メタ学習入門

岩田具治

NTTコミュニケーション科学基礎研究所

目次

- メタ学習とは
- 代表的メタ学習法
  - Gradient-based (Model-agnostic meta-learning)
  - Black-box adaptation (Neural process)
  - Model-based (Prototypical network)
- 異なる特徴量空間を持つデータからのメタ学習

深層学習

- 画像処理，自然言語処理，音声認識など様々なタスクで高い性能を達成
- 高い性能を達成するためには，膨大な数のパラメータを持つモデルを膨大なデータで学習する必要がある
  - GPT-3（言語モデル）
    - 5兆語の学習データ（英語のWikipediaは37億語）
    - 1750億個のパラメータ
  - AlphaGo（囲碁）
    - 過去の16万回の対戦における3000万の局面を教師あり学習データ
  - AlphaGo Zero（ゲーム）
    - 490万回の自己対戦
  - Visual Transformer（画像認識）
    - 3億枚の学習画像

メタ学習

問題

- 大量の学習データが得られない場合が多々ある
- ラベル付けにコストがかかる，プライバシー保護，データ収集に時間がかかる

メタ学習

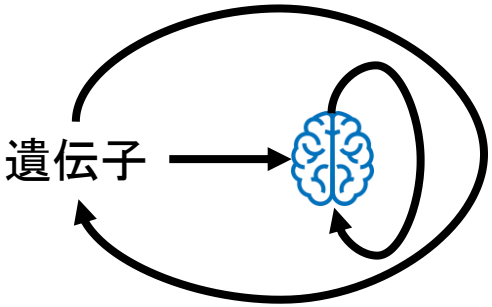
- 関連する他のデータから学習の仕方を学習することで，目的のタスクにおいて，少数の学習データしかなくても，性能を高める

効果

- 機械学習の適用領域の拡大
  - › 少量and/or質の悪いデータしかない場面でも機械学習を使えるようになる
- 「学習可能性」の理解
  - › 人間のように様々なコトを学習できるためには，どのような仕組みが必要？

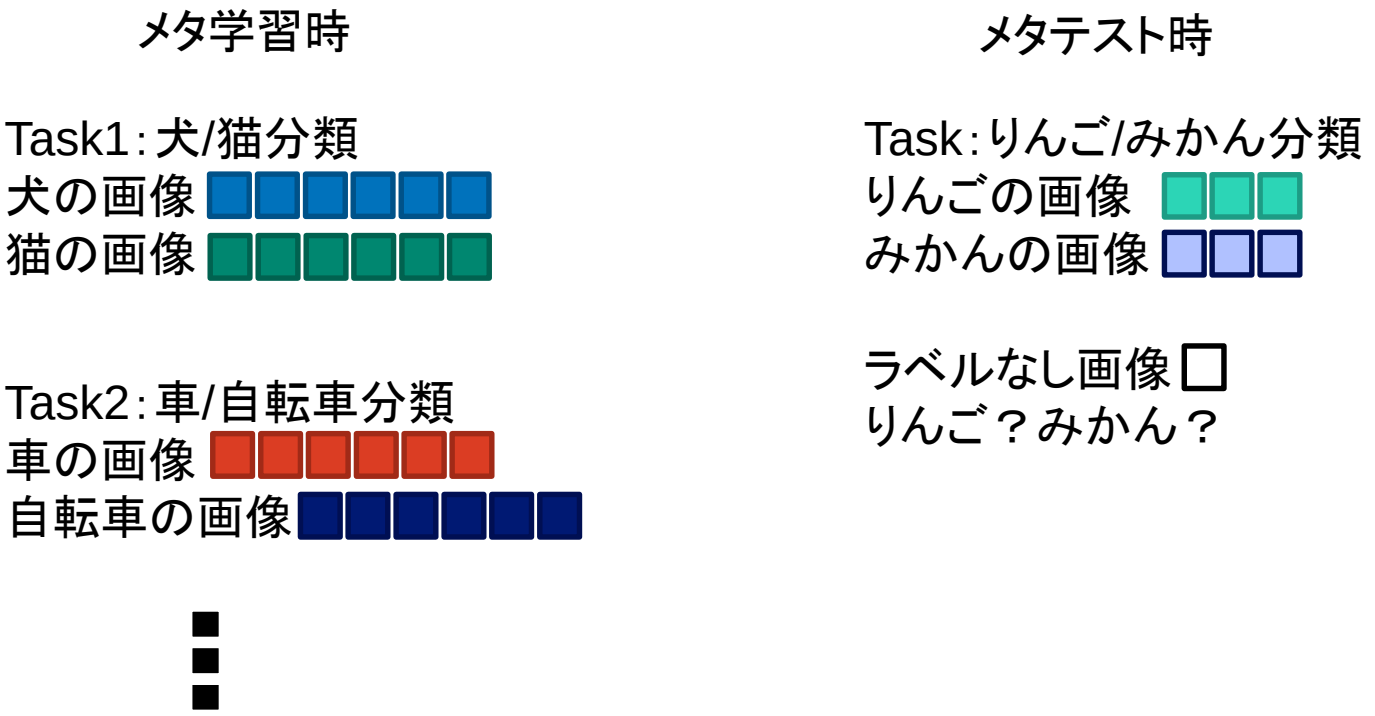
実世界におけるメタ学習例

- 人間は様々な経験から学習し，未知の状況にその経験を生かせる
  - 英語とイタリア語を話せたら，スペイン語を早く習得できる
  - 将棋がうまい人はチェスの上達が早い
- 動物
  - 各個体は様々な環境に置かれて経験から学習する
  - 学習により適応度が高くなった個体は多くの遺伝子を残す
  - For each 世代
    1. 遺伝子から神経ネットワークの初期値を生成
    2. 経験により神経ネットワークを更新（学習）
    3. うまく学習した個体の遺伝子が増える（遺伝的アルゴリズム）



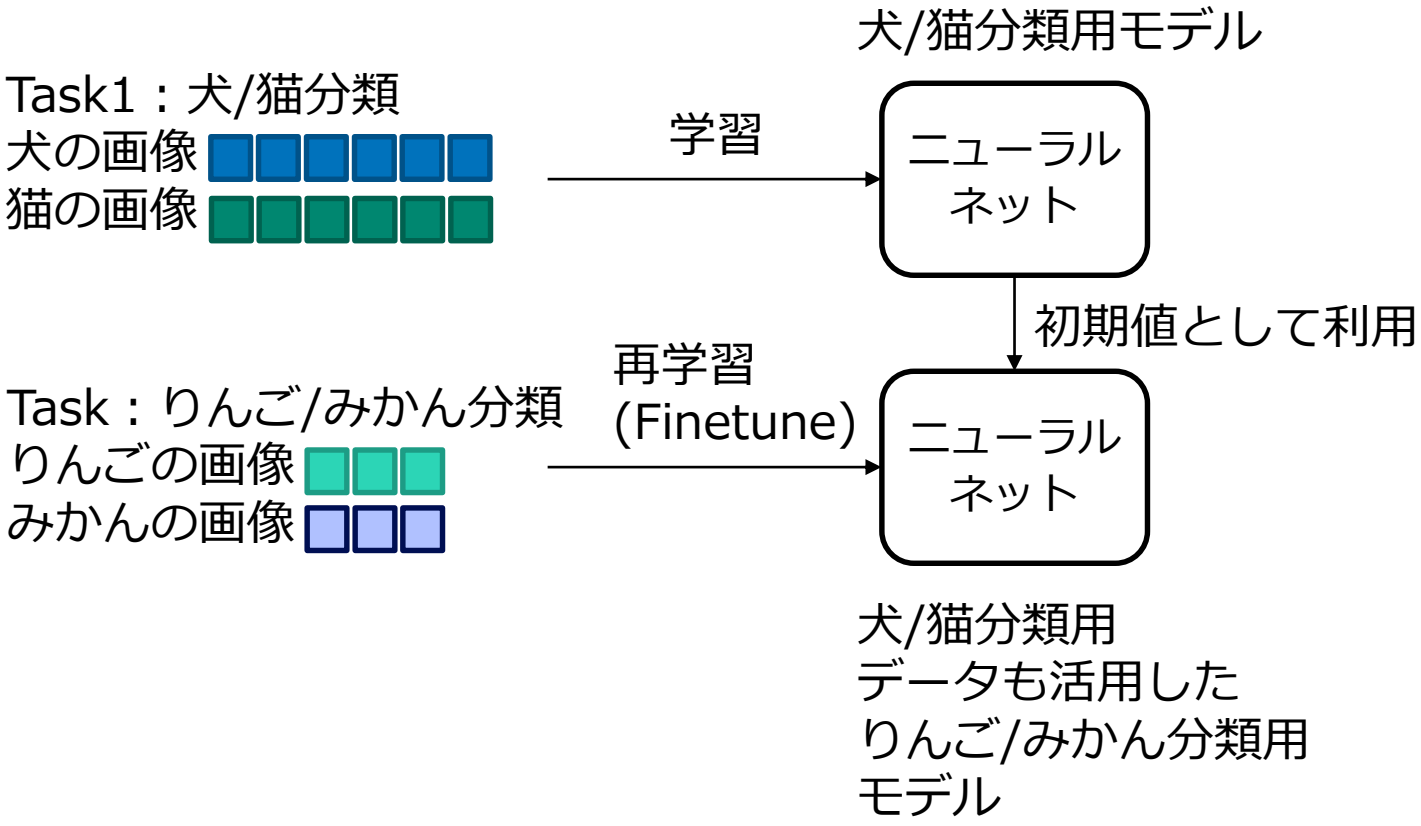
メタ学習の基本的設定

- 学習時とは違うタスク（異なるカテゴリの分類タスク）の性能を，少数のラベルありデータのみで向上させたい
- 例：新しい人の話者認識，新しい言語の翻訳，新しい分類体系での文書分類，新しい行動種類での行動識別



