Supervised machine learning I

Computational Biomedicine, WS 2023, Prof. Dr. Michael Altenbuchinger

December 20, 2023

Learning objectives: become familiar with supervised machine learning techniques and learn how these methods circumvent overfitting.

(1) Linear lasso regression

About 40% of patients receiving allogeneic stem cell transplants (ASCT) develop a systemic acute graft versus host disease. About 54% of these diseases affect the gastrointestinal tract. A candidate for a protective substance in the gut is the tryptophan microbial fermentation product indole (3-indoxy sulfate, 3-IS). The question is, can we predict 3-IS levels from patients intestinal microbiomes? The microbiomes were measured by sequencing the hypervariable V3 region of the 16S ribosomal RNA gene in patient stool and mapping the sequences to operational taxonomic units (OTUs).

- Install and load the package "glmnet", which allows us to learn penalized regression models.
- Load the microbiome data "OTU_data_spike.rds" and the respective 3-IS levels "indoxylSulfate.csv" (`read.csv()`, `readRDS()`) and save them as object `otus` and `indoxyl`, respectively. Check the data. Verify that patients IDs are in accordance between both datasets.
- Let's first do a naive analysis! Learn a linear model which predicts 3-IS levels (`lm()`, ignore the warnings for a moment):

  ```
  lm(formula = indoxyl ~ ., data = otus)
  ```

  Apply this linear model to your training data and convince yourself that you perfectly predict 3-IS levels (`predict()`, `cor()`). Note, the great performance is meaningless since we are interested in a low generalization error!!!
- Write a leave-one-out cross validation. In each cross-validation step you can now learn a linear model and record the prediction for your left-out-sample. You need to write a for-loop for this. Check out how these predictions agree with the ground truth. Note, this is now an estimate of your generalization error!
- Now let's try out a penalized regression method using "glmnet". Learn a lasso regression model using `cv.glmnet()` on the full dataset, save the fit and apply it to your training data using `predict()`.
- Apply `plot()` to your `cv.glmnet`-fit object. What do you see?
- Add the lasso regression to your cross validation and check out its performance.

Take home message: a low training error can be meaningless. We need to estimate the generalization error. Methods like the lasso do an internal cross validation to calibrate the regularization parameter λ. In such a way we end up with a model which generalizes much better. However, also here, our prediction error might be over-optimistic as we saw when we included the lasso regression in our own cross validation (we did a nested cross validation). There, the accordance is much lower than previously for the model calibrated on all samples.

(2) Logistic lasso and ridge regression

We have seen in the tutorial about unsupervised machine learning methods that we can nicely identify molecular Burkitt lymphomas from molecular data. Here, we are interested in developing a diagnostic assay to identify the ABC/GCB subtype in diffuse large B-cell lymphomas, which turned out to be more difficult.

- Load the data "mmml_vsn.rda" and extract your data matrix and your phenotypic data.
- Check your data as usual.
- Split your data into a training cohort (about 2/3 of the samples) and a test cohort (the rest) and define them as `x.training` and `x.test`, respectively.
- Generate an object `y.training` and `y.test` which encodes the diagnosis as integers (`0` for GCB, and `1` for ABC) for the training and validation data, respectively.
- Use `cv.glmnet` to learn a classifier on your training data. Here, you need to specify the option `family="binomial"`.
- Use `predict()` to get prediction for the test data. You can also predict probabilities. For this you need to specify `type="response"` in `predict()`.
- To assess the performance install and load the package "ROCR".
- Use the command `prediction()` and `performance()` to verify the performance. Here, you need "fpr" (false-positive rate) and "tpr" (true-positive rate).
- Plot the ROC curve using `plot()`. What is the area under the curve ("auc")?
- Extract the most important molecular features of your `cv.glmnet()` fit using `coef()`.
- You can also use a so-called ridge regression for this analysis. Repeat the `cv.glmnet` fit for `alpha=0`, which corresponds to a ridge regression and compare the performance to the lasso. Assess the cross-validation plots for both cases using `plot()` on the fit objects.
- `cv.glmnet()` uses an internal cross validation. It is also interesting to analyse the ridge and lasso regression in more detail by visualizing their regularization paths. Use `glmnet()` instead of `cv.glmnet()`, save the fits and apply the `plot()` command, which returns the regularization path. What does it show you?