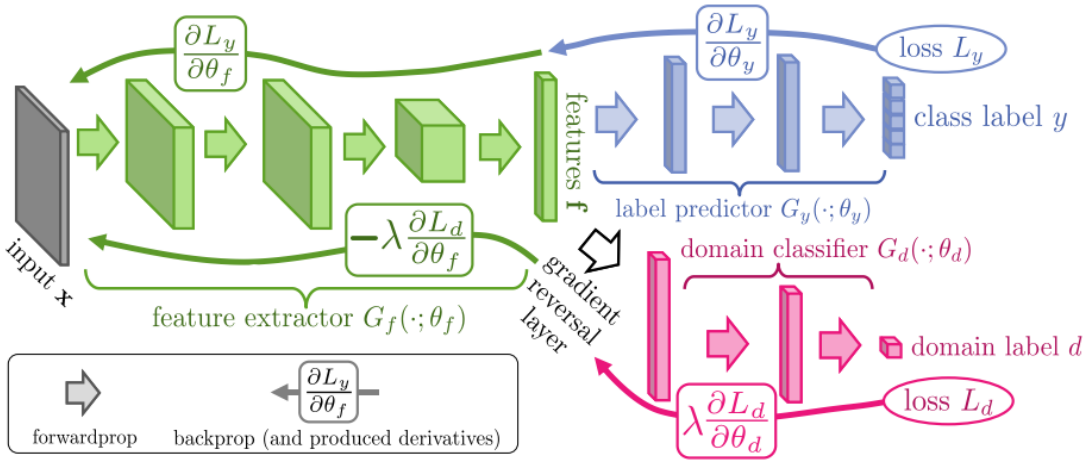
DANN



④DANN～詳細

■ $\min_{G_y,G_f,G_d} \sum_{i=1}^{n} L_y\left(G_y\left(G_f(\boldsymbol{x}_i)\right), y_i\right) + \lambda \sum_{i=1}^{N} L_d\left(G_d\left(R\left(G_f(\boldsymbol{x}_i)\right)\right), d_i\right)$

- $\boldsymbol{x}_i, y_i, d_i$: 入力データ, クラスラベル, ドメインラベル
  - $i \sim n$: ソースドメイン
  - $i+1 \sim N$: ターゲットドメイン
- G_f: 特徴抽出器
- G_y: 分類器
- L_y, L_d: 分類損失, 識別損失。いずれも交差エントロピー
- R: **勾配反転層**, $R(\boldsymbol{x}) = \boldsymbol{x}$, $\frac{dR}{dx} = -\boldsymbol{I}$
- G_d: 識別器
- λ: バランシングパラメータ
- ➡SGD (確率的勾配降下法) 等で最適化



■ 最適化における特徴抽出器の挙動

- 分類損失は小さく、識別損失は大きくしようとする
- 抽出される特徴表現を、入力データの分類はうまくできるが、ドメインの見分けはつかないものにする
- ➡ドメイン間の「ズレ」を「気にしない」モデルが得られる

④DANN～コーディング

https://github.com/antoinedemathelin/adapt/blob/master/adapt/feature_based/_dann.py

```python
# Unpack the data.
Xs, Xt, ys, yt = self._unpack_data(data)

# loss
with tf.GradientTape() as task_tape, tf.GradientTape() as enc_tape, tf.GradientTape() as disc_tape:
    # Forward pass
    Xs_enc = self.encoder_(Xs, training=True)
    ys_pred = self.task_(Xs_enc, training=True)
    ys_disc = self.discriminator_(Xs_enc, training=True)

    Xt_enc = self.encoder_(Xt, training=True)
    yt_disc = self.discriminator_(Xt_enc, training=True)

    # Reshape
    ys_pred = tf.reshape(ys_pred, tf.shape(ys))

    # Compute the loss value
    task_loss = self.task_loss_(ys, ys_pred)

    disc_loss = (-tf.math.log(ys_disc + EPS)
                 -tf.math.log(1-yt_disc + EPS))

    task_loss = tf.reduce_mean(task_loss)
    disc_loss = tf.reduce_mean(disc_loss)

    enc_loss = task_loss - self.lambda_ * disc_loss

    task_loss += sum(self.task_.losses)
    disc_loss += sum(self.discriminator_.losses)
    enc_loss += sum(self.encoder_.losses)
```

