# Covariate-shift Robust Adaptive Transfer Learning for High-dimensional Regression

Zelin He

Dept. of Statistics,
The Pennsylvania State University

# Outline

# Outline

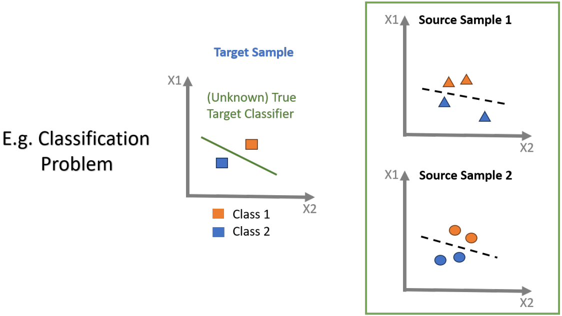
# Transfer Learning

- **Target sample:** $(\boldsymbol{X}^{(0)}, \boldsymbol{y}^{(0)}) \sim P^{(0)}(\boldsymbol{x}, y) = P^{(0)}(y|\boldsymbol{x})P^{(0)}(\boldsymbol{x})$.
- **Source samples:**

$$(\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)}) \sim P^{(k)}(\boldsymbol{x}, y) = P^{(k)}(y|\boldsymbol{x})P^{(k)}(\boldsymbol{x}),\ k = 1, \ldots, K.$$

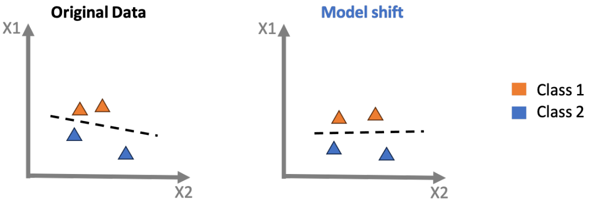
- **Goal of transfer learning:**

Learn the target model $P^{(0)}(y|\boldsymbol{x})$, by incorporating source information.



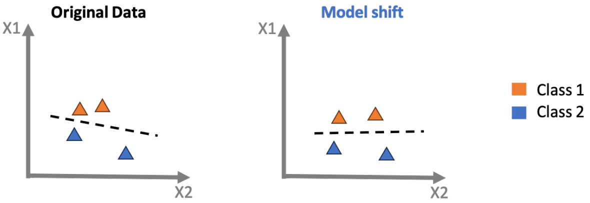
# Model shift in Transfer Learning

- **Model shift:** $P^{(i)}(y|\mathbf{x}) \neq P^{(j)}(y|\mathbf{x}),\ i,j = 0,1,\ldots,K.$



# Model shift in Transfer Learning

- **Model shift:** $P^{(i)}(y|\mathbf{x}) \neq P^{(j)}(y|\mathbf{x}),\ i,j = 0,1,\ldots,K.$



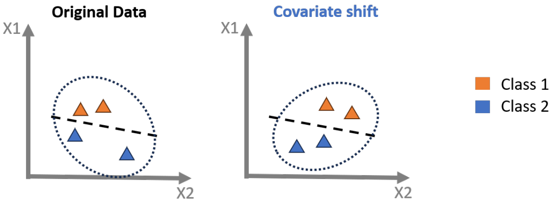

- **Typical strategy to deal with HD model shift:**

**“Pooling training + debiasing”**

- Linear model: Li et al. (2022a,b)
- Generalized linear model: Tian et al. (2022)
- Quantile regression model: Jin et al. (2022), Cao and Song (2024)
- ...

# Covariate Shift in Transfer Learning

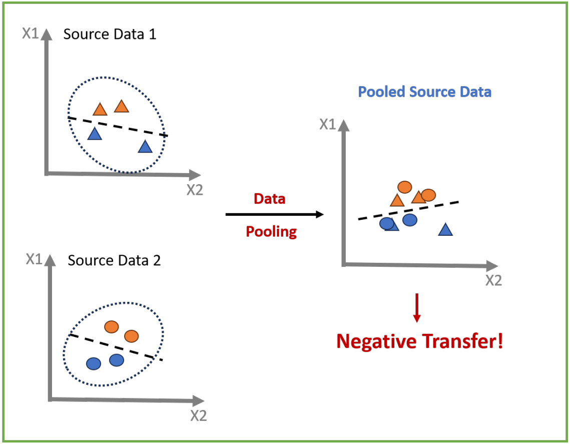
- **covariate shift:** $P^{(i)}(\boldsymbol{x}) \neq P^{(j)}(\boldsymbol{x}),\ i, j = 0, 1, \ldots, K.$




Under high-dimensionality, covariate shift is often inevitable.

# Covariate Shift in Transfer Learning

- **Impact of covariate shift on pooling training:**

# Covariate Shift in Transfer Learning

PennState

## Existing works to deal with covariate shift:

- **Domain adaptation:** Chen et al. (2016), Redko et al. (2020), etc.

  $\rightarrow$ typically assumes no model shift;

  $\rightarrow$ struggles with high-dimensional covariates.

- **Constrained $\ell_1$-minimization:** Li et al. (2023), etc.

  $\rightarrow$ involves multiple nonsmooth constraints;

  $\rightarrow$ restricted parameter space/strong theoretical assumptions;

  $\rightarrow$ computationally intractable.

# Covariate Shift in Transfer Learning

- **Existing works to deal with covariate shift:**
  - **Domain adaptation:** Chen et al. (2016), Redko et al. (2020), etc.

    $\rightarrow$ typically assumes no model shift;

    $\rightarrow$ struggles with high-dimensional covariates.
  - **Constrained $\ell_1$-minimization:** Li et al. (2023), etc.

    $\rightarrow$ involves multiple nonsmooth constraints;

    $\rightarrow$ restricted parameter space/strong theoretical assumptions;

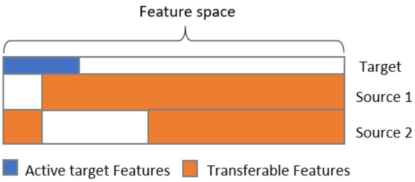
    $\rightarrow$ computationally intractable.

$\Rightarrow$ **Our first question:**

*How to develop a computationally tractable method that effectively handles model shift in high-dimensional transfer learning, while being robust to covariate shift?*

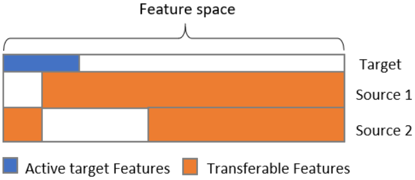
# Feature-specific Transferable Structure

- **Feature-specific transferable structure:**
  - In high-dimensional transfer learning, the transferable structure often varies across features within the same source sample.
  - E.g. Whole brain functional connectivity pattern analysis (Li et al., 2018): Each source may have a distinct set of non-transferable features due to variations in brain conditions.



# Feature-specific Transferable Structure

- **Feature-specific transferable structure:**
  - In high-dimensional transfer learning, the transferable structure often varies across features within the same source sample.
  - E.g. Whole brain functional connectivity pattern analysis (Li et al., 2018): Each source may have a distinct set of non-transferable features due to variations in brain conditions.




