Learning with Whom to Share in Multi-task Feature Learning

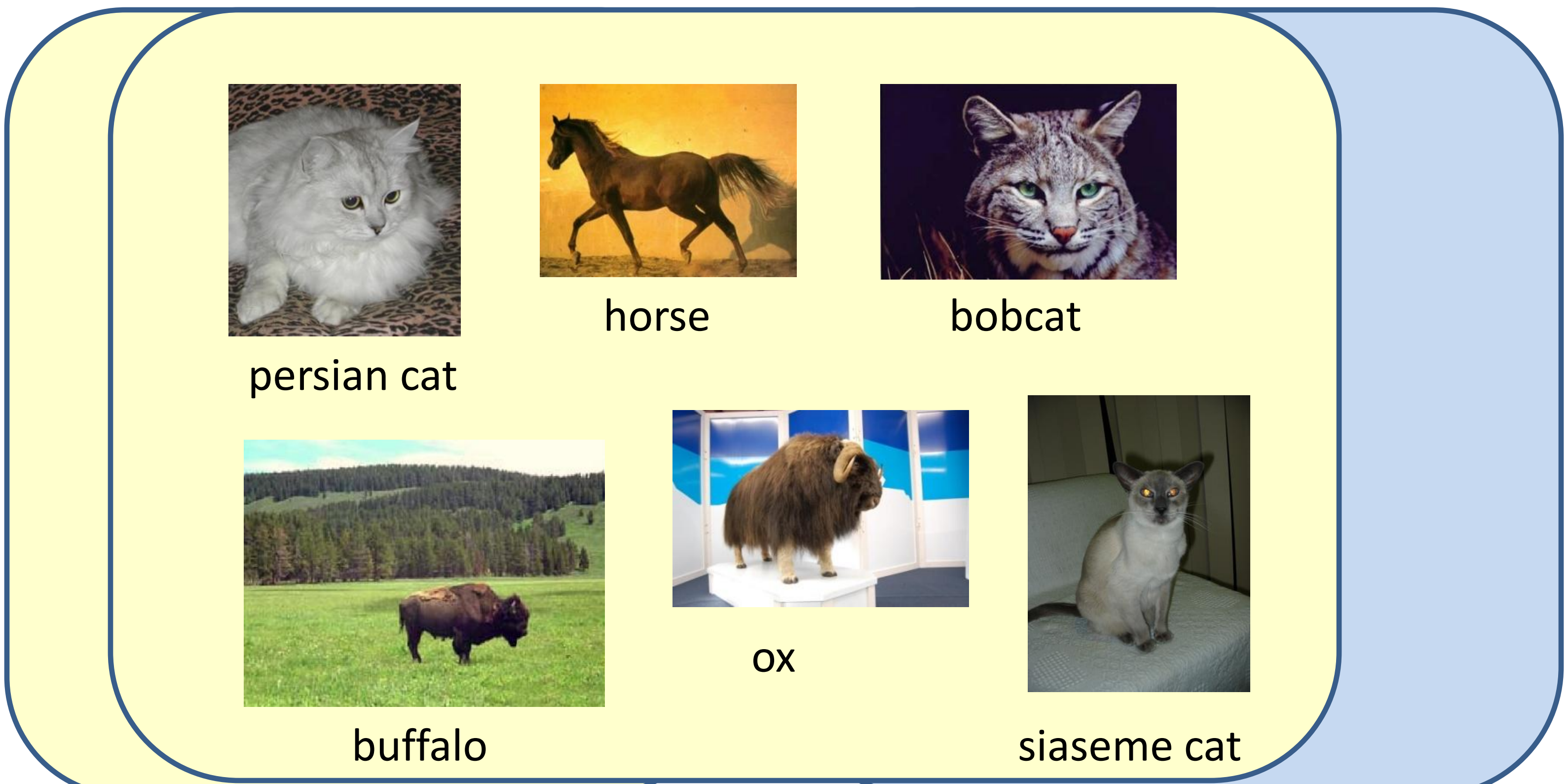
Zhuoliang Kang
Department of Computer Science
University of Southern California

[Joint work with Kristen Grauman @ U. of Texas Austin and Fei Sha @ U. of Southern California]

This is a talk about

- ***Multi-task learning.***
- ***Automatic tasks grouping.***

E.g. multiple animal recognition tasks.

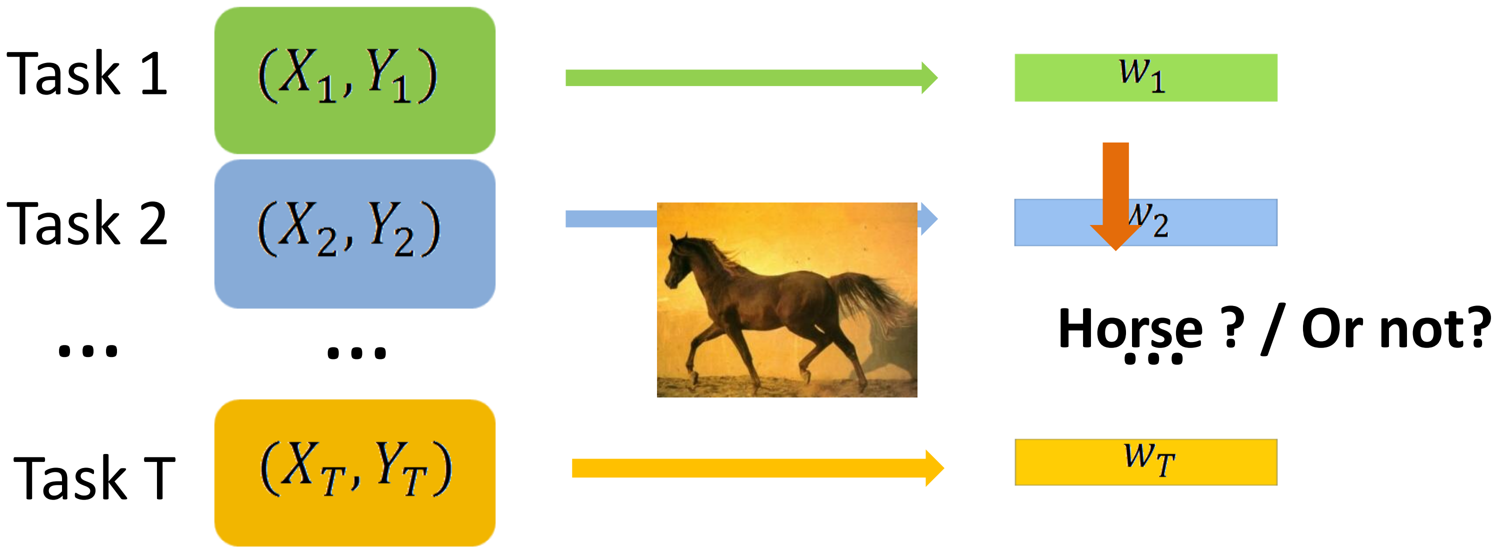


Outline

- **Background**
  - What is multi-task learning
- **Motivation**
  - Why we want to group tasks
- **Algorithms**
  - How to discover the grouping
- **Empirical results**
  - Validate our approach
- **Conclusion**
  - Summary
  - Future work

Supervised learning

- Given training data and label
  - Learn parameters for future prediction.
- Given ***multiple*** tasks.
  - Learn parameters ***independently***.

