Machine Learning 2020–2021

Exam

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

You must submit your individual solution of the exam electronically via the **Digital Exam / Digital Eksamen** system. The deadline for submitting the exam is **Friday, 29 January 2021, at 16:00**. The exam must be solved **individually**. You are **not allowed** to work in groups or discuss the exam questions with other students. For fairness reasons any questions about the exam will be answered on Absalon. If your question may reveal the answer to other students, please, email it personally to the lecturers and we will either answer it on Absalon or tell you that we cannot answer your question.

WARNING: The goal of the exam is to evaluate your personal achievements in the course. We believe that take-home exams are most suitable for this evaluation, because they allow to test both theoretical and practical skills. However, our ability to give take-home exams strongly depends on your honesty. Therefore, any suspicion of cheating, in particular collaboration with other students, will be directly reported to the head of studies and prosecuted in the strictest possible way. It is also strictly prohibited to post the exam questions or parts thereof on the Internet or on discussion forums and to seek help on discussion forums. And you are not allowed to store your solutions in open access version control repositories or to post them on the Internet or on discussion forums. Be aware that if proven guilty you may be expelled from the university. Do not put yourself and your fellow students at risk.

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 this PDF file. Do *not* include the task description or parts thereof in your report.
- Please, use the provided LaTeX template for typing your report.

- 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.
- Your code should be structured such that there is one main file 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 also include a README text file describing how to run your program.

1 Perceptron (12 points)

In this assignment (inspired by Blum et al., 2020), we extend the perceptron algorithm as introduced and analyzed on slides 27–33 from the "Linear Classification" lecture.

Let $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ be a non-trivial training set (i.e., containing patterns of both classes), $\boldsymbol{w}_0 = \boldsymbol{0}$, and let

$$R \leftarrow \max_{1 \leq i \leq N} \|\boldsymbol{x}_i\| \ .$$

Suppose that there exists $\boldsymbol{w}_{\text{opt}}$ with $\|\boldsymbol{w}_{\text{opt}}\| = 1$ such that the training patterns are separable by margin $\rho > 0$, that is,

$$y_i \langle \boldsymbol{w}_{\text{opt}}, \boldsymbol{x}_i \rangle \geq \rho > 0$$

for $1 \leq i \leq N$. Then Novikoff's theorem tells us that the number of updates k made by the online perceptron algorithm on S is at most

$$\left(\frac{R}{\rho}\right)^2 \ .$$

However, the standard perceptron algorithms typically does not find a separator of large margin. If we also want to find a separator of large margin, a natural alternative is to update the parameter vector whenever an example $(\boldsymbol{x}, y)$ does not meet the target margin, that is, for a target margin of 1 if $y\langle \boldsymbol{w}, \boldsymbol{x} \rangle < 1$. This leads to the following algorithm:

```
Algorithm 1: Margin Perceptron
  Input: separable data $\{(\boldsymbol{x}_1, y_1), \dots\} \subseteq (\mathbb{R}^n \times \{-1, 1\})^N$
  Output: hypothesis $h(\boldsymbol{x}) = \operatorname{sgn}(\langle \boldsymbol{w}_k, \boldsymbol{x} \rangle) = \operatorname{sgn}(f(\boldsymbol{x}))$
1 $\boldsymbol{w}_0 \leftarrow \boldsymbol{0}; k \leftarrow 0$
2 repeat
3     for $i = 1, \dots, N$ do
4         if $y_i \langle \boldsymbol{w}_k, \boldsymbol{x}_i \rangle < 1$ then
5             $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k + y_i \boldsymbol{x}_i$
6             $k \leftarrow k + 1$
7 until no mistake made within for loop
```

Recall the hinge loss

$$L_{\text{hinge}}(y, f(x)) = [1 - yf(x)]_+ = \max(0, 1 - yf(x))$$

from the lecture on "Support Vector Machines" as well as stochastic gradient descent (SGD) from the lecture "Linear Classification" and the slide "Online vs batch learning iteration" from the lecture "Deep Learning Part I: Neural Networks".

