Loss estimation

Aljaž Konec

2024-03-23

Estimating the performance of a model is a crucial step in the machine learning pipeline. Here we discuss common estimation metrics for estimating the loss of a given model. First, we run intial helper functions that we use throughout the analysis. We generate a large dataset with 100,000 observations to estimate the true risk of the model. The standard error drops at the rate of **sqrt(n)** where n is the size of the dataset. For n=100,000 this results in improving the standard error by a factor of 316 which is enough to reduce the error between actual performance and our proxy to the 3rd decimal place.

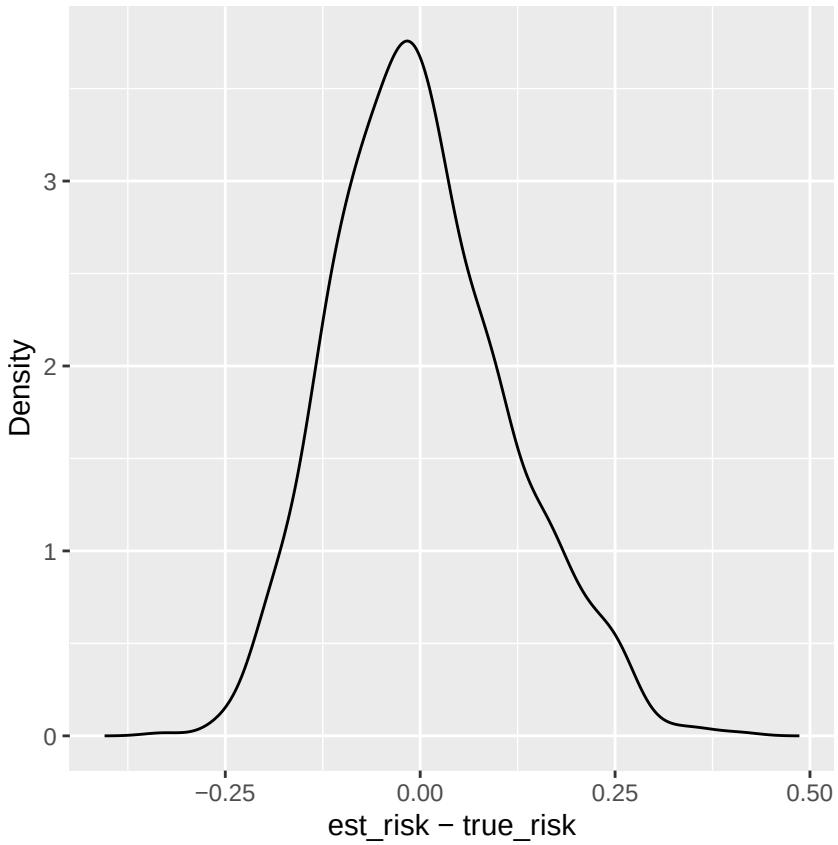
```
library(ggplot2)
library(patchwork)

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))
}

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

Holdout estimation

```
data <- toy_data(50,0)

h <- glm(y ~ ., data = data, family = binomial(link=logit))
pred <- predict(h, df_dgp, type = "response")
true_risk <- mean(log_loss(df_dgp$y, pred))

means <- c()
std_errors <- c()

contains_true_risk <- c()

for ( x in 1:1000){
    data <- toy_data(50, x+1000)

    pred <- predict(h, data, type = "response")
    loss <- log_loss(data$y, pred)
```

```
    means <- c(means, mean(loss))

    m <- mean(loss)
    std_error <- sd(loss)/sqrt(50)
    std_errors <- c(std_errors, std_error)
    ci.lower <- m - 1.96 * std_error
    ci.upper <- m + 1.96 * std_error

    contains_true_risk <- c(contains_true_risk, (ci.lower <= true_risk) & (ci.upper >= true_risk))

}
{
mean_difference <- abs(round(mean( means - true_risk), 4))
mean_observed_risk <- mean(means)

differences <- means - true_risk
density <- density(differences)
df <- data.frame(x = density$x, y = density$y)

always_05_risk <- mean(log_loss(df_dgp$y, rep(0.5, nrow(df_dgp))))
median_standard_error <- median(std_errors)

print(paste0("True risk proxy: ", round(true_risk, 4)))
print(paste0("Mean difference: ", round(mean_difference, 4)))
print(paste0("0.5-0.5 baseline true risk: ", round(always_05_risk, 4)))
print(paste0("Median standard error: ", round(median_standard_error, 4)))
print(paste0("Percentage of 95CI that conatin true risk proxy: ", mean(contains_true_risk)*100))
ggplot(df, aes(x, y)) +
    geom_line() +
    theme(aspect.ratio=1) +
    ylab("Density") +
    xlab("est_risk - true_risk")
}
```

