Homework 2: Loss estimation

Vito Založnik

Generating dataset

```r
toy_data <- function(n, seed = NULL) {
set.seed(seed)
x <- matrix(rnorm(8 * n), ncol = 8)
z <- 0.4 * x[,1] - 0.5 * x[,2] + 1.75 * x[,3] - 0.2 * x[,4] + x[,5]
y <- runif(n) > 1 / (1 + exp(-z))
return (data.frame(x = x, y = y))
}
log_loss <- function(y, p) {
-(y * log(p) + (1 - y) * log(1 - p))
}
```

A Proxy For True Risk

How did I determine that 100000 is enough to reduce the error to the 3rd decimal digit?

Because of asymptotic normality. Error is decreasing with the "speed" of sqaure root of number of samples. Sqrt(100.000) > 100 thus error is reduced to the 3rd decimal digit.

```r


df_dgp <- toy_data(100000, seed=0)
```

Helper functions

Model loss estimator variability due to test data variability

Generate a toy dataset with 50 observations.

```r
toy_dataset50 <- toy_data(50, seed=0)
```

Train model h using a learner (logistic regression).

```r
model <- function(data) {
 h <-glm(y ~ .,data=data, family = binomial(link = "logit"))
 return(h)
}
```

Compute the true risk proxy for h using the huge dataset

Computing log loss risk of the model with input data model and test set.

```r
log_risk_model <- function(model, test_set){
  log_loss(test_set$y, predict(model, newdata=test_set, type="response"))
}
```

Evaluation of a true risk proxy of a model on a huge dataset:

```r
model_proxy <- function(model){
  log_risk_model(model, df_dgp)
}
```

Computing on data:

```r
model50 <- model(toy_dataset50)
true_risk <- mean(model_proxy(model50))

sprintf("True risk proxy: %.4f", true_risk)
```

```
## [1] "True risk proxy: 0.5755"
```

Generate a toy dataset with 50 observations, estimate the risk of h using this dataset, compute the standard error of the estimate, record if the 95% CI contains the true risk.

Below is implementation of calculating standard error and 0.95 confidence intervals from scratch. We used unbiased estimator for variance.

```r
serror <- function(x){
  n <- length(x)
  mu <- mean(x)
  var <- sum((x - mu)^2)/(n-1) #unbiased

  s <- sqrt(var/n)
  return(s)
}

ci95 <- function(x){
  mu <- mean(x)
  se <- serror(x)
  min <- mu - 1.96*se
  max <- mu + 1.96*se
  return(c(min, max))

}
```

Holdout estimation

```r
toy_dataset50_1 <- toy_data(50)

log_risk <- log_risk_model(model50, toy_dataset50_1 )


sprintf("SE of risk of h using new dataset: %.4f", serror(log_risk))
```

```
## [1] "SE of risk of h using new dataset: 0.1007"
```

```
ci_min <- ci95(log_risk)[1]
ci_max <- ci95(log_risk)[2]

sprintf("CI95 lower bound is: %.4f", ci_min)
```

```
## [1] "CI95 lower bound is: 0.2615"
```

```
sprintf("CI95 upper bound is: %.4f", ci_max)
```

```
## [1] "CI95 upper bound is: 0.6564"
```

```
bool <- true_risk >= ci_min & true_risk <= ci_max
sprintf("0.5-0.5 baseline true risk: %.4f", mean(log_loss(df_dgp$y, 0.5)))
```

```
## [1] "0.5-0.5 baseline true risk: 0.6931"
```

```
print("true risk proxy contained in confidence interval 0.95:")
```

```
## [1] "true risk proxy contained in confidence interval 0.95:"
```