⇒ **Our second question:**

*Is it possible to auto-detect nontransferable features/learn transferable structure for each source while transferring source information?*

# Outline

# Key Contributions of the work

In the **high-dimensional regression** context:

- To tackle the covariate shift issue, we
  1. develop a new transfer learning framework via fused regularization, named **TransFusion**;
  2. extend TransFusion to a distributed setting, called **D-TransFusion**.
- To further address the feature-specific transferable structure, we
  3. propose a feature-adaptive transfer learning framework, named **AdaTrans**, to auto-detect non-transferable feature.
- For each method, we establish the corresponding non-asymptotic bound.

# Outline

2 Covariate-shift Robust Transfer Learning
- TransFusion: Transfer Learning with a Fused Regularization
- D-TransFusion: TransFusion under Distributed Setting

# High-dimensional Linear Regression Model

Sample-level **target** model (with sample size $n_T$):

$$\boldsymbol{y}^{(0)} = \boldsymbol{X}^{(0)}\boldsymbol{\beta}^{(0)} + \boldsymbol{\epsilon}^{(0)},$$

Sample-level **source** model (with sample size $n_S$):

$$\boldsymbol{y}^{(k)} = \boldsymbol{X}^{(k)}\boldsymbol{\beta}^{(k)} + \boldsymbol{\epsilon}^{(k)} \equiv \boldsymbol{X}^{(k)}(\boldsymbol{\beta}^{(0)} + \boldsymbol{\delta}^{(k)}) + \boldsymbol{\epsilon}^{(k)}, \ \ k = 1, \dots, K.$$

- $E(\boldsymbol{\epsilon}^{(k)}) = \boldsymbol{0}$, $\mathrm{Cov}(\boldsymbol{\epsilon}^{(k)}) = \sigma^2\boldsymbol{I}$, $\boldsymbol{\epsilon}^{(k)} \perp\!\!\!\perp \boldsymbol{X}^{(k)}$, $k = 0, 1, \dots, K$.
- $\boldsymbol{X}_i^{(k)}$ i.i.d sub-Gaussian across $i$, with covariance matrix $\boldsymbol{\Sigma}^{(k)}$.
- $p \geq n_S \geq n_T$, $\boldsymbol{\beta}^{(0)} \in \mathbb{R}^p$ is high-dimensional yet sparse.
- **Covariate shift**: $\mathrm{Cov}(\boldsymbol{X}_i^{(k)}) = \boldsymbol{\Sigma}^{(k)}$ varies across $k$.
- **Model shift**: $\boldsymbol{\delta}^{(k)} \in \mathbb{R}^p$ varies across $k$; take $\boldsymbol{\delta}^{(0)} = \boldsymbol{0}$.

# Objective function of TransFusion

**Formulation:** Estimate $\boldsymbol{\beta} := ((\boldsymbol{\beta}^{(0)})^\top, (\boldsymbol{\beta}^{(1)})^\top, \ldots, (\boldsymbol{\beta}^{(K)})^\top)^\top \in \mathbb{R}^{(K+1)p}$ by solving

$$\operatorname*{argmin}_{\boldsymbol{\beta} \in \mathbb{R}^{(K+1)p}} \left\{ \frac{1}{2N} \sum_{k=0}^{K} \|\boldsymbol{y}^{(k)} - \boldsymbol{X}^{(k)} \boldsymbol{\beta}^{(k)}\|_2^2 + \lambda_0 \left( \|\boldsymbol{\beta}^{(0)}\|_1 + \sum_{k=1}^{K} \nu \|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(0)}\|_1 \right) \right\},$$

**Reformulation:** Estimate $((\boldsymbol{\beta}^{(0)})^\top, (\boldsymbol{\delta}^{(1)})^\top, \ldots, (\boldsymbol{\delta}^{(K)})^\top)^\top \in \mathbb{R}^{(K+1)p}$ by solving

$$\operatorname*{argmin}_{\boldsymbol{\beta} \in \mathbb{R}^{(K+1)p}} \left\{ \frac{1}{2N} \left( \sum_{k=0}^{K} \|\boldsymbol{y}^{(k)} - \boldsymbol{X}^{(k)} (\boldsymbol{\beta}^{(0)} + \boldsymbol{\delta}^{(k)})\|_2^2 \right) + \lambda_0 \left( \|\boldsymbol{\beta}^{(0)}\|_1 + \sum_{k=1}^{K} \nu \|\boldsymbol{\delta}^{(k)}\|_1 \right) \right\},$$

- $N = Kn_S + n_T$ is the total sample size, $\lambda_0$ and $\nu$ are the tuning parameters.
- The first term measures the average fitness of the models.
- $\lambda_0$ and $\lambda_0 \nu$ respectively take charge of achieving sparsity of $\boldsymbol{\beta}^{(0)}$ and shrinking the contrast $\boldsymbol{\delta}^{(k)} = \boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(0)}$ for information transfer.

# Two-step TransFusion Estimator

### Step 1. Joint training:

1. Obtain $\hat{\beta} := ((\hat{\beta}^{(0)})^\top, (\hat{\beta}^{(1)})^\top, \ldots, (\hat{\beta}^{(K)})^\top)^\top \in \mathbb{R}^{(K+1)p}$ using a newly advocated proximal gradient descent-based algorithm with message-passing iteration.
2. Construct the first-step TransFusion estimator

$$\hat{\beta}^{(0)}_{\mathrm{TF1}} = \sum_{k=1}^{K} \frac{n_S}{N} \hat{\beta}^{(k)} + \frac{n_T}{N} \hat{\beta}^{(0)}.$$

# Two-step TransFusion Estimator

**Step 1. Joint training:**

1. Obtain $\hat{\boldsymbol{\beta}} := ((\hat{\boldsymbol{\beta}}^{(0)})^\top, (\hat{\boldsymbol{\beta}}^{(1)})^\top, \ldots, (\hat{\boldsymbol{\beta}}^{(K)})^\top)^\top \in \mathbb{R}^{(K+1)p}$ using a newly advocated proximal gradient descent-based algorithm with message-passing iteration.
2. Construct the first-step TransFusion estimator

$$\hat{\boldsymbol{\beta}}^{(0)}_{\mathrm{TF1}} = \sum_{k=1}^{K} \frac{n_S}{N} \hat{\boldsymbol{\beta}}^{(k)} + \frac{n_T}{N} \hat{\boldsymbol{\beta}}^{(0)}.$$

**Step 2. Local debiasing:**

3. If necessary, correct the bias of $\hat{\boldsymbol{\beta}}^{(0)}_{\mathrm{TF1}}$ using the target sample through

$$\hat{\boldsymbol{\delta}} \in \underset{\boldsymbol{\delta} \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2n_T} \left\| \boldsymbol{y}^{(0)} - \boldsymbol{X}^{(0)} \hat{\boldsymbol{\beta}}^{(0)}_{\mathrm{TF1}} - \boldsymbol{X}^{(0)} \boldsymbol{\delta} \right\|_2^2 + \tilde{\lambda} \|\boldsymbol{\delta}\|_1 \right\},$$

4. Define the second-step TransFusion estimator

$$\hat{\boldsymbol{\beta}}^{(0)}_{\mathrm{TF2}} = \hat{\boldsymbol{\beta}}^{(0)}_{\mathrm{TF1}} + \hat{\boldsymbol{\delta}}.$$

# Theoretical Guarantee for TransFusion

**Theorem (Error rate of TransFusion estimators)**

*Consider the parameter space*

$$\Theta(s,h)=\left\{B=(\beta^{(0)},\beta^{(1)},\ldots,\beta^{(K)}):\ \|\beta^{(0)}\|_0\le s,\left\|\beta^{(k)}-\beta^{(0)}\right\|_1\le h\right\}.$$

