Machine Learning B

2021-2022

Home Assignment 2

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

The deadline for this assignment is **7 December 2021, 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 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.
- 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. 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 The airline question (40 points)

1. An airline knows that any person making a reservation on a flight will not show up with probability of 0.05 (5 percent). They introduce a policy to sell 100 tickets for a flight that can hold only 99 passengers. Bound the probability that the number of people that show up for a flight will be larger than the number of seats (assuming they show up independently).

2. An airline has collected an i.i.d. sample of 10000 flight reservations and figured out that in this sample 5 percent of passengers who made a reservation did not show up for the flight. They introduce a policy to sell 100 tickets for a flight that can hold only 99 passengers. Bound the probability of observing such sample and getting a flight overbooked.

   There are multiple ways to approach this question. We will guide you through two options. You are asked to solve the question in both ways.

   (a) Let p be the true probability of showing up for a flight (remember that p is unknown). In the first approach we consider two events: the first is that in the sample of 10000 passengers, where each passenger shows up with probability p, we observe 95% of show-ups. The second event is that in the sample of 100 passengers, where each passenger shows up with probability p, everybody shows up. Note that these two events are independent. Bound the probability that they happen simultaneously assuming that p is known. And then find the worst case p (you can do this numerically). With a simple approach you can get a bound of around 0.61. If you are careful and use the right bounds you can get down to around 0.0068.

   It is advised to visualize the problem (the $[0, 1]$ interval with 0.95 point for the 95% show-ups and 1 for the 100% show-ups and p somewhere in $[0, 1]$). This should help you to understand the problem; to understand where the worst case p should be; and to understand in what direction of inequalities you need.

   Attention: This is a frequentist rather than a Bayesian question. In case you are familiar with the Bayesian approach, it cannot be applied here, because we do not provide a prior on p. In case you are unfamiliar with the Bayesian approach, you can safely ignore this comment.

   (b) The second approach considers an alternative way of generating the two samples, using the same idea as in the proof of the VC-bound. Consider the following process of generating the two samples:

      i. We sample 10100 passenger show up events independently at random according to an unknown distribution p.

ii. And then we split them into 10000 passengers in the collected sample and 100 passengers booked for the 99-seats flight.

Bound the probability of observing a sample of 10000 with 95% show ups and a 99-seats flight with all 100 passengers showing up by following the above sampling protocol. If you do things right, you can get a bound of about 0.0062 (there may be some variations depending on how exactly you do the calculation).

2 The growth function (10 points)

1. Let $\mathcal{H}$ be a finite hypothesis set with $|\mathcal{H}| = M$ hypotheses. Prove that $m_{\mathcal{H}}(n) \leq \min\{M, 2^n\}$.
2. Let $\mathcal{H}$ be a hypothesis space with 2 hypotheses (i.e., $|\mathcal{H}| = 2$). Prove that $m_{\mathcal{H}}(n) = 2$. (Put attention that you are asked to prove the equality, $m_{\mathcal{H}}(n) = 2$, not an inequality.)
3. Prove that $m_{\mathcal{H}}(2n) \leq m_{\mathcal{H}}(n)^2$.

3 Support Vector Machines

In this part of the assignment, you should get familiar with support vector machines (SVMs).

SVM Software You need an SVM implementation, and you are welcome to use existing SVM software.[1] You are supposed to use Python in this course, still we make some comments on SVM implementations available in other languages, which may be useful in other contexts. We might get back to this question in future assignments, so it may pay off to write clean, reusable code.

