# *Machine Learning A*

*2022-2023*

## Home Assignment 4

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

The deadline for this assignment is **4 October 2022, 18:00**. 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 full source code in the PDF file, only selected lines if you are asked to do so.
- 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. The programming language of the course is Python.
- IMPORTANT: Do NOT zip the PDF file, since zipped files cannot be opened in 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.
- Handwritten solutions will not be accepted, please use the provided latex template to write your report.

# 1 How to Split a Sample into Training and Test Sets (20 points)

In this question you will analyze one possible approach to the question of how to split a dataset $S$ into training and test sets, $S^{\texttt{train}}$ and $S^{\texttt{test}}$. As we have already discussed, overly small test sets lead to unreliable loss estimates, whereas overly large test sets leave too little data for training, thus producing poor prediction models. The optimal trade-off depends on the data and the prediction model. So can we let the data speak for itself? We will give it a try.

So, we want to find a good balance between the sizes of $S^{\texttt{train}}$ and $S^{\texttt{test}}$. We consider $m$ possible splits $\{(S_1^{\texttt{train}}, S_1^{\texttt{test}}), \ldots, (S_m^{\texttt{train}}, S_m^{\texttt{test}})\}$, where the sizes of the test sets are $n_1, \ldots, n_m$, correspondingly. For example, it could be $(10\%, 90\%), (20\%, 80\%), \ldots, (90\%, 10\%)$ splits or anything else with a reasonable coverage of the possible options. We train $m$ prediction models $\hat{h}_1^*, \ldots, \hat{h}_m^*$, where $\hat{h}_i^*$ is trained on $S_i^{\texttt{train}}$. We calculate the test loss of the $i$-th model on the $i$-th test set $\hat{L}(\hat{h}_i^*, S_i^{\texttt{test}})$. Derive a bound on $L(\hat{h}_i^*)$ in terms of $\hat{L}(\hat{h}_i^*, S_i^{\texttt{test}})$ and $n_i$ that holds for all $\hat{h}_i^*$ simultaneously with probability at least $1 - \delta$.

*Comment: No theorem from the lecture notes applies directly to this setting, because they all have a fixed sample size $n$, whereas here the sample sizes vary, $n_1, \ldots, n_m$. You have to provide a complete derivation.*

# 2 From a lower bound on the expectation to a lower bound on the probability [Pedagogical question, but not for submission]

The purpose of this question is to help you understand how a lower bound on the expectation of a random variable $X$ can be used to derive a lower bound on the probability that $X$ is not too small.

Let $X$ be a random variable that is always upper bounded by $b$. Let $c < a < b$. Prove that if $\mathbb{E}[X] = a$, then $\mathbb{P}(X \geq c) \geq \frac{a-c}{b}$.

Hint: see the illustration in Figure 1 and check the proof of the lower bound on the probability in Yevgeny's slides.

Informal conclusion: If we show that the expectation of a *bounded* random variable $X$ is large, then the probability that $X$ is large cannot be too small. (Note that if $X$ is unbounded, the claim does not hold. You are very welcome to think why.)

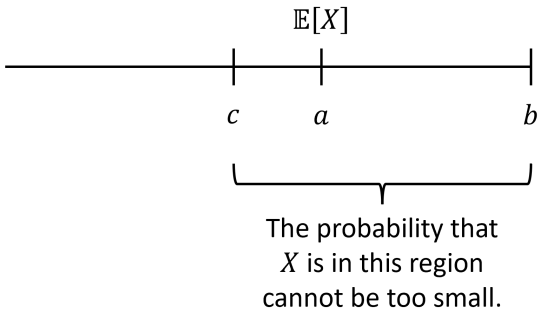


Figure 1: Illustration of how to obtain a lower bound on the probability from a lower bound on the expectation.

# 3 Early Stopping (30 points)

Early stopping is a widely used technique to avoid overfitting in models trained by iterative methods, such as gradient descent. In particular, it is used to avoid overfitting in training neural networks. In this question we analyze several ways of implementing early stopping. The technique sets aside a validation set $S_{\text{val}}$, which is used to monitor the improvement of the training process. Let $h_1, h_2, h_3, \ldots$ be a sequence of models obtained after $1, 2, 3, \ldots$ epochs of training a neural network or any other prediction model (you do not need to know any details about neural networks or their training procedure to answer the question). Let $\hat{L}(h_1, S_{\text{val}})$, $\hat{L}(h_2, S_{\text{val}})$, $\hat{L}(h_3, S_{\text{val}}), \ldots$ be the corresponding sequence of validation errors on the validation set $S_{\text{val}}$.

1. Let $h_{t^*}$ be the neural network returned after training with early stopping. In which of the following cases is $\hat{L}(h_{t^*}, S_{\text{val}})$ an unbiased estimate of $L(h_{t^*})$ and in which cases is it not. Please, explain your answer.

   (a) Predefined stopping: the training procedure always stops after 100 epochs and always returns the last model $h_{t^*} = h_{100}$.