Finetune（メタ学習ではない）

1. あるタスク（元タスク）でニューラルネットを学習
2. 新しいタスク（目的タスク）でそのニューラルネットを再学習



Finetuneアルゴリズム

- 入力
  - 学習済ニューラルネットのパラメータθ
  - 目的タスクの学習データ$S = \{(x, y)\}$

特徴量　ラベル

- 出力
  - 目的タスク用のニューラルネットのパラメータθ'

- パラメータを初期化$\theta' = \theta$
- For each 学習エポック

損失関数　パラメータθ'の ニューラルネット

  1. 損失 $\mathcal{L}(\theta'|S) = \sum_{(x,y)\in S} L(\hat{y}(x;\theta'), y)$ とその勾配$\nabla\mathcal{L}(\theta'|S)$を計算
  2. 勾配法によりパラメータを更新 $\theta' = \theta' - \alpha\nabla\mathcal{L}(\theta'|S)$

学習率

Finetuneの問題点

- 元タスクでの学習では，目的タスクの初期値に利用されることを考慮しない
  - 元タスクでの性能は向上するが，目的タスクの性能が向上するとは限らない

- 目的タスクでの学習データがとても少ない場合（例：各クラス１個），検証用（validation）データを作成できない
  - エポック数（early stop）や学習率などのハイパーパラメータを調整できない
  - 過学習や未学習になりやすい

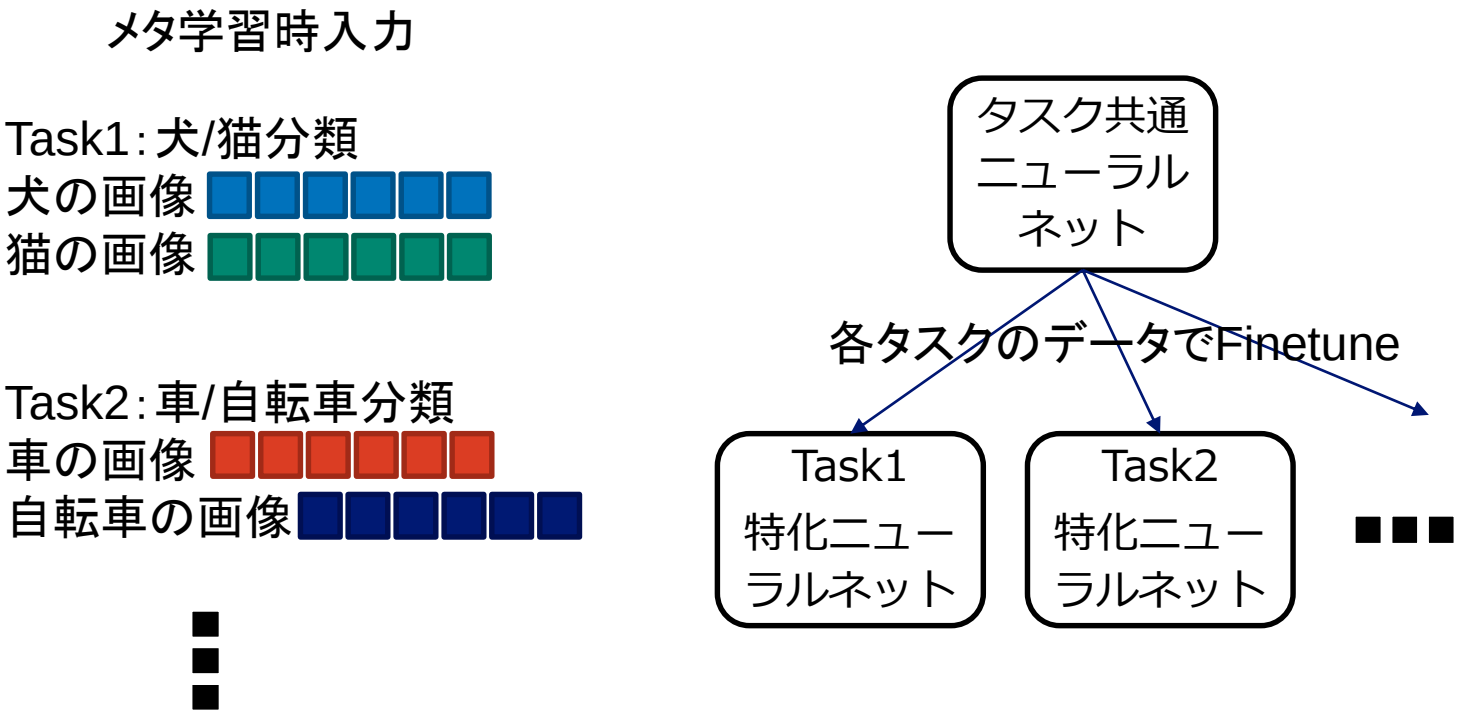
代表的メタ学習法

1. Gradient-based (Model-agnostic meta-learning)
2. Black-box adaptation (Neural process)
3. Model-based (Prototypical network)

Gradient-basedメタ学習

- Model-agnostic meta-learning (MAML)
- 様々なタスクに勾配法で適合（Finetune）したときにテスト性能がよくなるようにタスク共通ニューラルネットのパラメータを更新

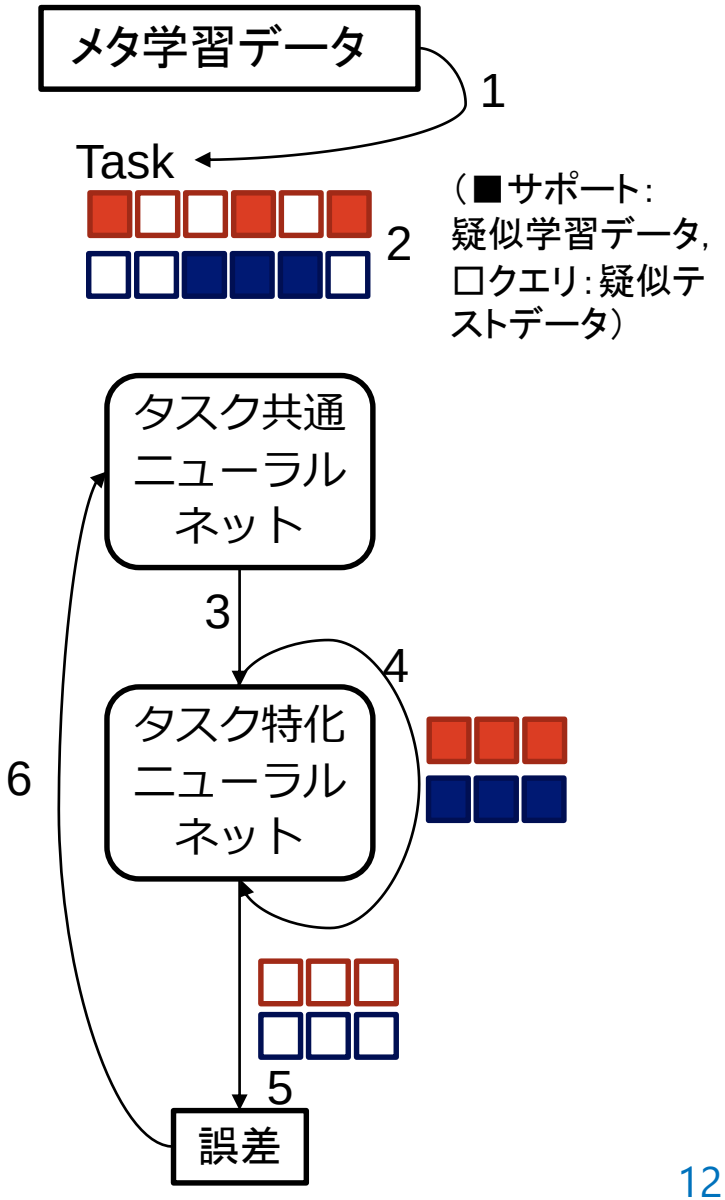
[Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks," *International Conference on Machine Learning*, 2017]



Gradient-basedメタ学習アルゴリズム

- タスク共通パラメータθをランダムに初期化
- For each メタ学習エポック
  1. タスクdをサンプリング
  2. タスクdからサポート集合$\mathcal{S}=\{(x,y)\}$とクエリ集合$\mathcal{Q}=\{(x,y)\}$をサンプリング
  3. タスクd用のニューラルネットのパラメータをタスク共通パラメータで初期化$\theta'=\theta$
  4. For each 学習エポック
     - › サポート集合に対する損失$\mathcal{L}(\theta'|\mathcal{S})=\sum_{(x,y)\in\mathcal{S}}L(\hat{y}(x;\theta'),y)$とその勾配$\nabla\mathcal{L}(\theta'|\mathcal{S})$を計算
     - › 勾配法によりパラメータを更新$\theta'=\theta'-\alpha\nabla\mathcal{L}(\theta'|\mathcal{S})$
  5. クエリ集合に対する損失$\mathcal{L}(\theta'|\mathcal{Q})=\sum_{(x,y)\in\mathcal{Q}}L(\hat{y}(x;\theta'),y)$とその勾配$\nabla\mathcal{L}(\theta'|\mathcal{Q})$を計算
  6. 勾配法によりタスク共通パラメータを更新$\theta=\theta-\beta\nabla\mathcal{L}(\theta'|\mathcal{Q})$

Fineturnと同じ



Gradient-basedメタ学習

- 疑似的に学習データ（サポート集合），テストデータ（クエリ集合）を作成することで期待テスト誤差を評価/最小化できる
  - Episodic training framework
- 自動微分ライブラリを利用することにより，勾配の勾配も計算可能

○ Finetuneしたときにタスク共通モデルを学習

○ 任意のモデルをメタ学習化できる

○ 経験的に性能がよいFinetuneを利用

× 内側のループ（Finetune）のエポック数に線形に必要メモリが増大
  - Finetuneのステップ数は少ない場合のみ計算可能（例：５）
    - › 陰関数定理を使うことでステップ数が多い場合の勾配計算を可能にする手法もある（implicit MAML）

[Rajeswaran, Aravind, et al. "Meta-Learning with Implicit Gradients." *Advances in Neural Information Processing Systems* (2019).]

