Report on Transfer Learning for Non-parametric Regression

Adrian Cao, Chuanhui Peng

May 3, 2023

1 Motivation and Goal of the Project

Transfer learning is a popular technique that is used in Machine Learning and Statistical Learning. Specifically, transfer learning involves training a model on a source task or domain and then using the learned knowledge to improve the performance of the model on a related but slightly different target task or domain. In nonparametric statistics, transfer learning can be particularly useful because it can help to leverage the information learned from one set of data to improve the analysis of another set of related data. In nonparametric regression, transfer learning can be used to improve the prediction accuracy of a model on a target dataset by incorporating knowledge from a source dataset that has similar features or patterns. Here, we illustrate how we could use the idea of **Confidence Threshold Algorithm** by **Tony Cai** to achieve transfer learning in nonparametric regression.

The common setting is transfer learning for a single source domain is given by the following. Let's denote the source domain by $\mathcal{S}$ and the target domain by $\mathcal{T}$. Let $X_S = \{x_{S,1}, x_{S,2}, ..., x_{S,n_S}\}$ and $Y_S = \{y_{S,1}, y_{S,2}, ..., y_{S,n_S}\}$ denote the source dataset with n_S observations, where $x_{S,i}$ is a vector of predictor variables and $y_{S,i}$ is the corresponding response variable. In the target domain, the samples are independent and identically distributed(i.i.d.), $(X_{S,i}, Y_{S,i}) \overset{\text{iid}}{\sim} S, i = 1, \ldots, n_S$ with

$$Y_{S,i} = f(X_{S,i}) + \epsilon_i, \epsilon_i \overset{\text{iid}}{\sim} (0, \sigma).$$

Here, $f(x)$ is the unknown function of interest that demonstrates the relationship between X_S and Y_S and ϵ_i is the i.i.d random noise with conditional noise that is equal to zero.

Let $X'_T = \{x'_{T,1}, x'_{T,2}, ..., x'_{T,n_T}\}$ denote the target dataset with n_T observations, where $x'_{T,i}$ is a vector of predictor variables. And $Y'_T = \{y'_{T,1}, y'_{T,2}, ..., y'_{T,n_T}\}$ is the response vectors, where $(X'_{T,i}, Y'_{T,i}) \overset{\text{iid}}{\sim} T, i = 1, \ldots, n_T$ with

$$Y'_{S,i} = g(X'_{S,i}) + \epsilon_i, \epsilon_i \overset{\text{iid}}{\sim} (0, \sigma).$$

Our goal is to learn a regression function g for the target domain through the information we got from f.

There are two popular settings for transfer learning, which are covariate shift and posterior shift. Here, the covariate shift means the conditional mean functions from the target and the source domain, $f\&g$, are the same, but the marginal distributions of the covariates, X_i and X_i', are different. For the posterior shift, the mean function, $f\&g$, are different, but the marginal distributions of the covariates, X_i and X_i', are the same. Here, we are using the posterior shift settings and assume that the difference between the mean functions, f and g, can be well approximated by a polynomial function of a given order in L_1 distance. And we measure this distance by the bias strength, ε. If $\varepsilon = 0$, then we identify f and g differ by a polynomial, which means transfer learning could be really beneficial. We called such g as a "perfect reference" of f.

There are basically two goals of transfer learning:

- Accurately quantify the contribution of observations from the source domain to the regression task in the target domain.
- Develop an optimal transfer learning algorithm.

2 Problem Statement and Setup

Definition 2.1 (Hölder class) *For any positive numbers β, L, the Hölder function class $\mathcal{H}(\beta, L)$ is defined to be the set of all functions h with contin-*

uous partial derivative of order $\omega(\beta)$ that satisfy

$$\max_{|t|\le\omega(\beta)} \sup_{x\in[0,1]^d} |D^t h(x)| + \max_{|t|=\omega(\beta)} \sup_{x\neq y\in[0,1]^d} \frac{|D^t h(x) - D^t h(y)|}{\|x-y\|^{\beta-|t|}} \le L \qquad (1)$$

Here, both the source functions $f \in \mathcal{H}(\beta_S, L_S)$ and the target function $g \in \mathcal{H}(\beta_T, L_T)$ belongs to the Hölder class and we call f β_S-smooth and g β_T-smooth.