   (b) Non-adaptive stopping: the training procedure is executed for a fixed number of epochs $T$, and returns the model $h_{t^*}$ with the lowest validation error observed during the training process, i.e., $t^* = \arg\min_{t \in \{1,\ldots,T\}} \hat{L}(h_t, S_{\text{val}})$.

   (c) Adaptive stopping: the training procedure stops when no improvement in $\hat{L}(h_t, S_{\text{val}})$ is observed for a significant number of epochs. It then returns the best model observed ever during training. (This procedure is proposed in Goodfellow et al. (2016, Algorithm 7.1) or `https://www.quora.`

com/How-does-one-employ-early-stopping-in-TensorFlow, but again, you do not need to know the details of the training procedure.)

2. Derive a high-probability bound (a bound that holds with probability at least $1-\delta$) on $L(h_{t^*})$ in terms of $\hat{L}(h_{t^*}, S_{\text{val}})$, $\delta$, and the size $n$ of the validation set $S_{\text{val}}$ for the three cases above. In the second case the bound may additionally depend on the total number of epochs $T$, while in the third case the bound may additionally depend on the index $t^*$ of the epoch providing the optimal model. Please, solve the last case using the series $\sum_{t=1}^{\infty} \frac{1}{i(i+1)} = 1$.[1]

3. The adaptive approach suggests stopping when "no improvement in $\hat{L}(h_t, S_{\text{val}})$ is observed for a significant number of epochs". A natural way of redefining the stopping criterion once we have the generalization bound is to stop when "no improvement in the generalization bound is observed for a significant number of epochs". The adaptive approach does not limit the number of epochs in advance, but what is the maximal number of epochs $T_{\max}$, after which it makes no sense to continue training according to the bound you derived in Point 2? Express $T_{\max}$ in terms of the number of validation samples $n$. It is sufficient to provide an order of magnitude of $T_{\max}$ in terms of $n$, you do not have to calculate the explicit constants.

4. How would your answer to the previous point change if you use the series $\sum_{i=1}^{\infty} \frac{1}{2^i} = 1$ for deriving the bound? (You should get that with these series you can run significantly less epochs in the adaptive approach compared to the series used in Point 2. Thus, unlike in the case of decision trees from the previous question, here the choice of the series has a significant impact.)

5. In this question we compare the adaptive procedure with non-adaptive. Assume that the two procedures use the same initialization, so that the corresponding models at epoch $t$ are identical, and assume that the adaptive procedure has reached $T_{\max}$ (but $t^*$ may be smaller than $T_{\max}$). Show that the generalization bound for adaptive stopping in Point 2 is never much worse than the generalization bound for non-adaptive stopping, but in some cases the adaptive bound can be significantly lower.

   Guidance: To simplify the analysis, throughout the question we assume that the confidence parameter $\delta \leq \frac{1}{2}$. For $T \geq 1$ it gives $\delta \leq \frac{1}{2} \leq \frac{T}{T+1}$.

   (a) First, assume that $T \leq T_{\max}$. Let $t^*$ be the index of the epoch selected by the adaptive procedure and $T^*$ be the index of the epoch selected by the non-adaptive procedure. Since the adaptive procedure has selected $t^*$ we know that the adaptive bound for epoch $t^*$ is lower than the

[1] We have $\sum_{i=1}^{\infty} \frac{1}{i(i+1)} = \sum_{i=1}^{\infty} \left(\frac{1}{i} - \frac{1}{i+1}\right) = 1$.

adaptive bound for epoch $T^*$. We also know that $T^* \leq T$, where $T$ is the number of epochs in the non-adaptive approach. Use this information and do some bounding to show that for any confidence parameter $\delta \leq \frac{1}{2}$, the adaptive bound can be at most a multiplicative factor of $\sqrt{2}$ larger than the non-adaptive bound.

(b) Now consider the case $T > T_{\max}$. Show that in this case the adaptive bound is trivially 1 and the non-adaptive bound is at least $\frac{1}{\sqrt{2}}$. So in this case the adaptive bound also cannot exceed the non-adaptive bound by more than a multiplicative factor of $\sqrt{2}$.

(c) You have shown that under the assumption that $\delta \leq \frac{1}{2}$ the adaptive bound never exceeds the non-adaptive bound by more than a multiplicative factor of $\sqrt{2}$. Now explain in which situations the adaptive bound can be significantly smaller than the non-adaptive bound. You should have two cases. In both cases you should have $T < T_{\max}$ and $\delta \leq \frac{1}{2}$.

Conclusion: depending on the data, the generalization bound for adaptive stopping can be significantly smaller than the generalization bound for non-adaptive stopping and at the same time it is guaranteed that it is never worse by more than a multiplicative factor of $\sqrt{2}$.

# 4 Random Forests (50 points)

## 4.1 Analyzing Satellite Data (35 points)

In the first part of this exercise, you will deal with data from the *Landsat 8* satellite. The satellite takes images via *multiple bands*, which yields so-called *multi-spectral images* with multiple values given for each pixel. Using such multi-spectral images, one can generate "normal" true color images, see Figure 2.[2] On Absalon, you can find three preprocessed Landsat 8 files: `landsat_train.csv`, `landsat_validation.csv`, and `landsat_area.csv`. The training file contains $n = 5,000,000$ lines. Each line contains a label (first value) and $d = 9$

Figure 2: Landsat 8: Example image (true color) that is derived from the multi-spectral bands.