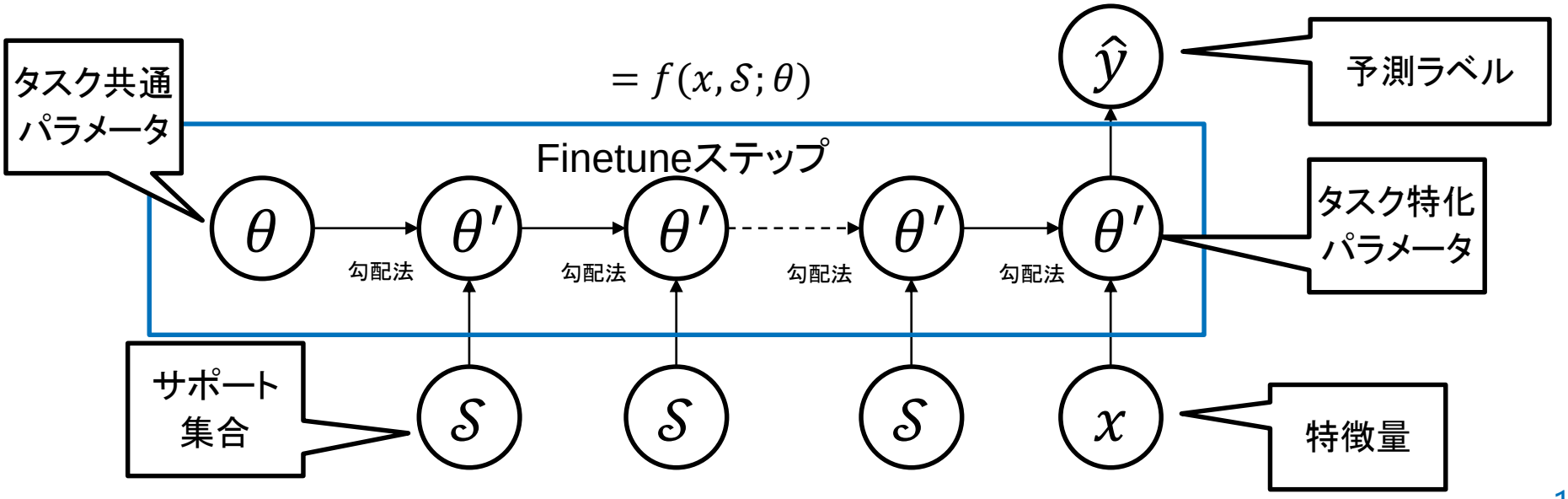
Gradient-basedメタ学習

入力：サポート集合Sと特徴量x

出力：予測ラベル$\hat{y}$

の勾配法に基づく計算を内部に組み込んだ

特殊なニューラルネット$f(x, S; \theta)$とみなせる



Black-box adaptationメタ学習

Black-boxなニューラルネット$f(x, \mathcal{S}; \theta)$でサポート集合$\mathcal{S}$への適合をモデル化

- For each メタ学習エポック
  1. タスクdをサンプリング
  2. タスクdからサポート集合 $\mathcal{S} = \{(x, y)\}$とクエリ集合 $\mathcal{Q} = \{(x, y)\}$ をサンプリング
  3. クエリ集合に対する損失 $\mathcal{L}(\theta|\mathcal{Q}) = \sum_{(x,y)\in\mathcal{Q}} L(f(x, \mathcal{S}; \theta), y)$ とその勾配 $\nabla\mathcal{L}(\theta|Q)$を計算
  4. 勾配法によりタスク共通パラメータを更新 $\theta = \theta - \beta\nabla\mathcal{L}(\theta|\mathcal{Q})$

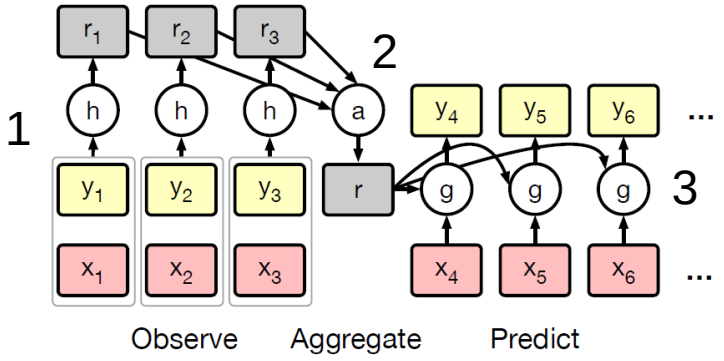
SNAIL [Mishra, Nikhil, et al. "A Simple Neural Attentive Meta-Learner." *International Conference on Learning Representations*. 2018.]
MANN [Santoro, A., et al. "Meta-Learning with Memory-Augmented Neural Networks." *33rd International Conference on Machine Learning, ICML 2016*.]
Neural Process [Gordon, Jonathan, et al. "Convolutional Conditional Neural Processes." *International Conference on Learning Representations*. 2019.]

Black-box関数例：Neural process

入力：サポート集合$S = \{(x_n, y_n)\}_{n=1}^{N}$と特徴量$x$

出力：Sに適合したxの予測ラベル$\hat{y}$

1. サポート集合の各ラベルありデータ(x_n, y_n)からニューラルネットhを用いて表現r_nを計算：$r_n = h(x_n, y_n)$
2. サポート集合の表現r_nを平均してタスクの表現rを計算：
   $r = \frac{1}{N}\sum_{n=1}^{N} r_n$
3. タスクの表現rと特徴量xからニューラルネットgを用いて予測ラベルを計算：$\hat{y} = g(x, r)$



[Gordon, Jonathan, et al. "Convolutional Conditional Neural Processes." *International Conference on Learning Representations*. 2019.]

Black-box adaptationメタ学習

○ 勾配の勾配の計算が必要がない

○ ニューラルネットで柔軟に適合がモデル化可能

- Gradient-basedメタ学習の場合は，共通の初期値から数ステップの勾配降下でタスク特化にたどり着けることが必要

× 適合のモデル化 $\hat{y} = f(x, S)$ は一般に教師あり学習 $\hat{y} = f(x)$ より複雑

- サポート集合に含まれるデータでも適切に予測できるとは限らない

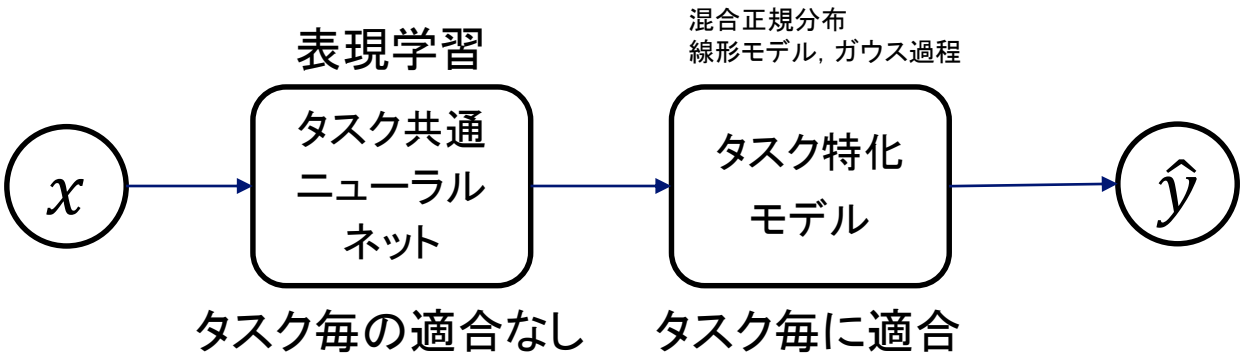
Model-basedメタ学習

- ニューラルネットの適合（Gradient-based）は勾配法による繰り返し計算が必要
- Finetuneとその微分の計算が容易なモデルを組み合わせる
  - 混合正規分布
    - $p(k|x) \propto \exp\left(-\left|\left|x-\mu_k\right|\right|^2\right), \mu_k = \frac{1}{N_k}\sum_{n:y_n=k} x_n$
  - 線形モデル
    - $\hat{y} = Wx, W = \left(X^\top X\right)^{-1} X^\top y$
  - ガウス過程

Prototypical networks [Snell, Jake, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." *Neural Information Processing Systems*. 2017]

R2D2[Bertinetto, Luca, et al. "Meta-learning with differentiable closed-form solvers." *International Conference on Learning Representations*. 2018]

[Iwata, Tomoharu, and Yusuke Tanaka. "Few-shot Learning for Spatial Regression." *arXiv preprint arXiv:2010.04360*, 2020]



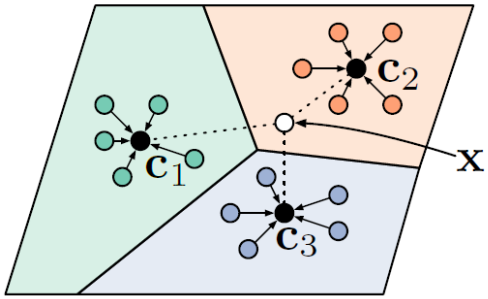
Model-basedメタ学習

Prototypical network

データの表現を計算するニューラルネット$f(x;\theta)$

表現空間における混合正規分布

[Snell, Jake, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." *Neural Information Processing Systems*. 2017]



- For each メタ学習エポック
  1. タスクdをサンプリング
  2. タスクdからサポート集合 $\mathcal{S} = \{(x, y)\}$とクエリ集合 $\mathcal{Q} = \{(x, y)\}$ をサンプリング
  3. 表現空間での各クラスの平均値をサポート集合で計算（Finetune）

$$c_k = \frac{1}{N_k} \sum_{(x,y)\in\mathcal{S}, y=k} f(x;\theta)$$

  4. クエリ集合に対して分類$p(k|x) \propto \exp(-\| f(x;\theta) - c_k \|^2)$
  5. クエリ集合に対する損失$\mathcal{L}(\theta|\mathcal{Q}) = \sum_{(x,y)\in\mathcal{Q}} L(p(k|x), y)$ とその勾配$\nabla\mathcal{L}(\theta|Q)$を計算
  6. 勾配法によりタスク共通パラメータを更新 $\theta = \theta - \beta\nabla\mathcal{L}(\theta|\mathcal{Q})$

