Machine Learning 2019-2020

Home Assignment 5

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

The deadline for this assignment is **7 January 2020 22:00**. 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 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.
- Handwritten solutions will not be accepted, please use the provided latex template to write your report.

1 How Variance Influences Concentration (14 points)

1. Make 1,000,000 repetitions of the experiment of drawing 20 i.i.d. Bernoulli random variables $X_1, \ldots, X_{20}$ (20 coins) with bias 0.05. ***Put attention, the bias is not 0.5, but 0.05, the coins are heavily biased!***
2. Plot the empirical frequency of observing $\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha$ for $\alpha \in (0.05, 0.1, 0.15, \ldots, 0.95, 1)$.
3. In the same figure plot Markov's, Chebyshev's, and Hoeffding's bounds on $\mathbb{P}\big(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha\big)$.
4. Repeat the experiment with unbiased coins and $\alpha \in (0.5, 0.55, 0.6, \ldots, 0.95, 1)$. (You can cut-and-paste your plot from Home Assignment 2. No need to explain the details of how you obtained this plot, but a copy of the plot should be in the report to make it self-sufficient.)
5. Compare the two plots and report your observations.

Comment: Hoeffding's inequality does not take variance into account, so in the case of sums of Bernoulli random variables it works well when the bias is close to 0.5 [which maximizes the variance], but it is not as tight when the bias is close to 0 or 1 [when the variance is significantly smaller than the worst-case]. There are other concentration of measure inequalities, for example the kl-inequality [which we study in Advanced Topics in Machine Learning] and Bernstein's inequality, which take variance into account and provide much tighter bounds when the variance is significantly smaller than the worst-case.

2 A bit more on the VC-dimension (14 points)

1. What is the VC-dimension of the hypothesis space $\mathcal{H}_d$ of binary decision trees of depth d?
2. What is the VC-dimension of the hypothesis space $\mathcal{H}$ of binary decision trees of unlimited depth?

3 Separating Hyperplanes (14 points)

1. You have a sample of 100,000 points and you have managed to find a linear separator that achieves $\hat{L}_{\texttt{FAT}}(h, S) = 0.01$ with a margin of 0.1. Provide a

bound on its expected loss that holds with probability of 99%. The input space is assumed to be within the unit ball and the hypothesis space is the space of linear separators.

4 The fine details of the lower bound (14 points)

We have shown that if a hypothesis space $\mathcal{H}$ has an infinite VC-dimension, it is possible to construct a worst-case data distribution which will lead to overfitting, i.e., with probability at least $\frac{1}{8}$ it will be possible to find a hypothesis for which $L(h) \geq \hat{L}(h, S) + \frac{1}{8}$. But does it mean that hypothesis spaces with infinite VC-dimension are always deemed to overfit? Well, the answer is that it depends on the data distribution. If the data distribution is not the worst-case for $\mathcal{H}$, there may still be hope.

Construct a data distribution $p(X, Y)$ and a hypothesis space $\mathcal{H}$ 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 $\mathcal{H}$.

5 Random Forests (44 points)

5.1 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.

If a transformation of the input (e.g., component-wise normalization or flipping and rotation of input images) does not change the behaviour of a classifier, then we say that the classifier is invariant under this transformation. When devising a machine learning algorithms for a given task, invariance properties can be an important design/selection criterion. If we know that the prediction of an input should not change if we a apply a certain transformation to it, then it is a plus if an algorithm is invariant under this transformation – the generalization from an input to its transformed version(s) is directly given and need not be learnt.

5.2 Random forests in practice

Apply a random forest classifier to the SVM example problem from the previous assignment. Use an ensemble of 50 trees. Report the perfromance on the test data. Show the line in which you define the classifier in the report.

Both R and Python provide good random forest implementations. In Python you can use `RandomForestClassifier` and `RandomForestRegressor`.[1]

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.

S. S. Lorenzen, C. Igel, and Y. Seldin. On PAC-Bayesian bounds for random forests. *Machine Learning*, 108(8-9):1503–1522, 2019.

[1] 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. If you are interested in our take on how to apply generalization bounds to random forests, please have a look at the paper by Lorenzen et al. (2019).