MTAT.03.227 Machine Learning
Spring 2015 / Exercise session IV
Nominal score: 10p
Maximum score: 15p
Deadline: 1st of March 16:15 EET

1. Given enough information about future data samples, it is possible to find a class with optimal accuracy. Describe the optimal classification rule $g_* : \{0, \ldots, 3\} \to \{0, 1\}$ when the future data sample is known to be chosen randomly from the following set of samples $(0, 0), (0, 1), (0, 0), (0, 1), (1, 0), (1, 1), (1, 0), (1, 0), (2, 0), (2, 1), (2, 1), (2, 1), (3, 1), (3, 1), (3, 1), (3, 1)$ where the first value is the input x and the second is the target y. What is the corresponding risk (probability of misclassification)? (**1p**)

2. The following exercise illustrates how well can we estimate misclassification probability given a limited holdout sample. Recall that the simplest possible loss function for a classification tasks:

$$L(y, \hat{y}) = \begin{cases} 0, & \text{if } y = \hat{y} \\ 1, & \text{if } y \neq \hat{y} \end{cases}$$

declares that the cost of an individual misclassification is always one. The corresponding risk is just the probability that we misclassify the next data sample:

$$R(g) = \Pr\left[(x, y) \leftarrow \mathcal{D} : g(x) \neq y\right] \ .$$

To estimate the misclassification probability, we must compute the empirical risk on the holdout sample $(x_1, y_1), \ldots, (x_N, y_N)$:

$$R_N(g) = \frac{1}{N} \cdot \sum_{i=1}^{N} L(g(x_i), y_i) = \frac{1}{N} \cdot \sum_{i=1}^{N} [g(x_i) \stackrel{?}{=} y_i] \ .$$

Let us now consider a particular classification rule: $y = \text{sign}(x_2 - x_1)$ and a data distribution defined as `DataSource` in the file `holdout-error.R`.

   (a) Write a function `EmpiricalRisk` that computes the empirical risk on the holdout sample. Study empirically whether the empirical risk indeed converges to some value for large enough sample sizes N. For that generate 100 datasets for $N = 1, 10, 50, 100, 500, 1000$ and compute empirical risk estimates. Draw corresponding boxplots and try to estimate to which value the empirical risk converges. (**1p**)

   (b) One way to estimate the approximation quality is to sample 1000 datasets for $N = 1, 10, 50, 100, 500, 1000$ and compute $R_N(g) - R(g)$ for each parameter value. In particular, we can find the smallest ε_N

such that $|R_N(g) - R(g)| \leq \varepsilon_N$ at least for 95% datasets of size N. The value ε_N characterises the approximation precision – only in 5% cases the approximation error is larger. Fix your best guess for $R(g)$ and tabulate the values of ε_N. Interpret results. (**1p**)

(c) It is possible to show that $N \cdot R_N(g)$ is distributed according to binomial distribution $\mathcal{B}(N, p)$, where N is the number of data samples and $p = \Pr[g(\boldsymbol{x}_i) = y_i] = R(g)$ is the sought risk. The latter allows us to find ε_N analytically instead of doing simulation experiments. Indeed, note that theoretically we would like to find ε_N such that

$$\Pr[|R_N(g) - R(g)| \geq \varepsilon_N] = 5\% \ .$$

Since the $R_N(g)$ is just a scaled binomial distribution, we can solve the inverse problem by solving the equation

$$\Pr[Y \leq N(p - \varepsilon_N)] + \Pr[Y \geq N(p + \varepsilon_N)] = 5\%$$

for a random variable $Y \sim \mathcal{B}(N, p)$ where $p = R(g)$. As binomial distribution is symmetric, we can simplify

$$\Pr[Y \leq N(p - \varepsilon_N)] = 2.5\%$$

and use the function `qbinom`, which solves equations of type

$$\Pr[Y \leq \beta] = \alpha \ ,$$

to get numerical values of ε_N. Using this information write a function that analytically computes ε_N. Compare obtained results with simulation-based estimates found in the previous subtask. (**1p**)

(d) Generalise the solution from the previous subtask so that it can work with any risk value p and draw a scatter plot that illustrates how the approximation precision ε_N depends on p and N. Look at the parameter sets $p = 0.0, 0.1, \ldots, 0.9, 1.0$ and $N = 1, 10, 50, 100, 500, 1000$ and interpret results. (**1p**)

3. Most classifiers internally compute a numeric decision-value and then convert it into a binary decision by using a pre-described threshold. By changing the threshold we can change the classifiers output towards the positive or the negative class. Receiver operating characteristic or ROC curve for short is a two-dimensional plot, which allows you to choose the right threshold value. Modify the file `roc-curve.R` to solve the following subtasks.