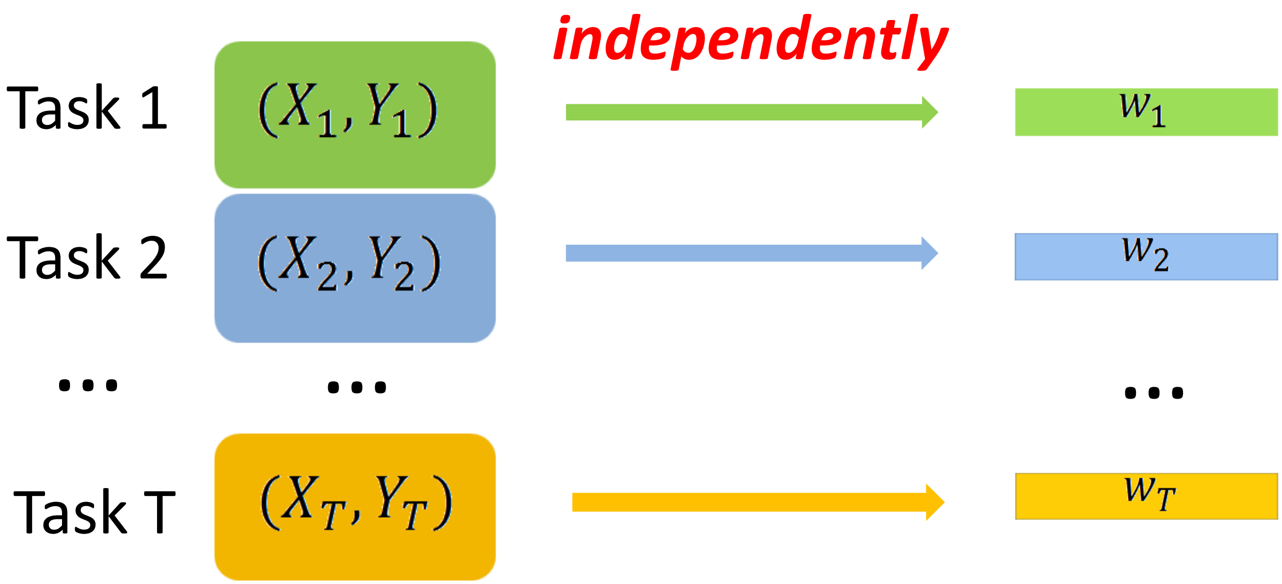


Regularization based framework

For ***each task***, solve an optimization problem

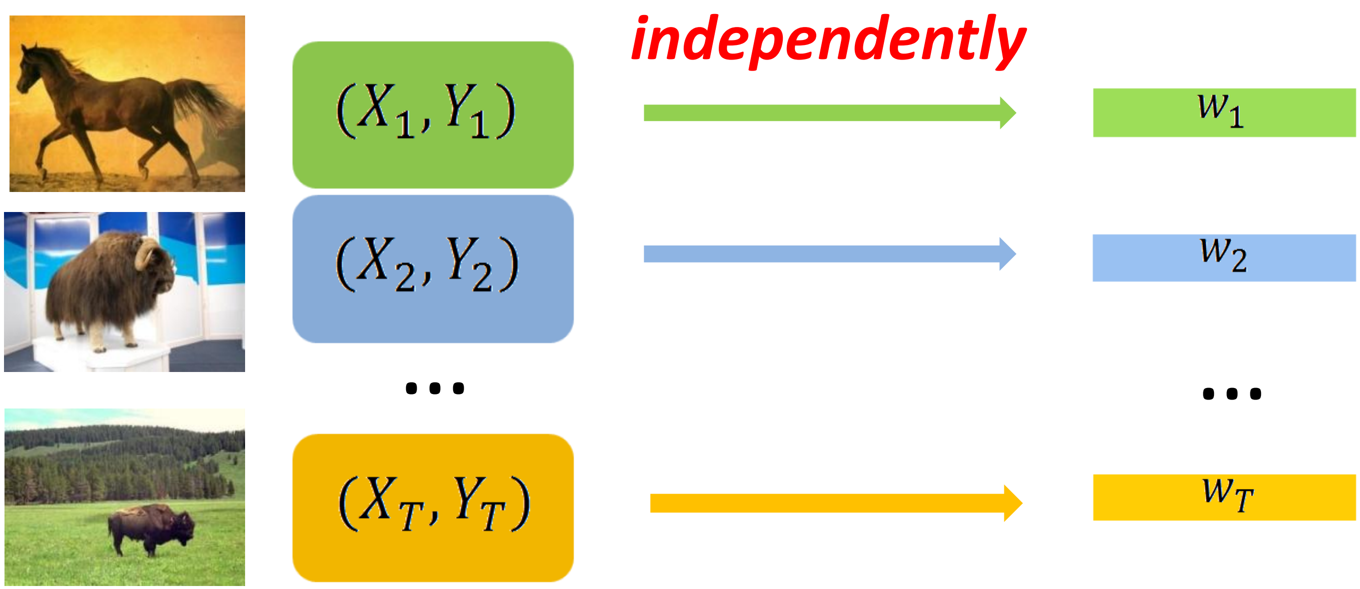
Balance empirical risk and model complexity

$$\min_{w_t} \quad loss(w_t, X_t, Y_t) + R(w_t)$$



How to solve a group of related tasks?

- Example
  - Recognizing similar animals.
  - Recognizing similar handwritten digits.
- We can do better than learning independently.



Multi-task learning (MTL)

- Main idea
  - Learn multiple tasks ***jointly***.
  - Take the advantage of ***relatedness***.
- Benefits
  - Improve ***generalization*** performance.
  - Require ***less*** training data.

Related work

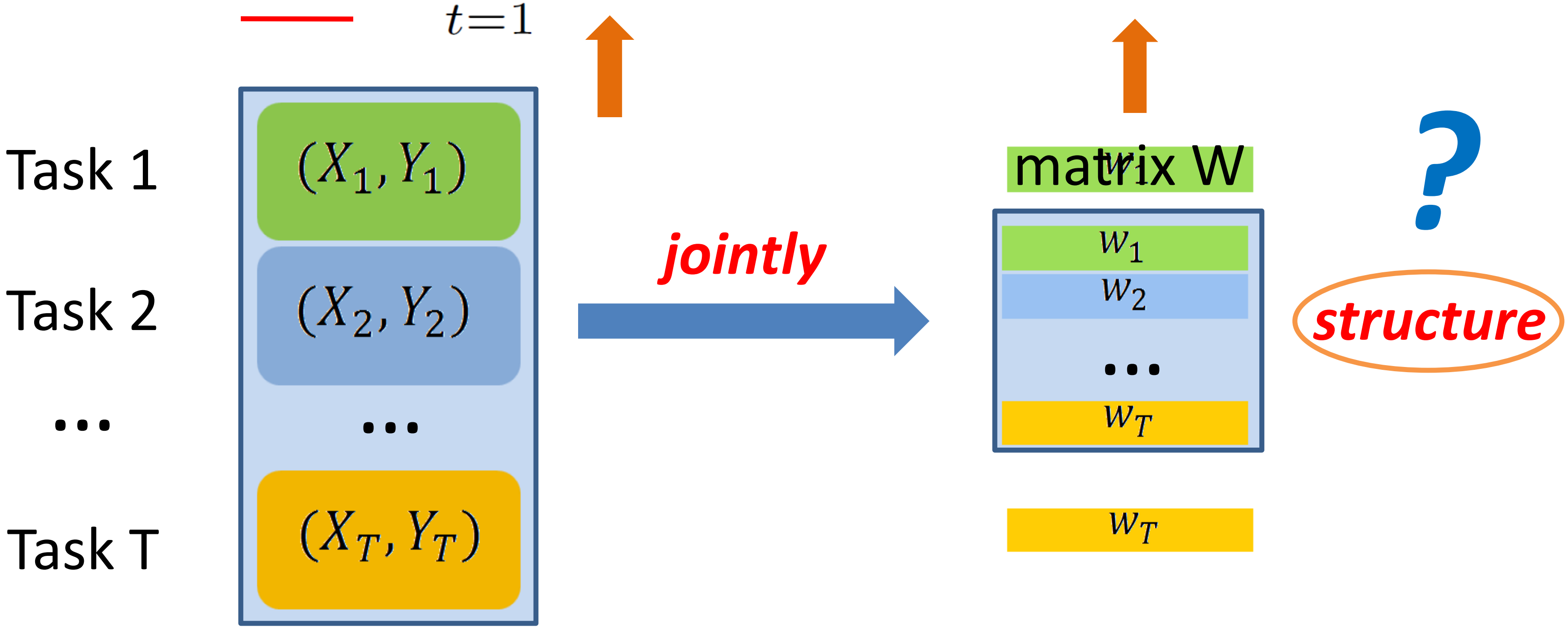
[Caruana, 97. Bakker and Heskes, 03. Evgeniou, et al. 04. Ando and Zhang. 05. Yu, et al., 05. Lee, et al., 07. Argyriou, et al. 08, Daumé, 09. Parameswaran, S. and Weinberger, K.Q. 2009. …]

Regularization based approach

Solve a joint optimization problem ***for all tasks***.