We recommend using LIBSVM Chang and Lin (2011), which can be downloaded from `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. Interfaces to LIBSVM for many programming languages exist, including Matlab and Python. The SVM implementation `sklearn.svm.SVC` (see `https://scikit-learn.org/stable/modules/svm.html`) of Scikit-learn Pedregosa et al. (2011) is also based

[1]Carefully study what the SVM implementation you use is doing under the hood. Some implementations consider C/ℓ instead of C in the SVM objective function, where C denotes the regularization parameter. The SVM from the Matlab Bioinformatics Toolbox may by default use different regularization parameters depending on the class and the class frequency.

on LIBSVM.[2] For this exercise, use Gaussian kernels of the form

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{z}\|^2) \ . \tag{1}$$

Here $\gamma > 0$ is a bandwidth parameter that has to be chosen in the model selection process. Note that instead of γ often the parameter $\sigma = \sqrt{1/(2\gamma)}$ is considered.

Variable Stars A variable star changes its intensity, as observed by a telescope, over time. This can be caused extrinsically, for example by other objects temporarily occluding it, but also intrinsically, when the star changes its physical properties over time. Figure 1 shows an example. The graph of the varying intensity as a function of time is called the light curve. Variable stars can be further divided into many classes depending on other physical properties. The task we are trying to solve is to predict the class of a variable star by its light curve. To achieve this, we train a classifier in a supervised setting using labeled data from the All Sky Automated Survey Catalog of Variable Stars (ACVS) (Pojmanski, 2000).

The data considered in the following is based on the study by Richards et al. (2012). Each sample encodes the astronomical properties of a variable star in a 61-dimensional feature vector. The features are listed in Table 3, for a detailed description of their meaning we refer to Dubath et al. (2011) and Richards et al. (2011). From the originally 25 different classes, we consider just two classes, namely “Beta Persei” and “Beta Lyrae”, in order to evaluate binary SVMs.

3.1 Data understanding and preprocessing (8 points)

Read the data:

```
y_train = np.loadtxt('y_train1-1.csv', delimiter=',')
y_test = np.loadtxt('y_test1-1.csv', delimiter=',')
X_train = np.loadtxt('X_train_binary.csv', delimiter=',')
X_test = np.loadtxt('X_test_binary.csv', delimiter=',')
```

The are four files, the input patterns and their labels for training and testing, respectively. Report the class frequencies, that is, for each class report the number of data points belonging to that class divided by the total number of data points in the training data.

Then normalize the input data to zero mean and unit variance on the training data set. Consider the training data. Compute the mean and the variance of the

[2]For R, package such as E1071 (recommended) and KERNLAB are available. If you are comfortable with C++, you are encouraged to use the SVM implementation within the SHARK machine learning library Igel et al. (2008). Get the latest snapshot from `http://www.shark-ml.org/`.

