Assignment 8

Statistical Machine Learning

Moritz Haas / Prof. Ulrike von Luxburg

Summer term 2022 — due **June 20th at 12:00**

Exercise 1 (Bootstrap, 2+1+1+2+3 points)
Let $X_1, \dots, X_n$ be a sample from the uniform distribution $U[0, \theta]$ with $\theta > 0$.
In this exercise we will see an example where the nonparametric bootstrap fails, while the parametric bootstrap works.

(a) Show that the maximum likelihood estimator for θ is given by $\hat{\theta}_n = \max_{i \in [n]} X_i$. Derive the distribution function $F_{\hat{\theta}_n}(z) = P(\hat{\theta}_n \leq z)$ of $\hat{\theta}_n$.

(b) Sample $n = 25$ independent observations from $U[0, 1]$. Implement the *nonparametric bootstrap*, which draws n samples $X_1^*, \dots, X_n^*$ from $X_1, \dots, X_n$ independently with replacement and calculates $\hat{\theta}_n^* = \max_{i \in [n]} X_i^*$.

Draw $B = 5000$ bootstrap samples, compute $\hat{\theta}_{n,i}^*$, $i = 1, \dots, B$ and make a histogram of these values (see Jupyter notebook).

(c) Let $T_n = n(\theta - \hat{\theta}_n)$ and $T_n^* = n(\hat{\theta}_n - \hat{\theta}_n^*)$. Show that

$$P(T_n^* \leq 0) = 1 - (1 - 1/n)^n.$$

(d) With the bootstrap procedure we want to learn something about the distribution of $\hat{\theta}_n$. Therefore we try to estimate the distribution of T_n using T_n^*. Show that the nonparametric bootstrap does not behave as it should by showing

$$\limsup_{n \to \infty} \sup_{t \in \mathbb{R}} |P(T_n \leq t) - P(T_n^* \leq t)| \geq 1 - e^{-1}.$$

Hint: You can use $(1 + x/n)^n \to e^x$ *for* $n \to \infty$ *and* $x \in \mathbb{R}$. *Why does*

$$\sup_{t \in \mathbb{R}} |P(T_n \leq t) - P(T_n^* \leq t)| \geq P(T_n^* \leq 0)$$

hold?

(e) Instead of only drawing from the data set, the *parametric bootstrap* generates bootstrap samples $\tilde{X}_1, \dots, \tilde{X}_n$ by drawing independently from a uniform distribution on $[0, \hat{\theta}_n]$. Then θ is estimated via $\tilde{\theta}_n = \max_{i \in [n]} \tilde{X}_i$.

Repeat the following for $N = 500$ independent realizations of our training set $X_1, \dots, X_n$ from $U[0, 1]$: Draw $B = 200$ bootstrap samples using the nonparametric and the parametric bootstrap and compute T_n, T_n^* and $\tilde{T}_n = n(\hat{\theta}_n - \tilde{\theta}_n)$.

Then compute the empirical distribution function of T_n across realizations

$$\hat{F}_{T_n}(z) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(T_{n,i} \leq z),$$

across the realizations $T_{n,1}, \dots, T_{n,N}$. For the predefined array `zs` of z-values, save $\hat{F}_{T_n}$(`zs[i_z]`) to `Tn[i_z]`.

Compute the empirical distribution function of T_n^*

$$\hat{F}_{T_n^*}(z) = \frac{1}{B} \sum_{i=1}^{B} \mathbb{1}(T_{n,i}^* \leq z),$$

across the bootstrap samples $T^*_{n,1}, \ldots, T^*_{n,B}$, for each realization. For the i-th realization and `zs[i_z]`, save $\hat{F}_{T^*_n}$(`zs[i_z]`) to `Tn_nb[i_z,i]`. Repeat the same procedure for $\tilde{T}_n$ and `Tn_pb`.

The Jupyter notebook has preimplemented code to jointly plot the empirical distribution functions with uncertainty bands across realizations.

