Machine Learning 2019-2020

Final 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, 24 January 2019, 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.
- 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 Logistic Regression (4 points)

We consider binary logistic regression. Let the input space be $\mathbb{R}^d$ and the label space be $\{0, 1\}$. Let our model f with parameters $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ model:

$$f(\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) = P(Y = 1 \,|\, X = \boldsymbol{x}) \tag{1}$$

Prove that if the (affine) linear part of the model encodes the log-odds, that is, if

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = \ln \frac{P(Y = 1 \,|\, X = \boldsymbol{x})}{P(Y = 0 \,|\, X = \boldsymbol{x})} \ , \tag{2}$$

then σ is the logistic function. That is, if $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$ encodes on log-scale how frequent class 1 occurs relative to class 0, then σ is the logistic function.

2 Linear Decision Boundaries from Logistic Regression and SVMs (18 points)

We consider the regularized loss function used in logistic regression

$$L((\mathbf{w}, b), S) = \frac{1}{N} \sum_{n=1}^{N} \ln P(Y = y_n \,|\, X = \boldsymbol{x}_n) + C\|\mathbf{w}\|^2 \tag{3}$$

with positive $C \in \mathbb{R}$. Here $(\mathbf{w}, b)$ are the parameters of the model and $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ are some data. That is, we want to minimize the logarithmic likelihood of the training data plus a two-norm regularizer.

Consider $N = 3$ and the three training data (x, y) pairs $S_1 = \{(0, 0), (2, 1), (2, 1)\}$.[1] and the alternative training data $S_2 = \{(0, 0), (0, 0), (2, 1)\}$.

[1] Note that S_1 has entries that are the same. Thus, it is strictly speaking a multi-set. Still, in machine learning we simply call it a data set (with three elements) and use the set notation $\{\ldots\}$.

1. Assume that you found a logistic regression model defined by the parameters $(\boldsymbol{w}, b)$ by minimizing (3) for S_1 that separates the data. Is the model also optimal (in the sense of (3)) for the data set S_2? Is the decision boundary also optimal (in the sense of (3)) for the data set S_2? Provide a rigorous proof for your answer.

2. Consider a hard-margin SVM with linear kernel and offset parameter. What is the decision boundary (the hyperplane) if you train it on S_1 and S_2, respectively? When you report the hyperplanes, remember to use the correct dimensionality. Do the hyperplanes differ? Give a short explanation why or why not.

3 Kernels (6 points)

We consider a binary classification task with input space $\mathbb{R}^{2d}$ for $d > 0$. Assume that a colleague suggests the following kernel

$$k(\boldsymbol{x}, \boldsymbol{z}) = (1-\gamma)e^{-\|[\boldsymbol{x}]_{1:d}-[\boldsymbol{z}]_{1:d}\|^2} + \gamma\langle[\boldsymbol{x}]_{d+1:2d}, [\boldsymbol{z}]_{d+1:2d}\rangle \tag{4}$$

for $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^{2d}$, where $0 \leq \gamma \leq 1$ and $[\boldsymbol{x}]_{1:d}$ denotes the first d elements of a vector $\boldsymbol{x}$ (i.e., $[\boldsymbol{x}]_{1:d} = (x_1, \ldots, x_d)^{\mathrm{T}}$) and accordingly $[\boldsymbol{x}]_{d+1:2d}$ the last d. The idea behind this suggestion is that the first half of the input is fed into a Gaussian kernel and the second half into a linear kernel and the importance of the two parts can be controlled by a hyperparameter γ.

Is (4) really a proper positive definite kernel? If your answer is yes, provide a proof using the techniques presented in the lecture (see also the slides). If your answer is no, provide a proof, for example by providing a (simple) counterexample.

4 Sleep Staging (35 points)

Sleep is one of the most fundamental physiological processes, and abnormal sleeping patterns are associated with poor health. They may, for example, indicate brain- & heart diseases, obesity and diabetes. During sleep our brain goes through a series of changes between different *sleep stages*, which are characterized by specific brain and body activity patterns. *Sleep staging* refers to the process of mapping these transitions over a night of sleep. This is of fundamental importance in sleep medicine, because the sleep patterns combined with other variables provide the basis for diagnosing many sleep related disorders (Kales and Rechtschaffen, 1968, Iber and AASM, 2007). The stages can be determined by measuring the neuronal activity in the cerebral cortex (via electroencephalography, EEG), eye movements (via electrooculography, EOG), and/or the activity of facial muscles

(via electromyography, EMG) in a *polysomnography* (PSG) study. The classification into stages is done manually. This is a difficult and time-consuming process, in which expert clinicians inspect and segment the typically 8–24 hours long multi-channel signals. Contiguous, fixed-length intervals of 30 seconds are considered, and each of these *segments* is classified individually. Algorithmic sleep staging aims at automating this process. The state-of-the-art in algorithmic sleep staging is marked by deep neural networks, which can be highly accurate and robust, even compared to human performance, see the recent work by Perslev et al. (2019) and references therein.

This assignment considers algorithmic sleep staging. The data is based on a single EEG channel from the Sleep-EDF-15 data set (Kemp et al., 2000, Goldberger et al., 2000). The input is given by an intermediate representation from the U-Time neural network architecture (Perslev et al., 2019), the targets are sleep stages. We created a training and test set, the inputs and the corresponding labels can be found in `X_train.csv` and `y_train.csv` and `X_test.csv` and `y_test.csv`, respectively.

4.1 Data understanding and preprocessing

Download and extract the data.