G_f, G_y, G_d

L_y, L_d

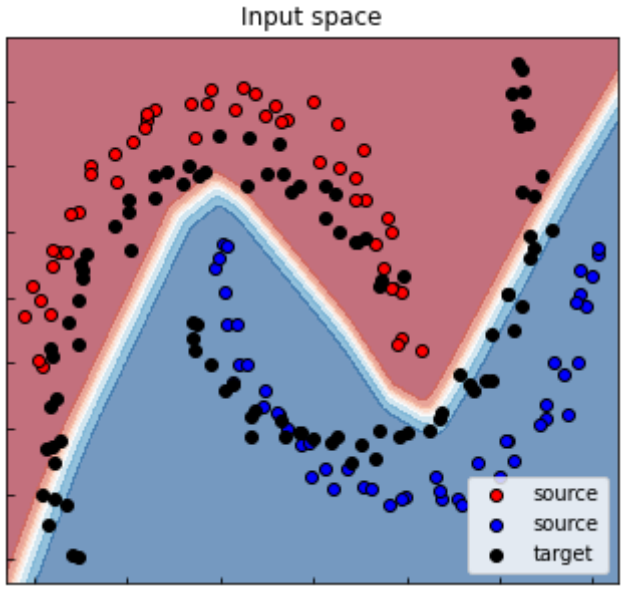
勾配反転層を用いる代わりに、特徴抽出器向けに符号を調整した損失を定義

勾配算出、モデルパラメータ更新

```python
# Compute gradients
trainable_vars_task = self.task_.trainable_variables
trainable_vars_enc = self.encoder_.trainable_variables
trainable_vars_disc = self.discriminator_.trainable_variables

gradients_task = task_tape.gradient(task_loss, trainable_vars_task)
gradients_enc = enc_tape.gradient(enc_loss, trainable_vars_enc)
gradients_disc = disc_tape.gradient(disc_loss, trainable_vars_disc)

# Update weights
self.optimizer.apply_gradients(zip(gradients_task, trainable_vars_task))
self.optimizer_enc.apply_gradients(zip(gradients_enc, trainable_vars_enc))
self.optimizer_disc.apply_gradients(zip(gradients_disc, trainable_vars_disc))
```

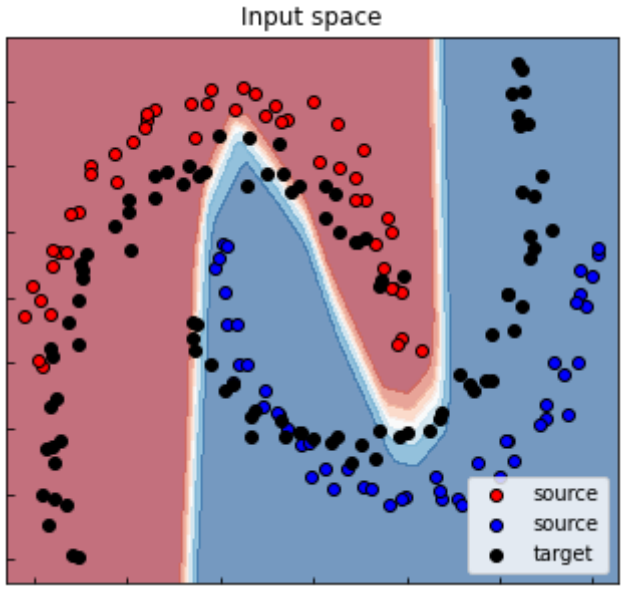
④DANN～結果

元ドメインのみ(λ=0)



