Loss Estimation

Ursa Zrimsek

11/05/2021

Setup

Generating toy dataset First, we prepare the generator for our toy binary classification data, which has 8 independent variables, 3 of which are unrelated to the target variable. Because we generate the data, we know all the properties of the data generating process, which will allow us to study the quality of our loss estimation. We'll be using negative log-loss (smaller is better) throughout this notebook.

```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 We'll be using this huge dataset as a proxy for the DGP and determining the ground-truth true risk of our models. Of course, this is itself just an estimate, but the dataset is large enough so that a model's risk on this dataset differs from its true risk at most on the 3rd decimal digit. The reason for this is that we have 100000 observations (log losses, calculated on the dataset), and we are calculating their mean. By CLT we know that its probability distribution will approximate normal distribution, with mean equal to the ground-truth and standard deviation equal to standard deviation of observations divided by sqrt(100000). Our standard error is equal to sd(losses)/sqrt(100000) = 0.003 * sd(losses). That means that if we check that the standard deviation of observations on our big dataset is smaller than one, the error will be small enough. That's why we will check standard error in observations every time we are calculating true risk. If they are smaller than 1.5, our estimation is OK (smaller error than 0.005), otherwise we will print a warning and also know how much our estimation of the true risk can differ from the true value.

```r
df_dgp <- toy_data(100000, 0)
options(digits=4)
```

Holdout estimation

Holdout estimation or, as it is commonly referred to, train-test splitting, is the most common approach to estimating a model's risk. The first and most important thing to understand is that the model's risk on the test data is just an estimate of the model's true risk. As such, it should always contain some sort of quantification of uncertainty, such standard errors or 95% confidence intervals. We've learned a couple of techniques for doing this, but in this homework we'll use the most simple one – standard errors and 95% CI based on the asymptotic argument that the mean loss will be normally distributed with variance n times lower than the variance of the loss. When interpreting the results, we will also inspect the coverage of these confidence intervals to see how this simple approach does in practice. In the remainder of this section we will investigate some of 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.

Model loss estimator variability due to test data variability We will generate a toy dataset with 50 observations and train a Bernoulli logit GLM on it. True risk proxy will be computed on the huge dataset with 100000 observations. Then we will generate new toy datasets with 50 observations 1000 times and analyze how the test data risk varies.

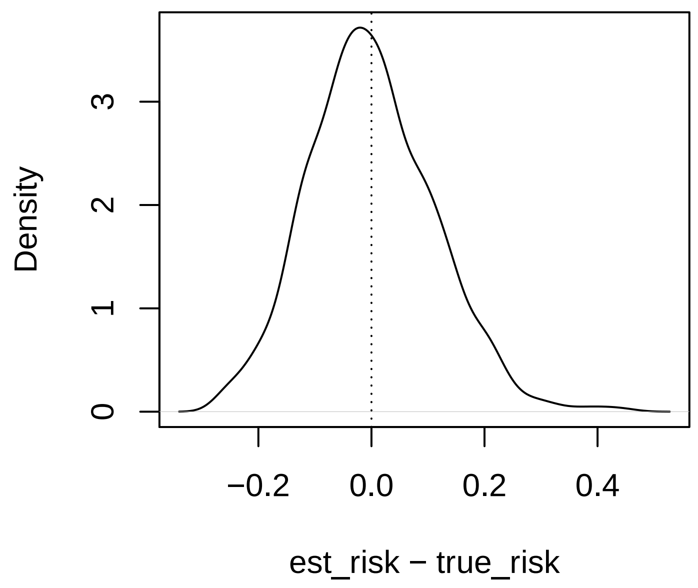
```
## True risk proxy: 0.5755
```