```
## [1] "True risk proxy: 0.5755"
## [1] "Mean difference: 0.001"
## [1] "0.5-0.5 baseline true risk: 0.6931"
## [1] "Median standard error: 0.1098"
## [1] "Percentage of 95CI that conatin true risk proxy: 93.9"
```



We can observe that the estimated risk is very close to the true risk of the model. If the training set was larger the estimated risk would be even closer to actual risk. Increasing the size of the testing set would impact the standard error as it would decrease with sqrt(n).

Overestimation of the deployed models risk

```
diff_in_risk <- c()

for(i in 1:50){
    dataset1 <- toy_data(50, i)
    dataset2 <- toy_data(50, i+50)

    h1 <- glm(y ~ ., data = dataset1, family = binomial(link= logit))
    combined <- rbind(dataset1, dataset2)
    h2 <- glm(y ~ ., data = combined, family = binomial(link= logit))

    pred1 <- predict(h1, df_dgp, type = "response")
    pred2 <- predict(h2, df_dgp, type = "response")

    risk1 <- mean(log_loss(df_dgp$y, pred1))
    risk2 <- mean(log_loss(df_dgp$y, pred2))

    diff_in_risk <- c(diff_in_risk, risk1 - risk2)
}

summary(diff_in_risk)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.07669   0.02786   0.12974   0.17926   0.21089   0.86056
```

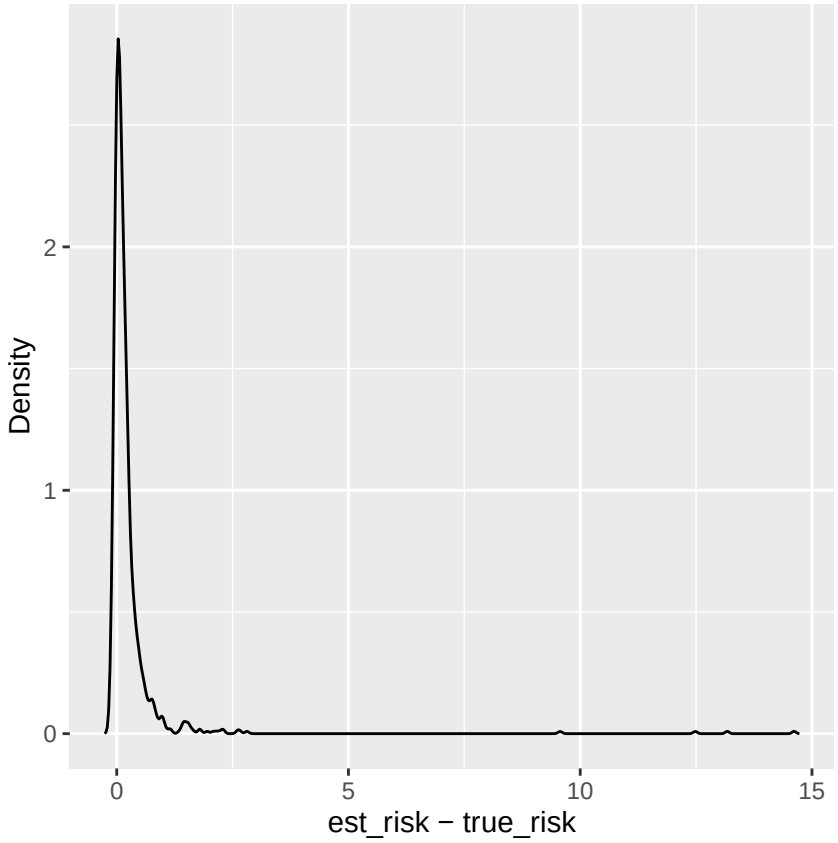
We observe that the risk of model **h1** is higher, which is due to it being trained on less data. This shows that a model trained on less data produces a pessimistic estimate of the risk of the model. This is due to the model being trained on less data and therefore having a higher bias in the estimated risk. Increasing the size of the dataset would produce a more representative sample of the data and therefore decrease the difference in risk between the two models.

Loss estimator variability due to split variability

```
data <- toy_data(100, 0)

h0 <- glm(y ~ ., data = data, family = binomial(link=logit))
true_risk <- mean(log_loss(df_dgp$y, predict(h0, df_dgp, type = "response")))

contains_true_risk <- c()
means <- c()
std_errors <- c()

for (i in 1:1000) {

    t <- sample(1:100, 50)

    training <- data[t, ]
    testing <- data[-t, ]

    h <- glm(y ~ ., data = training, family = binomial(link=logit))
    pred <- predict(h, testing, type = "response")
    loss <- log_loss(testing$y, pred)
    m <- mean(loss)
    means <- c(means, m)

    std_error <- sd(loss)/sqrt(50)
    std_errors <- c(std_errors, std_error)

    ci.lower <- m - 1.96 * std_error
    ci.upper <- m + 1.96 * std_error

    contains_true_risk <- c(contains_true_risk, (ci.lower <= true_risk) & (ci.upper >= true_risk))