Exercise 2 (Boosting, 1+3+1 points)
Consider a binary classification setting with training data $S = \{(x_i, y_i) \in \mathcal{X} \times \{-1, +1\} \mid i = 1, \ldots, n\}$, and a finite class of weak base classifiers $\mathcal{H} = \{h_t : \mathcal{X} \to \{-1, +1\} \mid t = 1, \ldots, T\}$. As introduced in the lecture, AdaBoost is an iterative algorithm, that combines several weak classifiers in a weighted sum. In each round, data points that have been classified falsely in the previous round receive a larger weight, and correctly classified points a smaller one. We denote the weighted errors in each round by $\varepsilon_t = P_{i\sim D_t}(y_i \neq h_t(x_i))$. Concretely the weights are updated as

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-y_i \alpha_t h_t(x_i)\right)}{Z_t},$$

where $i = 1, \ldots, n$, $\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$ and Z_t is a normalization constant so that D_{t+1} becomes a probability distribution. The final classifier is given by $h_S(x) = \operatorname{sign} f(x)$, where f is the 'confidence function'

$$f(x) = \frac{\sum_{t=1}^T \alpha_t h_t(x)}{\sum_{t=1}^T \alpha_t}.$$

As for SVMs, under mild assumptions, it has been found that a classifier h with many 'confident' correct predictions $yh(x) > \theta$ for some margin $\theta \in (0, 1)$ has desirable generalization properties. The intuition is that when a sample is classified with a large margin, a small perturbation of the classifier will not change the classification of that sample, increasing the classifier's robustness.
In this exercise, we will show that AdaBoost produces such large margins with high probability, when the weak classifiers are better than random guessing. The quantity of interest is the probability, that the margin θ is violated in the (fixed) training set, $P_{(x,y)\sim S}\big(yf(x) \leq \theta\big)$ in (1). Part (a) prepares the main result in (b) and part (c) ensures the usefulness of the bound (1).

(a) Show that $Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$.

Hint: Which values can $\exp\left(-y_i \alpha_t h_t(x_i)\right)$ *attain?*

(b) By $P_{(x,y)\sim S}$ we denote the distribution where each element of S has the same weight $1/n$. For our 'confidence' function $f(x)$ from above, show that for all $\theta \in (0, 1)$,

$$P_{(x,y)\sim S}\big(yf(x) \leq \theta\big) \leq 2^T \prod_{t=1}^T \sqrt{\varepsilon_t^{1-\theta}(1-\varepsilon_t)^{1+\theta}}. \tag{1}$$

Hint: First show

$$P_{(x,y)\sim S}\big(yf(x) \leq \theta\big) \leq \mathbb{E}_{(x,y)\sim S} \exp\left(-y \sum_{t=1}^T \alpha_t h_t(x) + \theta \sum_{t=1}^T \alpha_t\right).$$

Then you can use the identity

$$\exp\left(-y_i \sum_{t=1}^T \alpha_t h_t(x_i)\right) = D_T(i) \frac{\prod_{t=1}^T Z_t}{D_0(i)},$$

where $D_0(i) = 1/m$, *which holds by definition.*

(c) Show that the right hand side of (1) decreases with increasing number of rounds T, by showing that the function $f : [0, 1] \to [0, 1]$, $f(x) = x^{1-\theta}(1-x)^{1+\theta}$ has its unique maximum $1/4$ at $x = \frac{1-\theta}{2}$.

Exercise 3 (Random forests, 2+1+2 points)
In this exercise we explore the influences of various hyperparameter choices on the interpolation and generalization performance of random forests in a high-dimensional regression task. Download the data sets `rfdata_train` and `rfdata_test` from the website. You may import `RandomForestRegressor` from `sklearn.ensemble`.

(a) Fit random forests with the number of trees in [1, 10, 100, 1000] and the minimum number of samples in each leaf in [1, 5, 10]. Plot heat maps of the coefficient of determination evaluated on the train and on the test set respectively (implemented in the `.score`-method of the class `RandomForestRegressor`). State three observations you draw from these 2 heatmaps.

(b) Now use random forests with 1000 trees and at least 1 sample per leaf. Repeat the procedure of (a), now varying the number of features used in each split and the number of samples to train each base estimator. Which conclusions do you draw from these 2 heatmaps?

(c) Only 12 features are actually informative. The index list of informative features is given in the notebook. Can you recover the correct features using a random forest? Explain your procedure and one alternative procedure, that you do not need to apply.

Exercise 4 (Feedback Poll: Week 8, 0 bonus points but 1 gratitude)
Weekly feedback survey on Ilias about the lecture, tutorials and exercise sheets. If you participate, it helps us in understanding what we can improve. Future polls will be anonymous to ensure open feedback. The poll asks about the lectures that cover the topics treated in the respective sheet.