   (a) Consider a classification algorithm, which computes its output as $\text{sign}(x_2 - x_1 + b)$ for some fixed threshold b. Compute the ratio of true positives and the ratio of false positives for parameter values $b = -5, -4, \ldots, 5$ and draw the corresponding ROC curve. Use Wikipedia to find out how the ROC curve is drawn. Since you cannot compute the false positive and false negative ratio analytically, compute these values on the holdout dataset of size 100. (**1p**)

(b) In the previous exercise, we computed the approximation of the ROC curve on the holdout dataset of size 100. Study how precise the corresponding estimate is by repeating the same computations on 100 datasets sampled from the same source and drawing all these ROC curves on the same plot. You should see a peculiar effect. Describe it and explain why it occurs. Repeat the same experiment with datasets of size 10 and 1000. Compare the resulting plots and interpret results. Is there a minimal size of the holdout sample for which the ROC curve makes sense? (**1p**)

(c) Use the knowledge from the previous subtask to draw the ROC curves of appropriate precision for the classifiers $g_0 = \text{sign}(x_2 - x_1 + b)$ and $\text{sign}(x_1 + x_2 + b)$. Which one is better and why? (**1p**)

4. The following exercise illustrates how the training error bias occurs and how it is related to flexibility of the classifier. To solve this exercise you have to modify the file `training-error-bias.R`. Note that the data source draws x uniformly from the set $\{0, 7\}$ and y uniformly from the set $\{0, 1\}$ and thus any classifier g must have the risk $R(g) = \frac{1}{2}$.

(a) Consider a simple learning algorithm that either outputs $g_0(x) \equiv 0$ or $g_8(x) \equiv 1$ such that the training error is minimal. Implement this learning algorithm and measure its training error for training sets of size $N = 1, 5, 25, 125, 625$. For each parameter value N, generate 100 datasets and draw a corresponding boxplot. Interpret results. (**1p**)

(b) Do the following computing experiment to see how the flexibility of a classifier impacts the training error bias. Consider nine functions $g_i(x) = [x < i]$ for $i = 1, \ldots, 8$, i.e., their tabulated values are

	0	1	...	7
g_0	0	0	...	0
g_1	1	0	...	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
g_7	1	1	...	0
g_8	1	1	...	1

and four learning algorithms. The first learning algorithm chooses the the function that minimises the training error form the set $\{g_0\}$. The second learning algorithm choses the function from the set $\{g_0, g_8\}$. The third learning algorithm chooses the function form $\{g_0, g_8, g_2, g_6\}$ and the fourth learning algorithm chooses the function form the set $\{g_0, g_1, \ldots, g_8\}$. Repeat the same experiment as in the first subtask for each of these learning algorithms and draw corresponding boxplots. Interpret results. (**1p**)

Hint: You do not have to implement learning algorithms which return the best classification rule, it is sufficient if you find the function with the best training error.

(c) To study the training error bias in more detail, re-examine the experiment done for the dataset size $N = 25$. Generate 1000 training sets and tabulate the probabilities that the training error bias $\Delta = 0.1, 0.05, 0.01, 0.005$ for different learning algorithms discussed in the previous subtask. What happens if you use a dataset with a strong signal `DataSourceWithStrongSignal`. Draw corresponding graphs and interpret results. (**1p**)

5. Cross validation methods reuse the same data more than once. As a result, individual error estimates on different data splits are not independent and therefore we do not know apriori whether we can approximate average cross validation error with normal distribution or not. Nor do we know whether the naive variance estimate

$$\hat{\sigma}^2(\overline{E}) = \frac{1}{k^2} \cdot \sum_{i=1}^{k} (E_i - \overline{E})^2$$

for the average cross validation error $\overline{E} = \frac{1}{k} \cdot (E_1 + \cdots + E_k)$ is correct or not. The aim of this exercise is to test whether we can use these theoretically incorrect assumptions in practice.