Prove that the margin perceptron algorithm above is equivalent to running stochastic gradient descent on the class of linear decision functions $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$ using hinge loss as the loss function and using a constant learning rate of 1.

2 Kernels (12 points)

This assignment tests the understanding of kernel functions (inspired by Blum et al., 2020). In the following, you may use all rules stated on slide 30 from the "Kernels" lecture.

1. Let $k_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel and $f : \mathcal{X} \to \mathbb{R}$. Prove that

$$k(x, z) = f(x)f(z)k_1(x, z)$$

   is a valid kernel.

   Comment: You can solve this directly without reference to the rules on slide 30 from the lecture. You can start from here: Because k_1 is a valid kernel, there exists a feature map Φ_1 such that $k_1(x, z) = \langle \Phi_1(x), \Phi_1(z) \rangle$.

2. Use the above result to prove that the Gaussian kernel

$$k(x, z) = e^{-\gamma \|x - z\|^2}$$

   for $\gamma > 0$ is a valid kernel.

3 Variable Stars (36 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 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]. 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 3, 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 3.

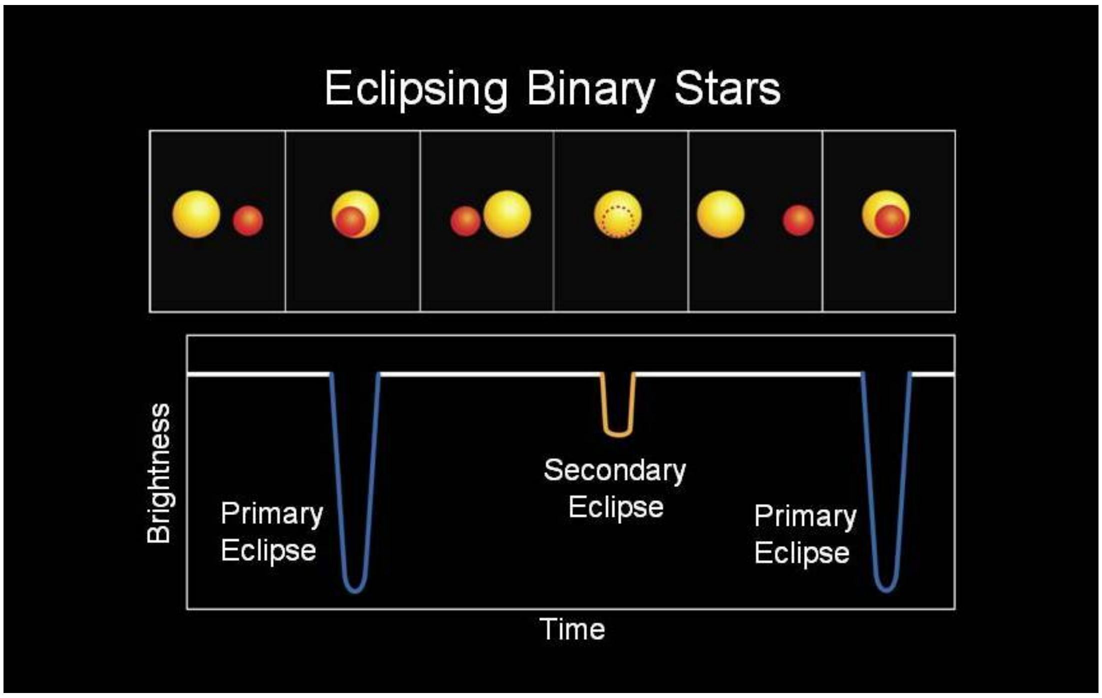

Figure 1: A variable star changes its intensity as observed from a telescope due to another smaller orbiting star. The image is taken from the NASA, `http://kepler.nasa.gov/news/nasakeplernews/index.cfm?FuseAction=ShowNews&NewsID=152`.

#	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.

3.1 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.

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.

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.