*Under mild conditions,*

- *if* $n_S \gg (K^2h^2 \vee s)\log p$, *with a proper choice of* $\lambda_0$, *with probability at least* $1-c_1\exp(-c_2 n_T)-c_3\exp(-c_4\log p)$, *we have*

$$\|\hat{\beta}^{(0)}_{TF1}-\beta^{(0)}\|_2^2 \lesssim \frac{s\log p}{N}+h\sqrt{\frac{\log p}{n_S}}+\varepsilon_D^2.$$

- *if* $n_T \gtrsim s\log p$, $n_S \gg K^2(h^2\vee s)\log p$ *and* $h\sqrt{\log p/n_T}=o(1)$, *with probability at least* $1-c_2\exp(-c_3\log p)$, *we have*

$$\|\hat{\beta}^{(0)}_{TF2}-\beta^{(0)}\|_2^2 \lesssim \frac{s\log p}{N}+h\sqrt{\frac{\log p}{n_T}},$$

# Theoretical Guarantee for TransFusion

**Theorem (Error rate of TransFusion estimators)**

$$\|\hat{\beta}^{(0)}_{TF1} - \beta^{(0)}\|_2^2 \lesssim \frac{s\log p}{N} + h\sqrt{\frac{\log p}{n_S}} + \varepsilon_D^2;$$

$$\|\hat{\beta}^{(0)}_{TF2} - \beta^{(0)}\|_2^2 \lesssim \frac{s\log p}{N} + h\sqrt{\frac{\log p}{n_T}}.$$

# Theoretical Guarantee for TransFusion

**Theorem (Error rate of TransFusion estimators)**

$$\|\hat{\beta}^{(0)}_{TF1} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_S}} + \varepsilon_D^2;$$
$$\|\hat{\beta}^{(0)}_{TF2} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_T}}.$$

- $s \log p / N$: rate of estimating $\beta^{(0)}$ based on the full sample with size $N$.

# Theoretical Guarantee for TransFusion

**Theorem (Error rate of TransFusion estimators)**

$$\|\hat{\beta}^{(0)}_{TF1} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_S}} + \varepsilon_D^2;$$
$$\|\hat{\beta}^{(0)}_{TF2} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_T}}.$$

- $s \log p/N$: rate of estimating $\beta^{(0)}$ based on the full sample with size $N$.
- $h\sqrt{\log p/n_S}$: rate of estimating $\delta^{(k)}$'s based on source samples.

# Theoretical Guarantee for TransFusion

**Theorem (Error rate of TransFusion estimators)**

$$\|\hat{\beta}^{(0)}_{TF1} - \beta^{(0)}\|_2^2 \lesssim \frac{s\log p}{N} + h\sqrt{\frac{\log p}{n_S}} + \varepsilon_D^2;$$

$$\|\hat{\beta}^{(0)}_{TF2} - \beta^{(0)}\|_2^2 \lesssim \frac{s\log p}{N} + h\sqrt{\frac{\log p}{n_T}}.$$

- $s\log p/N$: rate of estimating $\beta^{(0)}$ based on the full sample with size $N$.
- $h\sqrt{\log p/n_S}$: rate of estimating $\delta^{(k)}$'s based on source samples.
- $\varepsilon_D = \|\sum_{k=1}^{K} \frac{n_S}{N}(\beta^{(k)} - \beta^{(0)})\|_1$: **task diversity** which measures the bias introduced by averaging.

# Theoretical Guarantee for TransFusion

**Theorem (Error rate of TransFusion estimators)**

$$\|\hat{\beta}^{(0)}_{TF1} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_S}} + \varepsilon_D^2;$$

$$\|\hat{\beta}^{(0)}_{TF2} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_T}}.$$

- $s \log p/N$: rate of estimating $\beta^{(0)}$ based on the full sample with size $N$.
- $h\sqrt{\log p/n_S}$: rate of estimating $\delta^{(k)}$'s based on source samples.
- $\varepsilon_D = \| \sum_{k=1}^{K} \frac{n_S}{N}(\beta^{(k)} - \beta^{(0)})\|_1$: **task diversity** which measures the bias introduced by averaging.
- $h\sqrt{\log p/n_T}$: rate of estimating $\delta^{(k)}$'s using the target sample.

# Theoretical Guarantee for TransFusion

**Theorem (Error rate of TransFusion estimators)**

$$\|\hat{\beta}^{(0)}_{TF1} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_S}} + \varepsilon_D^2;$$

$$\|\hat{\beta}^{(0)}_{TF2} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_T}}.$$

- $s \log p/N$: rate of estimating $\beta^{(0)}$ based on the full sample with size $N$.
- $h\sqrt{\log p/n_S}$: rate of estimating $\delta^{(k)}$'s based on source samples.
- $\varepsilon_D = \|\sum_{k=1}^{K} \frac{n_S}{N}(\beta^{(k)} - \beta^{(0)})\|_1$: **task diversity** which measures the bias introduced by averaging.
- $h\sqrt{\log p/n_T}$: rate of estimating $\delta^{(k)}$'s using the target sample.
- **Baseline:** target-only lasso with rate $O(s \log p/n_T)$.

# Theoretical Guarantee for TransFusion

**Theorem (Error rate of TransFusion estimators)**

$$\|\hat{\beta}^{(0)}_{TF1} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_S}} + \varepsilon_D^2;$$

$$\|\hat{\beta}^{(0)}_{TF2} - \beta^{(0)}\|_2^2 \lesssim \frac{s \log p}{N} + h\sqrt{\frac{\log p}{n_T}}.$$

- $s \log p / N$: rate of estimating $\beta^{(0)}$ based on the full sample with size $N$.
- $h\sqrt{\log p / n_S}$: rate of estimating $\delta^{(k)}$'s based on source samples.
- $\varepsilon_D = \| \sum_{k=1}^K \frac{n_S}{N}(\beta^{(k)} - \beta^{(0)})\|_1$: **task diversity** which measures the bias introduced by averaging.
- $h\sqrt{\log p / n_T}$: rate of estimating $\delta^{(k)}$'s using the target sample.
- **Baseline:** target-only lasso with rate $O(s \log p / n_T)$.
- $\hat{\beta}^{(0)}_{\mathrm{TF1}}$ is preferred when $\varepsilon_D$ or $n_T$ is small, while $\hat{\beta}^{(0)}_{\mathrm{TF2}}$ is preferred when $\varepsilon_D$ is non-negligible and $n_T$ is adequately large.