Definition 2.2 *For any $\epsilon > 0$, $M > 0$, positive integer D, the class of functions $\Psi(\epsilon, M, D)$ is defined as all the functions h such that there exists a polynomials function $\psi(x) \in Poly(M, D)$, i.e., the polynomial functions whose degree is smaller than or equal to D and coefficients have absolute values upper bounded by M, s.t.,*

$$\|\psi - h\|_1 = \int_{[0,1]^d} |\psi(x) - h(x)| dx \le \epsilon$$

Here, we assume $f - g \in \Psi(\varepsilon, M, D)$. This assumption requires that g is close in L_1 distance to f plus a polynomial of order D

Definition 2.3 *For any $u_1 > 0, u_2 > 0$, the class of sub-Gaussian random variables $G(u_1, u_2)$ are defined as all random variables Z such that for any $t > 0$,*

$$\mathbb{P}(|z| > t) \le u_1 \cdot e^{-u_2 t^2}$$

The assumption is that for any $x \in [0,1]^d$, the random noises $\epsilon_1 | X_1 = x$ and $\epsilon_1' | X_1' = x$ are sub-Gaussian to some constant u_1, u_2. This assumption ensures the outcome is not heavy-tailed. Assume the marginal distribution of $X_1 \sim f^T$ and $X_1' \sim f^S$, that is bounded by C^L and C^U. Hence, the parameter space is thus defined

$$\begin{aligned}
&F(\beta_S, \beta_T, \epsilon, M, D, \mu_1, \mu_2, L_S, L_T, C^L, C^U) = \\
&\qquad \{(S,T) : f \sim \mathcal{H}(\beta_T, L_T), g \sim \mathcal{H}(\beta_S, L_S), f - g \sim \Psi(\epsilon, M, D), \\
&\qquad\quad z_1 | X_1, z_1' | X_1' \sim G(u_1, u_2), C^L \leq f^S, f^T \leq C^U\}
\end{aligned}$$

For our problem, we are mostly interested in investigating how β_T,β_S, and ϵ, as the most controllable parameters, would influence the estimation risk

$$R_F(n_T, n_S) = \inf_{\hat{f}} \sup_{(S,T) \in F} \mathbb{E}((\hat{f} - f))^2$$

over this parameter space, which provides understanding about when and how the transfer learning would improve the estimation accuracy in the target domain.

3 Formulation of the Algorithm

3.1 Local Polynomial Regression

Local polynomial regression (LPR) is a non-parametric statistical method used for modeling the relationship between a response variable and one or more predictor variables. In LPR, the regression function is approximated by a polynomial function of a certain degree, with the coefficients of the polynomial varying locally depending on the nearby data points. The degree of the polynomial is typically chosen based on the trade-off between bias and variance and can be selected using techniques such as cross-validation or information criteria.

For one-dimensional observations $\mathcal{D} = \{(x_i, y_i), i = 1, 2, \cdots, n\}$, polynomial degree l, the local polynomial regression estimate $\hat{f}_{lpr}$ is given by

$$\hat{f}_{lpr} = \sum_{j=0}^{l} \hat{\beta}_j (x - x_0)^j$$

where

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=0}^{n} (y_i - \sum_{j=0}^{l} \beta_j (x_i - x_0)^j) K_h(x_0 - x_i)$$

3.2 The Confidence Thresholding(CT) Algorithm

3.2.1 The Confidence Thresholding Estimator

As we have data from both the target domain and the source domain, the most important step is integrating the information from both domains. This step is called **confidence thresholding**. Confidence thresholding estimator is designed to estimate a function when one has two different estimates. Suppose we have two different estimators $\hat{f}_1$ and $\hat{f}_2$ for some unknown function f. $\hat{f}_1$ converges slower and $\hat{f}_2$ converges faster but is slightly biased, which means $\hat{f}_2$ converges to a function that is different from but close to f in L_1