}

density <- density(means - true_risk)
df <- data.frame(x = density$x, y = density$y)
{

print(paste0("True risk proxy: ", round(true_risk, 4)))
print(paste0("Mean difference: ", round(mean(means) - true_risk, 4)))
print(paste0("Median standard error: ", round(median(std_errors), 4)))
print(paste0("Percentage of 95CI that conatin true risk proxy: ", mean(contains_true_risk)*100))

ggplot(df, aes(x, y)) +
    geom_line() +
```

```
    theme(aspect.ratio=1) +
    ylab("Density") +
    xlab("est_risk - true_risk")
}
```

```
## [1] "True risk proxy: 0.5255"
## [1] "Mean difference: 0.2428"
## [1] "Median standard error: 0.1226"
## [1] "Percentage of 95CI that conatin true risk proxy: 85.8"
```



We capture the true risk in the 95% confidence interval only around 85% of the time. This suggests that estimated risk has high variability is dependant on the split of the data. Taking just one split and estimating the risk of the model would not be a good representation of the true risk of the model. If the dataset was larger, each data split would be more representative of the underlying distribution and therefore the variability would drop. Having a smaller dataset would have the opposite effect since the data splits would have a higher probability of a biased representation of the distribution.

Cross-validation

Below is the implementation of the cross-validation method and a utility function to run 500 repetitions of cross validation.

```
crossvalidation <- function(data, n, nfolds) {
    fold_size <- floor(n/nfolds)

    s <- sample(1:n, n)
    losses <- c()
```

```
    for (i in 1:nfolds) {
        test <- data[s[((i-1)*fold_size + 1):(i*fold_size)], ]
        train <- data[s[-(((i-1)*fold_size + 1):(i*fold_size))], ]

        h <- glm(y ~ ., data = train, family = binomial(link=logit))
        pred <- predict(h, test, type = "response")
        loss <- log_loss(test$y, pred)
        losses <- c(losses, loss)
    }

    return(losses[order(s)])
}


repeat500times <- function(n, nfolds, rep= FALSE){
    ctr <- c()
    means <- c()
    std_errors <- c()

    for (i in 1:500) {
        data <- toy_data(n, i)

        h0 <- glm(y ~ ., data = data, family = binomial(link=logit))
        true_risk <- mean(log_loss(df_dgp$y, predict(h0, df_dgp, type = "response")))

        if (rep){
            all_losses <- numeric(n)
            for (j in 1:20){
                losses <- crossvalidation(data, n, nfolds)
                all_losses <- all_losses + losses
             }
            losses <- all_losses/20
        }
        else {
            losses <- crossvalidation(data, n, nfolds)

        }

        mean_loss <- mean(losses)
        se <- sd(losses)/sqrt(n)