   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

3.2 Principal component analysis

Perform a principal component analysis (PCA) of the normalized training data.[1] Visualize the data by a scatter plot of the data projected on the first two principal 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

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

3.3 Clustering

Perform 4-means clustering 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

3.4 Classification

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

4 Theoretically justified cross-validation (20 points)

Cross-validation is arguably the most widely-used method for parameter tuning in machine learning applications. The standard cross-validation procedure prescribes to cross-validate the parameters using cross-validation split of the data, select the best parameter [according to the cross-validation error], and then train the final model using the best parameter and all the available data (see, for example, Abu-Mostafa et al. [2012, Page 149]). While this approach is fairly robust [as long as the number of cross-validated parameters is reasonable in relation to the number of training data points], it has no theoretical guarantees and may potentially lead to overfitting [Kearns et al., 1997, Kearns and Ron, 1999]. In this question you will develop an alternative approach, which is theoretically justified. It is not the best possible, but it is good enough to obtain non-trivial generalization guarantees.

The modified cross-validation procedure that you will analyze goes as follows. We have a dataset S of size n and a set of parameters $\theta_1, \dots, \theta_M$ we want to cross-validate (for example, the number of neighbors in K-NN). We make K splits of S into training and validation subsets. For the i-th split we have $S = S_i^{\texttt{train}} \cup S_i^{\texttt{val}}$, where $S_i^{\texttt{train}}$ is the i-th training subset and $S_i^{\texttt{val}}$ is the i-th validation subset. We denote the size of validation subsets by $n_{\texttt{val}}$ and assume it is the same for all i. We train models with parameters $\theta_1, \dots, \theta_M$ on the training subsets and validate them on the corresponding validation subsets. Let $\hat{L}(h_{ij}, S_i^{\texttt{val}})$ denote the validation error of the model h_{ij} trained on $S_i^{\texttt{train}}$ with parameter θ_j and validated on $S_i^{\texttt{val}}$. For each i we pick the parameter $\theta_{j_i^*}$, which minimizes the corresponding validation loss, $j_i^* = \arg\min_j \hat{L}(h_{ij}, S_i^{\texttt{val}})$. The predictions on new data points are then made by taking a majority vote of the K models, $h_{1j_1^*}, \dots, h_{Kj_K^*}$.

The procedure above has several differences with the standard cross-validation procedure: (1) we do not select a single best parameter θ^*, but select the best parameter for each split $\theta_{j_i^*}$; (2) at the end instead of training a new model with parameter θ^* using all the data S, we take a majority vote of the K selected models; and (3) in the standard cross-validation it is common to have small validation sets, up to the limit of leave-one-out cross-validation [validation sets of size 1], whereas in the proposed procedure it is beneficial to have large validation sets, roughly of order $n/2$. Due to the latter difference it is also more natural to take K random unrelated splits of the data into $S^{\texttt{train}}$ and $S^{\texttt{val}}$ rather than the "orderly" partition into K folds typically used in cross-validation (where each time $K-1$ folds are used for training and 1 fold for validation).

After this long introduction, it is your turn to do the analysis.

1. Let $L(h_{ij})$ denote the expected loss of prediction model h_{ij} trained on $S_i^{\texttt{train}}$ with parameter θ_j. Derive a bound on $L(h_{ij})$ in terms of $\hat{L}(h_{ij}, S_i^{\texttt{val}})$ that holds simultaneously for all the models h_{ij} involved in the modified cross-validation procedure with probability at least $1-\delta$.

2. Let MV denote the majority vote of $h_{1j_1^*}, \dots, h_{Kj_K^*}$ and let $L(\mathrm{MV})$ denote the expected loss of MV. Show that $L(\mathrm{MV}) \leq \frac{2}{K}\sum_{i=1}^K L(h_{ij_i^*})$. Hint: the majority vote errs when at least half of the classifiers $h_{1j_1^*}, \dots, h_{Kj_K^*}$ make