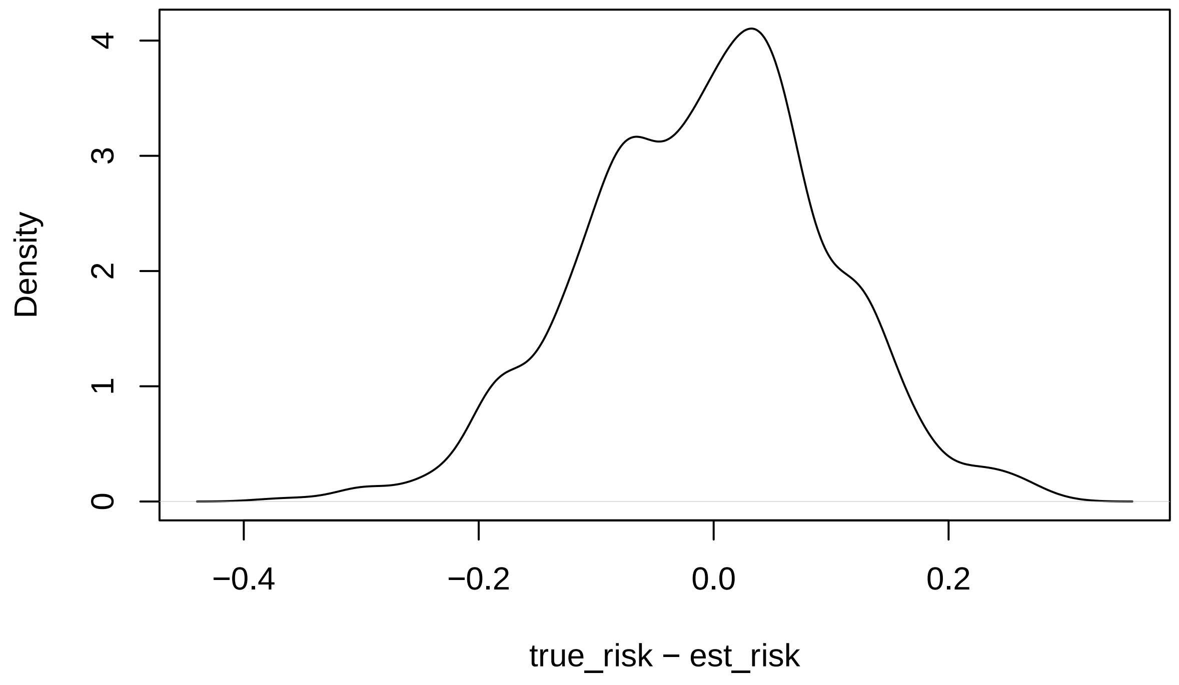
```
print(bool)
```

```
## [1] TRUE
```

Repeat it 1000 times:

```
diff <- c()
estimates <- c()
ci_contained <- c()
s_e <- c()
for (i in 1:1000) {
  data <- toy_data(50)
  log_risk <- log_risk_model(model50, data)
  mean_risk <- mean(log_risk)

  diff <- append(diff, true_risk - mean_risk, after = length(diff))
  s_e <- append(s_e, serror(log_risk))
  ci <- ci95(log_risk)
  mi <- ci[1]
  ma <- ci[2]
  if ((mi <= true_risk) & true_risk <= ma ){
    ci_contained <- append(ci_contained, 1)

  } else{
    ci_contained <- append(ci_contained, 0)
  }
}
```

```r
bias <- mean(diff)
median_se <- median(s_e)
plot(density(diff), xlab="true_risk - est_risk", main="")
```



```r
sprintf("bias: %.4f", bias)
```

```
## [1] "bias: -0.0061"
```

```r
sprintf("Median standard error: %.4f", median_se)
```

```
## [1] "Median standard error: 0.1106"
```

```r
sprintf("true risk: %.4f", true_risk)
```

```
## [1] "true risk: 0.5755"
```

```r
sprintf(" Percentage of 95CI that contain the true risk proxy: %.4f", mean(ci_contained)*100)
```

```
## [1] " Percentage of 95CI that contain the true risk proxy: 95.0000"
```

We can see that mean of differences between evaluated and true risk is almost centered at 0. This implies that holdout estimator is unbiased. Proxy is contained in the 95% confidence interval about 93% of the time. This implies that losses for test sample of size 50 are not yet normally distributed.

Increasing the size of train set would decrease differences between true and evaluated risk. Percentage of evaluated risk included in 95% CI should increase toward 0.95.

Overestimation of the deployed model's risk

```
differences <- c()
for (i in 1:50){
  data1 <- toy_data(50)
  data2 <- toy_data(50)
  data12 <- rbind(data1, data2)

  h1 <- model(data1)
  h12 <- model(data12)

  true_risk1 <- mean(model_proxy(h1))
  true_risk12 <- mean(model_proxy(h12))

  differences <- append(differences, true_risk1 - true_risk12)

}
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(differences)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.01944 0.00000 0.24114
```