```
## Percentage of CI containing true risk proxy: 93.4
```

```
## Mean difference: 0.0001925
```

```
## 0.5-0.5 baseline true risk: 0.6931
```

```
## Median standard error: 0.1094
```



```
## integer(0)
```

Above we can see plotted density of the difference between estimated and true risk of the model. If we look at the plot, we can see that it has longer tail on the right hand side, which means that our mistakes can be bigger when we underestimate our model's performance. That happens because there is a lower bound to how good our model can do, so there can't be such extreme cases where our estimated risk is lower than true risk than when it is higher (our train dataset can be chosen very poorly). The reason for this could also be the nature of the log loss function, as it punishes mistakes more than it rewards correct answers. We can also see that the mode of the difference is below 0, so most of the time we will overestimate the model's performance - but when we underestimate it, we underestimate it more (can see this from mean). The mean difference is 0 on the third decimal point, so this confirms that our estimate is unbiased (as we are choosing independent test sets with different seeds in every step). This happens because our test set is chosen independently from training set.

We can see that if we take 2 * median standard error around the true risk (or we could say our estimated risk, since it is in average the same), the risk of baseline model is included in this interval (we simulate the 95CI with median SE), which means that with this CI we are not exactly giving a very exact intervals. Another thing we notice when we look at the confidence intervals is, that we construct a 95CI, but our true value is contained in it in only 0.934 of repetitions. This means that we underestimate the uncertainty of our estimate if we construct the CI like this.

If our training set was bigger, the standard error would be smaller, because there would be less variability in the models. If it was smaller, the error would increase. Also, if we used bigger training set, the true risk (and then also its estimate) would be smaller, because the model would be better fitted (if we are using good learner) - oppositely to smaller training set. With bigger test set, we would decrease the difference between estimate and true risk, and decrease the standard error (CLT), oppositely to smaller test set. Mean difference would still be 0, since the holdout estimation is unbiased.

Overestimation of the deployed model's risk In practice 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. More data typically means that the learner will produce a model with lower true risk.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -0.015   0.046   0.085   0.676   0.223  10.642
```

If we translate the above into practical terms, we can see the summary of differences of risks of the model that we trained on training data (half), and the one that we trained on all the available data. As we said above, the model trained on more data, should typically have lower risk, but we can see that this is not a rule, as the minimum of the differences is negative, which means that in that case, we overestimated the model's performance. But in general, we can see that our estimate of the risk is overestimated, and the model that we deploy, is actually better than we estimate by training it on some smaller set. We can see that from all the other values in the summary. We can also see that the mean of the differences is now not equal to zero, so our estimate is biased (positively).

This difference become smaller if we take larger proportion of all available data for the training set, and even larger if we take less – our estimate will be even more biased. If our whole dataset is bigger, the difference will be smaller, as both risks will be smaller, and bigger with smaller dataset (worse models). Also variance would fall with bigger dataset and rise with smaller.

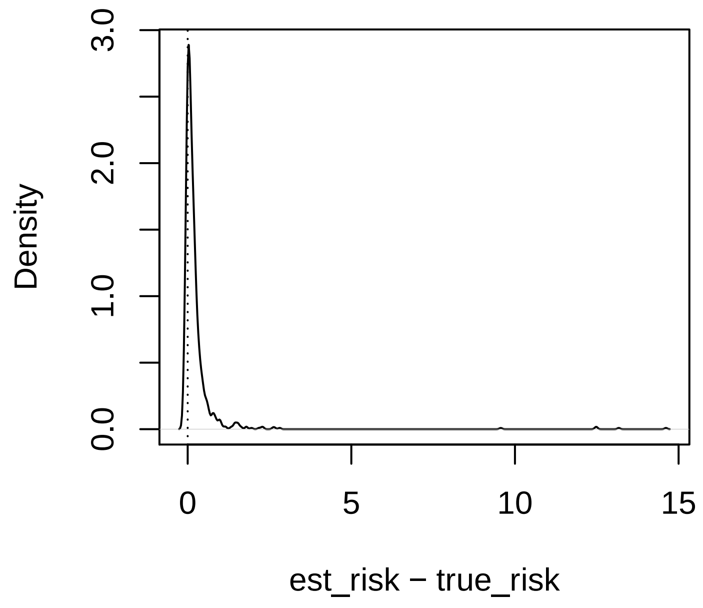
Loss estimator variability due to split variability In a practical application of train-test splitting, we would choose a train-test split proportion, train on the training data, and test on the test data. We would then use this result on the test data as an estimate of the true risk of the model trained on all data. From the experiments so far, we can gather that this estimate will be biased, because the tested model is trained on less data than the model we are interested in. It will also have variance due to which observations ended up in the training set and which in the test set. To that we can add the most basic source of variability – the variability of the losses across observations.

```
## True risk proxy of h0: 0.5255
```

```
## Percentage of CI containing true risk proxy: 0.869
```

```
## Mean difference: 0.2515
```

```
## Median standard error: 0.1223
```



```
## integer(0)
```

The first thing that we can confirm with above results is the hypothesis already written above – the true risk proxy of h0, that was trained on 100 observations is lower than the true risk of the model h trained on 50 observations.