Balance between ***total empirical risk*** and ***relatedness***.

$$\min_{W} \sum_{t=1}^{T} loss(w_t, X_t, Y_t) + R(W)$$



[Evgeniou, et al. 2004. Parameswaran, S. and Weinberger, K.Q. 2009. Zhang, et al. 2010. ...]

Alternatives to regularization based MTL

- Share a common layer in Neural Network
  - R. Caruana, 1997.
  - B. Bakker and T. Heskes, 2003.
- Share common priors
  - Yu, et al., 2005.
  - Lee, et al., 2007.
  - E. Bonilla, et al. 2008
  - Daumé, III, Hal. 2009.
- etc …

Multi-task feature learning (MTFL)

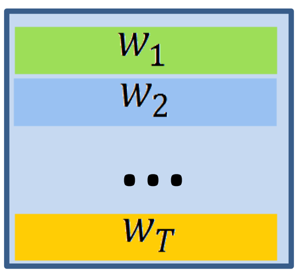
[Argyriou, et al. 2008.]

Task-relatedness

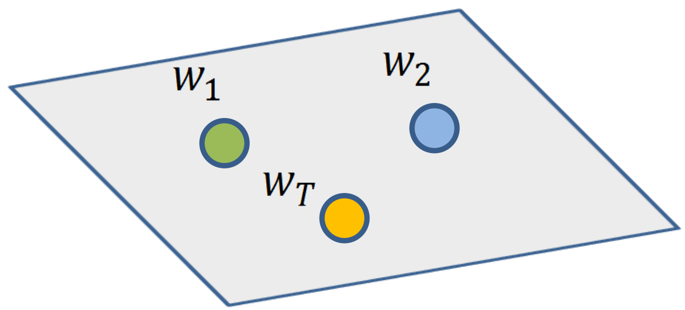
- Parameters lie on ***a common low-dimensional subspace.***
- Or equivalently, models share ***a common feature subspace.***

structure

matrix W

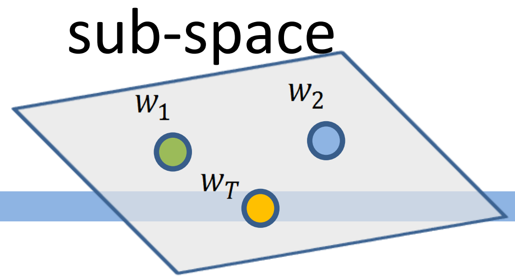


subspace



Structural constraint on W: **low rank**

Low-rank Regularization



Rank: number of none-zero singular values **(non-convex)**

$$W = U \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_D \end{pmatrix} V^T \qquad \mathrm{rank}(W) = \sum_d 1[\sigma_d \neq 0]$$

Singular Value Decomposition

Convex relaxation

- ***Trace norm***: $\boldsymbol{L_1}$-norm of singular values **(convex)**

$$\|W\|_{tr} = \sum_d |\sigma_d|$$

Outline

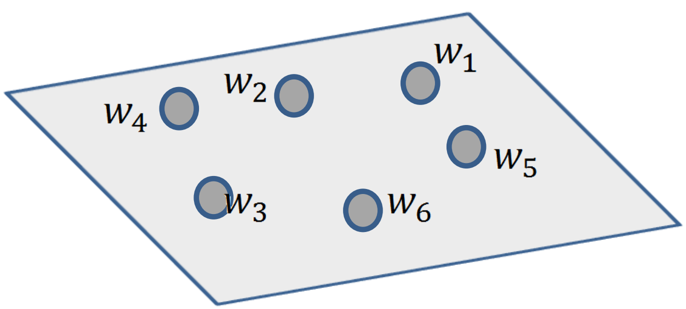
- **Background**
  - What is multi-task learning
- **Motivation**
  - Why we want to group tasks
- **Algorithms**
  - How to discover the grouping
- **Empirical results**
  - Validate our approach
- **Conclusion**
  - Summary
  - Future work

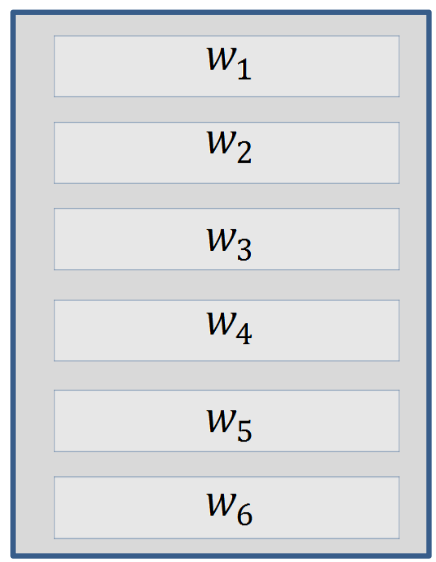
Motivation

Existing work on multi-task feature learning

- *single regularization term*
- *All* tasks are related.

$$\min_{W} \sum_{t=1}^{T} loss(w_t, X_t, Y_t) + \underline{\lambda \|W\|_{tr}^2}$$





matrix W

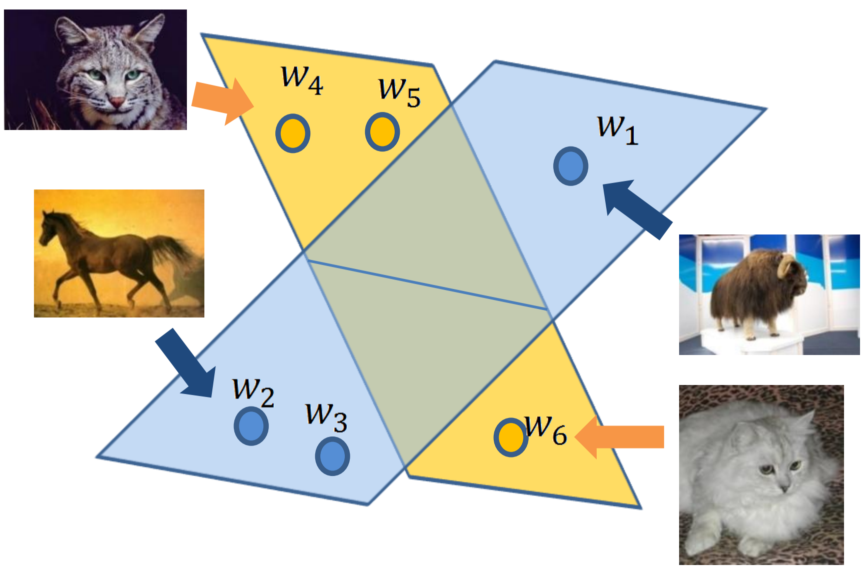
[Argyriou, et al. 2008.]