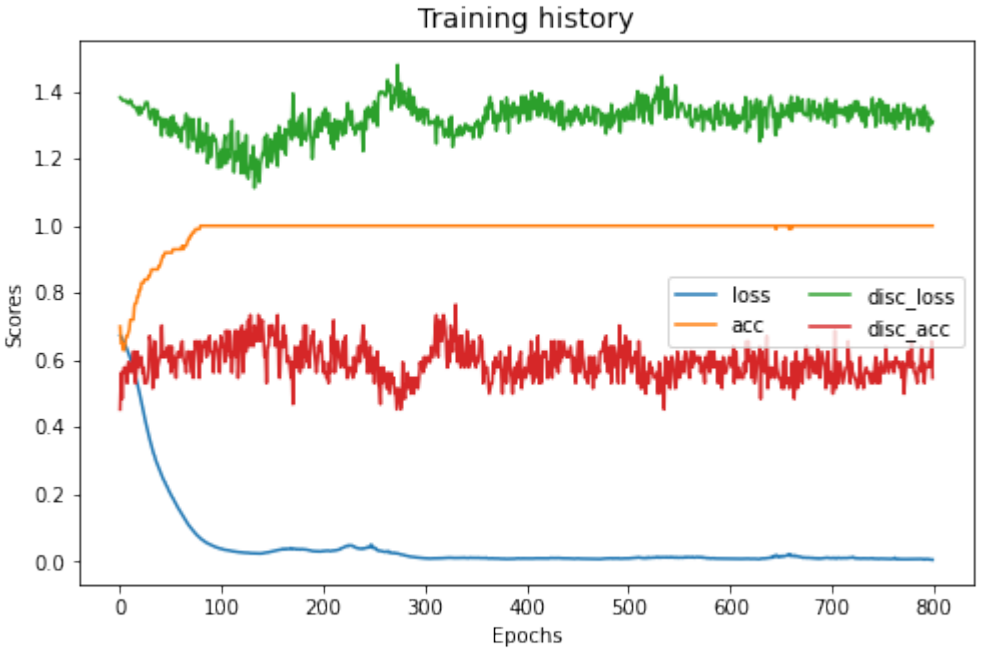
acc: 0.84

DANN



acc: 0.94

DANNの学習曲線



https://adapt-python.github.io/adapt/examples/Two_moons.html#DANN

異種転移～概要

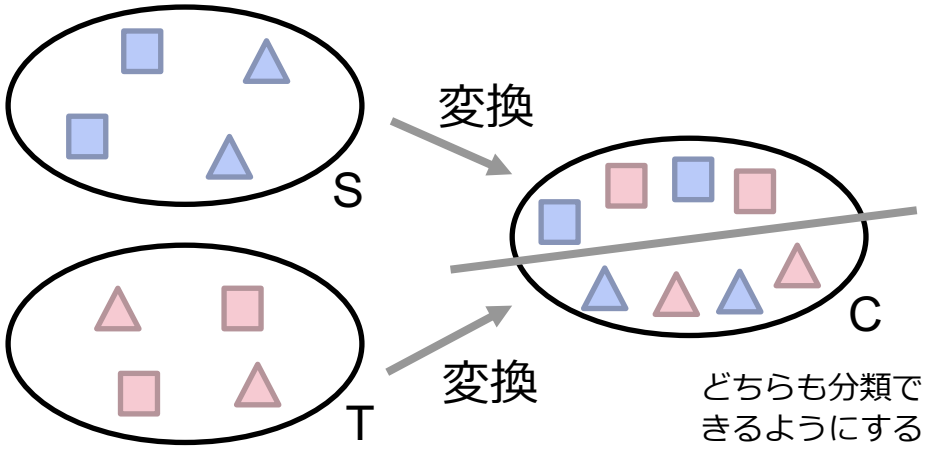
■ アプローチの特色

- 入力空間が異なるケース ($\mathcal{X}_S \neq \mathcal{X}_T$) を扱う
- 入力空間の違いに対処するため、特徴転移のアプローチをとる

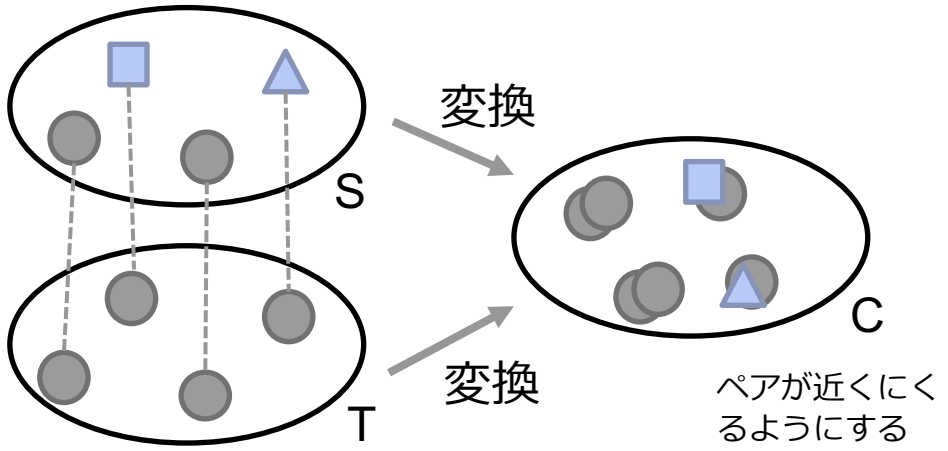
■ 分類

- ドメイン間で共通のラベルを用いる
- ドメイン横断で対応関係のある特徴ペア (ブリッジインスタンス) を用いる
- いずれの情報も用いない

ラベルの使用イメージ



ブリッジインスタンスの使用イメージ



異種転移～手法

■ ラベルを用いる手法

- **HFA (heterogeneous feature augmentation)** [Li+ 14]
  - FAM同様に特徴次元を拡張するが、ドメイン共通の次元を作るために変換を用いる
- **Arc-t (asymmetric regularized cross-domain transformation)** [Kulis+ 11]
  - ドメインの異なるサンプルペアのラベルの一致・不一致を識別するような変換を求める
- **HeDANN (heterogeneous DANN)** [Yonekawa +19]
  - DANNにおいてエンコーダをドメイン固有にし目標ドメインのラベルも用いるもの
  - ➡以降で詳説