Model-basedメタ学習

○ 大域的最適解に適合できる

○ 適合に勾配計算で繰り返し計算が不要

× 利用できるモデルに限りがあり，モデルを適切に選ぶ必要がある

× 適合の能力がそのモデルの能力に依存する

- ニューラルネットほど表現能力がない

Meta-learning from Tasks with Heterogeneous Attribute Space

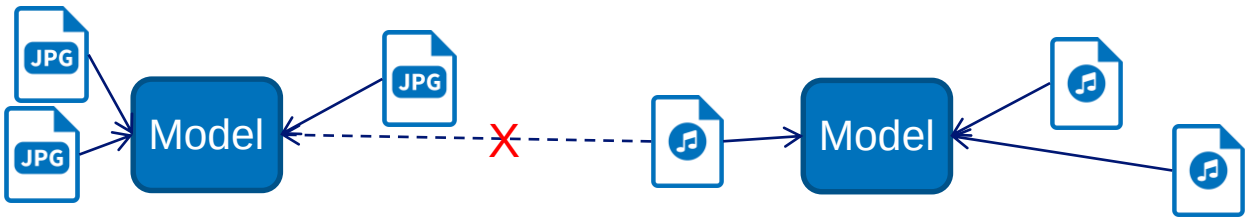
Tomoharu Iwata, Atsutoshi Kumagai

NeurIPS2020

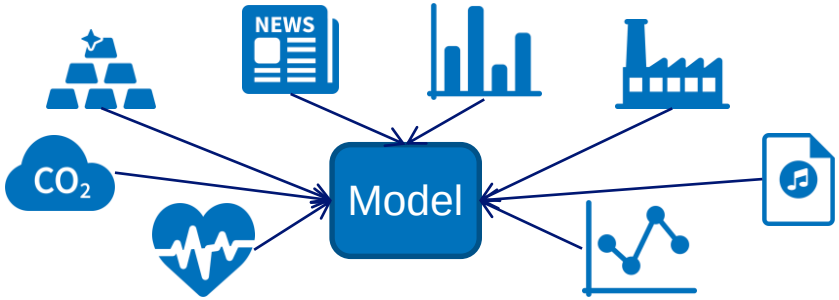
Introduction

- Neural networks require many labeled data.
- Meta-learning learns how to learn new tasks with small labeled data.

- Existing meta-learning methods assumes the attribute spaces are the same across tasks.



- We propose a meta-learning method that can learn from tasks with heterogeneous attribute spaces.

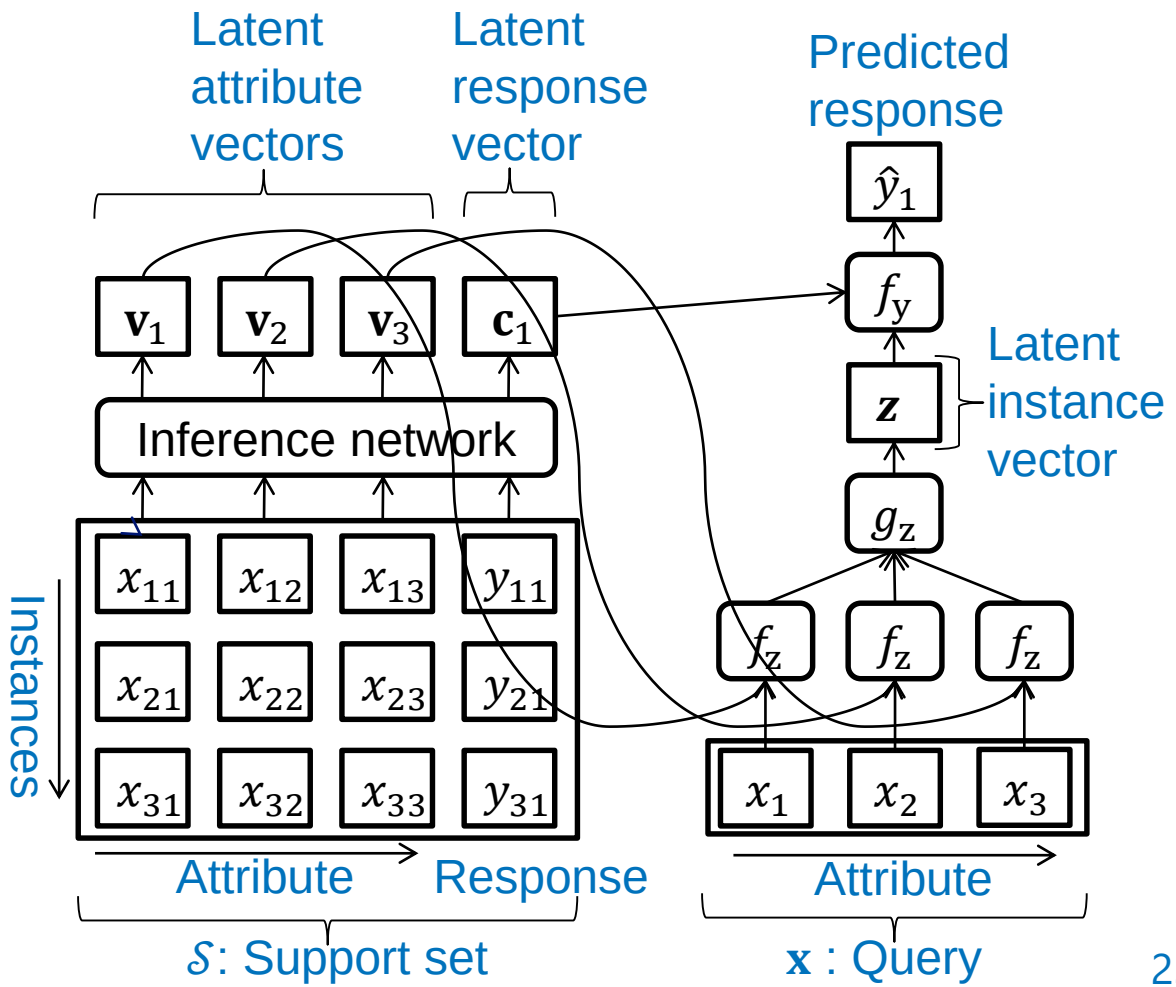


Our model: Overview

Input: Support set $\mathcal{S} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, query $\mathbf{x}$

Output: predicted response $\hat{y}$ for $\mathbf{x}$ adapted to $\mathcal{S}$

Our model can handle data with different numbers of attributes and responses using permutation invariant networks.

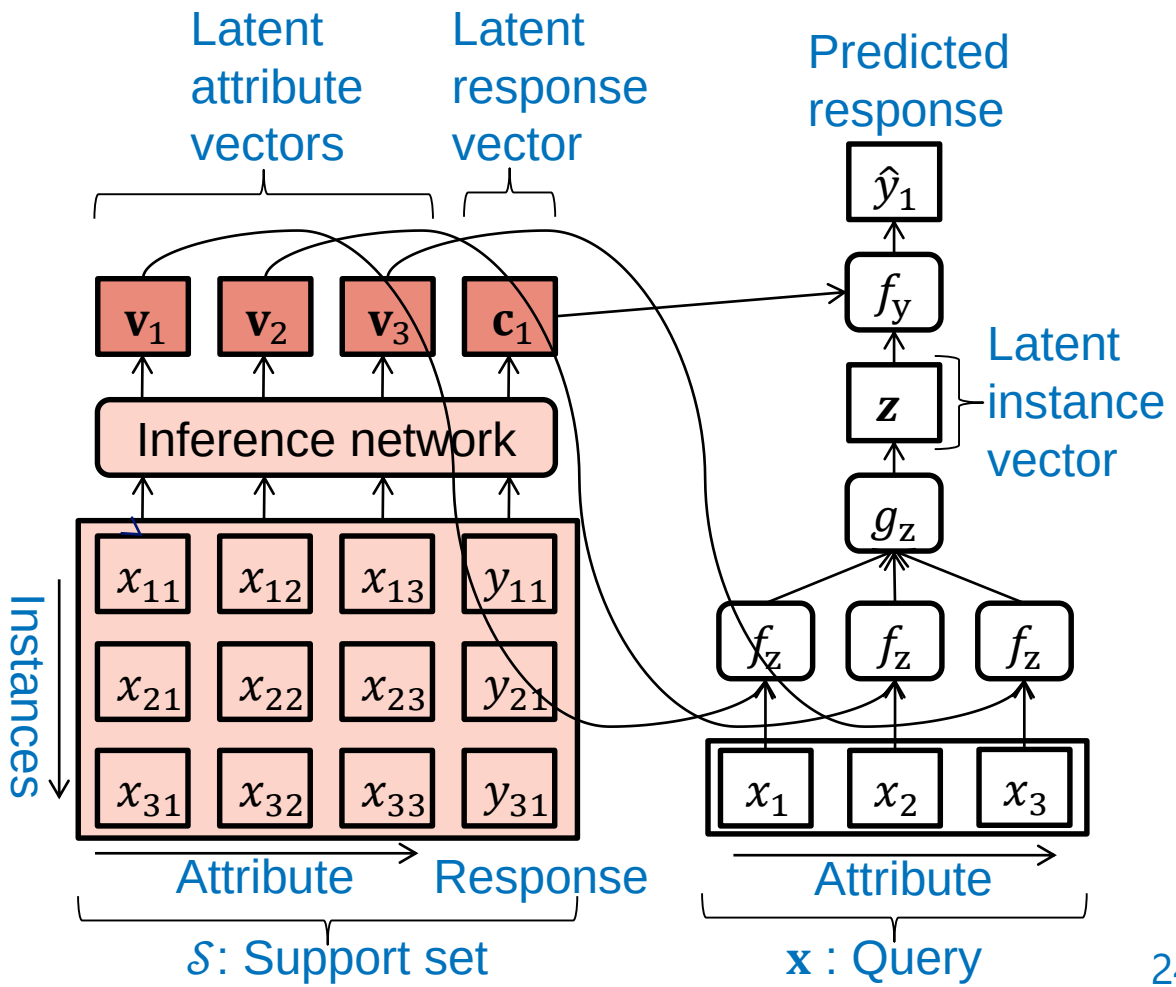


Our model: Overview

Input: Support set $\mathcal{S} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, query $\mathbf{x}$

Output: predicted response $\hat{y}$ for $\mathbf{x}$ adapted to $\mathcal{S}$

Our model can handle data with different numbers of attributes and responses using permutation invariant networks.