# Why TF is Robust to Covariate Shifts?

**TransFusion** utilizes task-specific parameters with a series of fused regularizers, resulting in

$$\hat{\boldsymbol{\beta}}^{(0)}_{TF1} - \boldsymbol{\beta}^{(0)} \to \frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(0)}).$$

# Why TF is Robust to Covariate Shifts?

**TransFusion** utilizes task-specific parameters with a series of fused regularizers, resulting in

$$\hat{\boldsymbol{\beta}}^{(0)}_{TF1} - \boldsymbol{\beta}^{(0)} \to \frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(0)}).$$

**Pooling training** first pools all data and trains a common model, resulting in

$$\hat{\boldsymbol{\beta}}^{(0)}_{\text{Pooling}} - \boldsymbol{\beta}^{(0)} \to \left(\sum_{k=1}^{K}\boldsymbol{\Sigma}^{(k)}\right)^{-1}\sum_{k=1}^{K}\boldsymbol{\Sigma}^{(k)}(\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(0)}).$$

# Why TF is Robust to Covariate Shifts?

**TransFusion** utilizes task-specific parameters with a series of fused regularizers, resulting in

$$\hat{\boldsymbol{\beta}}^{(0)}_{TF1} - \boldsymbol{\beta}^{(0)} \to \frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(0)}).$$

**Pooling training** first pools all data and trains a common model, resulting in

$$\hat{\boldsymbol{\beta}}^{(0)}_{\text{Pooling}} - \boldsymbol{\beta}^{(0)} \to \left(\sum_{k=1}^{K}\boldsymbol{\Sigma}^{(k)}\right)^{-1}\sum_{k=1}^{K}\boldsymbol{\Sigma}^{(k)}(\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(0)}).$$

The bias can be amplified by a factor (Li et al., 2022)

$$C_{\Sigma} := 1 + \max_{j\leq p}\max_{k}\left\| e_j^{\top}\left(\boldsymbol{\Sigma}^{(k)} - \boldsymbol{\Sigma}^{(0)}\right)\left(\sum_{1\leq k\leq K}\frac{1}{K}\boldsymbol{\Sigma}^{(k)}\right)^{-1}\right\|_1,$$

which could diverge with rate $\sqrt{p}$.

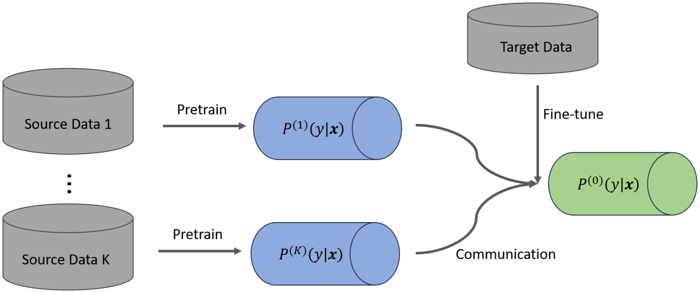
# Outline

PennState

# Distributed Transfer Learning in One-shot

**Distributed setting:** Source datasets are distributed across different machines.

- Privacy concern: raw data cannot be shared across machines;
- Communication bottleneck: inter-machine data communication is a significant source of latency;
- Pretraining & Fine-tuning strategy: quickly adapt to downstream tasks.



# Two-step D-TransFusion Estimator

**Step 1:** The $k$th source machine computes an initial estimator $\tilde{\beta}^{(k)}$ using $(\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)})$ and sends it to the target machine. Then the target machine solves

$$\hat{\beta}_D \in \underset{\boldsymbol{\beta} \in \mathbb{R}^{(K+1)p}}{\operatorname{argmin}} \left\{ \frac{n_S}{2N} \sum_{k=1}^{K} \|\tilde{\beta}^{(k)} - \boldsymbol{\beta}^{(k)}\|_2^2 + \frac{1}{2N} \|\boldsymbol{y}^{(0)} - \boldsymbol{X}^{(0)} \boldsymbol{\beta}^{(0)}\|_2^2 \right.$$
$$\left. + \lambda_0 \left( \|\boldsymbol{\beta}^{(0)}\|_1 + \sum_{k=1}^{K} \nu \|\tilde{\beta}^{(k)} - \boldsymbol{\beta}^{(0)}\|_1 \right) \right\},$$

and obtains $\hat{\beta}^{(0)}_{\textbf{D-TF1}} = \frac{n_S}{N} \sum_{k=1}^{K} \hat{\beta}_D^{(k)} + \frac{n_T}{N} \hat{\beta}_D^{(0)}$.

# Two-step D-TransFusion Estimator

**Step 1:** The $k$th source machine computes an initial estimator $\tilde{\boldsymbol{\beta}}^{(k)}$ using $(\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)})$ and sends it to the target machine. Then the target machine solves

$$\hat{\boldsymbol{\beta}}_D \in \underset{\boldsymbol{\beta}\in\mathbb{R}^{(K+1)p}}{\operatorname{argmin}} \left\{ \frac{n_S}{2N} \sum_{k=1}^{K} \|\tilde{\boldsymbol{\beta}}^{(k)} - \boldsymbol{\beta}^{(k)}\|_2^2 + \frac{1}{2N}\|\boldsymbol{y}^{(0)} - \boldsymbol{X}^{(0)}\boldsymbol{\beta}^{(0)}\|_2^2 \right.$$
$$\left. + \lambda_0\left(\|\boldsymbol{\beta}^{(0)}\|_1 + \sum_{k=1}^{K} \nu\|\tilde{\boldsymbol{\beta}}^{(k)} - \boldsymbol{\beta}^{(0)}\|_1\right) \right\},$$

and obtains $\hat{\boldsymbol{\beta}}^{(0)}_{\textbf{D-TF1}} = \frac{n_S}{N}\sum_{k=1}^{K} \hat{\boldsymbol{\beta}}^{(k)}_D + \frac{n_T}{N}\hat{\boldsymbol{\beta}}^{(0)}_D$.

**Step 2:** The target node corrects $\hat{\boldsymbol{\beta}}^{(0)}_{\text{D-TF1}}$ on its local sample $(\boldsymbol{X}^{(0)}, \boldsymbol{y}^{(0)})$ by

$$\hat{\boldsymbol{\delta}}_D \in \underset{\boldsymbol{\delta}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2n_T}\left\|\boldsymbol{y}^{(0)} - \boldsymbol{X}^{(0)}\hat{\boldsymbol{\beta}}^{(0)}_{\text{D-TF1}} - \boldsymbol{X}^{(0)}\boldsymbol{\delta}\right\|_2^2 + \tilde{\lambda}\|\boldsymbol{\delta}\|_1 \right\},$$

and outputs the second-step estimator $\hat{\boldsymbol{\beta}}^{(0)}_{\textbf{D-TF2}} = \hat{\boldsymbol{\beta}}^{(0)}_{\textbf{D-TF1}} + \hat{\boldsymbol{\delta}}_D$.

# Two-step D-TransFusion Estimator

**Step 1:** The $k$th source machine computes an initial estimator $\tilde{\beta}^{(k)}$ using $(\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)})$ and sends it to the target machine. Then the target machine solves

$$\hat{\beta}_D \in \underset{\beta \in \mathbb{R}^{(K+1)p}}{\operatorname{argmin}} \left\{ \frac{n_S}{2N} \sum_{k=1}^{K} \|\tilde{\beta}^{(k)} - \beta^{(k)}\|_2^2 + \frac{1}{2N} \|\boldsymbol{y}^{(0)} - \boldsymbol{X}^{(0)} \beta^{(0)}\|_2^2 \right.$$
$$\left. + \lambda_0 \left( \|\beta^{(0)}\|_1 + \sum_{k=1}^{K} \nu \|\tilde{\beta}^{(k)} - \beta^{(0)}\|_1 \right) \right\},$$

and obtains $\hat{\beta}^{(0)}_{\textbf{D-TF1}} = \frac{n_S}{N} \sum_{k=1}^{K} \hat{\beta}_D^{(k)} + \frac{n_T}{N} \hat{\beta}_D^{(0)}$.

