Machine Learning

2021-2022

Home Assignment 5

Fabian Gieseke **Christian Igel** **Yevgeny Seldin** **Sadegh Talebi**
Department of Computer Science
University of Copenhagen

The deadline for this assignment is **11 January 2022, 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.
- 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 VC dimension (38 points)

1. Let $\mathcal{H}$ be a finite hypothesis set with $|\mathcal{H}| = M$ hypotheses. Bound the VC-dimension of $\mathcal{H}$.

2. Let $\mathcal{H}$ be a hypothesis space with 2 hypotheses (i.e., $|\mathcal{H}| = 2$). Prove that $d_{\text{VC}}(\text{H}) = 1$.

3. What should be the relation between $d_{\text{VC}}(\text{H})$ and n in the VC generalization bound in Theorem 3.16 in Yevgeny's lecture notes in order for the bound to be non-trivial? [A bound on the loss that is greater than or equal to 1 is trivial, because we know that the loss is always bounded by 1. You do not have to make an exact calculation, giving an order of magnitude is sufficient.]

4. In the case of a finite hypothesis space, $|\mathcal{H}| = M$, compare the generalization bound that you can obtain with Theorem 3.16 in Yevgeny's lecture notes with the generalization bound in Theorem 3.2 in Yevgeny's lecture notes. In what situations which of the two bounds is tighter?

5. How many samples do you need in order to ensure that the empirical loss of a linear classifier selected out of a set of linear classifiers in $\mathbb{R}^{10}$ does not underestimate the expected loss by more than 0.01 with 99% confidence?

   Clarifications: (1) you are allowed to use the bound on the VC-dimension of linear classifiers mentioned in Yevgeny's lecture notes; (2) solving the question analytically is a bit tricky, you are allowed to provide a numerical solution. In either case (numerical or analytical solution), please, explain clearly in your report what you did.

6. Let $\mathcal{H}_+$ be the class of "positive" circles in $\mathbb{R}^2$ (each $h \in \mathcal{H}_+$ is defined by the center of the circle $c \in \mathbb{R}^2$ and its radius $r \in \mathbb{R}$; all points inside the circle are labeled positively and outside negatively). Prove that $d_{VC}(\mathcal{H}_+) \geq 3$.

7. Let $\mathcal{H} = \mathcal{H}_+ \cup \mathcal{H}_-$ be the class of "positive" and "negative" circles in $\mathbb{R}^2$ (the "negative" circles are negative inside and positive outside). Prove that $d_{VC}(\mathcal{H}) \geq 4$.

8. ***Optional question (0 points)*** Prove the matching upper bounds $d_{VC}(\mathcal{H}_+) \leq 3$ and $d_{VC}(\mathcal{H}) \leq 4$. [Doing this formally is not easy, but will earn you extra honor.]

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

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

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

12. **The fine details of the lower bound.** 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 that 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 $\mathcal{H}$.

    Hint: this can be achieved with an extremely simple example.

2 Random Forests (38 points)

2.1 Analyzing Satellite Data (30 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 1.[1] 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 $d = 9$ features separated via commas (each row

Figure 1: Landsat 8: Example image (true color) that is derived from the multi-spectral bands.

[1] 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/`

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 3000×3000 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 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? Given a random forest consisting of 10 trees, what is the smallest (guaranteed) upper bound for the asymptotic runtime to process a single new instance?

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).*

2.2 Normalization (8 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.

3 Principal Component Analysis (37.5 points)

Read section 9.2.1 in the online e-Chapter 9 of the textbook (Abu-Mostafa et al., 2012). The chapter can be downloaded from `http://book.caltech.edu/bookforum/showthread.php?t=4548`, the login is `bookreaders` and the password the first word on page 27 of the textbook. You can also find a scanned version of the section on Absalon.

3.1 PCA and preprocessing

3.1.1 By the book

Solve exercises 9.6 (b) and (c) on page e-Chap:9–8.

3.1.2 In the center

Assume a $d \times d$ matrix $\boldsymbol{S}$. Recall centering (e.g., from page e-Chap:9–2). Proof that if we center the matrix, then the rank of the resulting matrix is at most $d - 1$.