From mean difference (and the plot) we can confirm that our risk estimator is positively biased. On the plot we see that we have some extreme values that greatly overestimate the risk, which can lead to wrong selection of the model we think is best, only because of unfortunate train-test split. This shows us that it's necessary to include more different splits into model selection.

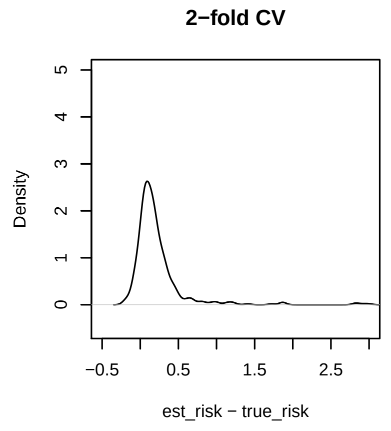
The percentage of 95CI that contain the true risk is much smaller that should be, which means we are again underestimating the uncertainty of our estimation.

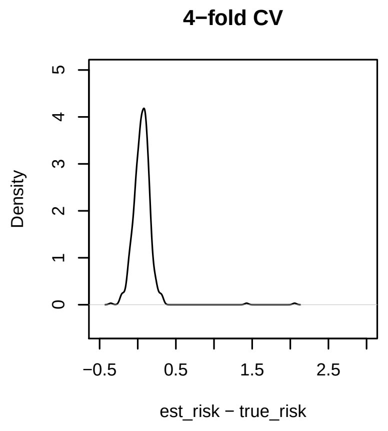
If the dataset was larger, our estimates would be better. First we would have lower true risk, because of better trained model. The estimate would have lower bias, as it would be trained on more data, and also lower error of estimation, because it would be tested on more data. If our dataset was smaller, the opposite would happen. If the proportion of training data would be bigger, we would again have lower bias because of more similar size of the training set to the whole set, but the standard error would be bigger, because the test size would be smaller. If we lower the proportion of training data, the bias would be bigger, because of more difference in the training set size. But we're not sure if the variance would drop, because even though we would have more test data, there would be more variability in how the models are trained.

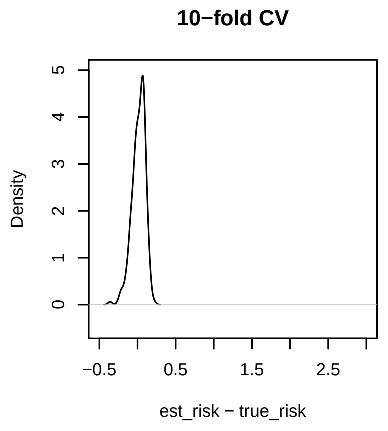
Cross-validation

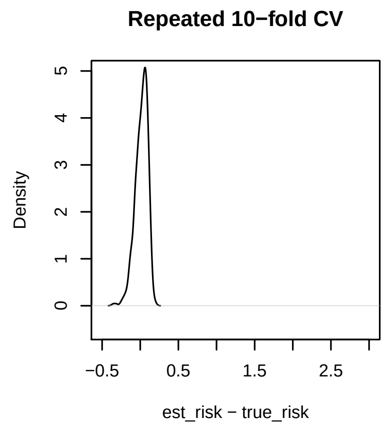
If we extrapolate the results so far to cross-validation, we can conclude that cross-validation estimates of true risk will also be biased and will contain a lot of variability if the dataset is relatively small. This variability will be both due to training set and due to test set variability on top of the inherent variability of the losses.