**Step 2:** The target node corrects $\hat{\beta}^{(0)}_{\text{D-TF1}}$ on its local sample $(\boldsymbol{X}^{(0)}, \boldsymbol{y}^{(0)})$ by

$$\hat{\delta}_D \in \underset{\delta \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \frac{1}{2n_T} \left\| \boldsymbol{y}^{(0)} - \boldsymbol{X}^{(0)} \hat{\beta}^{(0)}_{\text{D-TF1}} - \boldsymbol{X}^{(0)} \delta \right\|_2^2 + \tilde{\lambda} \|\delta\|_1 \right\},$$

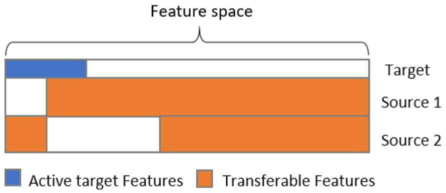
and outputs the second-step estimator $\hat{\beta}^{(0)}_{\textbf{D-TF2}} = \hat{\beta}^{(0)}_{\textbf{D-TF1}} + \hat{\delta}_D$.

- Only one-shot communication with the summary statistic is required.
- D-TransFusion has the same rate as TransFusion under mild conditions.

# Outline

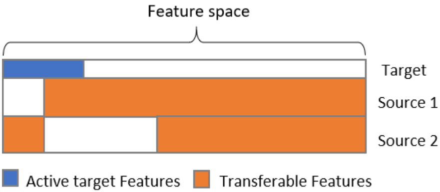
# Intuition of AdaTrans

On base of TransFusion, consider the **feature-specific transferable structure:**



# Intuition of AdaTrans

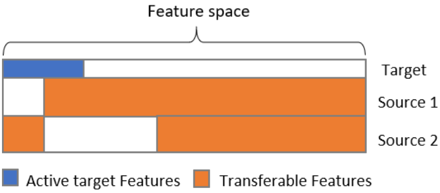
On base of TransFusion, consider the **feature-specific transferable structure:**




The transferability of the $j$th feature in the $k$-th source task can be assessed by the magnitude of model shift $\delta_j^{(k)}$. Ideally, we should...

# Intuition of AdaTrans

On base of TransFusion, consider the **feature-specific transferable structure:**




The transferability of the $j$th feature in the $k$-th source task can be assessed by the magnitude of model shift $\delta_j^{(k)}$. Ideally, we should...

- apply stronger penalties to transferable features with negligible $\delta_j^{(k)}$;
  $\rightarrow$ shrink $\delta_j^{(k)}$ to 0, i.e. pool $\beta_j^{(k)}$ and $\beta_j^{(0)}$, if the $j$-th feature from the $k$-th source is informative/transferable
- prevents excessive penalties to non-transferable features with large $\delta_j^{(k)}$.
  $\rightarrow$ prevent introducing bias from non-transferable signals

# Key Idea of AdaTrans

Estimate $\boldsymbol{\beta}^{(0)}$ by solving

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^{(K+1)p}}{\operatorname{argmin}} \left\{ \frac{1}{2N} \sum_{k=0}^{K} \|\boldsymbol{y}^{(k)} - \boldsymbol{X}^{(k)}(\boldsymbol{\beta}^{(0)} + \boldsymbol{\delta}^{(k)})\|_2^2 + \sum_{j=1}^{p} \hat{w}_{0j} |\boldsymbol{\beta}_j^{(0)}| + \sum_{k=1}^{K} \sum_{j=1}^{p} \hat{w}_{kj} |\boldsymbol{\delta}_j^{(k)}| \right\},$$

# Key Idea of AdaTrans

Estimate $\boldsymbol{\beta}^{(0)}$ by solving

$$\underset{\boldsymbol{\beta} \in \mathbb{R}^{(K+1)p}}{\operatorname{argmin}} \left\{ \frac{1}{2N} \sum_{k=0}^{K} \|\boldsymbol{y}^{(k)} - \boldsymbol{X}^{(k)}(\boldsymbol{\beta}^{(0)} + \boldsymbol{\delta}^{(k)})\|_2^2 + \sum_{j=1}^{p} \hat{w}_{0j} |\beta_j^{(0)}| + \sum_{k=1}^{K} \sum_{j=1}^{p} \hat{w}_{kj} |\delta_j^{(k)}| \right\},$$

**Choice of weight?**

# Key Idea of AdaTrans

Estimate $\beta^{(0)}$ by solving

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^{(K+1)p}}{\operatorname{argmin}} \left\{ \frac{1}{2N} \sum_{k=0}^{K} \|\boldsymbol{y}^{(k)} - \boldsymbol{X}^{(k)}(\boldsymbol{\beta}^{(0)} + \boldsymbol{\delta}^{(k)})\|_2^2 + \sum_{j=1}^{p} \hat{w}_{0j} |\beta_j^{(0)}| + \sum_{k=1}^{K} \sum_{j=1}^{p} \hat{w}_{kj} |\delta_j^{(k)}| \right\},$$

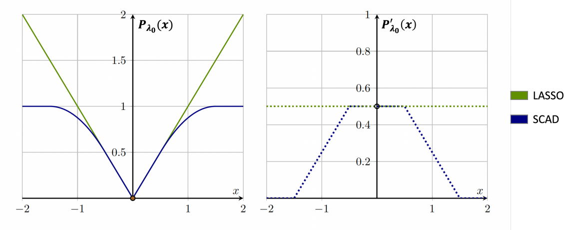
**Choice of weight?** $\Rightarrow$ **Folded-concave penalty function $\mathcal{P}_{\lambda_0}(\cdot)$:**

# Key Idea of AdaTrans

Estimate $\beta^{(0)}$ by solving

$$\underset{\beta \in \mathbb{R}^{(K+1)p}}{\operatorname{argmin}} \left\{ \frac{1}{2N} \sum_{k=0}^{K} \|\boldsymbol{y}^{(k)} - \boldsymbol{X}^{(k)}(\boldsymbol{\beta}^{(0)} + \boldsymbol{\delta}^{(k)})\|_2^2 + \sum_{j=1}^{p} \hat{w}_{0j} |\beta_j^{(0)}| + \sum_{k=1}^{K} \sum_{j=1}^{p} \hat{w}_{kj} |\delta_j^{(k)}| \right\},$$

**Choice of weight?** $\Rightarrow$ **Folded-concave penalty function $\mathcal{P}_{\lambda_0}(\cdot)$:**

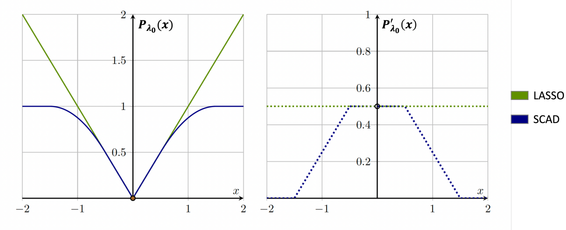


# Key Idea of AdaTrans

Estimate $\beta^{(0)}$ by solving

$$\underset{\beta \in \mathbb{R}^{(K+1)p}}{\operatorname{argmin}} \left\{ \frac{1}{2N} \sum_{k=0}^{K} \|\boldsymbol{y}^{(k)} - \boldsymbol{X}^{(k)}(\beta^{(0)} + \delta^{(k)})\|_2^2 + \sum_{j=1}^{p} \hat{w}_{0j} |\beta_j^{(0)}| + \sum_{k=1}^{K} \sum_{j=1}^{p} \hat{w}_{kj} |\delta_j^{(k)}| \right\},$$

