MTAT.03.227 Machine Learning
Spring 2012 / Exercise session III
Maximum score: 10p
Deadline: 6th of March 12:15 EET

1. Find how many samples are needed for estimating the test error in the holdout experiment. More precisely, tabulate necessary sample sizes for estimating accuracy in classification experiments

   Note that the empirical risk in the classification case

   $$R_n(g) = \frac{1}{n} \cdot \sum_{i=1}^{n} [g(\boldsymbol{x}_i) \stackrel{?}{=} y_i]$$

   is a scaled sum of independent zero-one variables drawn from the same distribution. As a consequence, $n \cdot R_n(g)$ is distributed according to binomial distribution $\mathcal{B}(n, p)$, where n is the number of data samples and p is the sought risk $R(g)$, i.e.,

   $$p = \Pr[g(\boldsymbol{x}_i) = y_i] = R(g) \ .$$

   Hence, the probability that estimate is out of tolerated bounds:

   $$\alpha = \Pr[|R_n(g) - R(g)| \geq \varepsilon]$$

   can be computed in closed form. More importantly, we can solve the inverse problem and find minimal error ε corresponding to $\alpha = 5\%$, i.e., find out what is the estimation error for 95% cases.

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

   $$\Pr[Y \leq n(p - \varepsilon)] + \Pr[Y \geq n(p + \varepsilon)] = \alpha$$

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

   $$\Pr[Y \leq n(p - \varepsilon)] = \frac{\alpha}{2} \ .$$

   The function `qbinom` allows you to solve equations of type

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

   (a) To illustrate this concept, perform the following experiment.
      1. Draw $u_1, \ldots, u_{20}$ uniformly from the set $[0, 1]$.
      2. Output $S_{20} = \frac{1}{20} \cdot \sum_{i=1}^{20} z_i$ for $z_i = [u_i \leq \frac{1}{2}]$.

Since u_i is drawn randomly, $z_i = 1$ with probability $\frac{1}{2}$ and $S_{20} \approx \frac{1}{2}$. Repeat this experiment $10,000$ times and tabulate on how many occasions (compute as a percentage of all runs) $\left|S_{20} - \frac{1}{2}\right| \geq \varepsilon$ where

$$\varepsilon \in \left\{ \frac{1}{20}, \ldots, \frac{10}{20} \right\} \ .$$

Find the largest value of ε for which the fraction of occurrences $\left|S_{20} - \frac{1}{2}\right| \geq \varepsilon$ is less than 5%, i.e. your approximation error is larger or equal to ε at most 5% cases.

Compare it with the theoretical estimate for ε that can be computed through the following algorithm:

1. Use `qbinom` to find β such that $\Pr[Y \leq \beta] \leq 2.5\%$ for $Y \sim \mathcal{B}(n, \frac{1}{2})$.
2. Find ε from the equation $\beta = n(\frac{1}{2} - \varepsilon)$. (1p)

(b) Generalise the theoretical approach from the part (a) and implement a function that computes estimation error $\varepsilon(n)$ of $R_n(g)$ for failure probability $\alpha = 5\%$. For that, you can use the fact $\varepsilon(n)$ is largest when $R(g) = \frac{1}{2}$. Use this function to compute minimal sample sizes for estimation errors $\varepsilon \in \{0.1, 0.05, 0.01, 0.005, 0.001\}$. What is practically achievable precision in most experiments. (1p)

Hint: If you do not want to write an algorithm that inverts $\varepsilon(n)$, then draw a scatter plot where on x-axis there is n and on y-axis is $\log_{10} \varepsilon(n)$ and find the values approximately by looking at the graph.

2. Consider a classification task where a class label Y must be predicted form binary features $X_1, \ldots, X_8$. The data samples are generated as follows:

   1. Flip a fair coin to get feature values for $X_1, \ldots, X_n$.
   2. Flip a fair coin to get a class value for Y.

   Now consider the following machine learning method that given a training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ performs following computations.

   1. It builds all 256 functions $g_i : \{0,1\}^8 \to \{0,1\}$.
   2. For each function g_i, it computes training error

   $$R_n(g_i) = \frac{1}{n} \cdot \sum_{i=1}^{n} [g_i(\boldsymbol{x}_i) \stackrel{?}{=} y_i] \ .$$

   3. It outputs a function g_* that has the minimal training error:

   $$R_n(g_*) = \min_i R_n(g_i) \ .$$

   (a) Implement this algorithm and construct the following table

Function	Training error	True risk	Optimism
g_1	$\cdots$	$\cdots$	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
g_{256}	$\cdots$	$\cdots$	$\cdots$

where the optimism is computed as $\Delta(g_i) = R(g_i) - R_n(g_i)$ for sample sizes $n \in \{100, 200, 400, 800\}$. Since the label y is chosen randomly then for any function the probability $y = f(\boldsymbol{x})$ is $\frac{1}{2}$ and thus $R(g) = \frac{1}{2}$. Alternatively, you can estimate empirically by drawing additionally $1,000$ samples from the data distribution and computing $R_{1000}(g)$ as an approximation of $R(g)$. (1p)

(b) Visualise the result by drawing scatter plots with $R_n(g_i)$ on x-axis and $\Delta(g_i)$ on the y-axis. Explain, why is the optimism $\Delta(g_*)$ so big, although for most functions g_i the optimism is in the bounds predicted by the first exercise. (1p)

Hint: State the third step of the algorithm in terms of $\Delta(g)$.

