Loss estimation homework

Žiga Trojer

7/5/2021

A proxy for true risk

we will be using huge generated data set as a proxy for the DGP and determining the ground-truth true risk of our models. First, we generate our data set with 100,000 data points.

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

Then we can check that a model's risk on this data set differs from it's true risk at most on the 3rd decimal digit. Let's have a general model that gives probabilities uniformly on [0,1]. Let's generate 100,000 such probabilities and let us estimate the error (using logistic regression learner)

```
set.seed(1)
p <- runif(100000)
log_l <- log_loss(df_dgp$y, p)
el <- mean(log_l)
tre <- sd(log_l) / sqrt(100000)
print(paste0("Expected loss: ", round(el, 4)))
```

```
## [1] "Expected loss: 1.0017"
```

```
print(paste0("Estimated error: ", round(tre, 4)))
```

```
## [1] "Estimated error: 0.0032"
```

We calculated expected loss with Monte Carlo integration and estimated the error, which is approximately 0.003. This means that the the model's risk on this data differs from it's true risk at most on the 3rd decimal digit.

Holdout estimation

Holdout estimation is the most common approach to estimating model's risk. Model's risk on the test data is just an estimate of the model's true risk and that's why we always need to report quantification of uncertainty, such as standard errors or 95% confidence intervals. We will investigate the sources of variability and bias that contribute to the difference between the test data risk estimate and the true risk of a model trained on the training data.

We will use Logistic Regression throughout the homework, so we will first demonstrate the process of fitting and predicting, as it is largely repeated over the next sections. Also, we will calculate true risk proxy of a model. First, we generate our train toy data set with 50 observations

```r
toy_50 <- toy_data(50, 0)
```

We are using Bernoulli-logit GLM - training model h using previously defined train data

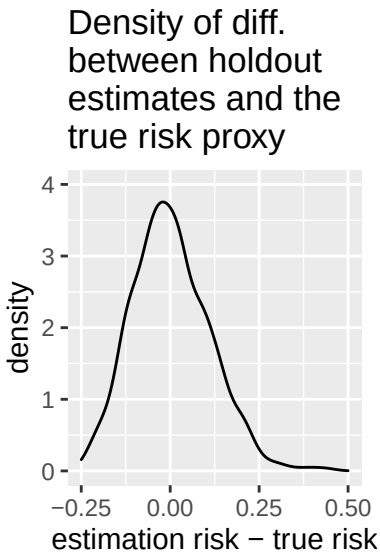
```r
h <- glm(y ~ ., family = binomial(link='logit'), data = toy_50)
```

Computing the true risk proxy for h using the huge data set df_dgp

```r
predictions <- predict(h, newdata = df_dgp[, -9], type = 'response')
rsk <- mean(log_loss(df_dgp$y, predictions))
print(paste0("True risk proxy: ", round(rsk, 4)))
```

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

The following plot shows differences differences between the estimates and the true risk proxy



We also computed true risk proxy, average difference between the estimate and the true risk, true risk of 0.5-0.5 predictions, median standard error and percentage of 95CI that contains the true risk proxy:

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

```
## [1] "Mean difference: 2e-04"
```

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

```
## [1] "Median standard error: 0.1094"
```

```
## [1] "Percentage of 95CI that contain the true risk proxy: 93.4"
```

First think we notice from the graph is that the mode is a bit to the left. Shape is asymmetric - a bit longer tails on the right side. First we could think that holdout estimation is biased, but because the distribution is skewed, this is not the case (mean difference around zero). Here we are using independent test set (as we are

generating a new toy data set with 50 observations), so this also tells us the holdout estimation is unbiased, as we are estimating loss on completely independent set. Also, holdout estimation is consistent (by the law of large number). Because of these two good properties, we should always use holdout estimation with the independent test set, if we have enough data.

Let us return to the longer tails on the positives. We think that this has to do something about the nature of log-loss function. It punishes mistakes more than it rewards good things (because logarithm diverges close to 0). From the longer tails we can also see that sometimes a bad model is picked before the good model, because sometimes it may happen that the generated test data is exactly such that the good model will fail on it. We see that 93.4 % of the times, true risk is in the 95CI - it underestimates the uncertainty of the estimator.