**Choice of weight?** $\Rightarrow$ **Folded-concave penalty function $\mathcal{P}_{\lambda_0}(\cdot)$:**

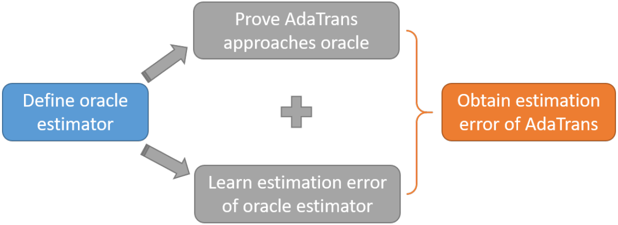



Borrowing the idea of local linear approximation, take $\hat{w}_{0j} \propto \mathcal{P}'_{\lambda_0}(\hat{\beta}^{(0)}_{\text{init},j})$ and $\hat{w}_{kj} \propto \mathcal{P}'_{\lambda_0}(\hat{\delta}^{(k)}_{\text{init},j})$, where $\hat{\beta}^{(0)}_{\text{init},j}$ and $\hat{\delta}^{(k)}_{\text{init},j}$ are initial estimators of $\beta_j^{(0)}$ and $\delta_j$.

# Outline

# Developing Theory for AdaTrans

Recall that under certain conditions, folded-concave penalization can obtain an **oracle estimator**, where the sparsity and transferable structures are known.



# Oracle AdaTrans Estimator

How to define oracle estimator for AdaTrans?

# Oracle AdaTrans Estimator

How to define oracle estimator for AdaTrans?

1. Define **sparsity structure:**
   - Active target feature set: $S_0 = \{j : \beta_j^{(0)} \neq 0\}$,
   - Inactive target feature set: $S_0^c = \{j : \beta_j^{(0)} = 0\}$;

# Oracle AdaTrans Estimator

How to define oracle estimator for AdaTrans?

1. Define **sparsity structure:**
   - Active target feature set: $S_0 = \{j : \beta_j^{(0)} \neq 0\}$,
   - Inactive target feature set: $S_0^c = \{j : \beta_j^{(0)} = 0\}$;
2. Define **transferability structure:**
   - Non-transferable set: $S_k = \{j : \delta_j^{(k)} \neq 0\}$, $k = 1, \ldots, K$,
   - Transferable set: $S_k^c = \{j : \delta_j^{(k)} = 0\}$, $k = 1, \ldots, K$.

# Oracle AdaTrans Estimator

How to define oracle estimator for AdaTrans?

1. Define **sparsity structure:**
   - Active target feature set: $S_0 = \{j : \beta_j^{(0)} \neq 0\}$,
   - Inactive target feature set: $S_0^c = \{j : \beta_j^{(0)} = 0\}$;
2. Define **transferability structure:**
   - Non-transferable set: $S_k = \{j : \boldsymbol{\delta}_j^{(k)} \neq 0\}$, $k = 1, \ldots, K$,
   - Transferable set: $S_k^c = \{j : \boldsymbol{\delta}_j^{(k)} = 0\}$, $k = 1, \ldots, K$.
3. Define **oracle AdaTrans estimator** $\hat{\beta}_{\text{ora}}^{(0)}, \hat{\boldsymbol{\delta}}_{\text{ora}}^{(1)}, \ldots, \hat{\boldsymbol{\delta}}_{\text{ora}}^{(K)}$ via

$$\begin{aligned} \min_{\beta^{(0)}, \{\boldsymbol{\delta}^{(k)}\}_{k=1}^K} \quad & \frac{1}{N} \sum_{k=0}^{K} \|\boldsymbol{y}^{(k)} - \boldsymbol{X}^{(k)}(\beta^{(0)} + \boldsymbol{\delta}^{(k)})\|_2^2 \\ \text{s.t.} \quad & \beta_{S_0^c}^{(0)} = 0, \ \boldsymbol{\delta}_{S_k^c}^{(k)} = 0, \forall k = 1, \ldots, K. \end{aligned} \tag{1}$$

# Oracle AdaTrans Estimator

**Theorem (Solution of oracle AdaTrans estimator)**

*If* $|S_0| < n_T$ *and* $\max_{1\le k\le K}|S_k| < n_S$*, the solution to problem (1) satisfies*

$$\hat{\beta}^{(0)}_{\mathrm{ora},S_0} = [\tilde{\boldsymbol{X}}_{S_0}^\top \tilde{\boldsymbol{X}}_{S_0}]^{-1}\tilde{\boldsymbol{X}}_{S_0}^\top \boldsymbol{y} \quad \textit{and} \quad \hat{\beta}^{(0)}_{\mathrm{ora},S_0^c} = \mathbf{0}. \tag{2}$$

- $\tilde{\boldsymbol{X}}_{S_0} = ((\boldsymbol{X}_{S_0}^{(0)})^\top, (\tilde{\boldsymbol{X}}_{S_0}^{(1)})^\top, \dots, (\tilde{\boldsymbol{X}}_{S_0}^{(K)})^\top)^\top$.
- $\tilde{\boldsymbol{X}}_{S_0}^{(k)} = (\boldsymbol{I} - \mathbf{H}_{S_k}^{(k)})\boldsymbol{X}_{S_0}^{(k)}$, where $\mathbf{H}_{S_k}^{(k)} := \boldsymbol{X}_{S_k}^{(k)}[(\boldsymbol{X}_{S_k}^{(k)})^\top \boldsymbol{X}_{S_k}^{(k)}]^{-1}(\boldsymbol{X}_{S_k}^{(k)})^\top$.

# Oracle AdaTrans Estimator

**Theorem (Solution of oracle AdaTrans estimator)**

*If* $|S_0| < n_T$ *and* $\max_{1\leq k\leq K} |S_k| < n_S$*, the solution to problem (1) satisfies*

$$\hat{\beta}^{(0)}_{\text{ora},S_0} = [\tilde{\boldsymbol{X}}_{S_0}^\top \tilde{\boldsymbol{X}}_{S_0}]^{-1} \tilde{\boldsymbol{X}}_{S_0}^\top \boldsymbol{y} \quad \textit{and} \quad \hat{\beta}^{(0)}_{\text{ora},S_0^c} = \mathbf{0}. \tag{2}$$

- $\tilde{\boldsymbol{X}}_{S_0} = ((\boldsymbol{X}_{S_0}^{(0)})^\top, (\tilde{\boldsymbol{X}}_{S_0}^{(1)})^\top, \ldots, (\tilde{\boldsymbol{X}}_{S_0}^{(K)})^\top)^\top$.
- $\tilde{\boldsymbol{X}}_{S_0}^{(k)} = (\boldsymbol{I} - \mathbf{H}_{S_k}^{(k)})\boldsymbol{X}_{S_0}^{(k)}$, where $\mathbf{H}_{S_k}^{(k)} := \boldsymbol{X}_{S_k}^{(k)}[(\boldsymbol{X}_{S_k}^{(k)})^\top \boldsymbol{X}_{S_k}^{(k)}]^{-1}(\boldsymbol{X}_{S_k}^{(k)})^\top$.
- $\tilde{\boldsymbol{X}}_{S_0}^{(k)}$ is indeed the projection of the active target feature onto the null space of the non-transferable feature in the $k$-th source sample.