Motivation

When models are in ***mixture*** of subspaces

$$\min_{W} \sum_{t=1}^{T} loss(w_t, X_t, Y_t) + \lambda \|W\|_{tr}^2$$

- Suboptimal to force with one regularizer
- Ex: synthetic data (later in the talk)



matrix W

Motivation

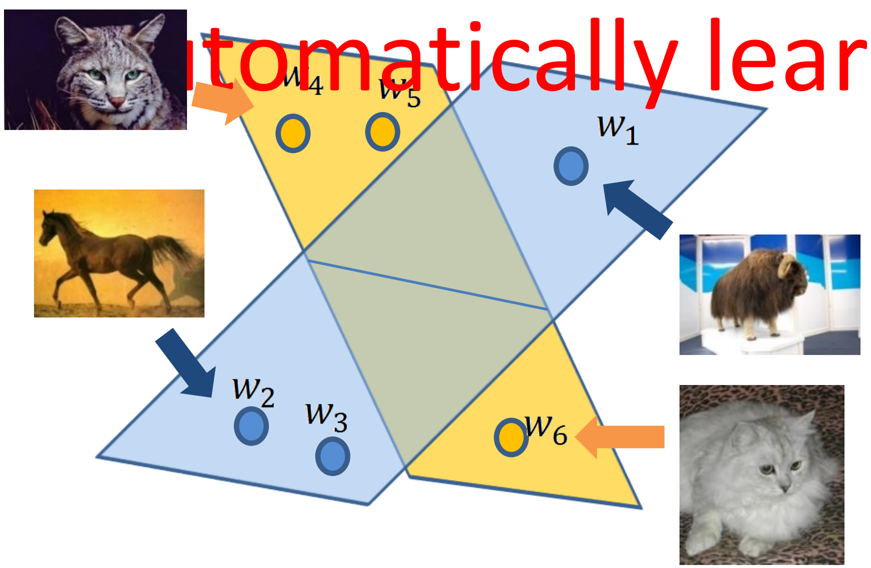
When groups are given

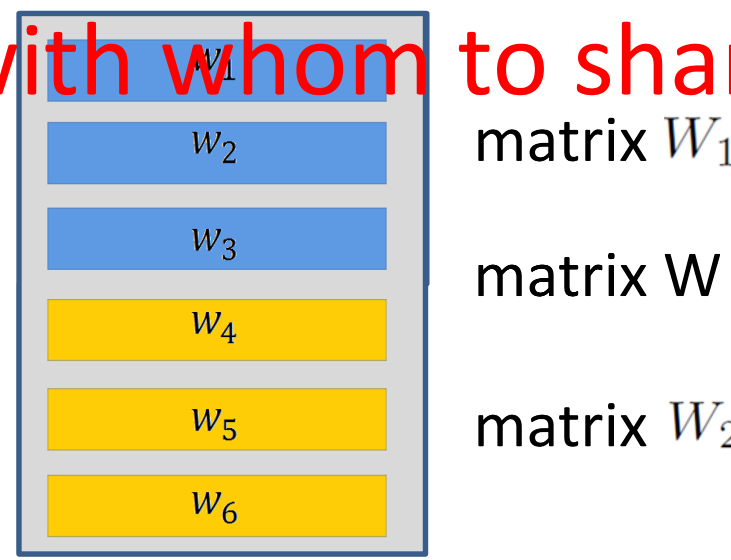
$$\min_{W_1, W_2} \sum_{t=1}^{T} loss(w_t, X_t, Y_t) + \underline{\lambda \|W_1\|_{tr}^2 + \lambda \|W_2\|_{tr}^2}$$

Desiderata

Regularize each group ***separately***.

tomatically learn with whom to share





Outline

- **Background**
  - What is multi-task learning
- **Motivation**
  - Why we want to group tasks
- **Algorithms**
  - How to discover the grouping
- **Empirical results**
  - Validate our approach
- **Conclusion**
  - Summary
  - Future work

Step1: use indicator matrix

Reformulate with task group assignment matrix **Q**

tasks

$$Q$$

groups

$$\begin{array}{|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 1 \\ \hline \end{array}$$

$$\|W_1\|_{tr}^2 = \|Q_1 W\|_{tr}^2$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{Q_1} \times \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix}}_{W} = \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ 0 \end{bmatrix}}_{Q_1 W}$$

matrix W_1

Integer programming for Inferring with whom to share

Re-formulate with matrix Q

- ***Integer*** constraint
- ***Hard*** group assignment

$$\min_{W,Q} \sum_{t=1}^{T} loss(w_t, X_t, Y_t)$$

$$+ \lambda \|Q_1 W\|_{tr}^2 + \lambda \|Q_2 W\|_{tr}^2$$

$$\text{s.t} \quad q_{gt} \in \{0, 1\}$$

$$Q_1 + Q_2 = I$$

$$Q = \begin{array}{cc} & \text{tasks} \\ \text{groups} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad Q_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 2: relax the constraint

- Approach 1:
  convex relaxation
  - *Continuous* constraint
  - Convex but *fractional* solutions
- Approach 2:
  non-convex relaxation
  - Use *square root of Q:*
    non-convex but *integer* solutions

$$\min_{W,Q} \sum_{t=1}^{T} loss(w_t, X_t, Y_t)$$

$$+ \lambda \| \underline{\sqrt{Q_1}} W \|_{tr}^2 + \lambda \| \underline{\sqrt{Q_2}} W \|_{tr}^2$$

$$\text{s.t} \quad \underline{0 \leq q_{gt} \leq 1}$$

$$Q_1 + Q_2 = I$$

Integer solutions guaranteed

Theorem 1. *Let $\{\boldsymbol{Q}_g^*\}$ be either the solution or a local optimum to the following optimization,*

$$\begin{aligned} \min \quad & T(\boldsymbol{Q}) = \sum_g \underline{\|\boldsymbol{W}\sqrt{\boldsymbol{Q}_g}\|_*^2} \\ \text{s.t} \quad & \sum_g \boldsymbol{Q}_g = \boldsymbol{I} \text{ with } \underline{0 \le q_{gt} \le 1} \end{aligned} \tag{9}$$

then either one of the following is true: i) $\{\boldsymbol{Q}_g^*\}$ is binary; ii) there exists another binary $\{\boldsymbol{Q}'_g\}$ such that $T(\boldsymbol{Q}^*) = T(\boldsymbol{Q}')$.

Proofs.

$$T(Q) = \sum_g \min_{\boldsymbol{\Omega}_g} \mathsf{Trace}\left[\boldsymbol{\Omega}_g^{-1}\boldsymbol{W}\sqrt{\boldsymbol{Q}_g}\sqrt{\boldsymbol{Q}_g}^{\mathrm{T}}\boldsymbol{W}^{\mathrm{T}}\right] \quad (1)$$

where $\boldsymbol{\Omega}_g$ is constrained to be positive definitive. Furthermore, $\mathsf{Trace}[\boldsymbol{\Omega}_g] = 1$. Let $\boldsymbol{\Psi}_g = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Omega}_g^{-1}\boldsymbol{W}$, we have

$$T(Q) = \min \sum_g \mathsf{Trace}\left[\boldsymbol{\Psi}_g\boldsymbol{Q}_g\right] \quad (2)$$

Since $\boldsymbol{Q}_g$ is a diagonal matrix, we have immediately

$$T(Q) = \min \sum_g \sum_t \psi_{tt}^g q_{gt} \quad (3)$$

Numerical Optimization

Optimize W and Q ***iteratively***

- Fix Q, update W
  - *For each group, we solve*

$$\min \sum_{t:q_{gt}=1} \ell(\mathcal{D}_t; \boldsymbol{w}_t) + \gamma \|\boldsymbol{W}_g\|_*^2$$

  - *Use existing algorithm*

cf: Argyriou, et al. ***Convex multi-task feature learning****. MLJ 2008.*

Numerical Optimization

Optimize W and Q ***iteratively***

- Fix W, update Q
  - *Use gradient descent*

$$\min_{Q} \quad \sum_{g} \| \sqrt{Q_g} W \|_{tr}^2$$

$$\text{s.t} \quad \sum_{g} Q_g = I \ \text{ with } 0 \leq q_{gt} \leq 1$$

  - *Remove constraints*
    - *by re-parameterization:* ***α is unconstrained***

$$q_{gt} = \frac{e^{\alpha_{gt}}}{\sum_{g=1}^{G} e^{\alpha_{gt}}}$$

(soft assigning)

