Post-Prediction Inference

Overview of Paper

- Main focus is improving statistical inference when machine learning-predicted labels are used.
- Particularly relevant in genomics where large, partially observed data is common.
- Evaluates classical and competing methods.
    - Highlight: assumption lean
- Proposed method achieves narrower intervals and outperforms other methods in simulations and applications.

Formal Statement

(X_i, Z_i, Y_i) denotes our labeled data, where i=1, 2, …, n

(X_i, Z_i) denotes our unlabeled data, where i=n+1, n+2, …, n+N.

Suppose that we train a machine learning model *f* on the labeled data, and use this to impute labels v_{n+1}, …, v_{n+N}.

Our challenges:

1. treating the imputed labels as true while maintaining appropriate coverage for confidence intervals
2. Using the imputed labels in some way, instead of disregarding unlabeled data during inference.

Example of problem

X_i = (age of the *i*th person, household income of the *i*th person, hours of study, …)

Y_i= real test score

Z_i = performance on a practice test

Suppose we only view the seniors' test scores. In this case, our unlabeled data would be junior students who took a practice test, but not the real test.

If we use some machine learning model (e.g. random forests or linear regression) to predict the juniors' test scores from (X_i, Z_i) we'd expect that it would give us *some* information about the juniors' real performance.

But, it would be wrong to use our predictions as if they were real, because our machine learning model could be wrong.

How do we use these predictions obtained from a machine learning model in a valid way?

Method

- R studio (packages tidyverse, randomForest)

Step 1: set up the variables X_i, Y_i, Z_i

```r
set.seed(1)

X <- rnorm(50, mean = 0, sd = 1)
Z <- rnorm(1, mean = 0, sd = 1)
theta <- c(rep(0.1/sqrt(10), 10), rep(0, 40))
r <- 0.8
tau_epsilon <- 1 - sum(theta^2)-r^2 # since we want Var[Y_i]=1
epsilon <- rnorm(1, mean = 0, sd = sqrt(tau_epsilon))
Y <- sum(X * theta) + r * Z + epsilon
#head(Y)
```

Generate a single observation for X, Y, Z, using a specific theta and epsilon to make sure $Var[Y_i]=1$

Step 2: Create labeled data

```r
n <- 500

X_l <- vector(mode = "list")
Y_l <- vector(mode = "list")
Z_l <- vector(mode = "list")

for (i in 1:500) {

  X_l[[i]] <- rnorm(50, mean = 0, sd = 1)
  Z_l[[i]] <- rnorm(1, mean = 0, sd = 1)
  theta <- c(rep(0.1/sqrt(10), 10),
             rep(0, 40))
  r <- 0.8
  tau_epsilon <- 1 - sum(theta^2)-r^2 # since we want Var[Y_i]=1
  epsilon <- rnorm(1, mean = 0, sd = sqrt(tau_epsilon))
  Y_l[[i]] <- sum(X_l[[i]] * theta) + r * Z_l[[i]] + epsilon

}
```

Now expand to a list of 500 observations of labeled data

Step 3: Create Unlabeled Data

```r
N <- 500

X_u <- vector(mode = "list")
Y_u <- vector(mode = "list")
Z_u <- vector(mode = "list")

for (i in 1:500) {

  X_u[[i]] <- rnorm(50, mean = 0, sd = 1)
  Z_u[[i]] <- rnorm(1, mean = 0, sd = 1)

}
```

Now create a list of 500 observations of unlabeled data

Step 4: Data Preparation

```r
# Convert list of X into a matrix
df_l <- data.frame(matrix(unlist(X_l), ncol = 50, byrow = TRUE))
# Name the columns X1, X2, ..., X50
colnames(df_l) <- paste0("X", 1:50)

df_l$Z <- unlist(Z_l)  # Add Z as a column
df_l$Y <- unlist(Y_l)  # Add Y as the outcome variable

head(df_l)
```

```r
df_u <- data.frame(matrix(unlist(X_u), ncol = 50, byrow = TRUE))
# Name the columns X1, X2, ..., X50
colnames(df_u) <- paste0("X", 1:50)

df_u$Z <- unlist(Z_u)  # Add Z as a column
# df$Y_u <- unlist(Y_u)  # Add Y as the outcome variable
# df_u[19,] == c(X_u[[19]], Z_u[[19]])
head(df)
```

Converts the list of labeled data and unlabeled data into a dataframe format suitable for model training, including adding variable names.

Step 5: Train the random forest model using labeled data

```r
# Train the random forest model
rf_l <- randomForest(Y~., data = df_l, ntree = 100)
# View model summary
print(rf_l)
```

```
Call:
 randomForest(formula = Y ~ ., data = df_l, ntree = 100)
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 17

          Mean of squared residuals: 0.4375333
                    % Var explained: 56.93
```

Step 6: Predict with the random forest model to impute labels

```
```{r fit unlabeled}

rf_predict <- predict(rf_l, newdata = df_u)
summary(rf_predict)

Y_u <- as.list(rf_predict) # a vector to list

# df_l$Y_hat <- predict(rf_l, select(df_l, -c("Y")))
```
```

Step 7: Compute PSPA estimator for regression coefficient

```{r theta_hat_PSPA}
# Assuming omega_opt is an identity matrix for simplification
# prepare data as matrices
X_l <- as.matrix(df_l[, -which(names(df_l) %in% c("Y", "Z"))])
Y_l <- matrix(unlist(Y_l), ncol = 1)
Z_l <- matrix(unlist(Z_l), ncol = 1)
X_u <- as.matrix(df_u[, -which(names(df_u) == "Z")])
Z_u <- matrix(unlist(Z_u), ncol = 1)
```

Set up the labeled and unlabeled data as matrices so we can put them in the estimator formula

```
w<- n/(n+N)
omega_opt <- diag(w, 50, 50)
sxx_l <- (t(X_l) %*% X_l) / n
sxy_l <- (t(X_l) %*% Y_l) / n
sxx_u <- (t(X_u) %*% X_u) / N
sxy_u <- (t(X_u) %*% rf_predict) / N
# dim(t(X_u)) should be 50 500 to multiply rf_predict which has length 500
# length(rf_predict)

term_1 <- solve(sxx_l + omega_opt %*% (sxx_u - sxx_l))
term_2 <- (sxy_l + omega_opt %*% (sxy_u - sxy_l))
```

Set up the terms for the estimator formula

```
theta_PSPA <-  term_1 %*% term_2

```

Takeaways

- Gained an understanding of how machine learning models can be used to predict outcomes in real-world applications.
- Learnt all about the computational processes behind machine learning predictions and how to design them effectively.

References

Miao, Jiacheng, et al. Assumption-Lean and Data-Adaptive Post-Prediction Inference. arXiv:2311.14220, arXiv, 16 Sept. 2024. arXiv.org, http://arxiv.org/abs/2311.14220.