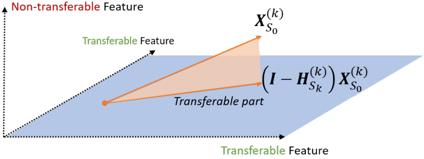
# Oracle AdaTrans Estimator

**Theorem (Solution of oracle AdaTrans estimator)**

*If* $|S_0| < n_T$ *and* $\max_{1\le k\le K} |S_k| < n_S$*, the solution to problem (1) satisfies*

$$\hat{\beta}^{(0)}_{\text{ora},S_0} = [\tilde{\boldsymbol{X}}_{S_0}^\top \tilde{\boldsymbol{X}}_{S_0}]^{-1} \tilde{\boldsymbol{X}}_{S_0}^\top \boldsymbol{y} \quad \text{and} \quad \hat{\beta}^{(0)}_{\text{ora},S_0^c} = \boldsymbol{0}. \tag{2}$$

- $\tilde{\boldsymbol{X}}_{S_0} = ((\boldsymbol{X}^{(0)}_{S_0})^\top, (\tilde{\boldsymbol{X}}^{(1)}_{S_0})^\top, \ldots, (\tilde{\boldsymbol{X}}^{(K)}_{S_0})^\top)^\top$.
- $\tilde{\boldsymbol{X}}^{(k)}_{S_0} = (\boldsymbol{I} - \mathbf{H}^{(k)}_{S_k})\boldsymbol{X}^{(k)}_{S_0}$, where $\mathbf{H}^{(k)}_{S_k} := \boldsymbol{X}^{(k)}_{S_k}[(\boldsymbol{X}^{(k)}_{S_k})^\top \boldsymbol{X}^{(k)}_{S_k}]^{-1}(\boldsymbol{X}^{(k)}_{S_k})^\top$.
- $\tilde{\boldsymbol{X}}^{(k)}_{S_0}$ is indeed the projection of the active target feature onto the null space of the non-transferable feature in the $k$-th source sample.



# Estimation Error of Oracle AdaTrans

**Theorem (Estimation error of oracle AdaTrans)**

*If* $|S_0| < n_T$, $\max_{1\le k\le K} |S_k| < n_S$ *and* $N \ge \log p$, *the error of* $\hat{\beta}^{(0)}_{\mathrm{ora}}$ *satisfies*

$$\|\hat{\beta}^{(0)}_{\mathrm{ora}} - \beta^{(0)}\|_2 \lesssim \kappa_F \left\| \left( \frac{\boldsymbol{X}_{S_0}^\top \boldsymbol{X}_{S_0}}{N} \right)^{-1} \right\|_\infty \sqrt{\frac{s \log s}{N}}, \tag{3}$$

*with probability larger than* $1 - \exp(c_1 \log p)$, *where* $\boldsymbol{X}_{S_0}$ *is column-submatrix indexed by* $S_0$ *of the full-sample design matrix* $\boldsymbol{X}$, *and*

$$\kappa_F := \frac{\left\| [\tilde{\boldsymbol{X}}_{S_0}^\top \tilde{\boldsymbol{X}}_{S_0}]^{-1} \tilde{\boldsymbol{X}}_{S_0}^\top \epsilon \right\|_\infty}{\left\| [\boldsymbol{X}_{S_0}^\top \boldsymbol{X}_{S_0}]^{-1} \boldsymbol{X}_{S_0}^\top \epsilon \right\|_\infty}.$$

# Estimation Error of Oracle AdaTrans

**Theorem (Estimation error of oracle AdaTrans)**

*If* $|S_0| < n_T$, $\max_{1\le k\le K} |S_k| < n_S$ *and* $N \geq \log p$, *the error of* $\hat{\beta}^{(0)}_{\mathrm{ora}}$ *satisfies*

$$\|\hat{\beta}^{(0)}_{\mathrm{ora}} - \beta^{(0)}\|_2 \lesssim \kappa_F \left\| \left( \frac{\boldsymbol{X}_{S_0}^\top \boldsymbol{X}_{S_0}}{N} \right)^{-1} \right\|_\infty \sqrt{\frac{s \log s}{N}}, \tag{3}$$

*with probability larger than* $1 - \exp(c_1 \log p)$, *where* $\boldsymbol{X}_{S_0}$ *is column-submatrix indexed by* $S_0$ *of the full-sample design matrix* $\boldsymbol{X}$, *and*

$$\kappa_F := \frac{\left\| [\tilde{\boldsymbol{X}}_{S_0}^\top \tilde{\boldsymbol{X}}_{S_0}]^{-1} \tilde{\boldsymbol{X}}_{S_0}^\top \epsilon \right\|_\infty}{\left\| [\boldsymbol{X}_{S_0}^\top \boldsymbol{X}_{S_0}]^{-1} \boldsymbol{X}_{S_0}^\top \epsilon \right\|_\infty}.$$

- $\kappa_F$ measures the transferability of source datasets. For $k = 1, \ldots, K$,
  - if $\boldsymbol{X}_{S_k}^{(k)} \perp \boldsymbol{X}_{S_0}^{(k)}$, all active features are transferable, then $\kappa_F = 1$;
  - if $S_0 \subset S_k$, all active features are non-transferable, then $\kappa_F \asymp \sqrt{N/n_T}$, and the final rate becomes $\sqrt{s \log s / n_T}$.

# Theoretical Guarantee of AdaTrans

**Theorem (Oracle property of AdaTrans)**

*Consider the parametric space*

$$\Theta_1 = \left\{ \left\|\delta_{S_k}^{(k)}\right\|_{\min} \geq h_k^{\wedge}, \left\|\delta_{S_k^c}^{(k)}\right\|_{\max} = 0, k = 1, \ldots, K; \left\|\beta_{S_0}^{(0)}\right\|_{\min} \geq h_0^{\wedge}, \left\|\beta_{S_0^c}^{(0)}\right\|_{\max} = 0 \right\}.$$

*Suppose for some $a > a_2 \geq 0$, the initial estimators satisfy*

$$\left\|\hat{\beta}_{init}^{(0)} - \beta^{(0)}\right\|_{\infty} \leq \frac{a_2}{2}\lambda_0, \ \left\|\hat{\delta}_{init}^{(k)} - \delta^{(k)}\right\|_{\infty} \leq \frac{a_2}{2}\lambda_1;$$

*the minimal target signal $h_0^{\wedge} \geq a\lambda_0 \gtrsim \sqrt{\frac{\log p}{N}}$, and the non-transferable signal $h_k^{\wedge} \geq a\lambda_1 \gtrsim \sqrt{\frac{n_S}{N}\frac{\log p}{N}}$, and $n_S \gtrsim \log p$. Then by choosing $w_{0j} = \mathcal{P}'_{\lambda_0}(\hat{\beta}_{init,j}^{(0)})/\lambda_0$ and $w_{kj} = \mathcal{P}'_{\lambda_1}(\hat{\delta}_{init,j}^{(k)})/\lambda_1$, with probability larger than $1 - \exp(-c_1 \log p)$, we obtain the oracle AdaTrans.*

# Outline

# Simulation settings for TransFusion

Recall the regression models

$$\boldsymbol{y}^{(0)} = \boldsymbol{X}^{(0)}\boldsymbol{\beta}^{(0)} + \boldsymbol{\epsilon}^{(0)},\ \boldsymbol{y}^{(k)} = \boldsymbol{X}^{(k)}(\boldsymbol{\beta}^{(0)} + \boldsymbol{\delta}^{(k)}) + \boldsymbol{\epsilon}^{(k)},\ k = 1, \ldots, K.$$

**General setup:**

- Target task: $n_T = 150$, $s = 10$, $\boldsymbol{\beta}^{(0)} = (\mathbf{1}_s^\top, \mathbf{0}_{p-s}^\top)^\top$, $\epsilon_i^{(0)} \sim N(0, 1)$.
- Source task: $n_S = 200$, $K \in \{1, 3, 5, 7, 9\}$, $\epsilon_i^{(k)} \sim N(0, 1)$.

