Machine Learning 2018-2019

Home Assignment 5

Yevgeny Seldin **Christian Igel**
Department of Computer Science
University of Copenhagen

The deadline for this assignment is **4 January 2019**. You must submit your *individual* solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your source code in the PDF file.
- A .zip file with all your solution source code (Matlab / R / Python scripts / C / C++ / Java / etc.) with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF.
- IMPORTANT: Do NOT zip the PDF file, since zipped files cannot be opened in the speed grader. Zipped pdf submissions will not be graded.
- Your PDF report should be self-sufficient. I.e., it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Your code should include a README text file describing how to compile and run your program, as well as a list of all relevant libraries needed for compiling or using your code.

1 Train-Validation Split Trade-off (40 points)

We address the question of how to split a dataset S into training $S^{\texttt{train}}$ and validation $S^{\texttt{val}}$. As we have already seen, leaving too little data for validation leads to unreliable loss estimates, whereas leaving too little data for training leads to poor prediction models. So how should we actually split the data into $S^{\texttt{train}}$ and $S^{\texttt{val}}$? In this question you will develop one possible approach to this question.

1. To warm up: assume that you have a fixed split of S into $S^{\texttt{train}}$ and $S^{\texttt{val}}$, where the size of $S^{\texttt{val}}$ is $n^{\texttt{val}}$. You train a model $\hat{h}^*_{S^{\texttt{train}}}$ on $S^{\texttt{train}}$ using cross-validation. Then you compute the validation loss $\hat{L}(\hat{h}^*_{S^{\texttt{train}}}, S^{\texttt{val}})$. (Do not get confused: the cross-validation losses are computed on $S^{\texttt{train}}$ as in the standard cross-validation procedure restricted to $S^{\texttt{train}}$. Once we have found the best parameter and trained $\hat{h}^*_{S^{\texttt{train}}}$ with that parameter setting on all of $S^{\texttt{train}}$ we validate $\hat{h}^*_{S^{\texttt{train}}}$ on $S^{\texttt{val}}$. So there are multiple internal cross-validations of intermediate models on parts of $S^{\texttt{train}}$ and one final validation of the final model $\hat{h}^*_{S^{\texttt{train}}}$ on $S^{\texttt{val}}$.) Derive a bound on $L(\hat{h}^*_{S^{\texttt{train}}})$ in terms of $\hat{L}(\hat{h}^*_{S^{\texttt{train}}}, S^{\texttt{val}})$ and $n^{\texttt{val}}$ that holds with probability at least $1-\delta$.

2. Now we want to find a good balance between the sizes of $S^{\texttt{train}}$ and $S^{\texttt{val}}$. We consider m possible splits $\{(S^{\texttt{train}}_1, S^{\texttt{val}}_1), \ldots, (S^{\texttt{train}}_m, S^{\texttt{val}}_m)\}$, where the sizes of the validation sets are $n_1, \ldots, n_m$, correspondingly. We train m prediction models $\hat{h}^*_1, \ldots, \hat{h}^*_m$, where $\hat{h}^*_i$ is trained on $S^{\texttt{train}}_i$ using cross-validation. We calculate the validation losses of the i-th model on the i-th validation set $\hat{L}(\hat{h}^*_i, S^{\texttt{val}}_i)$. (Again, do not get confused. There are multiple internal cross-validations of intermediate models on parts of $S^{\texttt{train}}_i$ and one final validation of the final model $\hat{h}^*_i$ that was trained on all of $S^{\texttt{train}}_i$ with the optimal parameter setting found in the i-th cross-validation procedure.) Derive a bound on $L(\hat{h}^*_i)$ in terms of $\hat{L}(\hat{h}^*_i, S^{\texttt{val}}_i)$ and n_i that holds for all $\hat{h}^*_i$ simultaneously with probability at least $1-\delta$.

3. Now we evaluate this approach in practice. Go back to Question 4 in Home Assignment 4 (the SVM question). Consider 9 splits, where $n_i \in \{9, 19, \ldots, 89\}$. Take the first $n - n_i$ points for training SVMs using cross-validation, as described in the previous assignment, and the last n_i points for the final validation, as described above. (The choice of n_i leaves roughly $10\%, 20\%, \ldots, 90\%$ of the data for validation.) Plot the validation losses $\hat{L}(\hat{h}^*_1, S^{\texttt{val}}_1), \ldots, \hat{L}(\hat{h}^*_9, S^{\texttt{val}}_9)$ and the generalization bound for $L(\hat{h}^*_1), \ldots, L(\hat{h}^*_9)$ you have derived in the previous point together with the test loss (the loss on the test set) of the models $\hat{h}^*_1, \ldots, \hat{h}^*_9$ you have obtained. Take $\delta = 0.05$. Discuss your results.

2 Contradiction between Upper and Lower Bound (30 points)

The Occam's razor bound provides a high-probability upper bound on the expected loss of all classifiers h within a countably infinite hypothesis class H (Theorem 3.3 in Yevgeny's lecture notes). The VC lower bound states that if the VC dimension of a hypothesis class H is infinite, then it is impossible to obtain a high-probability upper bound smaller than 0.125 that holds for all classifiers h (Corollary 3.19 of Theorem 3.18; if you cannot follow the formal argument in Theorem 3.19 it is sufficient to think about the intuitive explanation we gave in class). The two results may sound contradictory. For each of the following statements write whether the argument is correct or incorrect. Explain your answer.

1. The VC dimension is only defined for uncountably infinite hypothesis classes and, therefore, there is no contradiction between the two results.

2. The Occam's razor bound may be larger than 0.125 for some hypotheses in H and, therefore, there is no contradiction between the two results.

3. The Occam's razor bound holds for any data distribution $p(X, Y)$, whereas the data distribution in the construction of the lower bound was picked in a very specific way. Therefore, there is no contradiction between the two results.

4. It is possible to construct a data distribution $p(X, Y)$ and a hypothesis class H with infinite VC-dimension, such that for any sample S of more than 100 points with probability at least 0.95 we will have $L(h) \leq \hat{L}(h, S) + 0.01$ for all h in H.

3 Random Forests (30 points)

3.1 Normalization

As discussed, normalizing each component to zero mean and variance one (measured on the training set) is a common preprocessing step, which can remove undesired biases due to different scaling. Using this normalization affects different classification methods differently.

- Is nearest neighbor classification affected by this type of normalization? If your answer is yes, give an example. If your answer is no, provide convincing arguments why not.

- Is random forrest classification affected by this normalization? If your answer is yes, give an example. If your answer is no, provide convincing arguments why not.

3.2 Random forests in practice (not for submission)

Install – if necessary – and apply software implementing random forests. For example, both R and Python provide good random forest implementations. In Python you can use `RandomForestClassifier` and `RandomForestRegressor`. To our knowledge, the fastest random forest classifier is provided by the most recent version (i.e., you have to download from the repository) of the Shark machine learning library (Igel et al., 2008), which uses the engine from Woody (Gieseke and Igel, 2018). Still lacks a lot of features, but is fast as a shark.

Apply random forests to the example problems from the previous assignments. Play with the hyperparameters.

References

F. Gieseke and C. Igel. Training big random forests with little resources. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1445–1454. ACM Press, 2018.

C. Igel, T. Glasmachers, and V. Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.