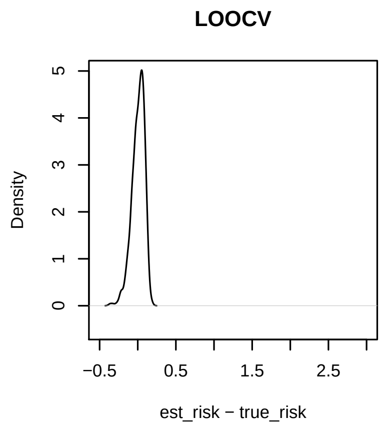
Plots from cross validation We are compared k-fold cross validation with different number of folds and with repetition. Below you can see the plots of differences between our estimation and true risk of the model and other statistics of different cross validations.











```
## [1] "2-fold CV:"
## Percentage of CI containing true risk proxy: 0.696
## Mean difference: 0.365
## Median standard error: 0.109
## [1] "4-fold CV:"
## Percentage of CI containing true risk proxy: 0.914
## Mean difference: 0.05856
## Median standard error: 0.0844
## [1] "10-fold CV:"
## Percentage of CI containing true risk proxy: 0.932
## Mean difference: 0.01529
## Median standard error: 0.07712
## [1] "Repeated 10-fold CV:"
## Percentage of CI containing true risk proxy: 0.95
## Mean difference: 0.01626
## Median standard error: 0.07646

## [1] "LOOCV:"

## Percentage of CI containing true risk proxy: 0.942

## Mean difference: 0.005411

## Median standard error: 0.07487
```

We can see, that by bigger number of folds, we are lowering bias and standard error of our estimate. Bias can be explained the same way as before, it is lower, when we have more examples in our train dataset (with more folds). But here also standard error lowers, when we have more folds, and the reason for that is that we always have the same number of observations that we calculate the risk on (the whole dataset), the only difference is in the model that we use to predict each point. And with bigger k, those models are more similar and better trained, so the standard error is lower.

From that we could conclude that LOOCV is the best, but here we don't take speed into account. This approach is much slower than others, because it needs to train as many models as we have points in our dataset.

From CI we can see that still in all CV without repetitions, we are underestimating the error of our estimate, but this becomes better with bigger number of folds. We reach the best estimation of the error of our estimate with repeated cross validation (it looks like it is unbiased), which could be explained by us averaging the loss on each observation and thereby getting results less dependent on the fold partitions, so the error is better estimated.

If we look at the tails on the plots (this can be seen even better if we don't limit the x-axis, but we did, for easier comparison of shape), we can also easily see that we are more likely to select a worst model with lower number of folds, as we can greatly overestimate the risk of our best model. With bigger folds the tails are smaller, which means that our model selection should be better, as we couldn't underestimate model's performance so much. Also with more folds we have sharper peaks, which means that the estimates are concentrated closer to the true value.

A different scenario If our learner wouldn't be smart, or our DGP would generate dataset that wouldn't make it possible for learner to train a good model, or our original dataset would be too small for a good fit, the above results probably wouldn't hold.

We tested this hypothesis for different DGP. Our new DGP is similar to the previous one, but here the target variable is dependant only on one of the input variables. This brings us to negative bias, which means that CV on such data underestimates the risk, and also our previous observations for CI and SE don't hold anymore. From this we can conclude, that the knowledge about model evaluation that we gathered holds for learners, DGPs and datasets that have nice properties. But as soon as one part of our model generation has some strange properties, that we don't detect, it can hurt our model evaluation and selection. So the important part is to analyze our datasets and know our learner's properties, so that we know when something goes wrong.

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

## Percentage of CI containing true risk proxy: 0.2

## Mean difference: 0.04204

## Median standard error: 0.06956

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

## Percentage of CI containing true risk proxy: 1

## Mean difference: -0.03272

## Median standard error: 0.05448
## [1] "10-fold CV:"

## Percentage of CI containing true risk proxy: 0.6

## Mean difference: -0.05337

## Median standard error: 0.04562

## [1] "Repeated 10-fold CV:"

## Percentage of CI containing true risk proxy: 0.6

## Mean difference: -0.05891

## Median standard error: 0.04239

## [1] "LOOCV:"

## Percentage of CI containing true risk proxy: 0.6

## Mean difference: -0.05461

## Median standard error: 0.04739
```