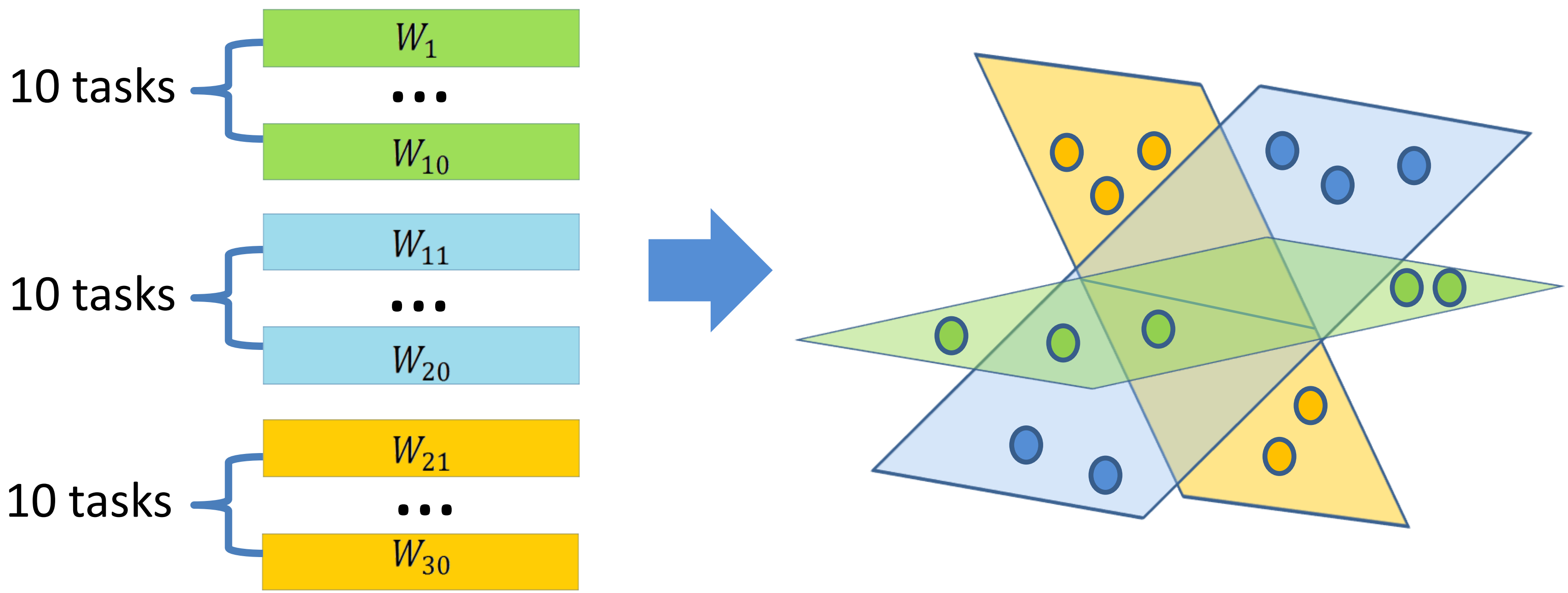
Outline

- **Background**
  - What is multi-task learning
- **Motivation**
  - Why we want to group tasks
- **Algorithms**
  - How to discover the grouping
- **Empirical results**
  - Validate our approach
- **Conclusion**
  - Summary
  - Future work

Results: synthetic data

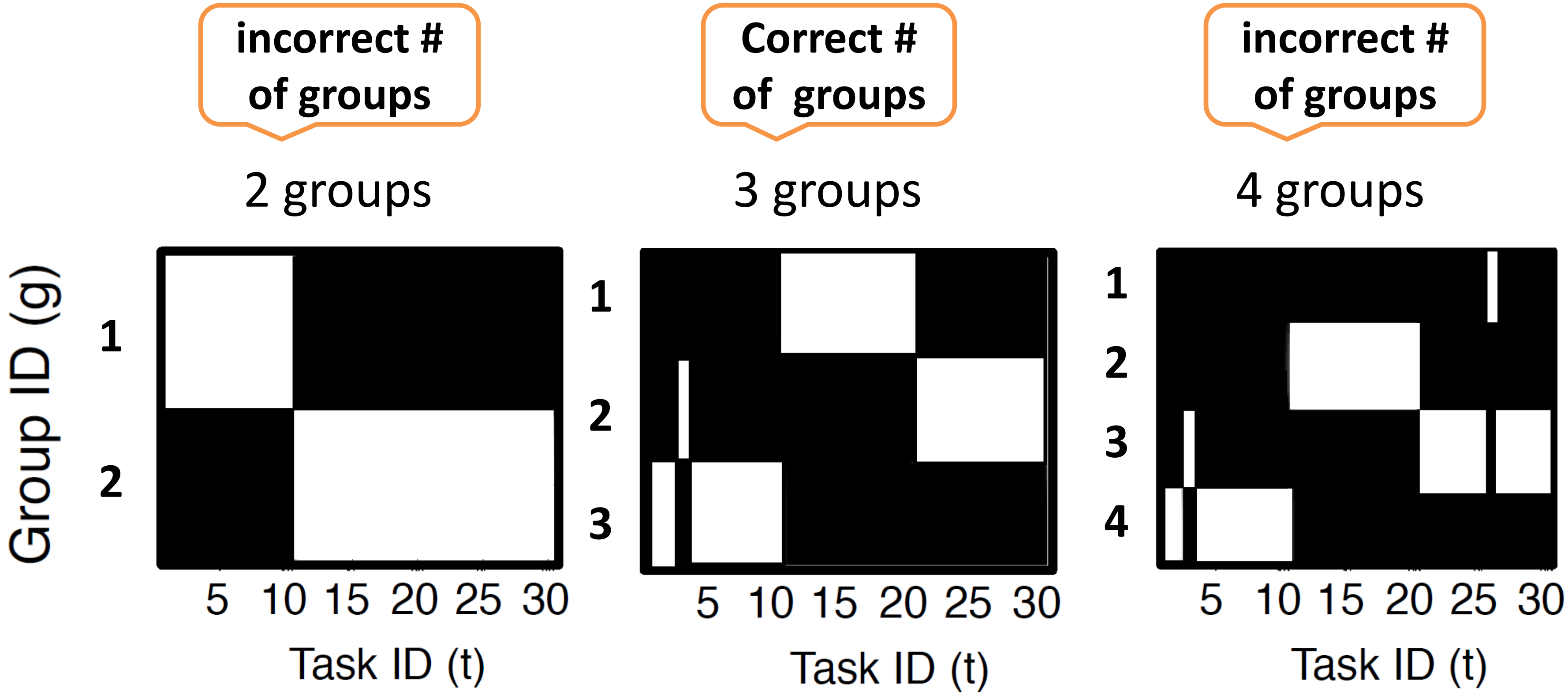
Setup

– We have 30 tasks with 3 groups (10 tasks per group).
– Each task is a regression problem.
– Tasks in the same group ***use the same feature.***