■ ブリッジインスタンスを用いる手法

- **CT-SVM (correlation-transfer SVM)** [Yeh+ 14]
  - カーネル正準相関分析による共通の特徴空間でSVMを学習する。その際、相関係数で正則化する
- **TTL (transitive transfer learning)** [Tan+ 15]
  - 行列因子分解ベースの同時クラスタリングにより共通の特徴空間に変換してラベル伝搬する

■ いずれの情報も用いない手法

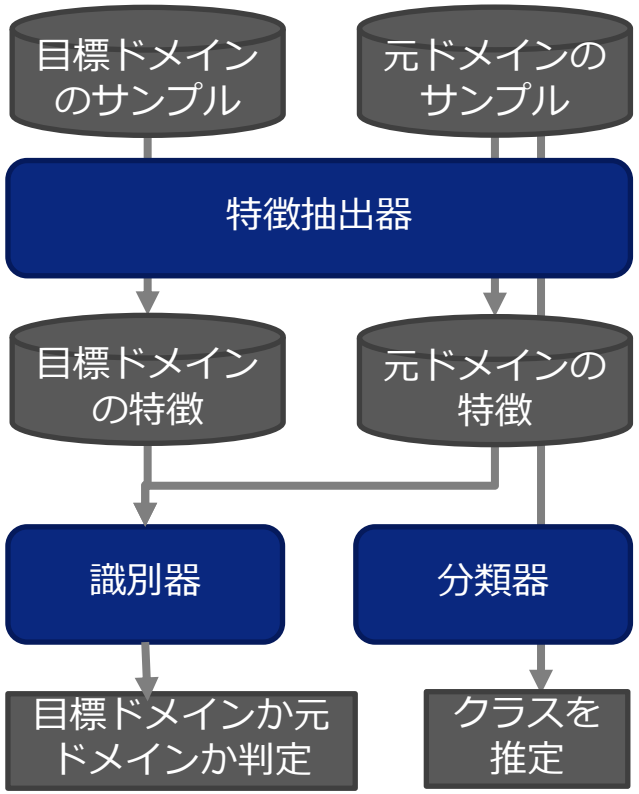
- **HeMap (heterogeneous spectral mapping)** [Shi+ 10b]
  - 再構成損失と共通の特徴空間の乖離度とを最小化するよう行列因子分解する