an error. You should turn this observation into a formal argument. There are two ways of doing it, you can pick any of the two: (a) Consider losses on individual examples, $\ell(\mathrm{MV}(X), Y)$, and bound $\ell(\mathrm{MV}(X), Y)$ in terms of the losses $\ell(h_{ij_i^*}(X), Y)$ in the case when $\ell(\mathrm{MV}(X), Y) = 0$ and $\ell(\mathrm{MV}(X), Y) = 1$. (b) We have $L(\mathrm{MV}) = \mathbb{E}\left[\ell(\mathrm{MV}(X), Y)\right] \leq \mathbb{P}\left(\frac{1}{K}\sum_{i=1}^K \ell(h_{ij_i^*}) \geq \frac{1}{2}\right)$. Take a random variable $Z = \frac{1}{K}\sum_{i=1}^K \ell(h_{ij_i^*})$ and apply Markov's inequality to Z.

3. Bound the expected error of the final model, $L(\mathrm{MV})$, in terms of the observed validation losses. The bound should hold with probability $1 - \delta$.

4. Discuss the dependence of the bound you have obtained above on the size of the validation sets $n_{\texttt{val}}$. Which terms in the bound are likely to increase as we increase $n_{\texttt{val}}$ and which terms are likely to decrease as we increase $n_{\texttt{val}}$? What can go wrong if $n_{\texttt{val}}$ is too small (say, $n_{\texttt{val}} = 1$) and if it is too large (say, $n_{\texttt{val}} = n - 1$)?

5 Properties of the VC dimension (20 points)

Let $d_{\mathrm{VC}}(\mathcal{H})$ denote the VC-dimension of a hypothesis class $\mathcal{H}$. For a finite hypothesis class $\mathcal{H}$ let $|\mathcal{H}|$ denote the size of $\mathcal{H}$ (the number of hypotheses in $\mathcal{H}$). All the questions below consider binary classification.

To avoid any confusion: a hypothesis h is a function from $\mathcal{X}$ to $\{\pm 1\}$ and if two hypotheses h_1 and h_2 label $\mathcal{X}$ identically then they are identical and count as a single hypothesis in $\mathcal{H}$.

1. Prove that if $|\mathcal{H}| = 2$ then $d_{\mathrm{VC}}(\mathcal{H}) = 1$.

2. For the following two statements prove whether they are true or false:

   (a) If $|\mathcal{H}|$ is infinite, then $d_{\mathrm{VC}}(\mathcal{H})$ is infinite.

   (b) If $d_{\mathrm{VC}}(\mathcal{H})$ is infinite, then $|\mathcal{H}|$ is infinite.

3. Let the sample space $\mathcal{X} = \{1, \ldots, 8\}$ be a finite set of size 8, let $\mathcal{H}_4$ be a hypothesis class with 4 hypotheses, and $\mathcal{H}_8$ be a hypothesis class with 8 hypotheses. We emphasize that $\mathcal{H}_4$ is *not* necessarily a subset of $\mathcal{H}_8$. What is the relation between $d_{\mathrm{VC}}(\mathcal{H}_4)$ and $d_{\mathrm{VC}}(\mathcal{H}_8)$? Your answer should be one of the following statements:

   (a) We always have $d_{\mathrm{VC}}(\mathcal{H}_4) < d_{\mathrm{VC}}(\mathcal{H}_8)$.

   (b) We always have $d_{\mathrm{VC}}(\mathcal{H}_4) \leq d_{\mathrm{VC}}(\mathcal{H}_8)$.

   (c) It is possible to construct an example of $\mathcal{H}_4$ and $\mathcal{H}_8$, such that $d_{\mathrm{VC}}(\mathcal{H}_4) > d_{\mathrm{VC}}(\mathcal{H}_8)$.

Pick the right statement and prove it. Answers without a proof will not be accepted.

4. Answer the same question when $\mathcal{X} = \{1, 2, 3\}$ is a finite set of 3 points.

References

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

A. Blum, J. Hopcroft, and R. Kannan, editors. *Foundations of Data Science*. Cambridge University Press, 2020.

P. Dubath, L. Rimoldini, M. Süveges, J. Blomme, M. López, L. 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.

M. Kearns, Y. Mansour, A. Ng, and D. Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27, 1997.

M. J. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11, 1999.

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.