If we would have larger training set, the true risk proxy and the variance of estimator would decrease. True risk proxy decreases due the fact we better understand DGP with more data. Both would increase when having smaller training set (having the same size of a test set).

In general, if we would increase the size of a test set, bias would increase, but in our case, holdout estimation is unbiased, so there would be no effect on bias (also when decreasing the size of a test set). But increasing the test data at the expense of training size, we decrease the 'capability' of a model to learn. The difference between the estimate and the true risk would decrease, as our estimate would be better.

Overestimation of the deployed model's risk

We rarely deploy the model trained only on the training data. Similarly, we never deploy any of the k models that are learned during k-fold cross-validation. Instead, we use the estimation as a means to select the best learner and then deploy the model trained on more data, typically all the available data. If a model is "smart", it means that with more data, it does not get worse in terms of performance.

We generated two data sets with 50 observations and trained first model h_1 on the first data set with 50 observations and model h_2 with combined data sets 1 and 2, so we end up model trained on 100 observations. We also calculated the true risk proxies for h_1 and h_2 and repeated this procedure 50 times. Here is the summary of the differences between the true risk proxies of h_1 and h_2:

```
## [1] "Summary of true risk h1 - true risk h2: "
```

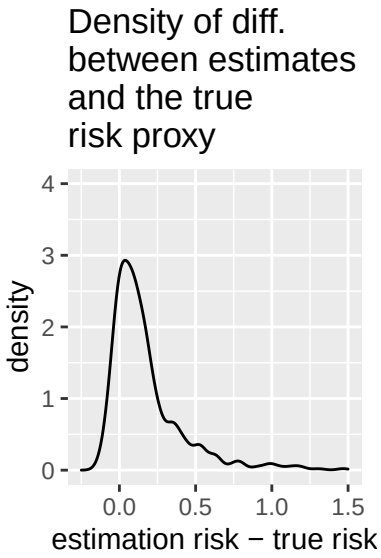
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.04257  0.06216  0.12419  0.65097  0.23298  8.60759
```

Because we trained model h_1 on a smaller set, true risk proxy for this model is higher than for h_2, which is trained using data set twice bigger. This is nicely seen from the summary above, where the mean average of differences is around 0.65. We also observe that the maximum difference is high (around 8.6) and minimum difference is almost zero. The estimation of the first model's risk has a lot of variance, compared to the variance of the second's model risk estimation. This nicely demonstrates that with more data, variance decreases and performance is better. Because the minimum is negative, we can conclude that it could happen that we get better estimation of model's risk with less data.

When choosing between two train data sets with different sizes, we should always take the larger one, as our variance will be lower. As we demonstrated on the case above, having the same train set size, picking larger train set makes sense and should be done always (at least when we have a "smart" model). If we would increase the difference between sizes of those sets, differences between performances would be even more visible, as the model trained on a bigger data set would outperform the one, trained on a smaller data set. It would be interesting to see the performance differences, when having two very small data set (for example, one with 10 data points and other with 15 data points). We think that for both estimations, the variance would be too big to see any differences between those two.

Loss estimator variability due to split variability

In practical application of train-test splitting, we would choose a train-test split proportion, train on the training data, and test on the test data. This results we then use on the test data as an estimate of the true risk of the model trained on all data. The estimates will be biased, because the tested model is trained on less data than the model we are interested in. Also, it would have variance due to which observations ended up in the training set and which in the test set. We can add another source of the variability - the variability of the losses across observations.