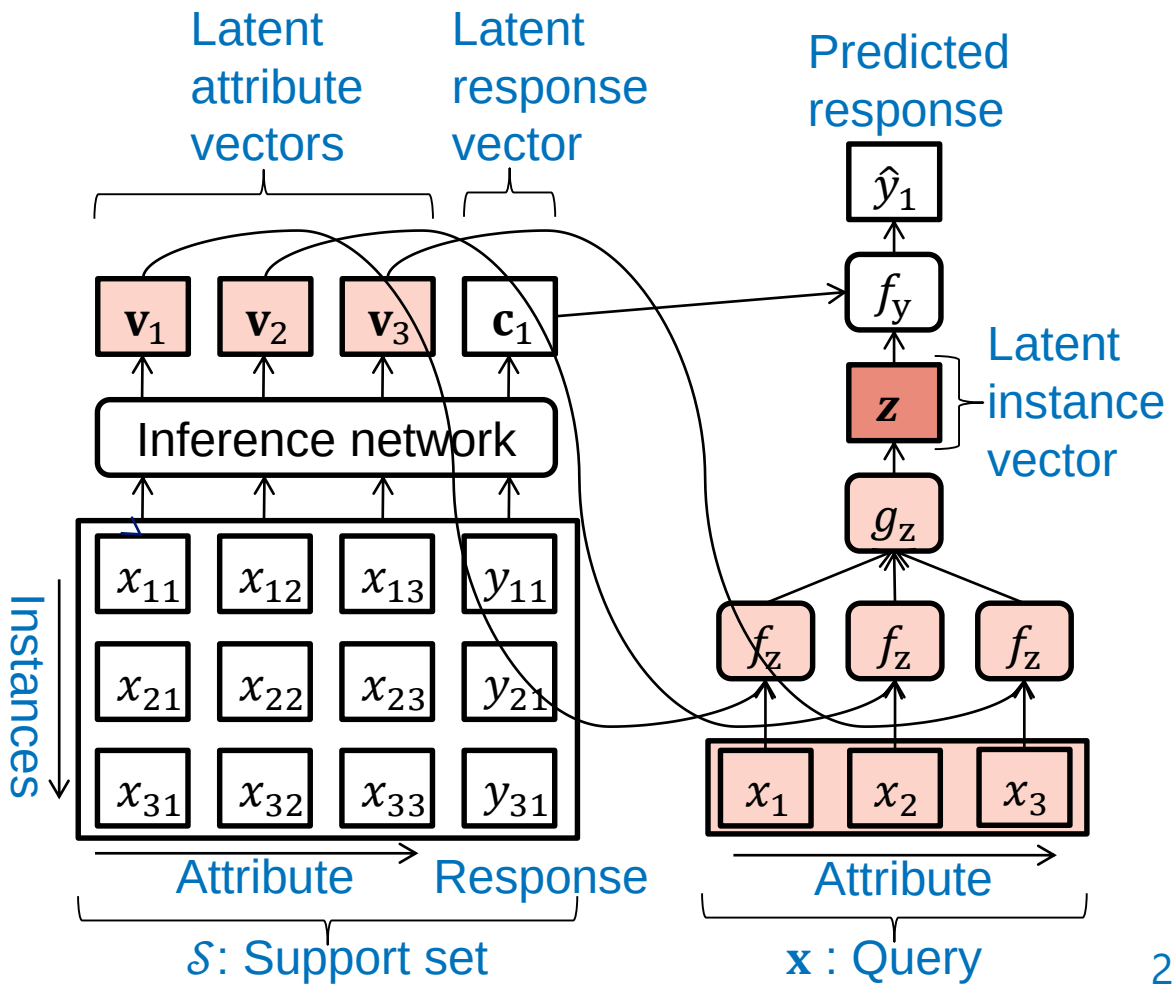


Our model: Overview

Input: Support set $\mathcal{S} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, query $\mathbf{x}$

Output: predicted response $\hat{y}$ for $\mathbf{x}$ adapted to $\mathcal{S}$

Our model can handle data with different numbers of attributes and responses using permutation invariant networks.

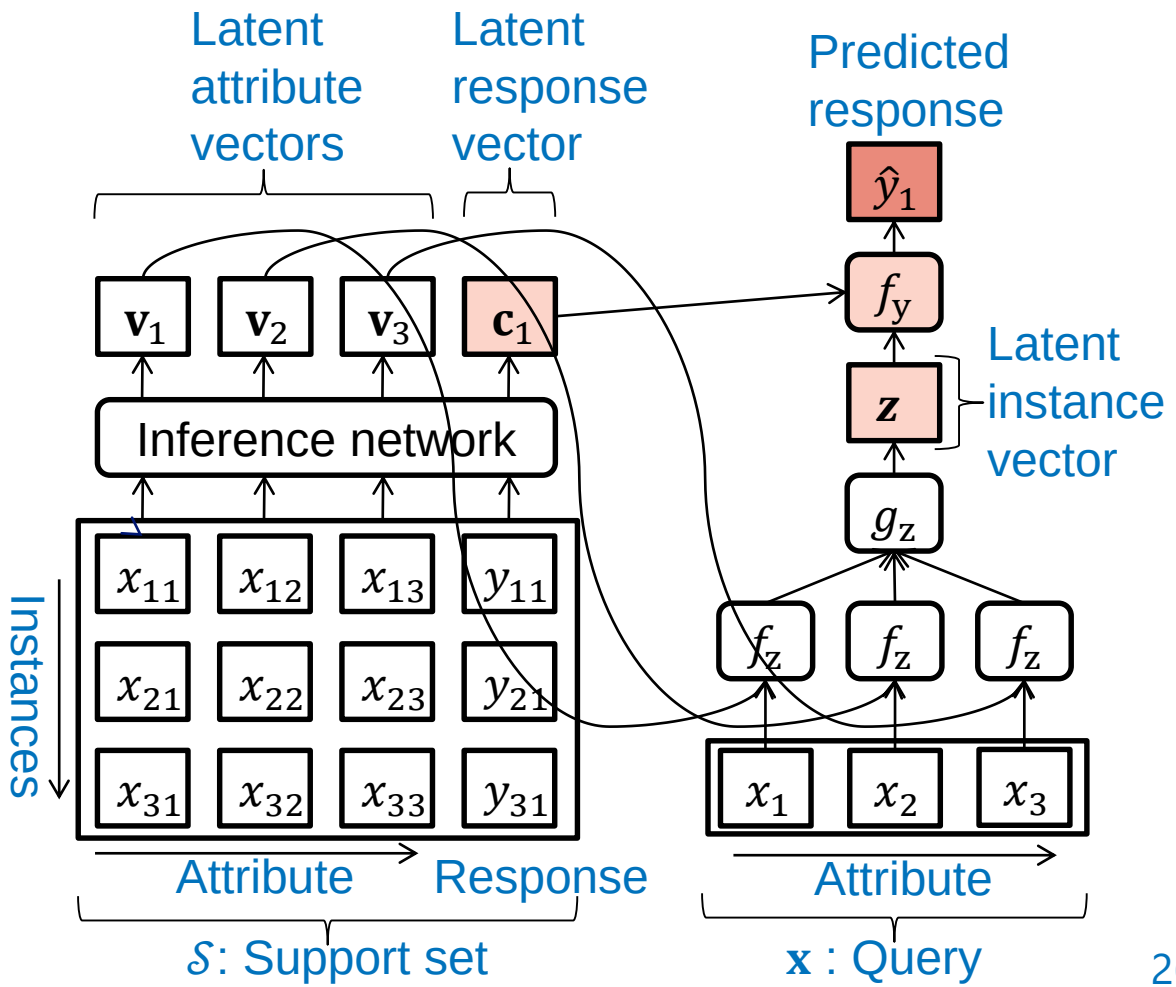


Our model: Overview

Input: Support set $\mathcal{S} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, query $\mathbf{x}$

Output: predicted response $\hat{y}$ for $\mathbf{x}$ adapted to $\mathcal{S}$

Our model can handle data with different numbers of attributes and responses using permutation invariant networks.