HeDANN～概要

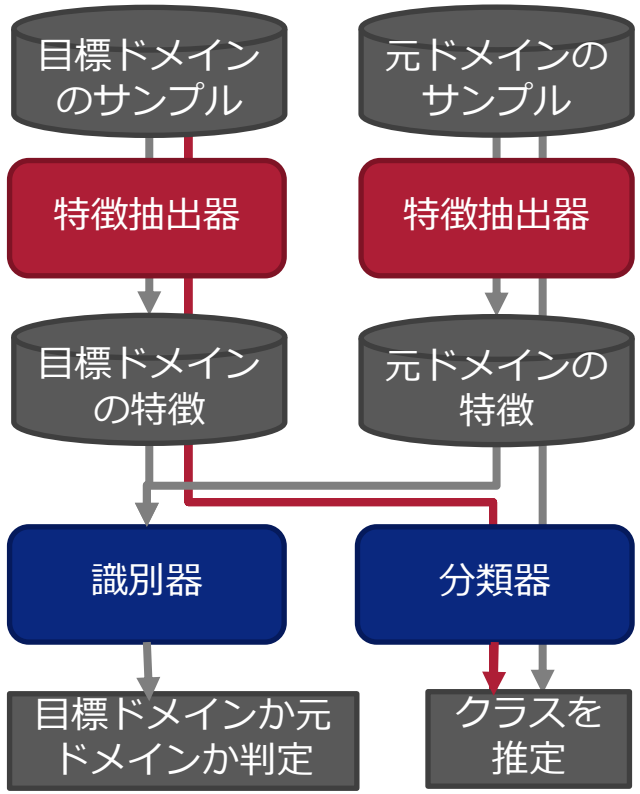
■ HeDANN

- DANNにおいてエンコーダをドメイン固有にし目標ドメインのラベルも用いるもの

DANN



HeDANN



HeDANN～コーディング

```
# Unpack the data.
Xs, Xt, ys, yt = self._unpack_data(data)

# loss
with tf.GradientTape() as task_tape, tf.GradientTape() as s_enc_tape, tf.GradientTape() as t_enc_tape, tf.GradientTape() as disc_tape:
    # Forward pass
    Xs_enc = self.encoder_(Xs, training=True)
    ys_pred = self.task_(Xs_enc, training=True)
    ys_disc = self.discriminator_(Xs_enc, training=True)

    Xt_enc = self.target_encoder_(Xt, training=True)
    yt_pred = self.task_(Xt_enc, training=True)
    yt_disc = self.discriminator_(Xt_enc, training=True)

    # Reshape
    ys_pred = tf.reshape(ys_pred, tf.shape(ys))
    yt_pred = tf.reshape(yt_pred, tf.shape(yt))

    # Compute the loss value
    s_task_loss = self.task_loss_(ys, ys_pred)
    t_task_loss = self.task_loss_(yt, yt_pred)
    task_loss = tf.concat([s_task_loss, t_task_loss], 0)

    disc_loss = (-tf.math.log(ys_disc + EPS)
                 -tf.math.log(1-yt_disc + EPS))

    task_loss = tf.reduce_mean(task_loss)
    s_task_loss = tf.reduce_mean(s_task_loss)
    t_task_loss = tf.reduce_mean(t_task_loss)
    disc_loss = tf.reduce_mean(disc_loss)

    enc_loss = task_loss - self.lambda_ * disc_loss

    task_loss += sum(self.task_.losses)
    disc_loss += sum(self.discriminator_.losses)
    enc_loss += sum(self.encoder_.losses) + sum(self.target_encoder_.losses)
```

目標ドメインのデータに、元ドメインとは別のエンコーダを適用し、元ドメインと同じ分類器でクラスの予測を実行

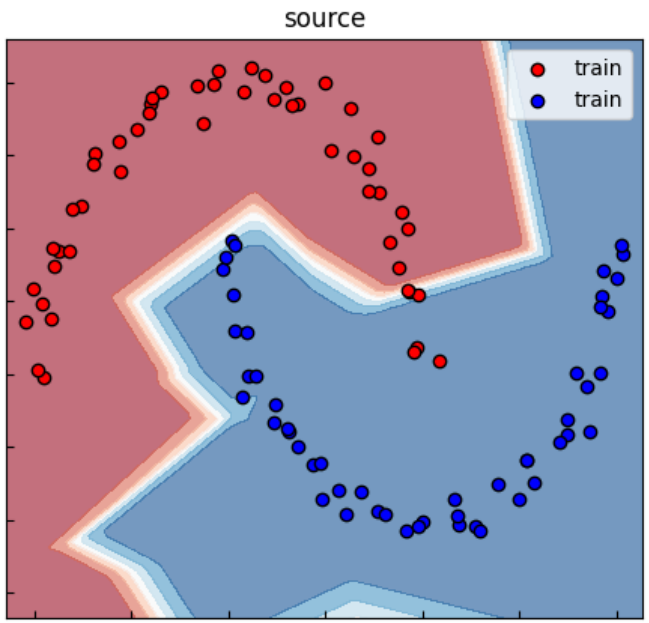
```
# Compute gradients
trainable_vars_task = self.task_.trainable_variables
trainable_vars_disc = self.discriminator_.trainable_variables
trainable_vars_s_enc = self.encoder_.trainable_variables
trainable_vars_t_enc = self.target_encoder_.trainable_variables

gradients_task = task_tape.gradient(task_loss, trainable_vars_task)
gradients_disc = disc_tape.gradient(disc_loss, trainable_vars_disc)
gradients_s_enc = s_enc_tape.gradient(enc_loss, trainable_vars_s_enc)
gradients_t_enc = t_enc_tape.gradient(enc_loss, trainable_vars_t_enc)

# Update weights
self.optimizer.apply_gradients(zip(gradients_task, trainable_vars_task))
self.optimizer_disc.apply_gradients(zip(gradients_disc, trainable_vars_disc))
self.optimizer_enc.apply_gradients(zip(gradients_s_enc, trainable_vars_s_enc))
self.optimizer_t_enc.apply_gradients(zip(gradients_t_enc, trainable_vars_t_enc))
```