```
## [1] "True risk proxy: 0.5255"
```

```
## [1] "Mean difference: 0.2377"
```

```
## [1] "Median standard error: 0.1245"
```

```
## [1] "Percentage of 95CI that contain the true risk proxy: 86.2"
```

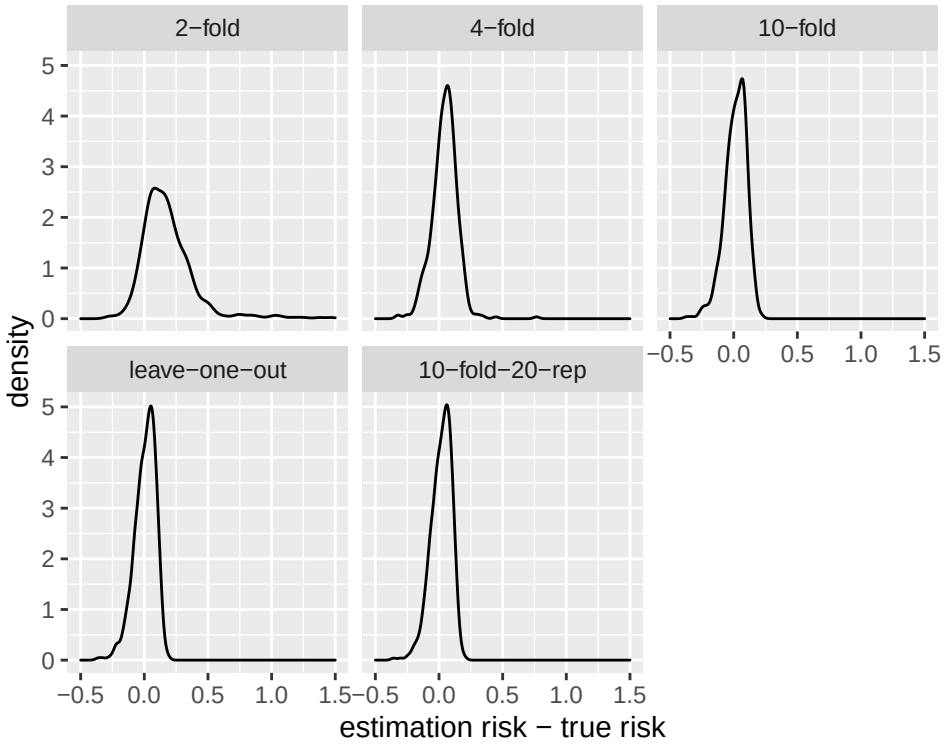
First thing we notice is that the distribution of differences is very skewed - very long tail on the positive and the mode is slightly on the positive side, which means that our model underestimates the performance, or overestimates the error. This could be also seen from the 95CI, as it contains true risk proxy 86.2 %. In some cases, the estimation of an error is really high, which is seen from long tails. We have already discussed longer tails on the right side before, where we said that this could mean that the best models could perform bad on some test cases. Also, the estimation of a risk is positively biased.

If we would have more data, the estimations would have lower bias and lower true risk proxy. Also, the variance of estimator would decrease. By increasing the proportion of train set, the bias would decrease and the variance would increase and vice versa. Increasing the proportion of a test set, the bias would increase, as we would not be able to fit the model properly. We should always consider taking such proportion that the size of train/test set would not be too low, as having small sets is not good practice - having small train set, the model can not learn properly and having too small test set, our estimations of a risk are very uncertain.

Cross-validation

The cross-validation estimates of true risk will be biased and will contain a lot of variability if the data set is relatively small. This variability will be both due to training set and due to test set variability of the inherent variability of the losses.

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

## [1] "Mean difference: 0.4826"

## [1] "Median standard error: 0.1034"

## [1] "Percentage of 95CI that contain the true risk proxy: 66.4"

## [1] "4-fold"

## [1] "Mean difference: 0.0447"

## [1] "Median standard error: 0.079"

## [1] "Percentage of 95CI that contain the true risk proxy: 86.6"

## [1] "10-fold"

## [1] "Mean difference: 0.0139"

## [1] "Median standard error: 0.073"

## [1] "Percentage of 95CI that contain the true risk proxy: 90.4"

## [1] "leave-one-out"

## [1] "Mean difference: 0.0054"

## [1] "Median standard error: 0.0749"

## [1] "Percentage of 95CI that contain the true risk proxy: 94.2"

## [1] "10-fold-20-rep"

## [1] "Mean difference: 0.0166"

## [1] "Median standard error: 0.0765"

## [1] "Percentage of 95CI that contain the true risk proxy: 94.8"

## Warning: Removed 38 rows containing non-finite values (stat_density).
```



The first thing we observe is that the estimation of a risk highly depends on number of folds. More folds we have, better the estimation of a risk - observing 2-fold cross-validation, we can see high average difference between estimates and the true risk proxy. Increasing number of folds, this difference is lowering and the get lowest difference with the leave-one-out cross-validation, which makes sense. We should always consider more splits, but we should be careful not to choose too many folds, as the computation time of a cross-validation increases a lot! The variance also decreases with the number of folds, but we can see that the variance of a 10-fold CV is lower than the variance of a leave-one-out. The most time consuming CV was the 10-fold-20-rep CV and this gives good results in terms of a percentage of 95CI that contains the true risk proxy the most times.