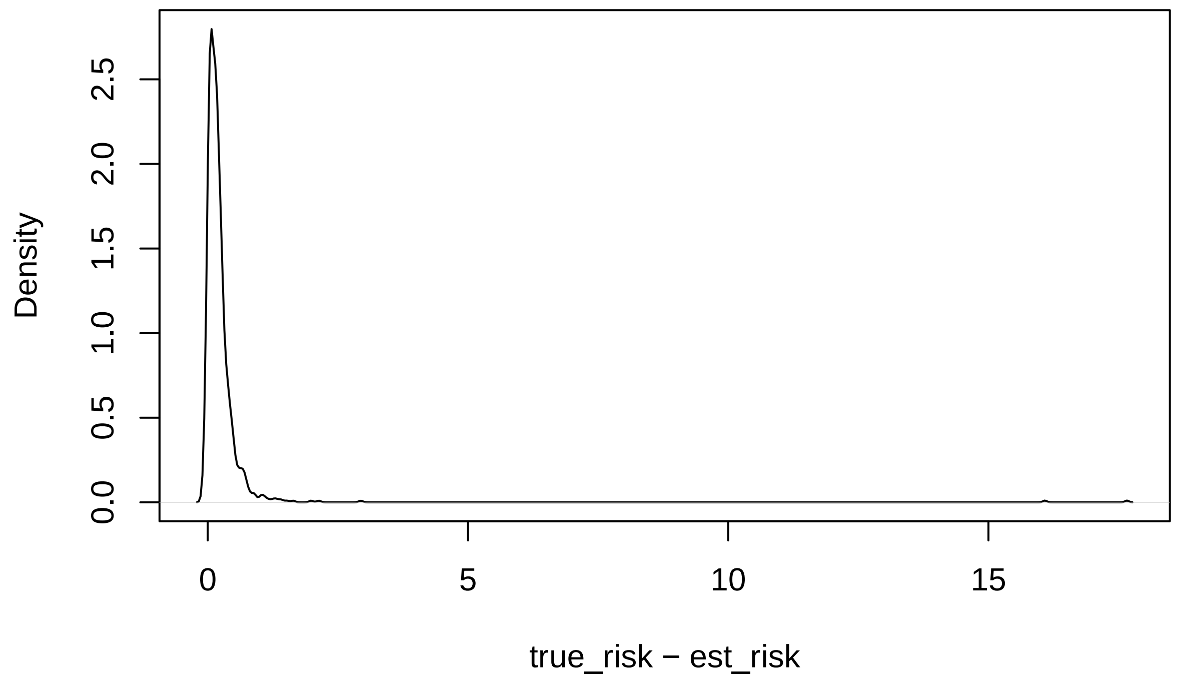
We can see that there is not much of a difference in risk between the models. But still the model trained on more data returned less risk. From this we can conclude that it's better to deploy a model using bigger (whole) dataset. Increasing the train datasets would result in decrease of the differences.

Loss estimator variability due to split variability

```r
data <- toy_data(100)
h0 <- model(data)
true_risk0 <- mean(model_proxy(h0))
differences <- c()
se <- c()
ci_contained <- c()

for (i in 1:1000){
  idx = sample(1:100, 50)
  vse <- c(1:100)
  idy <- vse[is.na(pmatch(vse,idx))]
  train <- data[idx,]
  test <- data[idy,]
  h_train <- model(train)
  log_risk <- log_risk_model(h_train, test)
  risk_est <- mean(log_risk)
  differences <- append(differences,   risk_est - true_risk0)
  se <- append(se, serror(log_risk))


  ci <- ci95(log_risk)
  mi <- ci[1]
  ma <- ci[2]
  if ((mi <= true_risk0) & true_risk0 <= ma ){
    ci_contained <- append(ci_contained, 1)

  } else{
    ci_contained <- append(ci_contained, 0)
  }

}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
bias <- mean(differences)
median_se <- median(se)
plot(density(differences), xlab="true_risk - est_risk", main="")
```



```r
sprintf("bias: %.4f", bias)
```

```
## [1] "bias: 0.2440"
```

```r
sprintf("Median standard error: %.4f", median_se)
```

```
## [1] "Median standard error: 0.1208"
```

```r
sprintf("true risk: %.4f", true_risk)
```

```
## [1] "true risk: 0.5755"
```

```r
sprintf(" Percentage of 95CI that contain the true risk proxy: %.4f", mean(ci_contained)*100)
```

```
## [1] " Percentage of 95CI that contain the true risk proxy: 80.6000"
```

From the low coverage of our 95% CI (84.4%), we can see that we overestimated our true risk based on the splits. Thus, the split greatly affect our risk estimation. With increasing of the size of dataset the variability across the splits would decrease, as the model would get more training instances.