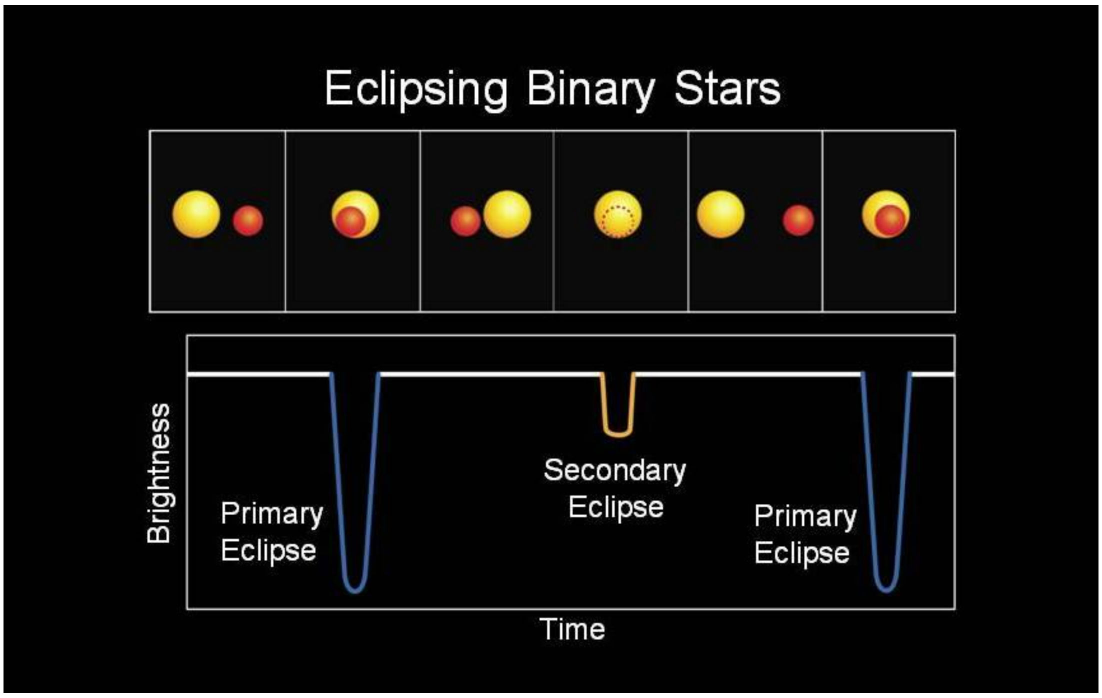

Figure 1: A variable star changes its intensity as observed from a telescope due to another smaller orbiting star. The image is taken from the NASA, `http://kepler.nasa.gov/news/nasakeplernews/index.cfm?FuseAction=ShowNews&NewsID=152`.

input features (i.e., for each component of the input vector independently). Find the affine linear mapping $f_{\text{norm}} : \mathbb{R}^{22} \to \mathbb{R}^{22}$ that transforms the training data such that the mean and the variance of every feature in the transformed data are 0 and 1, respectively (verify by computing these values).

Use the function f_{norm} to also encode the test data. Compute the mean and the variance of every feature in the transformed test data.

The normalization is part of the model building process. Thus, you may only use the training data for determining f_{norm} (always remember that you are supposed to not know the test data). See also Example 5.3 on pages 174-175 in the course textbook Abu-Mostafa et al. (2012).

Deliverables: number of training and test examples; code snippets for the normalization; mean and variance of features in the test data

#	Feature name
0	amplitude
1	beyond1std
2	flux_percentile_ratio_mid20
3	flux_percentile_ratio_mid35
4	flux_percentile_ratio_mid50
5	flux_percentile_ratio_mid65
6	flux_percentile_ratio_mid80
7	fold2P_slope_10percentile
8	fold2P_slope_90percentile
9	freq1_harmonics_amplitude_0
10	freq1_harmonics_amplitude_1
11	freq1_harmonics_amplitude_2
12	freq1_harmonics_amplitude_3
13	freq1_harmonics_freq_0
14	freq1_harmonics_rel_phase_1
15	freq1_harmonics_rel_phase_2
16	freq1_harmonics_rel_phase_3
17	freq2_harmonics_amplitude_0
18	freq2_harmonics_amplitude_1
19	freq2_harmonics_amplitude_2
20	freq2_harmonics_amplitude_3
21	freq2_harmonics_freq_0
22	freq2_harmonics_rel_phase_1
23	freq2_harmonics_rel_phase_2
24	freq2_harmonics_rel_phase_3
25	freq3_harmonics_amplitude_0
26	freq3_harmonics_amplitude_1
27	freq3_harmonics_amplitude_2
28	freq3_harmonics_amplitude_3
29	freq3_harmonics_freq_0
30	freq3_harmonics_rel_phase_1
31	freq3_harmonics_rel_phase_2
32	freq3_harmonics_rel_phase_3
33	freq_amplitude_ratio_21
34	freq_amplitude_ratio_31
35	freq_frequency_ratio_21
36	freq_frequency_ratio_31
37	freq_signif
38	freq_signif_ratio_21
39	freq_signif_ratio_31
40	freq_varrat
41	freq_y_offset
42	linear_trend
43	max_slope
44	median_absolute_deviation
45	median_buffer_range_percentage
46	medperc90_2p_p
47	p2p_scatter_2praw
48	p2p_scatter_over_mad
49	p2p_scatter_pfold_over_mad
50	p2p_ssqr_diff_over_var
51	percent_amplitude
52	percent_difference_flux_percentile
53	QSO
54	non_QSO
55	scatter_res_raw
56	skew
57	small_kurtosis
58	std
59	stetson_j
60	stetson_k

Table 1: Different features are used to describe the light curve of a variable star.