Our model: Inference network

1. Calculate initial attribute and response vectors using support set.

$$\bar{\mathbf{v}}_i = g_{\bar{\mathrm{v}}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\bar{\mathrm{v}}}(x_{ni})\right), \qquad \bar{\mathbf{c}}_j = g_{\bar{\mathrm{c}}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\bar{\mathrm{c}}}(y_{nj})\right)$$

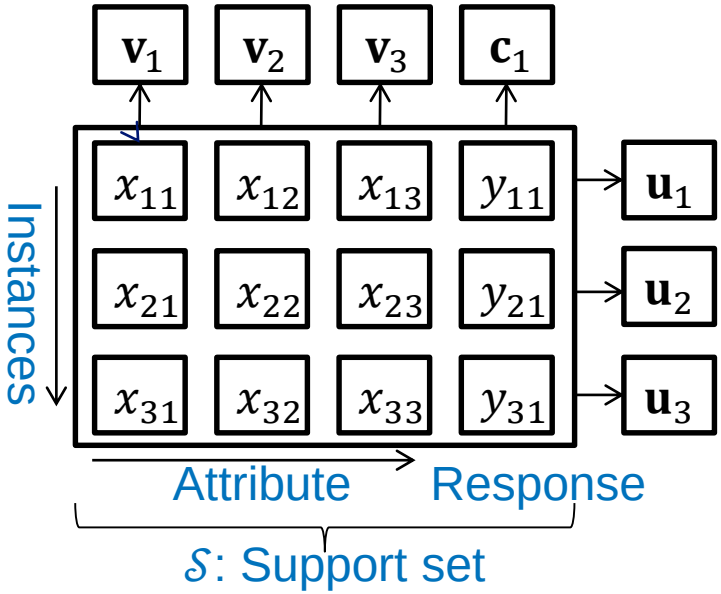
2. Calculate instance representation using attribute and response vectors and support set.

$$\mathbf{u}_i = g_{\mathrm{u}}\left(\frac{1}{I}\sum_{i=1}^{I} f_{\mathrm{u}}([\bar{\mathbf{v}}_i, x_{ni}]) + \frac{1}{J}\sum_{j=1}^{J} f_{\mathrm{u}}([\bar{\mathbf{c}}_j, y_{ni}])\right)$$

3. Calculate attribute and response vectors using instance representations and support set.

$$\mathbf{v}_i = g_{\mathrm{v}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\mathrm{v}}([\mathbf{u}_i, x_{ni}])\right), \qquad \mathbf{c}_j = g_{\mathrm{c}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\mathrm{c}}([\mathbf{u}_i, y_{nj}])\right)$$

f, g: neural networks
$[\cdot,\cdot]$: concatenation



Our model: Inference network

1. Calculate initial attribute and response vectors using support set.

$$\bar{\mathbf{v}}_i = g_{\bar{\mathrm{v}}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\bar{\mathrm{v}}}(x_{ni})\right), \qquad \bar{\mathbf{c}}_j = g_{\bar{\mathrm{c}}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\bar{\mathrm{c}}}(y_{nj})\right)$$

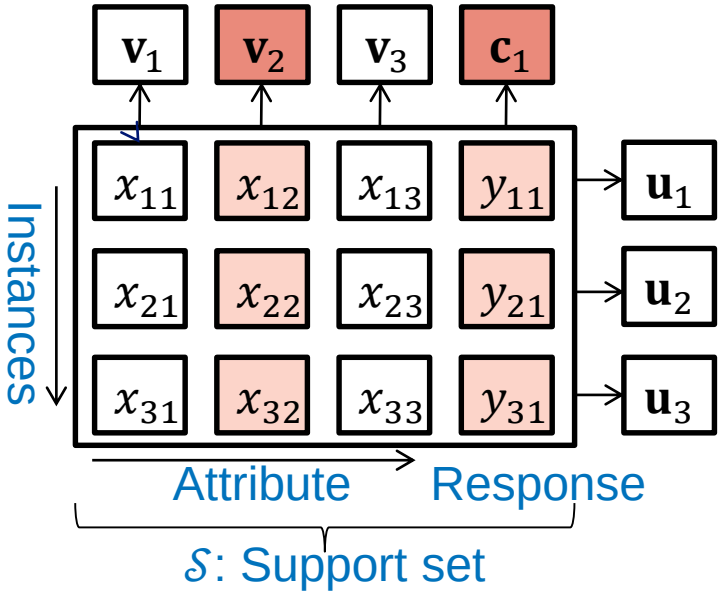
2. Calculate instance representation using attribute and response vectors and support set.

$$\mathbf{u}_i = g_{\mathrm{u}}\left(\frac{1}{I}\sum_{i=1}^{I} f_{\mathrm{u}}([\bar{\mathbf{v}}_i, x_{ni}]) + \frac{1}{J}\sum_{j=1}^{J} f_{\mathrm{u}}([\bar{\mathbf{c}}_j, y_{ni}])\right)$$

3. Calculate attribute and response vectors using instance representations and support set.

$$\mathbf{v}_i = g_{\mathrm{v}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\mathrm{v}}([\mathbf{u}_i, x_{ni}])\right), \qquad \mathbf{c}_j = g_{\mathrm{c}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\mathrm{c}}([\mathbf{u}_i, y_{nj}])\right)$$

f, g: neural networks
$[\cdot,\cdot]$: concatenation



Our model: Inference network

1. Calculate initial attribute and response vectors using support set.

$$\bar{\mathbf{v}}_i = g_{\bar{\mathrm{v}}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\bar{\mathrm{v}}}(x_{ni})\right), \qquad \bar{\mathbf{c}}_j = g_{\bar{\mathrm{c}}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\bar{\mathrm{c}}}(y_{nj})\right)$$

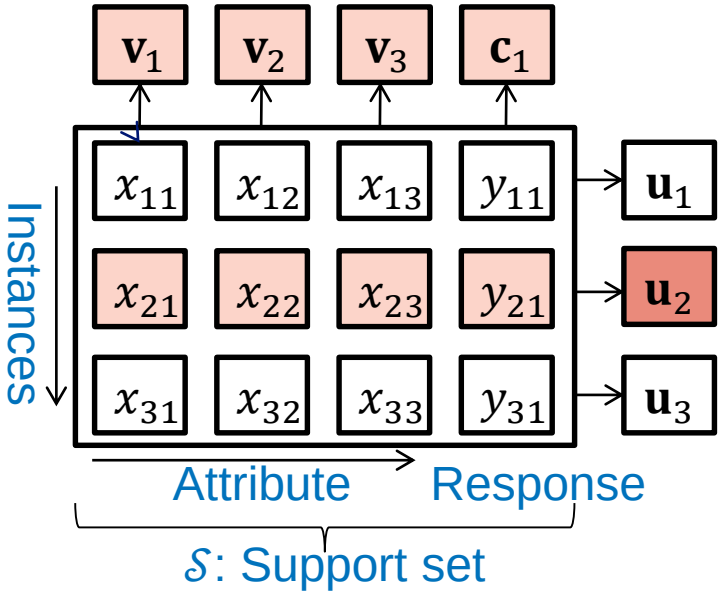
2. Calculate instance representation using attribute and response vectors and support set.

$$\mathbf{u}_i = g_{\mathrm{u}}\left(\frac{1}{I}\sum_{i=1}^{I} f_{\mathrm{u}}([\bar{\mathbf{v}}_i, x_{ni}]) + \frac{1}{J}\sum_{j=1}^{J} f_{\mathrm{u}}([\bar{\mathbf{c}}_j, y_{ni}])\right)$$

3. Calculate attribute and response vectors using instance representations and support set.

$$\mathbf{v}_i = g_{\mathrm{v}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\mathrm{v}}([\mathbf{u}_i, x_{ni}])\right), \qquad \mathbf{c}_j = g_{\mathrm{c}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\mathrm{c}}([\mathbf{u}_i, y_{nj}])\right)$$

f, g: neural networks
$[\cdot,\cdot]$: concatenation



Our model: Inference network

1. Calculate initial attribute and response vectors using support set.

$$\bar{\mathbf{v}}_i = g_{\bar{\mathrm{v}}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\bar{\mathrm{v}}}(x_{ni})\right), \qquad \bar{\mathbf{c}}_j = g_{\bar{\mathrm{c}}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\bar{\mathrm{c}}}(y_{nj})\right)$$

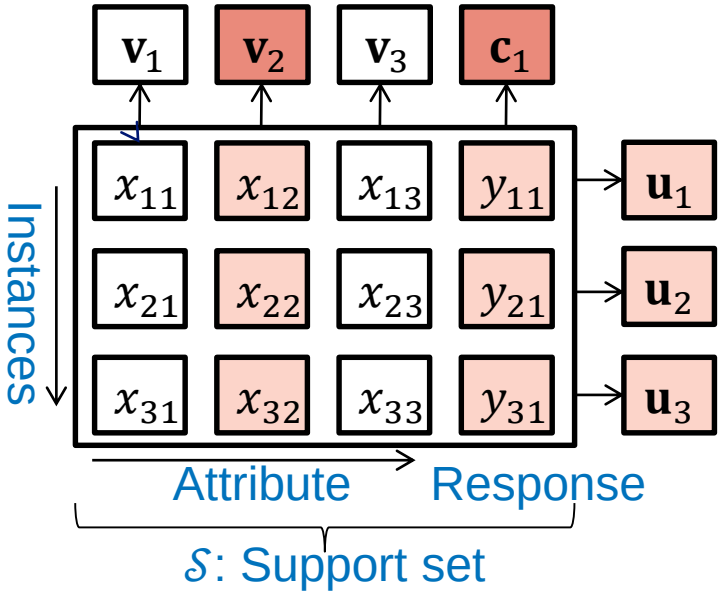
2. Calculate instance representation using attribute and response vectors and support set.

$$\mathbf{u}_i = g_{\mathrm{u}}\left(\frac{1}{I}\sum_{i=1}^{I} f_{\mathrm{u}}([\bar{\mathbf{v}}_i, x_{ni}]) + \frac{1}{J}\sum_{j=1}^{J} f_{\mathrm{u}}([\bar{\mathbf{c}}_j, y_{ni}])\right)$$