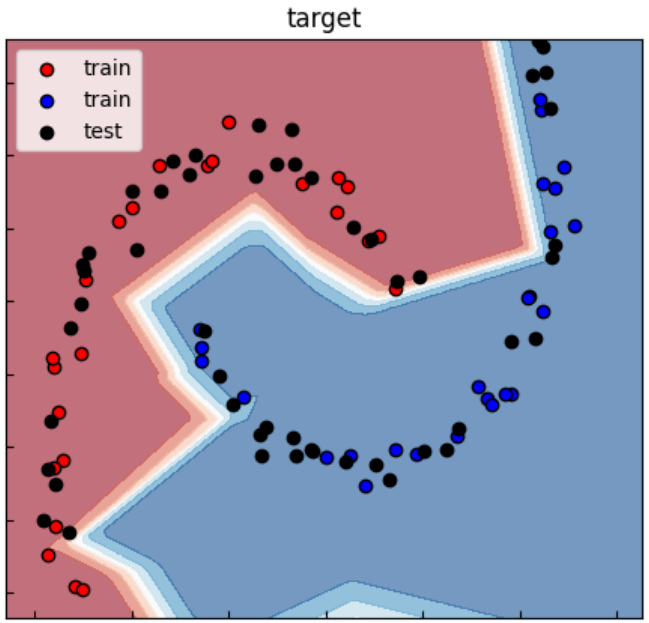
HeDANN～結果

Two Moonsデータセットを改変
・元ドメイン：そのまま。2次元
・目標ドメイン：XZ平面で反時計回りに30度回転。3次元

HeDANN

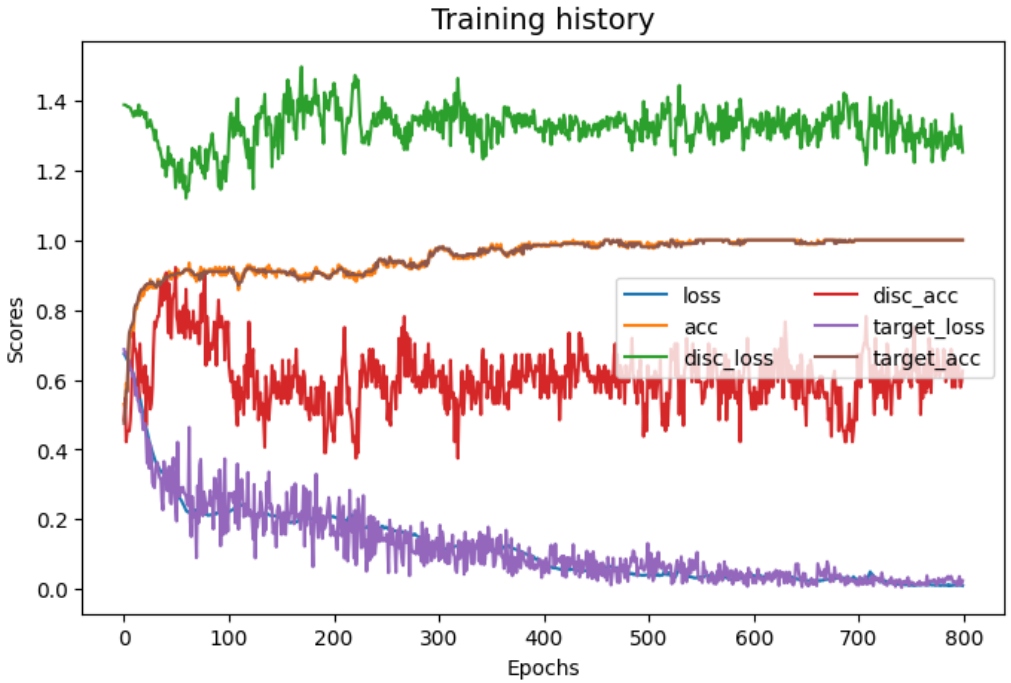
※XZ平面の回転を元に戻して描画





acc: 1.0

HeDANNの学習曲線



まとめ

■ 転移学習の問題設定等を説明後、各種手法をコードや動作結果を添えて説明

■ 同種転移

- ①事例転移（KDDI総合研究所 黒川）
- パラメータ転移（大阪大学 Li）
  - ②事前学習モデルの利用
  - ③メタ学習
- ④特徴転移（KDDI総合研究所 米川）