distance. The confidence thresholding estimator is constructed based on $\hat{f}_1$ and $\hat{f}_2$ as follows. Let e_1 be an upper bound of the L_∞ norm of $\hat{f}_1 - f$. Then a "confidence interval" for f_x is $[\hat{f}_1 - e_1, \hat{f}_1 + e_1]$ for all $x \in [0, 1)$. There are three different possible cases of the relationship between this confidence interval and $\hat{f}_2(x)$. If $\hat{f}_2(x)$ is greater than the upper bound of the confidence interval, then this confidence interval upper bound is better than $\hat{f}_2(x)$ in estimating $f(x)$ and in this case, the confidence interval upper bound is used as the estimate. If $\hat{f}_2(x)$ is in the confidence interval, then $\hat{f}_2(x)$ is acceptable and we use $\hat{f}_2(x)$. If $\hat{f}_2(x)$ is smaller than the lower bound of the confidence interval, then the confidence interval lower bound is better than $\hat{f}_2(x)$ in estimating $f(x)$, and the confidence interval lower bound is used as the estimate. We call this estimator the confidence thresholding estimator:

$$\hat{\mu}(\hat{f}_1(x), \hat{f}_2(x), e_1) = \hat{f}_1(x) + sgn(\hat{f}_2(x) - \hat{f}_1(x)) \cdot \min(|\hat{f}_2(x) - \hat{f}_1(x)|, e_1)$$

The paper introduces a lemma which we skip the claim here. With that lemma, there is a conclusion of the comparison between the confidence thresholding estimator with the two original estimators: the L_2 error of $\hat{f}_1$ is upper bounded by $2e_1$ and the L_2 error of $\hat{f}_1$ can be arbitrarily large since $\|\tilde{f}_2\|_2$ can be arbitrarily large. The L_2 error of the confidence thresholding estimator is upper bounded by $2e_1$, So it is at least as good as $\hat{f}_1$ up to a constant. Besides, under some regularity conditions, the confidence thresholding estimator outperforms both of two original estimators.

3.2.2 The Confidence Thresholding Algorithm

We utilize the confidence thresholding estimator and fit local polynomial regression twice to produce two elemantary estimators. First, we split the samples in the target domain $\mathcal{D}_{\mathcal{T}}$ randomly into two equalized subsets $\mathcal{D}_{T,1}$ and $\mathcal{D}_{T,2}$. We first fit local polynomial regression on $\mathcal{D}_{T,1}$ with some bandwidth $\tilde{b_t}$ and obtain an etimate $\hat{f}_{ref}$. We also construct a confidence interval and compute its length, which is denoted by L_{CI}. In the confidence thresholding estimator, $\hat{f}_{ref}$ will serve as $\hat{f}_1$ and $L_{CI}/2$ will serve as e_1. We then fit another local polynomial regression on $\mathcal{D}_{\mathcal{S}}$ to mimic $\hat{f}_2$. To get a faster convergence, we fit this local polynomial regression on the dataset with some larger bandwidth $\tilde{b}$. Let this estimate be $\hat{f}_{raw}$. // Note $f - g$ is close to a polynomial function in L_1 distance. Thus, the $\hat{f}_{raw}$ plus some polynomial function ϕ should be close to f. Since ϕ is unknown, we first setimate it and then plug the estimate into the confidence thresholding estimator. The estimator $\hat{\phi}$ is obtained by minimizing the empirical mean squared error on the validation set $\mathcal{D}_{\mathcal{T},\in}$. Formally,

$$\hat{\phi} = \underset{\phi \in Poly(\sqrt{\ln(n_T)}, l)}{\arg\min} \sum_{i=\lfloor \frac{n_T}{2} \rfloor + 1}^{n_T} [Y_i - \hat{f}_{ct}(X_i; \phi)]^2$$

. Finally, we get our confidence thresholding estimator:

$$\hat{f}_{ct}(x; \phi) = \tilde{\mu}_{ct}(\hat{f}_{ref}(x), \hat{f}_{raw}(x) + \phi(x), L_{CI/2})$$

4 Simulation Study

In this sectin, we evaluate the performance of the CT algorithm through simulations and compare it to existing methods.In the experiment, we fix other parameters and vary the sample size. We set the dimension to 1, the covariate distributions on both the source and target domains to uniform distribution on $[0, 1]$, and the random noise on both domains to normal random variables with zero mean and standard deviation 1/3. We evaluate the performance of all algorithm using mean squared error(MSE), which is the expected squared L_2 distance between the estimator and true mean function. Specifically we let the mean functions be

$$f(x) = \sin(10\pi x) + x^{3/2} - 0.1x + (0.1 - |x - 0.5|)_+$$

$$g(x) = \sin(10\pi x) + x^{3/2}$$

.