3. Calculate attribute and response vectors using instance representations and support set.

$$\mathbf{v}_i = g_{\mathrm{v}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\mathrm{v}}([\mathbf{u}_i, x_{ni}])\right), \qquad \mathbf{c}_j = g_{\mathrm{c}}\left(\frac{1}{N}\sum_{n=1}^{N} f_{\mathrm{c}}([\mathbf{u}_i, y_{nj}])\right)$$

f, g: neural networks
$[\cdot,\cdot]$: concatenation



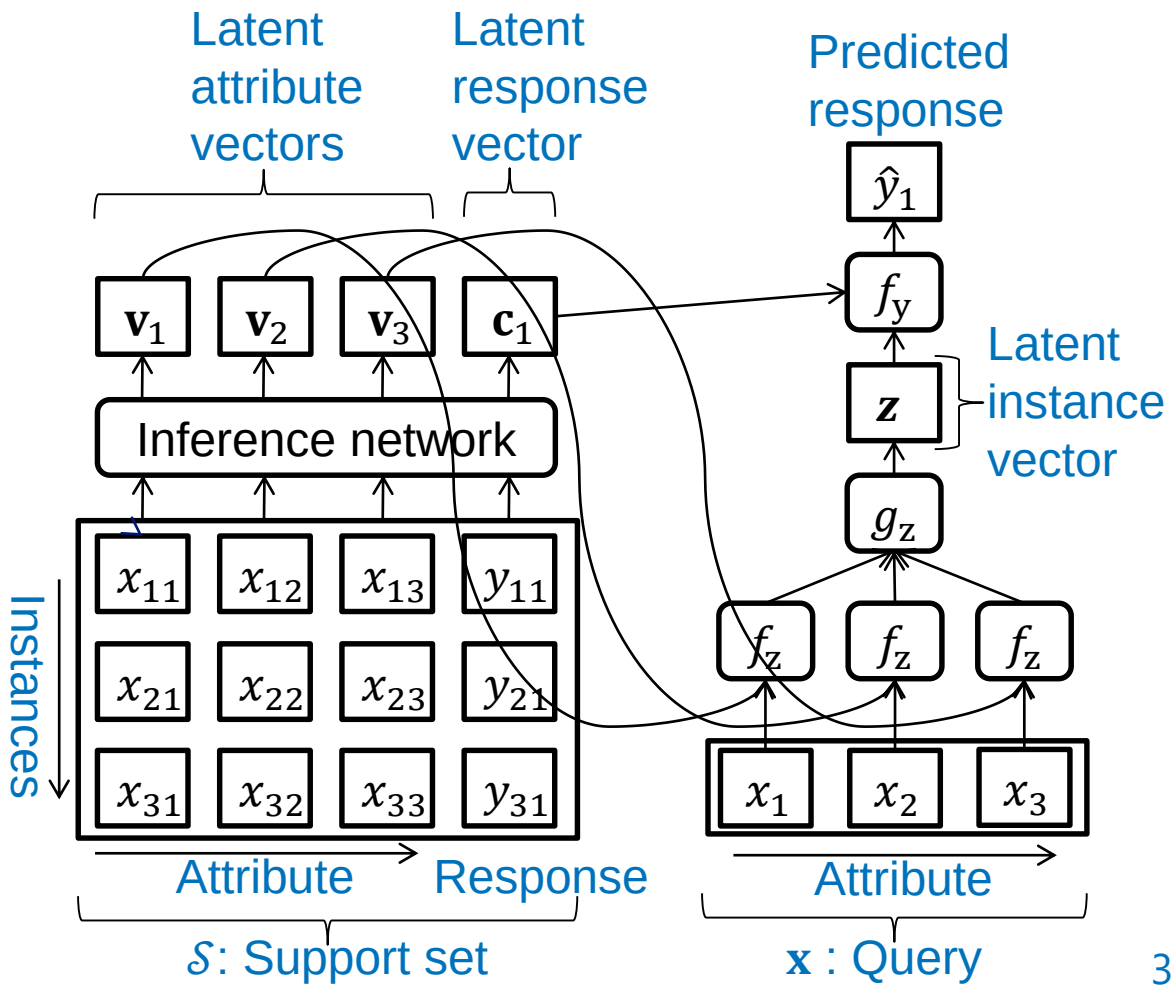
Our model: Prediction

1. Calculate latent instance vector using query and latent attribute vectors.

$$\mathbf{z}_i = g_z\left(\frac{1}{I}\sum_{i=1}^{I} f_z([\mathbf{v}_i, x_{ni}])\right)$$

2. Predict response using latent instance and response vectors

$$\hat{y}_{nj} = f_y(\mathbf{c}_j, \mathbf{z}_n)$$



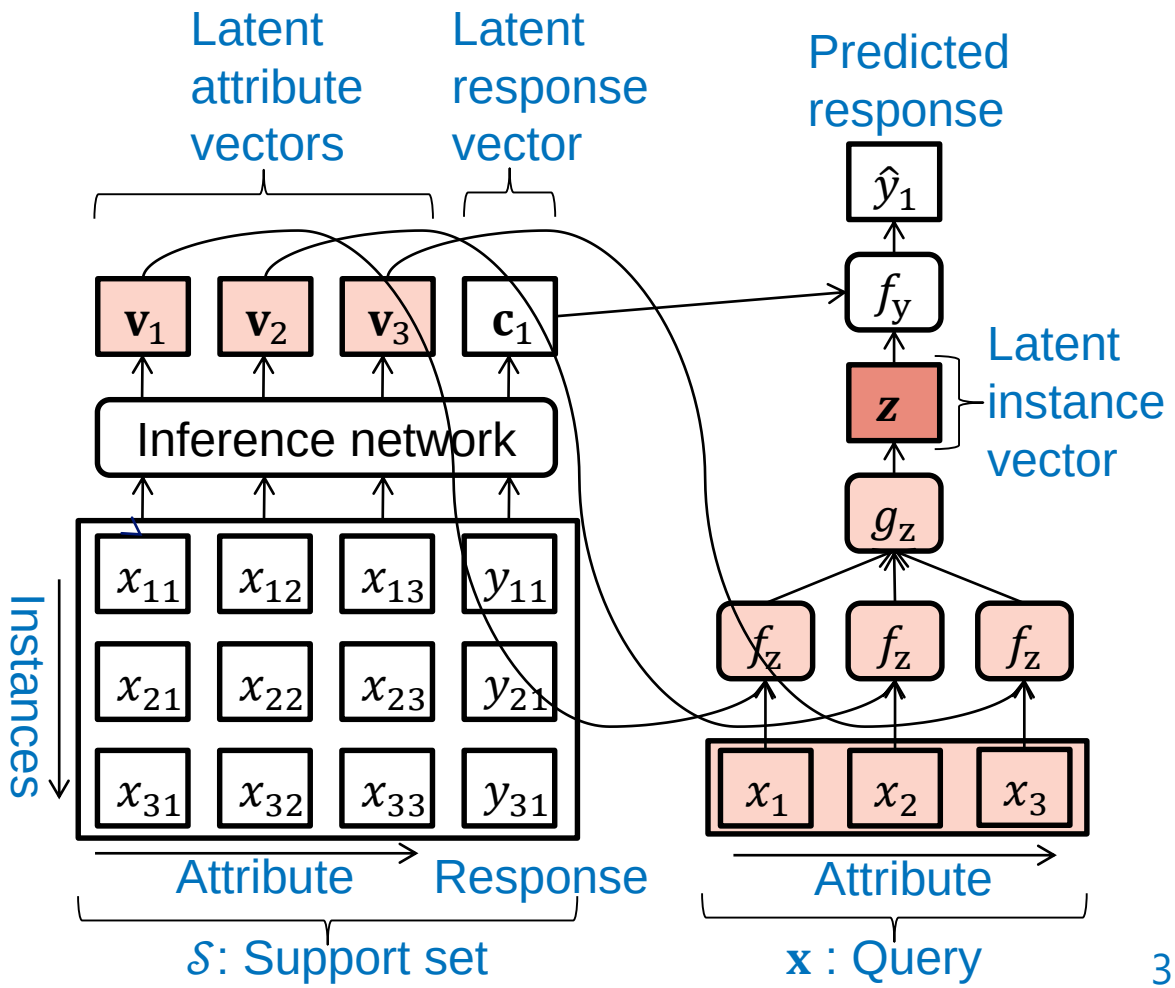
Our model: Prediction

1. Calculate latent instance vector using query and latent attribute vectors.

$$\mathbf{z}_i = g_{\mathrm{u}}\left(\frac{1}{I}\sum_{i=1}^{I} f_{\mathrm{u}}([\mathbf{v}_i, x_{ni}])\right)$$