3.2 PCA in practice

Consider the notebooks `PCA and explained variance using scikit-learn.ipynb` and `PCA Assignment 5.ipynb`. The first one shines some light on the Scikit-Learn implementation of PCA and also contains a simple PCA implementation in NumPy.

In this assignment, you are supposed to extend `PCA Assignment 5.ipynb`. Show the code you added to the notebook in the report as well as the plots and images you are asked to produce.

3.2.1 Explained variance

Plot the explained variance as in the plot on the PCA lecture slides. That is, for each eigenvalue, sorted by magnitude, plot the eigenvalue divided by the sum of all eigenvalues (see `PCA and explained variance using scikit-learn.ipynb`). Use a log-scale for the y-axis.

Answer the following question: Are 10 principal components enough to explain 80% of the variance? You can directly see this if you plot the cumulative explained variance (e.g., by simply using `np.cumsum`) or by simply summing the explained variances of the first ten eigenvalues.

3.3 Eigendigits

Plot the first 5 "eigendigits" similar to the "eigenfaces" in the lecture. You plot the first eigenvectors from the PCA as 2D images (e.g., using `.reshape(imshape)`, where `imshape` is defined in `PCA Assignment 5.ipynb`).

PyTorch self-study (0 points)

We will use PyTorch later in this course. If you have some time at your hands duringthe long break and if you have not worked with PyTorch before, you may want to go through the introductory tutorial:

`https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html`

The content of this introduction is important for later assignments (reading it will again be part of a later assignment).

In addition, work through the following basic programming section in the *Dive into Deep Learning* book Zhang et al. (2021), which will improve your general Python programming skills:

- 2.1 Data Manipulation (`https://d2l.ai/chapter_preliminaries/index.html`)
- 2.2 Data Preprocessing
- 2.3 Linear Algebra

- 2.4 Calculus
- 2.5 Automatic Differentiation
- 2.7 Documentation
- 11.5.1 Vectorization and Caches (`https://d2l.ai/chapter_optimization/minibatch-sgd.html#vectorization-and-caches`)

4 Variable Stars (37.5 points)

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 2 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). We have a training and a test set, in the file `VSTrain.dt` and `VSTest.dt`, respectively, with 771 labeled samples each. Each sample encodes the astronomical properties of a variable star in a 61-dimensional feature vector. The features are listed in Table 4, for a detailed description of their meaning we refer to Dubath et al. (2011) and Richards et al. (2011). The labels indicate the class a variable star has been assigned to. In total there are 25 different classes, see Table 4.

4.1 Data understanding and preprocessing

Download the data in `VSTrain.dt` and `VSTest.dt`. Each line contains the input and as last value in a row the target label.

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 conduct two preprocessing steps.

1. Remove all data points belonging to classes with less than 65 training examples. Report which classes and how many training and test examples remain.