The file `crossvalidation.R` contains the data source that generates data according to the following algorithm

1. Sample x uniformly from the range $[0, 1]$.
2. Compute uncorrupted response $y = x^2 - x$.
3. Generate random noise ε from normal distribution $\mathcal{N}(0, 1)$.
4. Output $(x, y + 0.1 \cdot \varepsilon)$.

(a) Search Wikipedia for the description of leave-one-out algorithm (LOO) and implement it. Generate $1,000$ data sets and estimate the mean square error with LOO. Alternatively, estimate mean square error with an independent test set that has comparable size. For instance, if training set is 100 elements then the test set should have also 100 elements, since LOO uses 100 data samples in error estimation. Study, whether the output of LOO algorithm is somehow biased compared to independent testing:

- compare mean and variance;
- draw a qq-plot for comparing the distributions of error estimates.

Hint: The complexity of this algorithm is linear in the dataset size. So choose the size of the dataset about 20-100. (**2p**)

(b) Search Wikipedia for the description of 10-fold cross validation algorithm (FCV) and implement it. Generate $1,000$ data sets and estimate the mean square error with 10-FCV. Alternatively, estimate mean square error with an independent test set that has comparable

size. For instance, if training set is 100 elements then the test set should have also 100 elements, since 10-FCV uses 100 data samples in error estimation. Study, whether the output of 10-FCV algorithm is somehow biased compared to independent testing:

- compare mean and variance;
- draw a qq-plot for comparing the distributions of error estimates.

Hint: To make results comparable with LOO experiment you should choose the same dataset size. (**1p**)

(c) Implement Monte-Carlo cross validation algorithm that makes 100 random splits with training-test data ratio $9:1$. Generate $1,000$ data sets and estimate the mean square error with MCCV. Alternatively, estimate mean square error with an independent test set that has comparable size. For instance, if training set is 100 elements then the test set should have also 10×100 elements, since MCCV uses $1,000$ data samples (counting also repetitions) in error estimation. Study, whether the output of MCCV algorithm is somehow biased compared to independent testing:

- compare mean and variance;
- draw a qq-plot for comparing the distributions of error estimates.

Hint: To make results comparable with LOO and FCV experiment you should choose the same dataset size. (**1p**)

6. Bootstrapping is commonly used to study stability properties of your algorithm. You can study how does the training error vary depending on training data. You can study how much does the training error overestimate the test error. You can study how much do coefficients of you model vary. How fragile is your learning algorithm to noise. This exercise illustrates all of these concepts. Modify the file `bootstrapping.R` to conduct the following experiments.

   (a) Implement basic bootstrapping algorithm that draws n samples randomly with replacement from n-element data set. (**1p**)

   (b) Study the stability of the following linear regression models $y\ x^2+x+1$ and $y\ x^8 + x^7 + \cdots + x + 1$. Fit these models on bootstrapped data and observe regression coefficients by drawing corresponding boxplots. Study their mean and variance. Declare that a coefficient is insignificant and set it to zero when its mean is not more than 3 standard deviations away from the zero. Interpret the results. Are both models similar? (**1p**)

   (c) To study robustness against noise, you can add additional Gaussian noise to y_i values of bootstrapped samples and later estimate how much did mean square error increase as a consequence. The latter should estimate how sensitive is your method to random noise. Study the stability of linear regression models $y\ x^2 + x + 1$ and $y\ x^8 + x^7 + \cdots + x + 1$. (**1p**)

(d) Find the description of *bootstrap .632* algorithm using Google and implement it. Compare its behaviour against standard bootstrap algorithm. (**3p**)

7. Describe a data generation algorithm such that seems plausible (not to artificial) such that cross validation methods clearly fail in error estimation. Since all cross validation estimates are unbiased, the average cross validation error over many data sets coincides with ideally conducted hold-out experiment. However, neither the variance nor the error distribution has to coincide with hold-out experiments. Cross validation can be either more conservative, i.e., overestimate the error or be overly optimistic, i.e., underestimate the error. Cross validation clearly fails, if the variance is significantly smaller (say $25-50\%$) than it is for comparable holdout experiment. Alternatively, you can search for a clearly visible overly optimistic discrepancy in qq-plot. (**10p**)