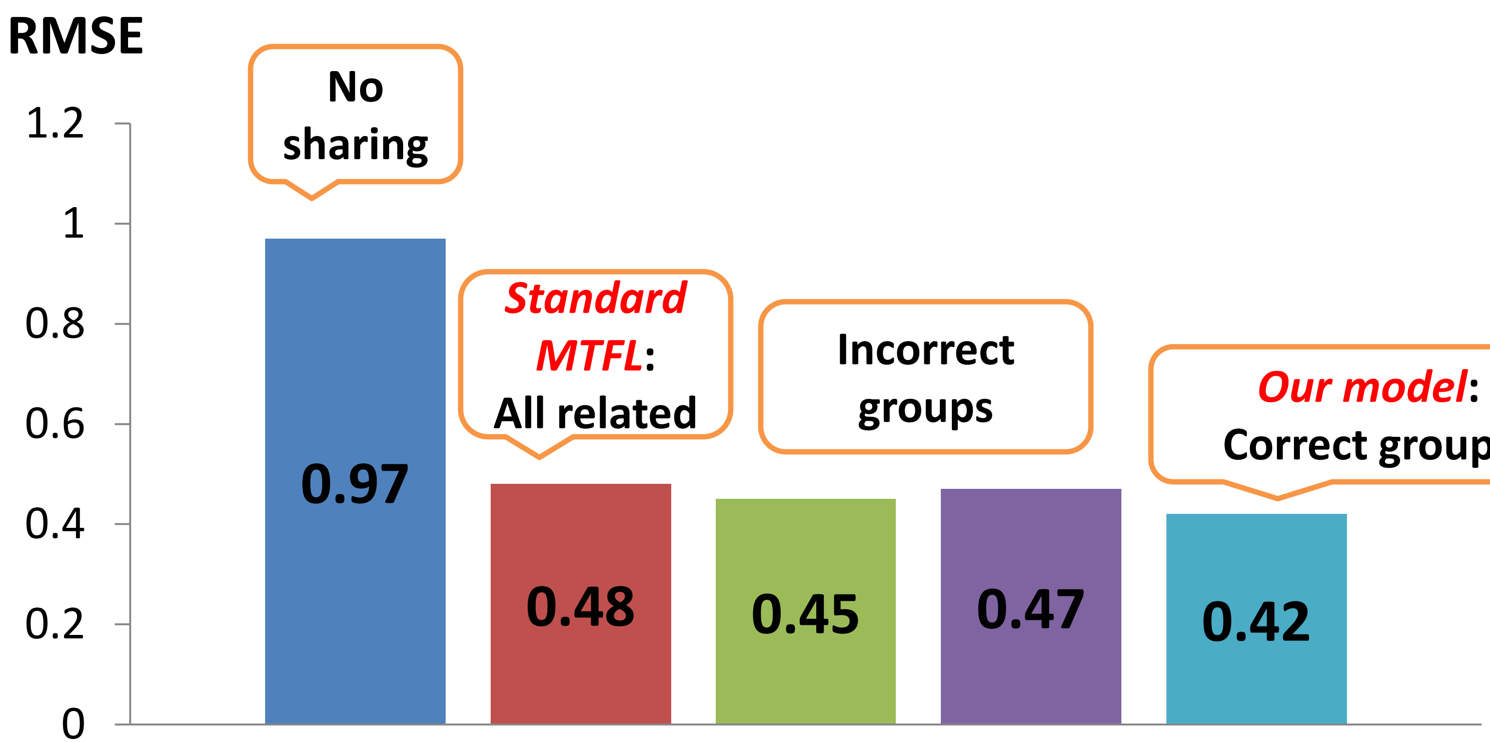
Grouping results of the tasks

- Specify the correct number of groups
- Identify the ***correct grouping***

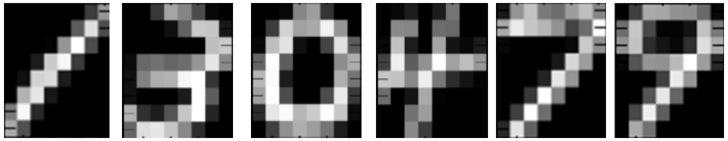


Also improve generalization

- Measure **average *root-mean-square error***.
- Obtain best performance with ***correct grouping***.