**Model shift:**

- $\delta_j^{(k)} \sim N(0.1, 0.2^2)$ for $1 \le j \le 50$ and $\delta_j^{(k)} = 0$ otherwise.

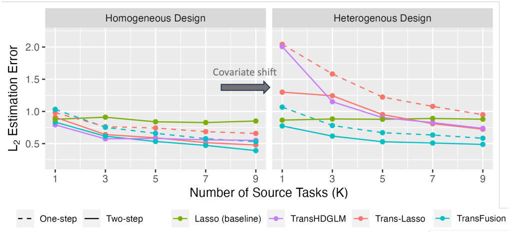
**Covariate shift**:

- Homogeneous design (without covariate shift): Each $\boldsymbol{X}_i^{(k)} \sim N(0, \boldsymbol{I})$.
- Heterogeneous design (with covariate shift): Each $\boldsymbol{X}_i^{(k)} \sim N(0, \boldsymbol{\Sigma}^{(k)})$, with $\boldsymbol{\Sigma}^{(k)} = (\boldsymbol{A}^{(k)})^\top(\boldsymbol{A}^{(k)}) + \boldsymbol{I}$, where $\boldsymbol{A}^{(k)}$ is a random matrix with each entry equals 0.3 with probability 0.3 and equals 0 with probability 0.7.

# Methods to be compared

- Lasso (baseline): LASSO regression on the target task.
- TransLasso (first-step) (Li et al., 2022): pooled estimator.
- TransLasso (two-step) (Li et al., 2022): debiased estimator.
- TransHDGLM (Li et al., 2023).
- TransFusion (first-step): the first step TransFusion estimator $\hat{\beta}^{(0)}_{\mathrm{TF1}}$.
- TransFusion (two-step): the debiased TransFusion estimator $\hat{\beta}^{(0)}_{\mathrm{TF2}}$.

# Simulation Results: TransFusion



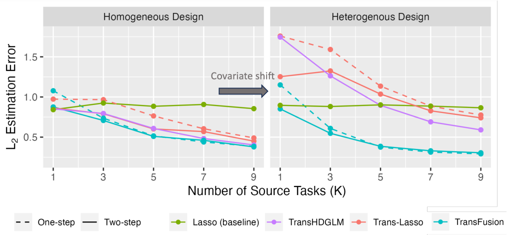




Figure: Estimation errors with/without covariate shift. Upper panel: task diversity $\epsilon_D \neq 0$; lower panel: $\epsilon_D = 0$.

# Simulation Results: D-TransFusion

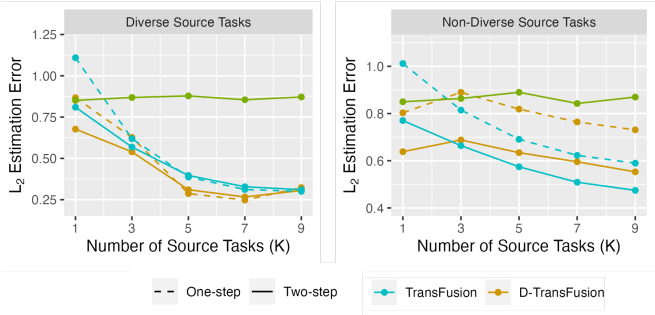



Figure: Estimation errors with $\epsilon_D = 0$ (left panel) and $\epsilon_D \neq 0$ (right panel).

# Outline

4 Numerical Studies
- Simulation Examples for TransFusion
- Simulation Examples for AdaTrans

# Simulation Settings for AdaTrans

Recall the regression models

$$\boldsymbol{y}^{(0)} = \boldsymbol{X}^{(0)}\boldsymbol{\beta}^{(0)} + \boldsymbol{\epsilon}^{(0)},\ \boldsymbol{y}^{(k)} = \boldsymbol{X}^{(k)}(\boldsymbol{\beta}^{(0)} + \boldsymbol{\delta}^{(k)}) + \boldsymbol{\epsilon}^{(k)},\ k = 1, \dots, K.$$

**General setup**:

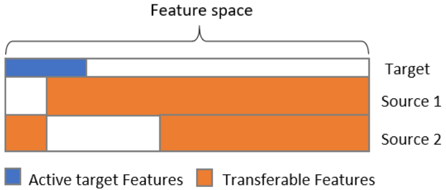
- Target task: $n_T = 50$, $s = 8$, $\boldsymbol{\beta}^{(0)} = (\mathbf{1}_s^\top, \mathbf{0}_{p-s}^\top)^\top$, $\epsilon_i^{(0)} \sim N(0,1)$.
- Source task: $n_S = 200$, $K = 2$, $\epsilon_i^{(k)} \sim N(0,1)$.

**Covariate shift:** Same as TransFusion.

# Simulation Settings for AdaTrans

**Model shift**:
We generate two source samples with non-overlapping transferable features:

- First source: the non-transferable $\delta^{(k)}$ is nonzero for the first $s/2$ elements;
- Second source: the non-transferable $\delta^{(k)}$ is nonzero from $(s/2+1)$-th to 25th elements.



# Methods to be compared

- Lasso (baseline): LASSO regression on the target task.
- TransGLM (Tian and Feng, 2022): TransLasso with source detection.
- AdaTrans: AdaTrans estimator.
- Oracle AdaTrans: Oracle AdaTrans estimator.

# Simulation Results: AdaTrans

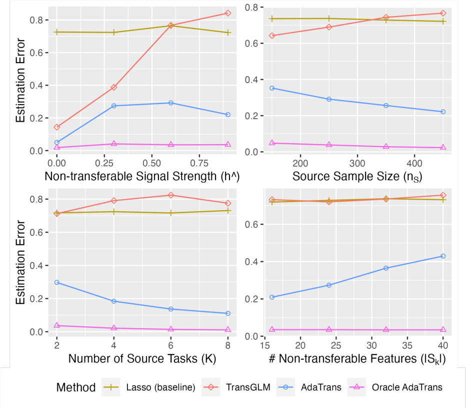



Figure: Estimation errors of different transfer learning methods.

- AdaTrans can also auto-detect and filter out non-transferable features.

# Conclusion

We proposed a new transfer learning framework that is robust to covariate shift and adaptive to feature-specific transferable structure.

- **TransFusion:** Conducting a fused-regularization based "joint training + debiasing" to achieve covariate-shift robustness.
- **D-TransFusion:** Incorporating intermediate estimators from different machines into TransFusion with one-shot communication.
- **AdaTrans:** Utilizing folded-concave penalization to auto-detect transferable structure while estimating parameters.
- Non-asymptotic bounds of estimation errors for all proposed estimators are established.

# Thank You!