        ci_lower <- mean_loss - 1.96 * se
        ci_upper <- mean_loss + 1.96 * se

        ctr <- c(ctr, (ci_lower <= true_risk) & (ci_upper >= true_risk))
        means <- c(means, mean_loss - true_risk)
        std_errors <- c(std_errors, se)
    }
    return (list(ctr,means, std_errors))
}
```

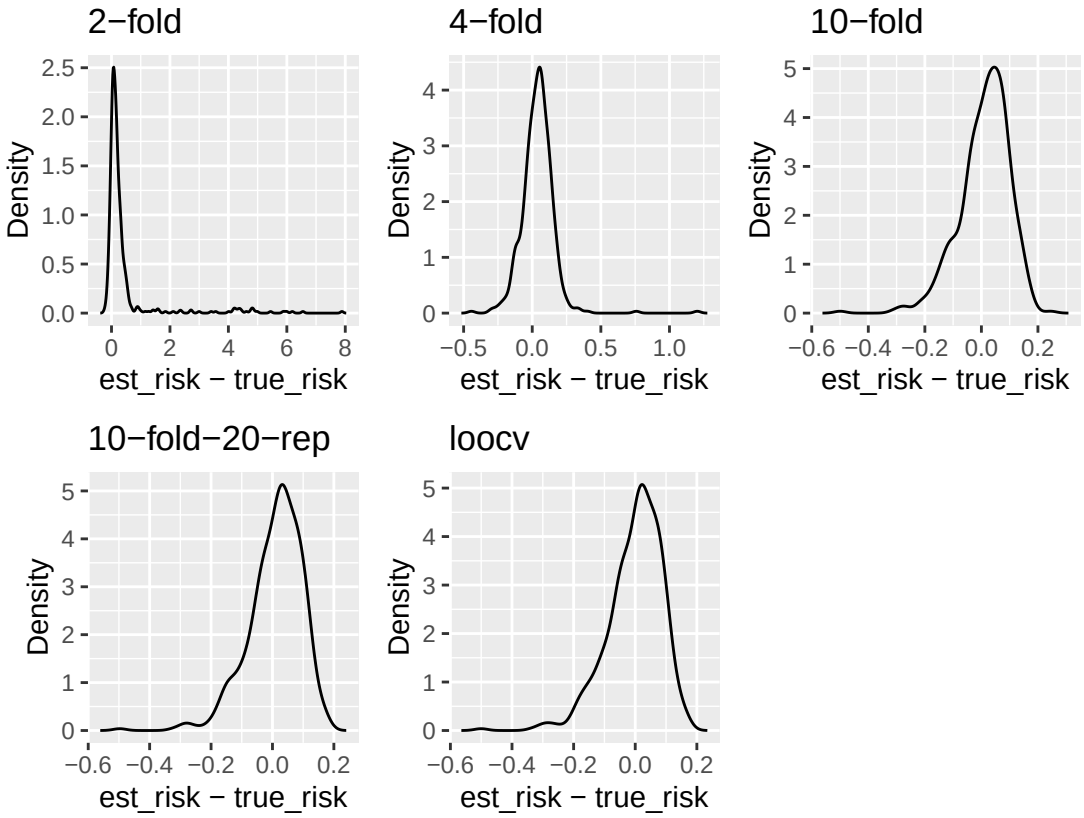
The size of the dataset is 100 and we perform cross-validation with 2, 4, 10 folds and 10 folds with 20 repetitions. Each cross-validation fold is run 500 times and the results are shown below.

```r
name <- list("2-fold" = 2, "4-fold" = 4, "10-fold" = 10, "10-fold-20-rep"= 10, "loocv" = 100)
densities <- list()

for (cv in names(name)) {

    if (cv == "10-fold-20-rep"){
        l <- repeat500times(100, name[[cv]], rep = TRUE)
        } else {
        l <- repeat500times(100, name[[cv]])
    }
    ctr <- l[[1]]
    means <- l[[2]]
    std_errors <- l[[3]]

    print(paste0("Number of folds: ", cv))
    print(paste0("Mean difference: ", mean(round(means, 4))))
    print(paste0("Median standard error: ", median(round(std_errors, 4))))
    print(paste0("Percentage of 95CI that conatin true risk proxy: ", mean(ctr)*100))
    d <- density(means)
    densities[[cv]] <- d

}
```

```
## [1] "Number of folds: 2-fold"
## [1] "Mean difference: 0.4250362"
## [1] "Median standard error: 0.1092"
## [1] "Percentage of 95CI that conatin true risk proxy: 68.8"
## [1] "Number of folds: 4-fold"
## [1] "Mean difference: 0.0387494"
## [1] "Median standard error: 0.083"
## [1] "Percentage of 95CI that conatin true risk proxy: 90.2"
## [1] "Number of folds: 10-fold"
## [1] "Mean difference: 0.0106702"
## [1] "Median standard error: 0.0777"
## [1] "Percentage of 95CI that conatin true risk proxy: 93.4"
## [1] "Number of folds: 10-fold-20-rep"
## [1] "Mean difference: 0.009884"
## [1] "Median standard error: 0.0767"
## [1] "Percentage of 95CI that conatin true risk proxy: 93.2"
## [1] "Number of folds: loocv"
## [1] "Mean difference: -0.0013174"
## [1] "Median standard error: 0.0747"
## [1] "Percentage of 95CI that conatin true risk proxy: 92"
```

```r
plots <- list()
for (cv in names(densities)) {
    # plot(densities[[cv]], main = cv, asp=1)
    d <- densities[[cv]]
    df <- data.frame(x = d$x, y = d$y)
    plots[[cv]] <-ggplot(df, aes(x, y)) +
                geom_line() +
                ggtitle(cv) +
                theme(aspect.ratio=1) +
                ylab("Density") +
```

```
                xlab("est_risk - true_risk")
}

plots[[1]] + plots[[2]] + plots[[3]] + plots[[4]] + plots[[5]]+ plot_layout(ncol = 3, nrow = 2)
```



Using cross validation with fewer folds decreases the amount of data that is available for training. This results in a higher bias in the estimated risk when training models. This is most apperant at with 2-fold CV where the mean difference is the highest. Increasing the number of folds to 10 reduces the bias to almost 0 while also decreasing the standard error. Peroforming 10-fold CV with 20 repetitions does not significantly change the estimation of 10-fold CV. The implications here would be to pick a suitable fold size based on the amount of available data.