We can also observe that mean differences for all CV are positive - we are overestimating the error, or underestimating the model's performance. Also, it is interesting that the difference of the estimated risk and true risk for 2-fold CV is high. We think that this is due to problems of fitting the model, as we got several warnings for the learning algorithm not converging.

Comparing distributions of differences we observe that the 2-fold CV has heavier tails and the standard deviation is higher, compared to all others. For other folds, the distributions of differences are similar.

A different scenario

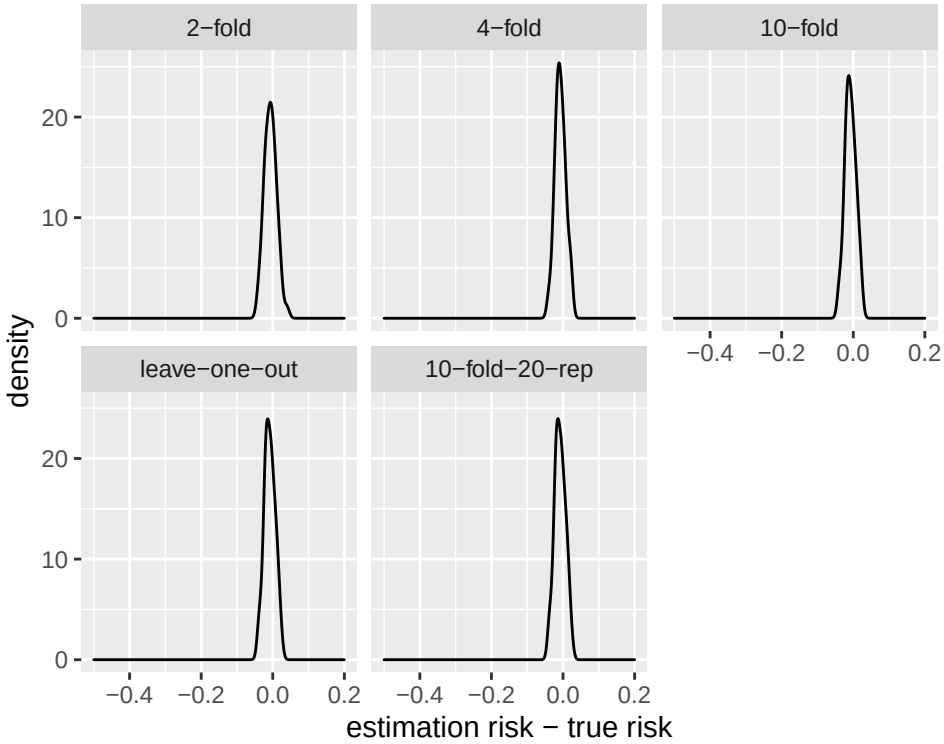
We can choose another DGP/learners such that the results will disagree with our last example. We hope that in such a way, the variance would increase with the increase of folds or maybe we even get more extreme cases. We will be dealing with DGP which gives only one variable and we will be using Poisson regression.

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

```
## [1] "Mean difference: -0.0064"
```

```
## [1] "Median standard error: 0.0085"
```

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

## [1] "Mean difference: -0.0074"

## [1] "Median standard error: 0.0133"

## [1] "10-fold"

## [1] "Mean difference: -0.008"

## [1] "Median standard error: 0.0143"

## [1] "leave-one-out"

## [1] "Mean difference: -0.0084"

## [1] "Median standard error: 0.0142"

## [1] "10-fold-20-rep"

## [1] "Mean difference: -0.0083"

## [1] "Median standard error: 0.0142"
```



As we can see, the mean standard error increases with the increase of the folds, which is exactly what we wanted to show. Also, we get an overestimation of a performance of a model, which is unusual. Observing the density of the risk differences, we can observe the mode for all folds is positive, but the average is negative, which tells us we are overestimating the model's performance.