Therefore g differ from f by a linear function and a small spike with width 0.2 and height 0.1. In this case $f \in \mathcal{H}(1, 40)$ and $g \in \mathcal{H}(3/2, 40)$.The sample size of the target domain is fixed at 200. The sample sizes of the source domain are taken to be (300,600,1200,2400,4800). In this series of experiments, g is smoother than f and n_S is greater than n_T, so g is easier to estimate. We compare the performance of the CT algorithm to that of local polynomial regression using only data from the target domain. By comparing the performance of CT to local polynomial regression, we are able to gauge the improvement gained through transfer learning with various sample sizes from the target domain.

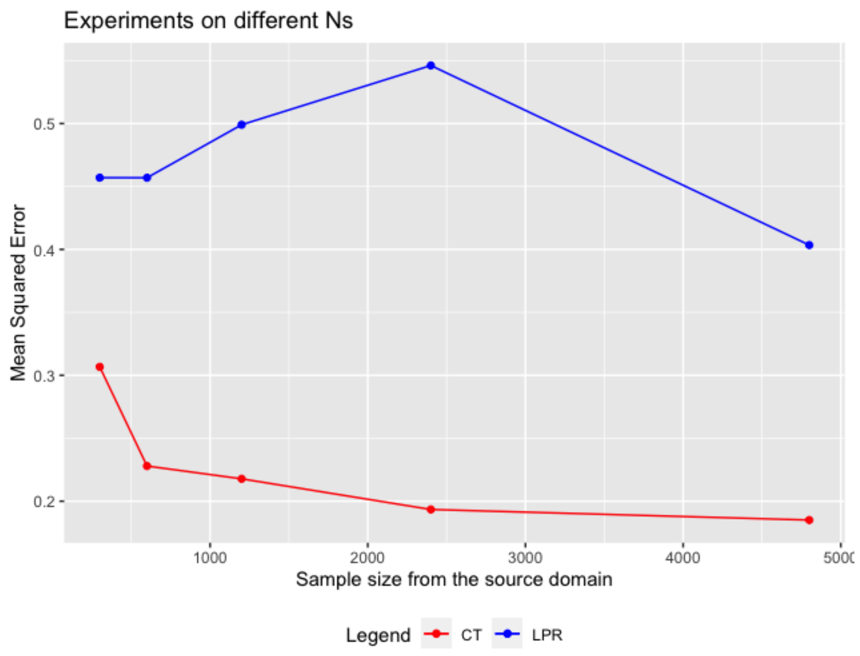

Figure 1: Experiments on different sample size of the source domain

Figure 1 presents the results of this experiment. The plot clearly demonstrates the gap in performance between the CT algorithm and local polynomial regression as predicted by theory. Additionally, the plot indicates that the CT algorithm's performance improves as the sample size from the source domain increases, however, this improvement seems to level off when n_t is large ($n_t > 2400$).

5 Further Discussion and Extensions

We found that our estimated mean squared error was not ideal due to our bandwidth selection. We conducted an experiment using cross-validation to select the bandwidth and found that transfer learning still improved the performance of single local polynomial regression while increasing the sample size of the source domain did not have a significant impact on convergence.

There were some limitations to our study, including our inability to investigate the bias strength's impact on the estimation or when there is no information about the function's smoothness. We also need to draw attention to some caveats, including that our confidence interval is entirely data-based, which could lead to bias. Additionally, our estimation of the distance between f and g is limited to simple polynomials contained in lower degrees.

A code

Code would be attached in the file 496 Project.R

References

[1] Cai T. Tony and Pu Hongming. "TRANSFER LEARNING FOR NON-PARAMETRIC REGRESSION: NON- ASYMPTOTIC MINIMAX ANALYSIS AND ADAPTIVE PROCEDURE". In: *Submitted to Annals of Statistics* ().