Cross-validation

```r
k_split <- function(d, k, true_risk0){
  results <- c()
  n <- nrow(d)
  vse <- 1:n

  fold_size <- n %/% k
  for (j in 1:k){
    idy <- 1:n
    idx = sample(vse, fold_size)

    vse <- vse[is.na(pmatch(vse,idx))];
    idy <- idy[is.na(pmatch(idy,idx))];
    train <- d[idy,];
    test <- d[idx,];


    h <- model(train)
    log_risk <- log_risk_model(h, test)
    risk_est <- mean(log_risk)
    results <- append(results, risk_est)
  }
  ci <- ci95(results)
  mi <- ci[1]
  ma <- ci[2]
  if ((mi <= true_risk0) & true_risk0 <= ma ){
    return(c(mean(results), 1))

  } else{
   return(c(mean(results), 0))
  }


}
```

```r
loocv <- function(d, true_risk0){
  results <- c()
  n <- nrow(d)
  vse <- 1:n

  fold_size <- n-1
  for (j in 1:n){

    test <- d[j,];
    train <- d[-j,]


    h <- model(train)
    log_risk <- log_risk_model(h, test)
    risk_est <- mean(log_risk)
    results <- append(results, risk_est)
  }
  ci <- ci95(results)
  mi <- ci[1]
  ma <- ci[2]
  if ((mi <= true_risk0) & true_risk0 <= ma ){
    return(c(mean(results), 1))

  } else{
   return(c(mean(results), 0))
  }


}
```

```
data <- toy_data(100);
h0 <- model(data);
true_risk <- mean(model_proxy(h0));
sprintf("true risk: %.4f", true_risk)
```

```
## [1] "true risk: 0.5712"
```

```
fold2 <- c()
fold4 <- c()
fold10 <- c()
fold10_20rep <- c()

ci_2 <- c()

print("2-fold:")
```

```
## [1] "2-fold:"
```

```
for (j in 1:500){
  res <- k_split(data,2,true_risk)
  risk <- res[1]
  ci <- res[2]
  ci_2 <- append(ci_2, ci)
  fold2 <- append(fold2, risk)



}
  se <- serror(fold2)
  sprintf("Median standard error: %.4f", median(se))
```

```
## [1] "Median standard error: 0.0690"
```

```
  fold2 <- fold2 - true_risk
  sprintf("Mean difference: %.4f", mean(fold2))
```

```
## [1] "Mean difference: 0.6288"
```

```
  sprintf(" Percentage of 95CI that contain the true risk proxy: %.4f", mean(ci_2)*100)
```

```
## [1] " Percentage of 95CI that contain the true risk proxy: 79.0000"
```

```
  #plot(density(fold2), xlab="true_risk - est_risk", xlim=c(-1,3), main="2-fold")
```

```
print("4-fold:")
```

```
## [1] "4-fold:"
```

```
ci_4 <- c()
fold4 <- c()
for (j in 1:500){
  res <- k_split(data,4,true_risk)
  risk <- res[1]
  ci <- res[2]
  ci_4 <- append(ci_4, ci)
  fold4 <- append(fold4, risk)



}
  se <- serror(fold4)
  sprintf("Median standard error: %.4f", median(se))
```

```
## [1] "Median standard error: 0.0037"
```

```
  fold4 <- fold4 - true_risk
  sprintf("Mean difference: %.4f", mean(fold4))
```

```
## [1] "Mean difference: -0.0859"
```

```
  sprintf(" Percentage of 95CI that contain the true risk proxy: %.4f", mean(ci_4)*100)
```

```
## [1] " Percentage of 95CI that contain the true risk proxy: 76.8000"
```

```
  #plot(density(fold4), xlab="true_risk - est_risk", xlim=c(-1,3), main="4-fold")
```

```
print("10-fold:")
```

```
## [1] "10-fold:"
```

```
ci_10 <- c()
fold10 <- c()
for (j in 1:500){
  res <- k_split(data,10,true_risk)
  risk <- res[1]
  ci <- res[2]
  ci_10 <- append(ci_10, ci)
  fold10 <- append(fold10, risk)



}
  se <- serror(fold10)
  sprintf("Median standard error: %.4f", median(se))
```

```
## [1] "Median standard error: 0.0012"
```

```
  fold10 <- fold10 - true_risk
  sprintf("Mean difference: %.4f", mean(fold10))
```

```
## [1] "Mean difference: -0.1170"
```

```
  sprintf(" Percentage of 95CI that contain the true risk proxy: %.4f", mean(ci_10)*100)
```

```
## [1] " Percentage of 95CI that contain the true risk proxy: 90.4000"
```

```
  #plot(density(fold10), xlab="true_risk - est_risk", xlim=c(-1,3), main="10-fold")
```

```
print("loocv:")
```

```
## [1] "loocv:"
```

```
ci_l <- c()
f <- c()
for (j in 1:500){
  res <- loocv(data,true_risk)
  risk <- res[1]
  ci <- res[2]
  ci_l <- append(ci_l, ci)
  f <- append(f, risk)



}
  se <- serror(f)
  sprintf("Median standard error: %.4f", median(se))
```

```
## [1] "Median standard error: 0.0000"
```

```
  f <- f - true_risk
  sprintf("Mean difference: %.4f", mean(f))
```