2. Predict response using latent instance and response vectors

$$\hat{y}_{nj} = f_{\mathrm{y}}(\mathbf{c}_j, \mathbf{z}_n)$$



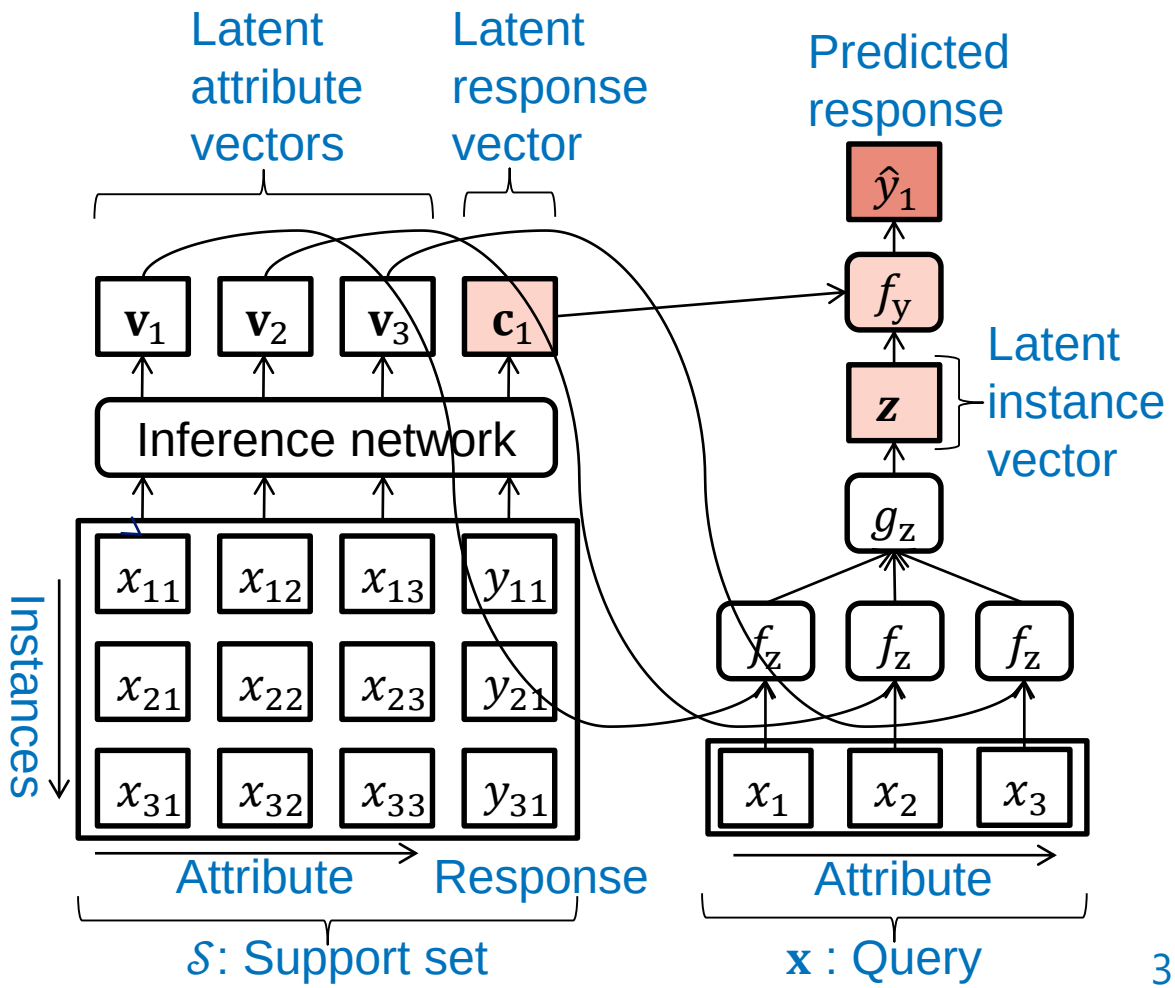
Our model: Prediction

1. Calculate latent instance vector using query and latent attribute vectors.

$$\mathbf{z}_i = g_{\mathrm{u}}\left(\frac{1}{I}\sum_{i=1}^{I} f_{\mathrm{u}}([\mathbf{v}_i, x_{ni}])\right)$$

2. Predict response using latent instance and response vectors

$$\hat{y}_{nj} = f_{\mathrm{y}}(\mathbf{c}_j, \mathbf{z}_n)$$

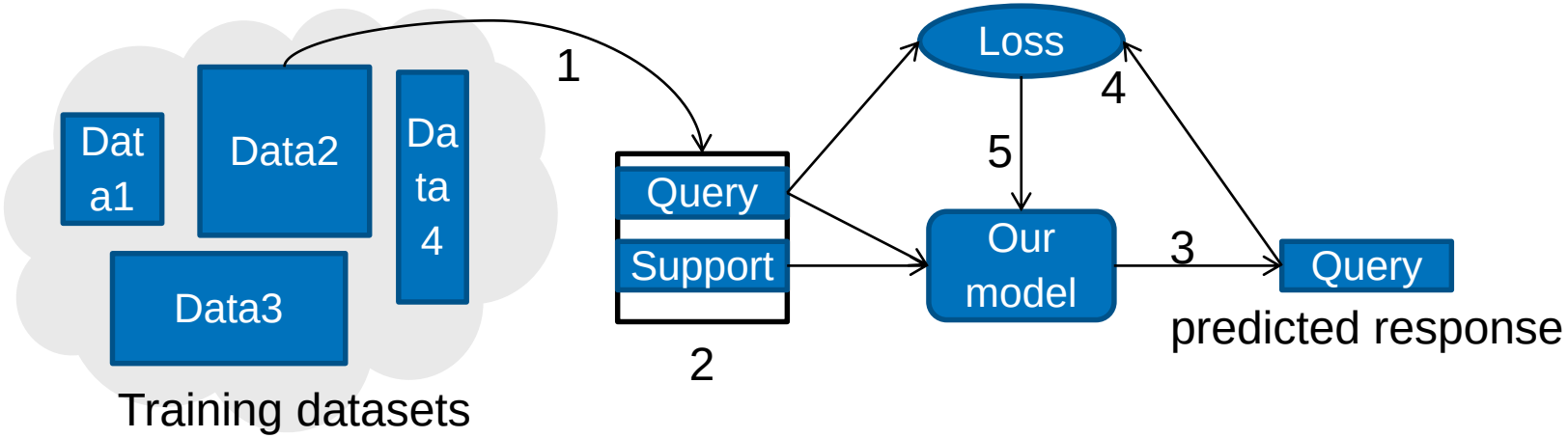


Training

Input: Data from tasks with heterogeneous attribute spaces

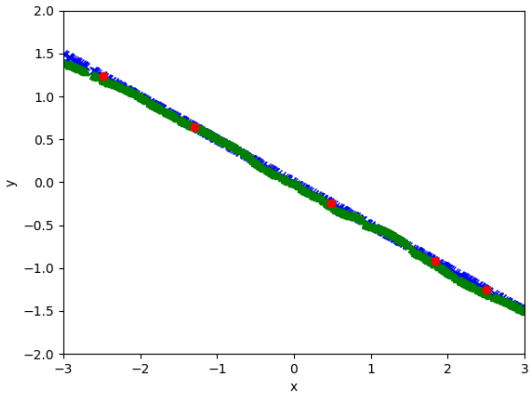
For each training epoch:

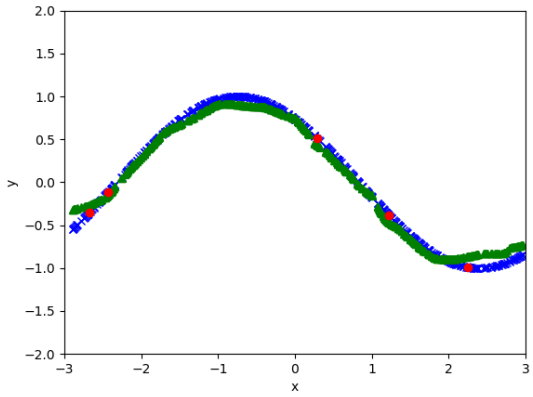
1. Randomly sample a task
2. Randomly generate support and query sets
3. Predict query set by our model using support set
4. Calculate loss between predicted and true query sets
5. Update our model by stochastic gradient descent

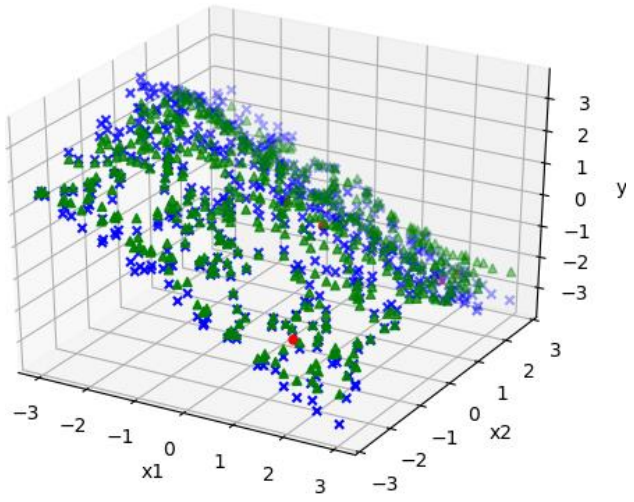


Experiments with synthetic data

- Data
  - 10,000 tasks generated by 1-dimensional linear and nonlinear models, and 2-dimensional nonlinear model with random parameters.
- Results
  - Our method appropriately learned unseen tasks with different dimensionality using 5 labeled instances.







Red: Labeled instances, Blue: True response, Green: Estimated response

Experiments with OpenML data

- Data
  - OpenML: open online platform for machine learning
  - 59 tasks with various attributes: #instances 10-300, #attributes 2-30
  - #labeled instances per task: 3
- Results
  - The proposed method achieved the lowest error compared with existing meta-learning and regression methods.

Method	MSE	Method	MSE	Method	MSE
Ours	**0.788 ± 0.011**	NP+FT	0.907 ± 0.013	KR	0.828 ± 0.021
DS	0.896 ± 0.011	NP+MAML	0.845 ± 0.012	GP	1.113 ± 0.112
DS+FT	0.887 ± 0.011	Ridge	1.179 ± 0.038	NN	1.107 ± 0.028
DS+MAML	0.854 ± 0.011	Lasso	1.281 ± 0.024	Mean	1.347 ± 0.025
NP	0.845 ± 0.012	BR	1.544 ± 0.134		

DS (deep set), FT (finetuning), MAML(model-agnostic meta-learning)
NP (conditional neural process) were trained using 59 tasks.
Ridge, Lasso, BR, KR, GP, NN (neural net), Mean were trained using target tasks.

Conclusion

- We proposed a neural network-based meta-learning method that learns from multiple tasks with different attribute spaces, and predicts a response given a few instances in unseen tasks.
- Our work is an important step for learning from a wide variety of datasets, and use the learned knowledge for new tasks.

- Future work: use different types of neural networks for calculating latent vectors, e.g., attentions.

おわりに

- メタ学習により深層学習の適用範囲を拡大
  - 多様なタスクから学習の仕方を学習
  - 少数データしかない場合での深層学習の性能を向上させる
- 代表的メタ学習手法
  - Gradient-based (Model-agnostic meta-learning)
    - Finetuneしたときに期待テスト損失が小さくなるように初期値を学習
  - Black-box adaptation (Neural process)
    - 各タスクへの適合をニューラルネットでモデル化
  - Model-based (Prototypical network)
    - 適合とその勾配計算が容易なモデルを利用
- メタ学習の研究がやりやすい環境
  - 多様なデータが収集可能
  - 自動微分ライブラリでメタ学習の実装も容易に
- 多様な場面でメタ学習が活用できる