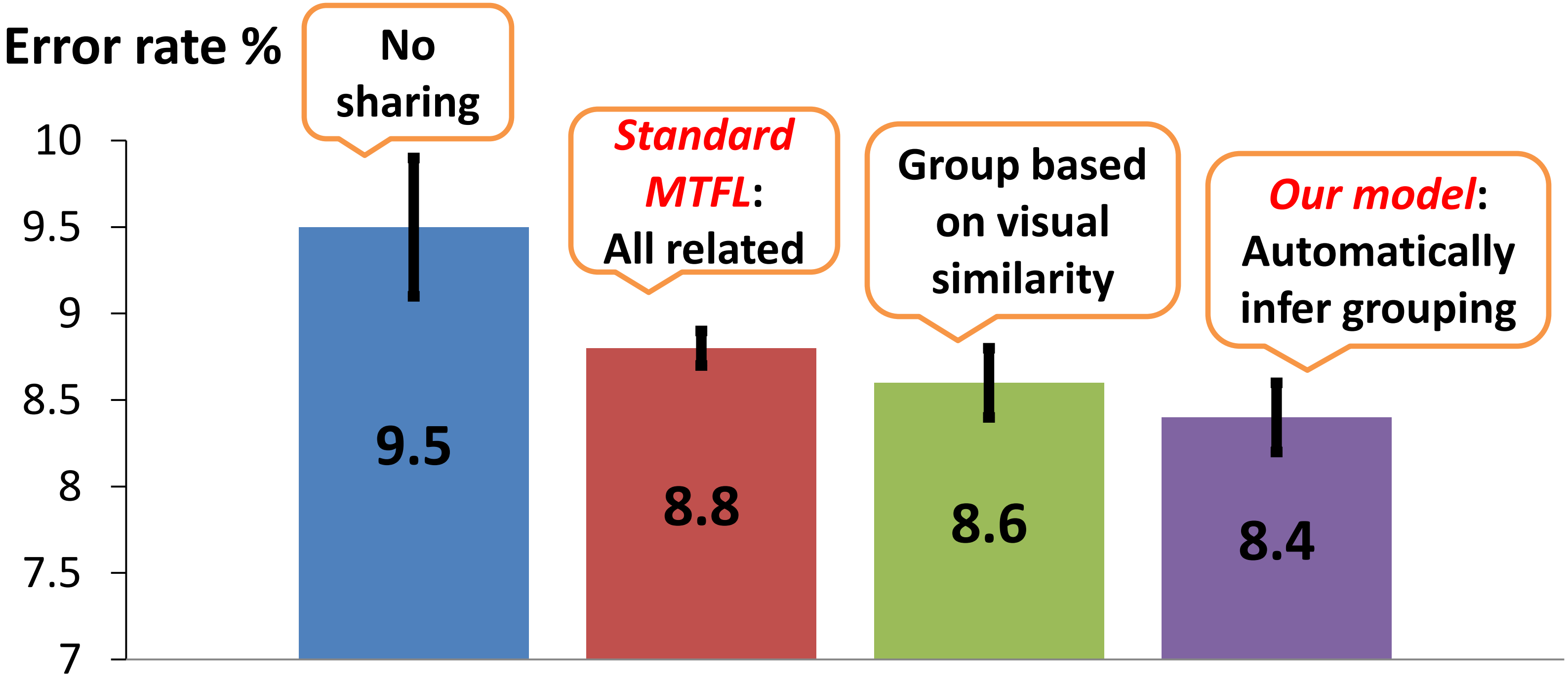


Results: USPS



Setup

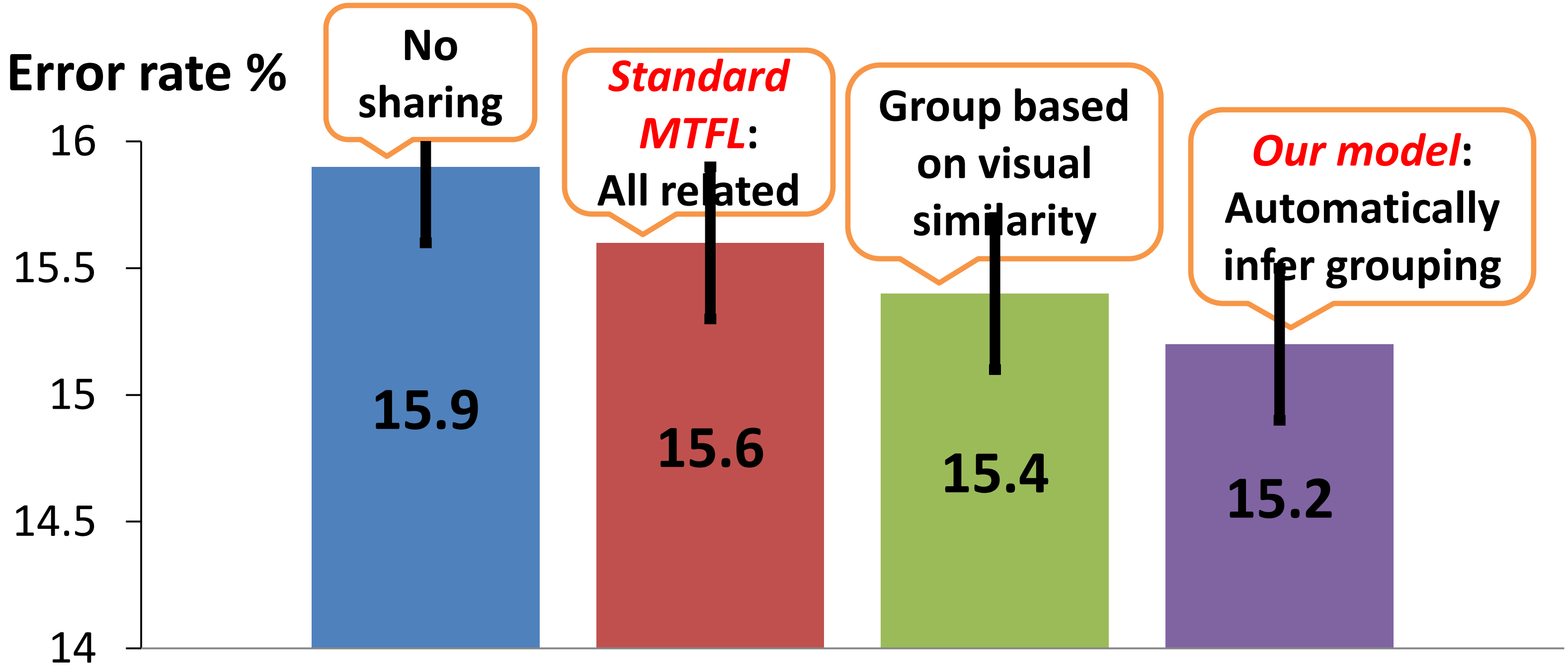
- 10-way classification on images of 10 handwritten digits
- 1000 training data
- Classifier: binary logistic regression



Results: MNIST

Setup

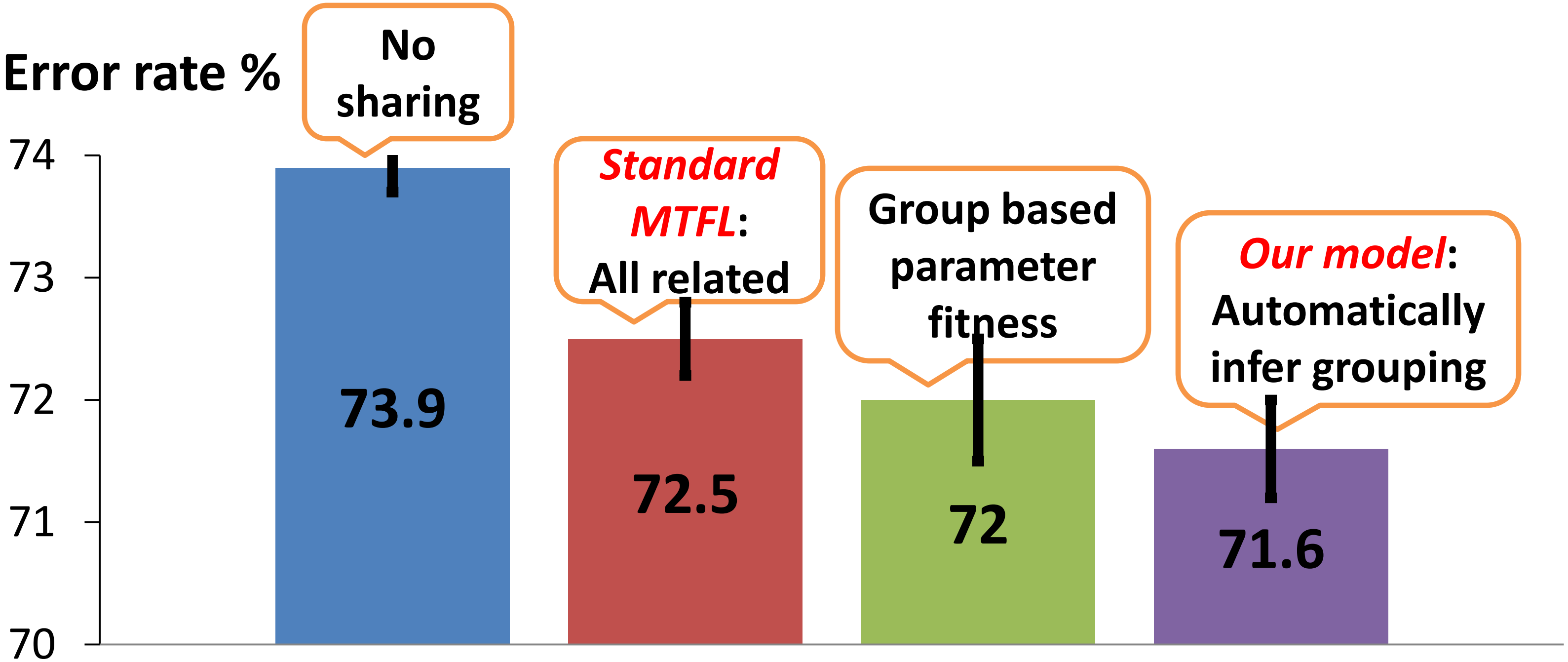
- 10-way classification on images of 10 handwritten digits
- 1000 training data
- Classifier: binary logistic regression



Results: recognize animals

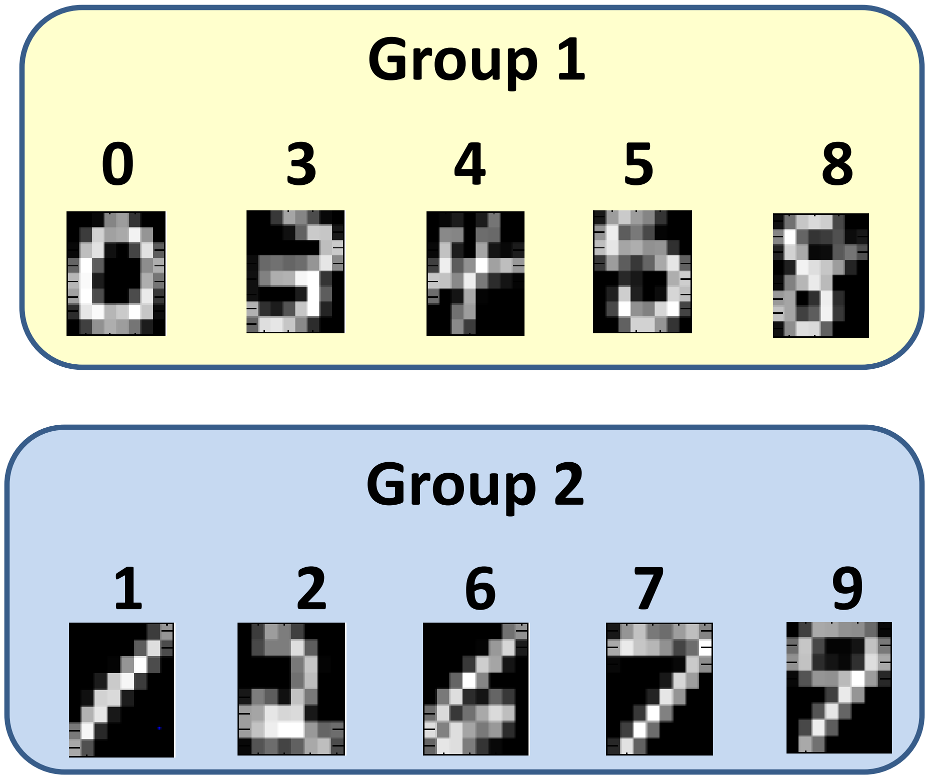
Setup

- Data set: Animal with Attributes (images of 20 classes)
- 1000 training data; Features: SIFT
- Classifier: binary logistic regression