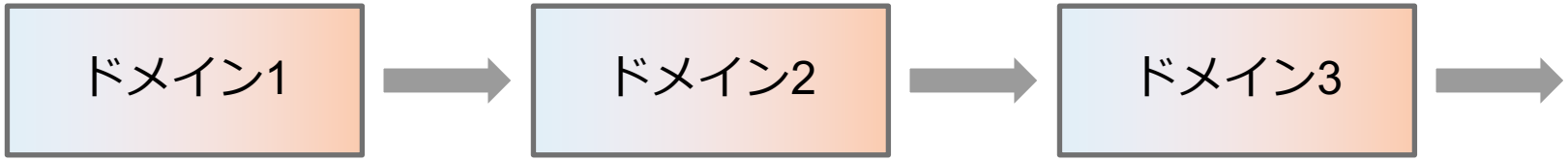
■ 異種転移（KDDI総合研究所 米川）

  - 特徴転移の異種拡張

発展的な話題

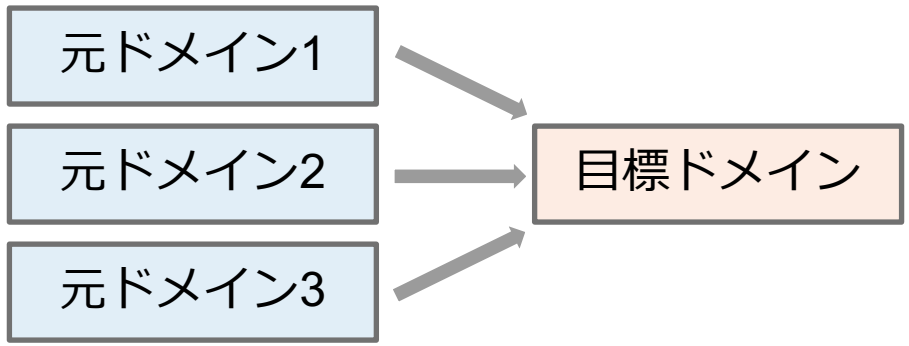
■ 2ドメイン以上への発展

- **継続学習** [Chen+ 18, Delange+ 18]
  - 次々と訪れるドメイン（またはタスク）に適応

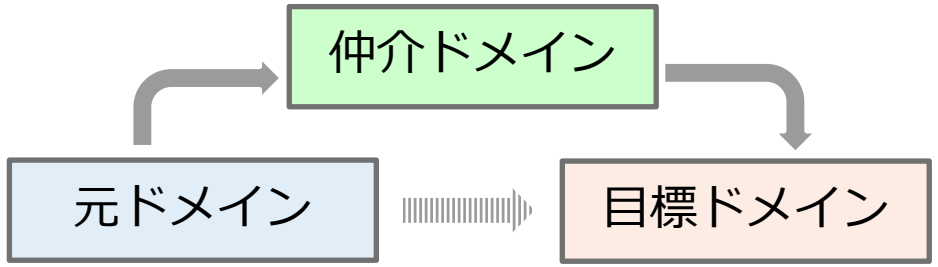


  - 過去のドメイン知識を忘れないように忘却抑止
    - 新ドメインの追加学習の際に過去データを再生する方法や、モデル上でのパラメータの正則化／分離によりパラメータの挙動を制御する方法により対応

- **マルチソース転移**
  - 複数ドメインからの転移 [Xu+ 12]



- **遷移的転移**
  - 仲介ドメインを介した転移 [Tan+ 15]



発展的な話題

■ 理論的な理解（When to Transferの理解）

● 不一致度：ドメインの類似度を分布間（擬）距離で測る

例：$\mathcal{H}$-ダイバージェンス [Ben-David+ 2010]

$$d_{\mathcal{H}\Delta\mathcal{H}}(P,Q) := 2 \sup_{h \in \mathcal{H}\Delta\mathcal{H}} \left|P(I_h) - Q(I_h)\right| \qquad \boldsymbol{x} \in I_h \Leftrightarrow h(\boldsymbol{x}) = 1$$

$$= \{h \oplus h' \mid h, h' \in \mathcal{H}\}$$：排他的論理和

- $d_{\mathcal{H}\Delta\mathcal{H}}(P,Q)$ が大きい→ PとQは分布として「似ていない」
- $d_{\mathcal{H}\Delta\mathcal{H}}(P,Q)$ が小さい→ PとQは分布として「似ている」

➡ 転移のしやすさの指標になる

● 目標ドメインの期待リスクの上界評価（汎化誤差解析）

例：$\mathcal{H}$-ダイバージェンスに基づく上界 [Ben-David+ 2010]

$$R_T(h) \leq R_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(P_{\mathcal{X}_T}, P_{\mathcal{X}_S}) + \min_{h \in \mathcal{H}} (R_S(h) + R_T(h))$$

- $R_T(h)$：目標ドメインのリスク
- $R_S(h)$：元ドメインのリスク
- $d_{\mathcal{H}\Delta\mathcal{H}}(P_{\mathcal{X}_T}, P_{\mathcal{X}_S})$：両ドメインの入力分布間の不一致度
- $\min_{h \in \mathcal{H}} (R_S(h) + R_T(h))$：両ドメインのラベリングルール間の不一致度

参考文献（1/2）

■ 全般・理論
- [統計的学習理論] 金森敬文, 統計的学習理論,機械学習プロフェッショナルシリーズ, 講談社.
- NIPS (NeurIPS) '05 Workshop, http://socrates.acadiau.ca/courses/comp/dsilver/Share/2005Conf/NIPS2005_ITWS/Website/
- [Ben-David+ 2010] Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. Machine learning, 79(1), 151-175.