3. Any machine learning method tries to adjust model parameters according to training data $\boldsymbol{s} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n), y_n\}$. Ideally, it would pick a model g_* with a minimal risk $R(g_*)$. However, due to the noise and randomness in the training data generation its hardly the case. Instead, the method outputs a predictor $g(\boldsymbol{s})$ that depends on the training data. As a result, the performance of prediction method fluctuates. Finally, the holdout estimate on the risk $R(g(\boldsymbol{s}))$ fluctuates, as the test set itself is random.

   To illustrate these concepts, consider the following data generator:

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

   And simple linear regression of type $Y \sim X + 1$.

   ($\star$) For quadratic loss function and additive noise $y = f(x) + \varepsilon$ with zero mean $\mathbf{E}(\varepsilon) = 0$, the risk can be split further

   $$R(g) = \mathbf{E}_{x,\varepsilon}\big((f(x) + \varepsilon - g(x))^2\big) = \mathbf{E}_x\big((f(x) - g(x))^2\big) + \mathbf{D}(\varepsilon)$$

   where the first term measures the difference between f and g known as *model bias* and the second *noise variance*. Estimate the *model bias* for the class of linear models $\mathcal{L}$:

   $$B_{\mathcal{L}} = \min_{g \in \mathcal{L}} \mathbf{E}\big((f(x) - g(x))^2\big) = \min_{a,b} \int_0^1 (x^2 - ax - b)^2 dx$$

and corresponding parameter values a and b. Compute the minimal possible risk we could achieve with linear models for this particular data generator algorithm. (1p)

Hint: If you do not know how to integrate or how to minimise the resulting formula, you can use the fact that linear models are asymptotically consistent, i.e., the more samples you use the closer to the minimising function you get. So if you generate $10,000$ data points without error term and train the linear model you get rather good estimate of a and b and use another $10,000$ samples for validation.

(a) Study how the risk of the linear regression model fluctuates due to the randomness in training set generation. For that generate 100 datasets for each data set size $n \in \{10, 20, \ldots, 100\}$. For each dataset, train a linear model and estimate its risk. Draw a graph which visualises the overall trends and fluctuations, e.g. use boxplot. (1p)

Hint: The simplest way is to use again some $10,000$ element validation set for estimating risk $R(g)$.

(b) Study how does the finite test set size influence the precision of the risk estimate. Fix a reasonably good prediction model g and compute the test error $R_m(g)$ for validation sets of size $m \in \{10, 20, \ldots, 100\}$. Use the same visualisation scheme as in the (b) part. (1p)

(c) Study the bias variance dilemma in the hold-out experiment. Consider two data set sizes $n \in \{40, 104\}$. Try different ratios between training and test sets ranging from $1:7$ to $7:1$ (e.g. for 40 data set possible sizes of a training sets are $8, 16, 24, \ldots, 32$). For each dataset size and split ratio do 100 holdout experiments with freshly generated data. Visualise the results similarly to the (b) part and interpret results. (1p)

4. 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.

Fix a some sort of data generation algorithm for which you know that linear or polynomial regression should work well but there is still a reasonable amount of noise in the response y. If you do not want to define

your own data generation model, use the data generation algorithm from the previous exercise.

(a) Implement leave-one-out algorithm. 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 (there should be $1,000$ test sets—one for each experiment). 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 liner in the dataset size. So choose the size of the dataset about 20-100. (2p)

(b) Implement 10-fold cross validation algorithm. 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 1000 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 100×10 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)

5. 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. Use the following data generation algorithm in all experiments.

1. Draw $x_1, \ldots, x_{30}$ from normal distribution $\mathcal{N}(0, 1)$.
2. Compute $y = x_1 + x_2 + x_3 + x_4 + x_5 + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, 1)$.
3. Output $(\boldsymbol{x}, y)$.

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

(b) Let $\mathcal{S}$ denote the original data and $\mathcal{S}_{bs}$ denote the bootstrap sample. Fit linear regression model on the bootstrap sample $\mathcal{S}_{bs}$ and estimate its mean square error on $\mathcal{S}$. Do 200 such experiments and output 95% confidence interval for the mean square error. Compute the optimism constant, i.e., the average difference between test and training error. Compute bootstrap and bootstrap .632 estimates. (2p)

Hint: The standard bootstrap estimate on the on the test error is computed

$$E_{test}(g) = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - g(\boldsymbol{x}_i))^2 \quad .$$

For the final test error you just output the average E_{test} over 200 experiments. For bootstrap .632 estimate you have to compute test error as out of training set error and training error

$$E_{ots}(g) = \frac{1}{n} \cdot \sum_{i \in \mathcal{I}} (y_i - g(\boldsymbol{x}_i))^2$$

$$E_{tr}(g) = \frac{1}{n} \cdot \sum_{i \notin \mathcal{I}} (y_i - g(\boldsymbol{x}_i))^2$$

where

$$\mathcal{I} = \{i : (\boldsymbol{x}_i, y_i) \notin \mathcal{S}_{bs}\} \quad .$$

Now you compute averages of training and out of training errors $\overline{E}_{tr}$ and $\overline{E}_{ots}$ over 200 examples and output

$$E = 0.368 \cdot \overline{E}_{tr} + 0.632 \cdot \overline{E}_{ots} \quad .$$

(c) Repeat the same experiment as in part (b) with the difference that $\mathcal{S}_{bs}$ is replaced with independently generated data sample. Compute 95% confidence interval for mean square error and the optimism constant. Compare results (1p).

(d) Observe regression coefficients coming form bootstrapped data samples. Study their mean and variance. Declare that a coefficient is zero when its mean is not more than 3 standard deviations away from the zero. Do you discover that only $\alpha_1, \ldots, \alpha_5$ are significant? (1p)

(e) 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.Do the analogous experiment as in (b) and (c) to estimate reliability of such bootstrap estimate. (1p)

6. 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)