Convergence of test and training error, bootstrapping

Machine Learning II 2019

1.3 Optimise the data generation process* (1p)

The data generation procedure is slow as it does not use `numpy` matrix operations. Fix this issue and measure the speedup.

Solution

An example piece of code

```
def data_sampler_numpy(n:int, k:int, weights:Series) -> DataFrame:
    columns = ['x_{}'.format(num) for num in range(1, k + 1)]

    dat = DataFrame(np.random.rand(n, k) >= 0.5, columns = columns)
    dat['y'] = np.random.rand(n) <= sigmoid(np.dot(dat.iloc[:,:len(weights)] - 0.5,
        weights))

    return(dat)
```

9.1 Comparison of bootstrap methods (2p)

Implement basic bootstrapping algorithm that draws n samples randomly with replacement from n-element data set. You can use `DataFrame.sample(n, replace=True)` for that. On top of that implement all bootstrap estimates and compare their behaviour on four example cases:

- For each data source and algorithm pair draw around 1000 datasets of size 100.
- For each of these datasets compute compute $E_b, E_b^*, E_t, E_{.632}, E_{.632+}$.
- For each of these datasets also sample n-element independent testset and compute holdout error E_h.
- Visualise results by drawing violin and boxplots.
- Interpret results. Which od those estimates is closest to E_h? Why some estimates are biased?

Solution

The results are in **Figure 1**. Notice how LOO can overestimate the error (apparent in the subfigure with normal data and logistic regression) and how the .632+-method rescues the ordinary .632-method in case we overfit.

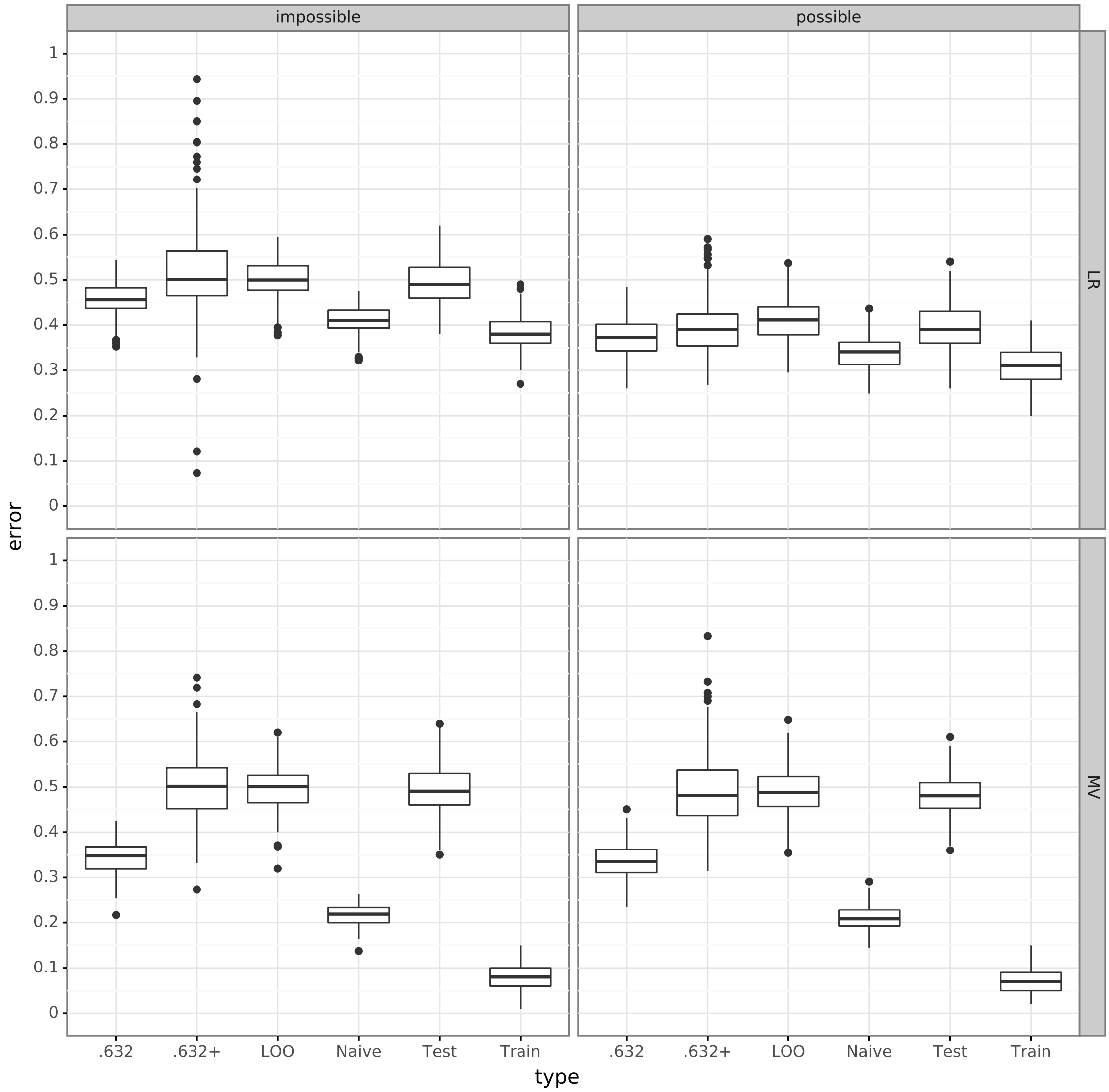

Figure 1: Errors with different bootstrap sampling methods for each data source (impossible – no signal in the data; possible – some signal in the data) and classification method (LR – logistic regression; MV – majority voting).