Consider the training data `X_train.csv` and the corresponding labels `y_train.csv`. Report the class frequencies, that is, for each of the 5 classes report the number of data points divided by the total number of data points) in the training data.

The ith row in `X_train.csv` are the features of the ith training pattern. The class label of the ith pattern is given in the ith row of `y_train.csv`.

Deliverables: description of software used; frequency of classes

4.2 Principal component analysis

Perform a principal component analysis of the training data `X_train.csv`. Plot the eigenspectrum (see the plot on slide 28 of the *PCA* slides for an example). How many components are necessary to "explain 90 % of the variance"? 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.

Deliverables: description of software used; plot of the eigenspectrum; number of components necessary to explain 90 % of variance; scatter plot of the data projected on the first two principal components with different colors indicating the 5 different classes

4.3 Clustering

Perform 5-means clustering of `X_train.csv`. 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.[2]

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

4.4 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 50, 100, and 200 trees. Report training and test loss (in terms of 0-1 loss).
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; description of regularization and model selection process

5 Cross-Testing (14 points)

You have a dataset with n sample points and you give it to two students. One of them applies 5-fold cross-testing: the data are split into 5 folds and then each time one fold is reserved for testing and the remaining four are used to train a prediction model. Each of the 5 folds is used for testing only once. This way the student obtains 5 prediction models and gives you the one that has achieved the lowest test error. The other student applies 10-fold cross-testing in the same way and returns you the model with the lowest test error.

[2] In this example, you do not get 5 nicely separated clusters in 2D. A better looking projection of the data can be achieved using a non-linear dimensionality reduction technique, for example *non-linear t-Distributed Stochastic Neighbor Embedding* (t-SNE) (van der Maaten and Hinton, 2008). Although not part of the exam, we recommend that you try it out.

The two students work independently of each other. Let h^{1^*} be the model produced by the first student and $\hat{L}(h^{1^*})$ its test loss and let h^{2^*} be the model produced by the second student and $\hat{L}(h^{2^*})$ its test loss.

1. You pick one of the two models given to you by the students, let us call it h^*. Derive a generalization bound for the expected loss of h^*. If you find it more convenient, you are allowed to provide separate expressions for the case $h^* = h^{1^*}$ and $h^* = h^{2^*}$.

2. How would you select among h^{1^*} and h^{2^*}? For example, if $\hat{L}(h^{1^*}) = 0.1$, $\hat{L}(h^{2^*}) = 0.05$, $n = 10000$, and you take confidence parameter $\delta = 0.01$. Which model would you pick?

6 Early Stopping (24 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 Goodfellow et al. (2016, Section 7.8)[3]. In this question we analyze several ways of implementing it. The technique sets aside a validation set S_{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 (you do not have to know the details of the 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_{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 (this is the procedure proposed in Goodfellow et al. (2016, Algorithm 7.1) or `https://www.quora.com/`

[3] `https://www.deeplearningbook.org/contents/regularization.html`

How-does-one-employ-early-stopping-in-TensorFlow). It then returns the best model observed ever during training.

2. Derive a high-probability bound (a bound that holds with probability at least $1-\delta$) on $L(h_{t^*})$ in terms of $\hat{L}(h_{t^*}, S_{\text{val}})$, δ, and the size n of the validation set S_{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 $\sum_{i=1}^{\infty} \frac{1}{i(i+1)} = 1$ or $\sum_{i=1}^{\infty} \frac{6}{\pi^2 i^2} = 1$.[4]

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?

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:

   (a) 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 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

[4]For the first series we have $\sum_{i=1}^{\infty} \frac{1}{i(i+1)} = \sum_{i=1}^{\infty} \left(\frac{1}{i} - \frac{1}{i+1}\right) = 1$, for summation of the second series see https://en.wikipedia.org/wiki/Basel_problem. You should only use one of the two series.

show that the adaptive bound is at most a multiplicative factor of $\sqrt{2}$ larger than the non-adaptive bound. Ignore small additive constants in the adaptive bound. (With the additive constants the relation is a very tiny bit worse than $\sqrt{2}$, but the difference is so small that we will ignore it.)

(b) Small technical detail: when you do the analysis above you have to assume that $T \leq T_{\max}$. If $T > T_{\max}$ then the adaptive bound is trivially 1 and you can show that the non-adaptive bound is at least $\frac{1}{\sqrt{2}}$ (again, if we ignore small additive constants). So in the latter case the adaptive bound is also worse by at most a multiplicative factor of $\sqrt{2}$. You do not have to analyze the case $T > T_{\max}$.

(c) 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 $T < T_{\max}$.

References

A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. doi: 10.1161/01.CIR.101.23.e215.

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

C. Iber and AASM. *The AASM manual for the scoring of sleep and associated events: rules, terminology and technical specifications*. American Academy of Sleep Medicine, Westchester, I. L., 2007.

A. Kales and A. Rechtschaffen. *A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects. Allan Rechtschaffen and Anthony Kales, editors*. U. S. National Institute of Neurological Diseases and Blindness, Neurological Information Network Bethesda, Md, 1968.

B. Kemp, A. H. Zwinderman, B. Tuk, H. A. C. Kamphuisen, and J. J. L. Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomedical Engineering*, 47(9): 1185–1194, 2000. doi: 10.1109/10.867928.

M. Perslev, M. Hejselbak Jensen, S. Darkner, P. J. Jennum, and C. Igel. U-time: A fully convolutional network for time series segmentation applied to sleep

staging. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.