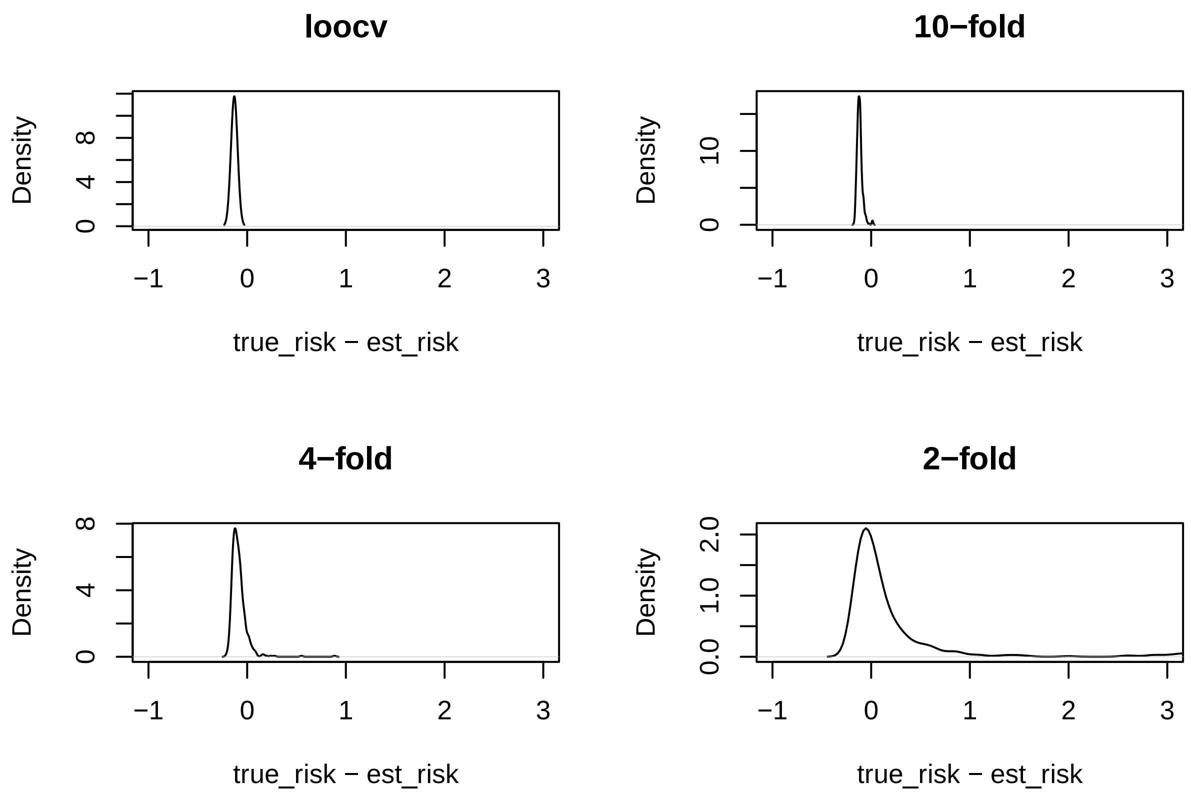
```
## [1] "Mean difference: -0.1305"
```

```
sprintf(" Percentage of 95CI that contain the true risk proxy: %.4f", mean(ci_l)*100)
```

```
## [1] " Percentage of 95CI that contain the true risk proxy: 100.0000"
```

```
#plot(density(f), xlab="true_risk - est_risk", xlim=c(-1,3), main="loocv")
```

```
par(mfrow=c(2,2))
plot(density(f), xlab="true_risk - est_risk", xlim=c(-1,3), main="loocv")
plot(density(fold10), xlab="true_risk - est_risk", xlim=c(-1,3), main="10-fold")
plot(density(fold4), xlab="true_risk - est_risk", xlim=c(-1,3), main="4-fold")
plot(density(fold2), xlab="true_risk - est_risk", xlim=c(-1,3), main="2-fold")
```



We can see that 2-fold cross-validation performs the worst with highest bias and overestimation of risk. 10-fold and loocv are quite consistent with very small bias. Increasing of the train set size resulted in increase of performance. Loocv and 10-fold cross validations computational time was far greater compared to others. we can also see that confidence intervals percentage increases and even exceeds 95% in some cases. With more folds we in general produce better results.