[2]If you are interested, you can check out the details related to this type of data, but it's not needed for doing the assignment: `https://landsat.gsfc.nasa.gov/landsat-data-continuity-mission/`

features separated via commas (each row
corresponds to a pixel in an associated multi-spectral image). The validation file is of the same form and contains $1,335,558$ instances. The last file contains $9,000,000$ lines each 9 features (i.e., no label).

1. Train a random forest with 10 trees and bootstrap samples using the training data. At each node, test all the $d$ features and consider the Gini index as evaluation criterion. Build full trees, i.e., do not set any maximum depth for the trees (which is the default for random forests). Afterwards, compute the validation accuracy using the instances provided in `landsat_validation.csv`. What is the validation accuracy?

2. Next, apply the model to all instances given in `landsat_area.csv` and visualize the predictions (e.g., one color per class). Here, each line of `landsat_area.csv` corresponds to a pixel in a $3000 \times 3000$ image, where the first 3000 lines correspond to the first row, the following 3000 ones to the second row, and so on. Do you recognize the area?

3. Assume you are given $n$ training points and that you have built a binary decision tree based on these points (full tree, i.e., you only stop once each leaf is pure). What is the maximum depth/height of such a tree, i.e., how many nodes might be, in the worst case, on the path from the root to a leaf?

**Deliverables:** Provide all your source code and add answers to the questions raised above. Also include the visualization/plot to your write-up.

*Hints: Make use of Python and the Numpy package to load the data. To train the model, make use of the* ***`RandomForestClassifier`*** *class provided by the Scikit-Learn package. Note that loading the data and training/applying the model* ***might take some time*** *(maybe a few minutes).*

## 4.2 Normalization (15 points)

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.

Comment: 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 Occam's Razor [Optional question, not for submission]

We want to design an application for bilingual users. The application should detect the language in which the person is typing based on the first few letters typed. In other words, we want to design a classifier that takes a short string (that may be less than a full word) as input and predicts one of two languages, say, Danish or English. For simplicity we will assume that the alphabet is restricted to a set $\Sigma$ of 26 letters of the Latin alphabet plus the white space symbol (so in total $|\Sigma| = 27$). Let $\Sigma^d$ be the space of strings of length $d$. Let $\mathcal{H}_d$ be the space of functions from $\Sigma^d$ to $\{0, 1\}$, where $\Sigma^d$ is the space of input strings and $\{0, 1\}$ is the label space (e.g., Danish or English). Let $\mathcal{H} = \bigcup_{d=0}^{\infty} \mathcal{H}_d$ be the union of $\mathcal{H}_d$-s.

1. Derive a high-probability bound[3] for $L(h)$ that holds for all $h \in \mathcal{H}_d$.
2. Derive a high-probability bound for $L(h)$ that holds for all $h \in \mathcal{H}$.
3. Explain the trade-off between picking short strings (small $d$) and long strings (large $d$). Which terms in the bound favor small $d$ (i.e., they increase with $d$) and which terms in the bound favor large $d$ (i.e., they decrease with $d$)?
4. We have presented a lower bound, where we constructed an example of a problem with a large hypothesis class (of size $2^{2n}$), where the empirical loss of the empirically best hypothesis was always zero, but the expected loss of

[3] A bound that holds with probability at least $1 - \delta$.

the empirically best hypothesis was at least 1/8 with probability at least 1/8. The hypothesis class $\mathcal{H}$ in this question is obviously infinite. Explain why there is no contradiction between the bound in Point 2 and the lower bound.

***Optional, not for submission*** You are very welcome to experiment with the influence of the string length $d$ on the performance. You can find a lot of texts in different languages at `http://www.gutenberg.org/`. Do you observe the effect of initial improvement followed by overfitting as you increase $d$?

# 6 Learning by discretization [Optional didactic question for understanding the Occam's razor bound, not for submission]

We want to learn an arbitrary binary function on a unit square by discretizing the square into a uniform grid with $d^2$ cells. The hypothesis space is the space of all possible uniform grids with $d^2$ cells for $d \in \{1, 2, 3, \dots\}$, where each cell gets a binary label.

We have a sample $S$ of size $n$ to learn the function. Let $\mathcal{H}$ be the hypothesis set of uniform grids and let $f(h)$ denote the number of cells in the hypothesis $h$. (If we take $d(h) = \sqrt{f(h)}$, then $d(h) \in \{1, 2, 3, \dots\}$.)

1. Derive a generalization bound for learning with $\mathcal{H}$.
2. Explain how to use the bound to select a prediction rule $h \in \mathcal{H}$.
3. What is the maximal number of cells as a function of $n$, for which your bound is non-vacuous? (It is sufficient to give an order of magnitude, you do not need to make the precise calculation.)
4. Explain how the density of the grid affects the bound. Which terms in the bound increase as the density of the grid increases and which terms in the bound decrease as the density of the grid increases?

# References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.