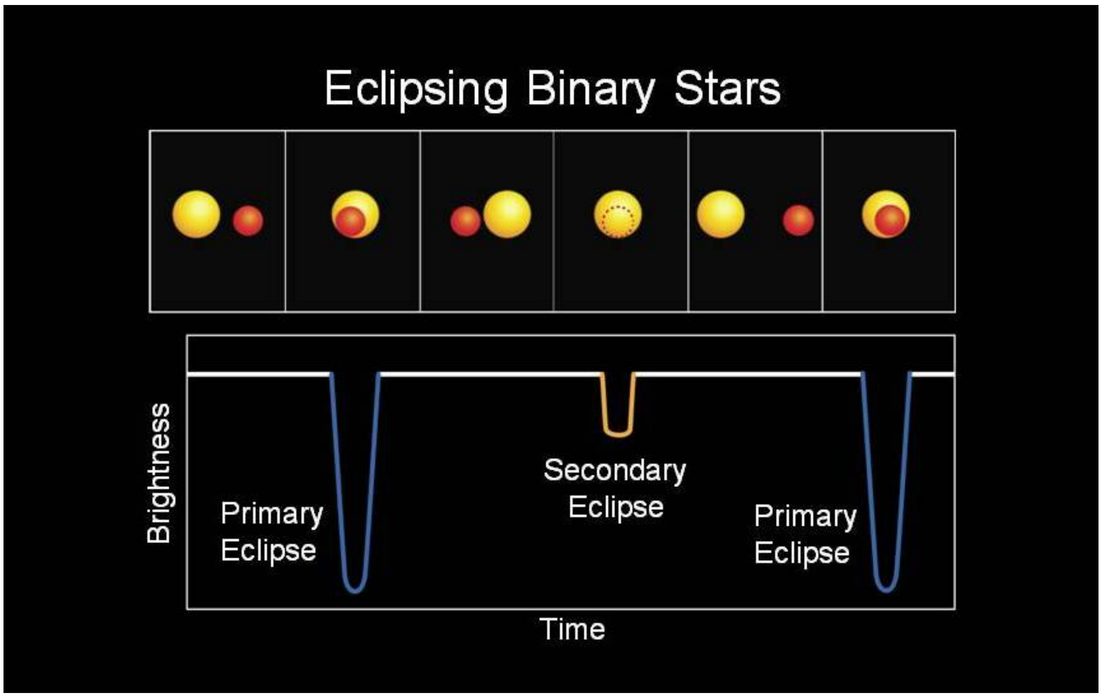

Figure 2: A variable star changes its intensity as observed from a telescope due to another smaller orbiting star. The image is taken from the NASA, `http://kepler.nasa.gov/news/nasakeplernews/index.cfm?FuseAction=ShowNews&NewsID=152`.

> *Hint:* Assuming `import numpy as np`, the Python functions `np.unique(..., return_counts=True)`, `np.where(...)`, and `np.delete(...)` (with `axis=0` on the input data) may be useful.

2. Normalize the data to zero mean and unit variance on the training data set.

Deliverables: frequency of classes in original; number of classes and number of training and test examples after removing small classes; code snippets for the normalization and data removal in the report

4.2 Principal component analysis

Perform a principal component analysis (PCA) of the normalized training data.[2] Visualize the data by a scatter plot of the data projected on the first two principal

[2]Note that PCA results including the eigenspectrum change due to the normalization.

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

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

components. Use different colors for the different classes in the plot so that the points belonging to different classes can be clearly distinguished.

Deliverables: scatter plot of the data projected on the first two principal components with different colors indicating the different classes

Label	Class name	Label	Class name
0	Mira	13	Gamma Doradus
1	Semireg PV	14	Pulsating Be
2	RV Tauri	15	Per. Var. SG
3	Classical Cep	16	Chem. Peculia
4	Pop. II Cephe	17	Wolf-Rayet
5	Multi. Mode C	18	T Tauri
6	RR Lyrae, FM	19	Herbig AE/BE
7	RR Lyrae, FO	20	S Doradus
8	RR Lyrae, DM	21	Ellipsoidal
9	Delta Scuti	22	Beta Persei
10	Lambda Bootis	23	Beta Lyrae
11	Beta Cephei	24	W Ursae Maj
12	Slowly Puls.		

Table 2: The 25 different classes a variable star can be assigned to.

4.3 Clustering

Perform clustering using `4-means` and `4-means++` of the training data.

After that, project the cluster centers to the first two principal components of the training data. Then visualize the clusters by adding the cluster centers to the plot from the previous exercise.

Briefly discuss the results.

Deliverables: description of software used; one plot with cluster centers and data points; short discussion of results

4.4 Classification (0 points, optional)

The task is to evaluate several multi-class classifiers on the data. Build the models using the training data only. The test data must only be used for final evaluation.

1. Apply multi-nominal logistic regression. If you use regularization, describe the type of regularization you used. Report training and test loss (in terms of 0-1 loss).

2. Apply random forests with 200 trees. In one setting, set the number of features considered when looking for the best split to the square root of the total number of features. In a second setting, set the number of features considered when looking for the best split to the total number of features. Report training and test loss (in terms of 0-1 loss) and the out-of-bag (OOB) *error.*

3. Apply k-nearest-neighbor classification. Use cross-validation to determine the number of neighbors. Report training and test loss (in terms of 0-1 loss). Describe how you determined the number of neighbors.

Deliverables: description of software used; training and test errors; OOB errors for the random forests; description of regularization and model selection process

References

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

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.

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.

A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. Dive into deep learning. *CoRR*, abs/2106.11342, 2021. URL `https://arxiv.org/abs/2106.11342`.