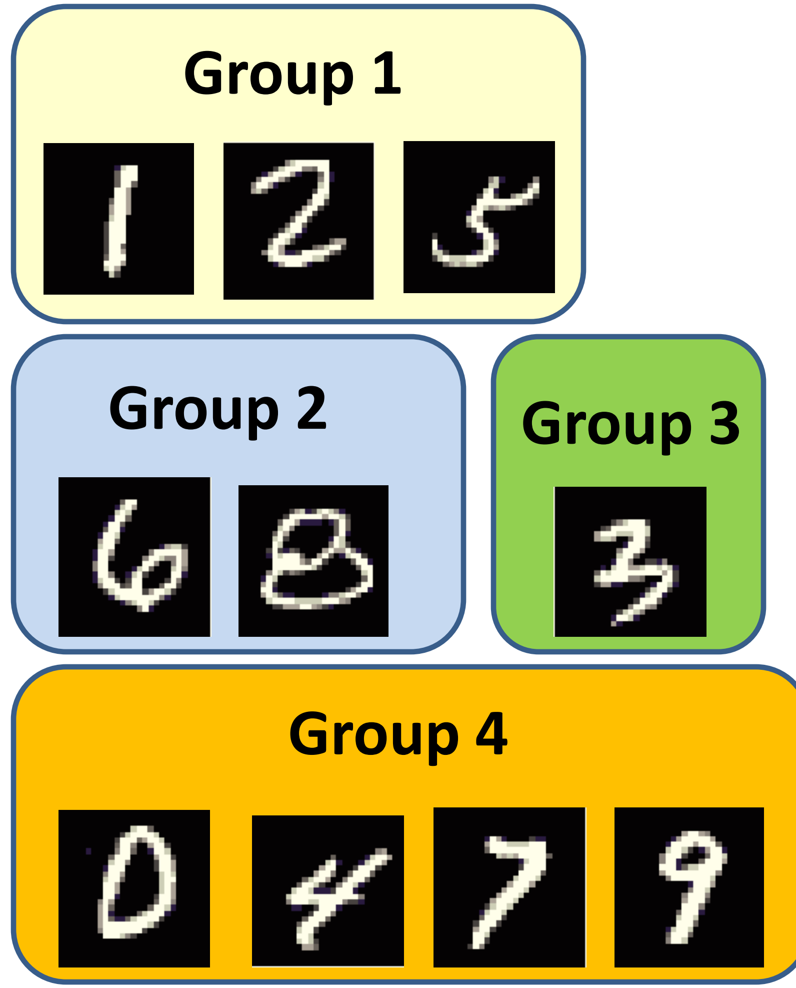


Grouping results on digits data

USPS: 10 digits



MNIST: 10 digits



Grouping results on animal data

Animal with Attributes data set: 20 classes are used

Group 1

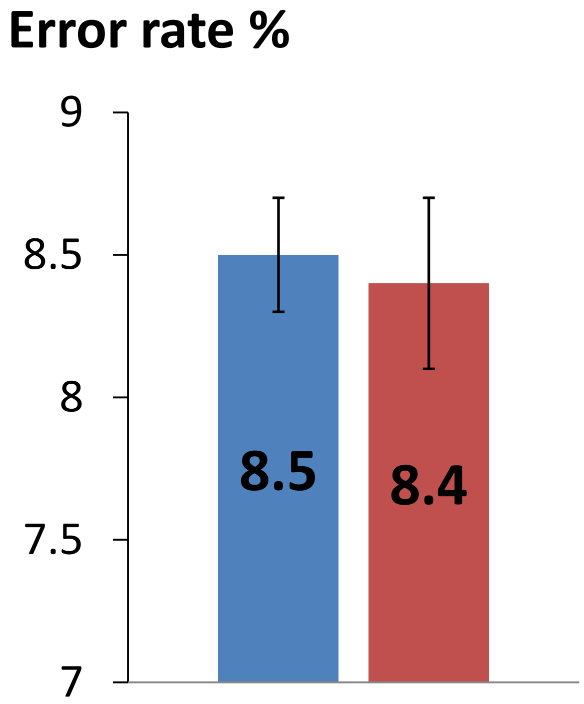
Group 2

Group 3

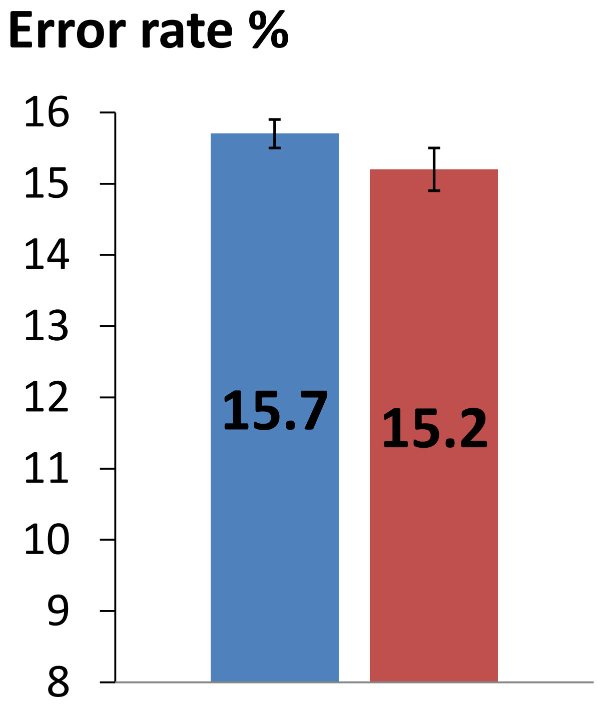
Comparison with other methods

- **Online learning of sets of kernels** [Argyiou, et al. ECML 2008]
- **Our method**

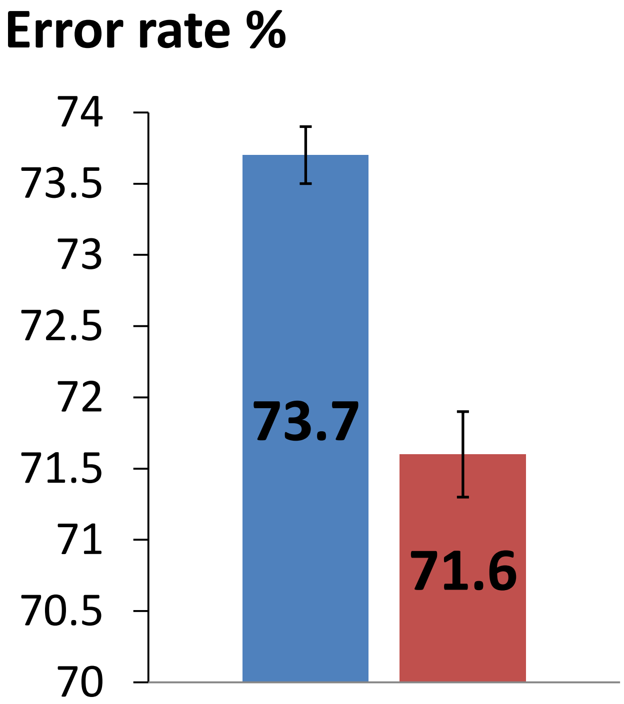
USPS



MNIST



Animals



Conclusions

Multi-task feature learning

- Beneficial to **identify related tasks.**
- **Sharing with related tasks** instead all of them helps.
- Effective joint inference of **shared structure** and **model parameters**.

Future work

- More complex structures.
- Investigation of the grouping robustness.
- Transfer for new tasks.

Transfer for new tasks

- Q and W of old tasks is enough
- keep Q and W of old tasks fixed
- update Q and W of new tasks