3.2 Model selection using grid-search (21 points)

The performance of your SVM classifier depends on the choice of the regularization parameter C and the kernel parameters (here γ). Adapting these *hyperparameters* is referred to as SVM *model selection*.

Use grid-search to determine appropriate SVM hyperparameters γ and C. Look at all combinations of $C \in \{c_1, c_2, \dots, c_7\}$ and $\gamma \in \{g_1, \dots, g_7\}$, where you have to choose proper grid points for yourself. Consider values around $C = 10$ and $\gamma = 0.1$ of different orders of magnitude. It is common to vary the values on a logarithmic scale (e.g., $0.001, 0.01, 0.1, 1, 10, 100$). For each pair, estimate the performance of the SVM using *5-fold cross validation.*

Cross validation is an important technique in machine learning and it is very important that you become familiar with it. You can read about cross-validation on pages 145-149 in the course textbook Abu-Mostafa et al. (2012).

Pick the hyperparameter pair with the lowest average 0-1 loss (classification error) and train an SVM with these hyperparameters using the complete training dataset. Only the training data must be used in the model selection process.

Report the values for C and γ you found in the model selection process, the 0-1 loss on the training data, as well as the 0-1 loss on the test data.

Deliverables: Description of software used; a short description of how you proceeded (e.g., did the cross-validation); training and test errors as well as the best hyperparameter configuration

3.3 Inspecting the kernel expansion (21 points)

A support vector x_i is bounded if the corresponding coefficient in the kernel expansion (usually denoted by α_i) has an absolute value of C. If $0 < \alpha_i < C$ then the support vector is free. Free support vectors lie on the margin (i.e., have a functional margin of 1). All wrongly classified patterns as well as the patterns that are correctly classified but are inside the margin area (i.e., they are too close to the decision boundary and have a functional margin less than 1) end up as bounded support vectors.

Let us consider the effect of the regularization parameter C. How do you expect the number of bounded and free support vectors change if C is drastically increased and decreased? Briefly justify your claim giving rigorous arguments.

In addition, verify and illustrate your claim experimentally by counting the number of free and bounded support vectors in solutions for the problem in this exercise for two (additional) different values of C. Report the C values and the (different) numbers of free and bounded support vectors. Show the part of the code that computes the number of free and bounded support vectors, respectively, in the report (in addition to the attached full source code in an archive).

Deliverables: Answer and rigorous argumentation, results of empirical validation including description of how these results were computed

References

Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin. *Learning from Data*. AMLbook, 2012.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions Intelligent Systems Technology*, 2(3):27:1–27:27, 2011.

P. Dubath, L. Rimoldini, M Süveges, J. Blomme, M López, L. M. Sarro, J. De Ridder, J. Cuypers, L. Guy, I. Lecoeur, et al. Random forest automated supervised classification of hipparcos periodic variable stars. *Monthly Notices of the Royal Astronomical Society*, 414(3):2602–2617, 2011.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

G. Pojmanski. The all sky automated survey. Catalog of about 3800 variable stars. *Acta Astronomica*, 50:177–190, 2000.

J. W. Richards, D. L. Starr, N. R. Butler, J. S. Bloom, J. M. Brewer, A. Crellin-Quick, J. Higgins, R. Kennedy, and M. Rischard. On machine-learned classification of variable stars with sparse and noisy time-series data. *The Astrophysical Journal*, 733(1):10, 2011.

J. W. Richards, D. L. Starr, H. Brink, A. A. Miller, J. S. Bloom, N. R. Butler, J. B. James, J. P. Long, and J. Rice. Active learning to overcome sample selection bias: Application to photometric variable star classification. *The Astrophysical Journal*, 744(2):192, 2012.