■ ツール
- [ADAPT] A. de Mathelin, F. Deheeger, G. Richard, M. Mougeot, and N. Vayatis, "ADAPT: Awesome Domain Adaptation Python Toolbox," arXiv preprint arXiv:2107.03049, 2021. URL: https://adapt-python.github.io/adapt/

■ 事例転移
- [Sugiyama+ 08] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe, "Direct importance estimation for covariate shift adaptation," Annals of the Institute of Statistical Mathematics, vol. 60, no. 4, pp. 699-746, 2008.
- [Kanamori+ 09] Kanamori, T., Hido, S., and Sugiyama, M., “A least-squares approach to direct importance estimation,” Journal of Machine Learning Research, vol.10 (Jul.), pp.1391-1445, 2009.
- [Yamada+ 13] Yamada, M., Suzuki, T., Kanamori, T., Hachiya, H., and Sugiyama, M., “Relative density-ratio estimation for robust distribution comparison,” Neural computation, 25(5), pp. 1324-1370, 2013.
- [Dai+ 07] Dai W., Yang Q., Xue G., and Yu Y., “Boosting for transfer learning,” ICML, 2007.
- [Shi+ 10] X. Shi, Q. Liu, W. Fan, Q. Yang, and P.S. Yu, “Predictive modeling with heterogeneous sources,” SDM, pp. 814--825, 2010.

■ 事前モデルの利用
- [Hinton+ 15] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," CoRR, vol. abs/1503.02531, 2015.
- [Romero+ 14] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in 3rd International Conference on Learning Representations, 2015.
- [Zhang+ 17] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in IEEE Conference on Computer Vision and Pattern Recognition, pp. 4320-4328, 2018.
- [Yim+ 17] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in IEEE Conference on Computer Vision and Pattern Recognition, pp. 7130-7138, 2017.

■ メタ学習
- [Shell+ 17] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, pp. 4077-4087, 2017.
- [Santoro+ 16] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap, "Meta-learning with memory-augmented neural networks," in Proceedings of the 33nd International Conference on Machine Learning, JMLR Workshop and Conference Proceedings, vol. 48, pp. 1842-1850, 2016.
- [Finn+ 17] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in Proceedings of the 34th International Conference on Machine Learning, PMLR, vol. 70, pp. 1126-1135, 2017.

参考文献（2/2）

■ 特徴転移

- [Daume III 07] H. Daume III, "Frustratingly Easy Domain Adaptation," in Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, 2007, pp. 256–-263.
- [Sun+ 16] B. Sun, J. Feng, and K. Saenko, "Return of Frustratingly Easy Domain Adaptation," Proc. AAAI Conf. Artif. Intell., vol. 30, no. 1, 2016.
- [Sun+ 16b] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," ECCV 2016 Work. ECCV 2016. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2016.
- [Ganin+ 16] Y. Ganin et al., "Domain-Adversarial Training of Neural Networks," J. Mach. Learn. Res., vol. 17, no. 1, pp. 2096–-2030, May 2016.
- [Tzeng+ 17] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial Discriminative Domain Adaptation," Proc. 2017 IEEE Conf. Comput. Vis. Pattern Recognit., pp. 2962-2971, Feb. 2017.
- [Bousmalis+ 16] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain Separation Networks," in Advances in Neural Information Processing Systems 29, 2016, pp. 343-351.

■ 異種転移

- [Li+ 14] W. Li, L. Duan, D. Xu, and I. W. Tsang, "Learning With Augmented Features for Heterogeneous Domain Adaptation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 6, pp. 1134-1148, Jun. 2014.
- [Kulis+ 11] B. Kulis, K. Saenko, and T. Darrell, "What You Saw is Not What You Get: Domain Adaptation Using Asymmetric Kernel Transforms," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1785-1792, 2011.
- [Yonekawa +19] K. Yonekawa et al., A Heterogeneous Domain Adversarial Neural Network for Trans-Domain Behavioral Targeting, vol. 11607 LNAI, 2019.
- [Yeh+ 14] Yi-Ren Yeh, Chun-Hao Huang, and Y.-C. F. Wang, "Heterogeneous Domain Adaptation and Classification by Exploiting the Correlation Subspace," IEEE Trans. Image Process., vol. 23, no. 5, pp. 2009-2018, 2014.
- [Tan+ 15] B. Tan, Y. Song, E. Zhong, and Q. Yang, "Transitive Transfer Learning," in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1155–-1164, 2015.
- [Shi+ 10b] X. Shi, Q. Liu, W. Fan, P. S. Yu, and R. Zhu, "Transfer Learning on Heterogenous Feature Spaces via Spectral Transformation," in Proceedings of the IEEE International Conference on Data Mining, pp. 1049-1054, 2010.

■ その他

- [Xu+ 12] Z. Xu, and S. Sun, "Multi-source transfer learning with multi-view adaboost," In International conference on neural information processing, pp. 332-339, 2012.
- [Chen+ 18] Z. Chen, and B. Liu, "Lifelong Machine Learning," Morgan & Claypool, 2018.
- [Delange+ 18] M. Delange et al., "A continual learning survey: Defying forgetting in classification tasks," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

謝辞

■ 本研究・調査の一部は、JST CREST JPMJCR21F2の支援、科研費（課題番号：20K19871）の助成を受けたものである。

KDDI Research