EXAM IN STATISTICAL MACHINE LEARNING
STATISTISK MASKININLÄRNING

DATE: June 8, 2023

RESPONSIBLE TEACHER: Jens Sjölund

NUMBER OF PROBLEMS: 5

AIDING MATERIAL: Calculator, mathematical handbooks

PRELIMINARY GRADES:	grade 3	23 points
	grade 4	33 points
	grade 5	43 points

Some general instructions and information:

- Your solutions should be given in English.
- Only write on one page of the paper.
- Write your exam code and a page number on all pages.
- Do not use a red pen.
- Use separate sheets of paper for the different problems (i.e. the numbered problems, 1–5).

With the exception of Problem 1, all your answers must be clearly motivated! A correct answer without a proper motivation will score zero points!

Good luck!

1. This problem is composed of 10 true-or-false statements. You only have to classify these as either `true` or `false`. For this problem *(only!)* no motivation is required. Each correct answer scores 1 point and each incorrect answer scores -1 point (capped at 0 for the whole problem). Answers left blank score 0 points.

   i. True. (1p)

   ii. False. (1p)

   The training error will generally be worse due to the extra penalty. However, the test error will most likely be better (if we use the right amount of regularization).

   iii. False. (1p)

   The decision boundaries of kNN will generally be non-linear regardless of k.

   iv. False. (1p)

   In bootstrapping you randomly sample from a data set *with replacement.*

   v. True. (1p)

   vi. True. (1p)

   vii. False. (1p)

   The ensemble members are conditionally independent (given the training data set) and the correlation between any pair of ensemble members is independent of the number of ensemble members.

   viii. False. (1p)

   ix. False. (1p)

   The term $\beta_0\beta_1$ is not linear in the parameters.

   x. False. (1p)

   Any mismatch between the postulated model and the true input-output relationship will result in a model bias which does not vanish as the number of data points becomes large.

2. (a) `Domestic` and `Color` are qualitative, whereas `Weight` and `Diameter` are quantitative. (1p)

   (b) The output y encodes `Domestic`. We let $y = 0$ denote `No` and $y = 1$ denote `Yes`. As the fist input $\mathbf{x}^{(1)}$ feature, we take `Weight`, and use its numerical value as it is. We do the same for `Diameter` which will be $\mathbf{x}^{(2)}$. For `Color`, we need to create an extra dummy variable, meaning we have to use both $\mathbf{x}^{(3)}$ and $\mathbf{x}^{(4)}$ for this. We use the following scheme: $\mathbf{x}^{(3)} = \mathbf{x}^{(4)} = 0$ represents `Green`, $\mathbf{x}^{(3)} = 1, \mathbf{x}^{(4)} = 0$ represents `Yellow`, and $\mathbf{x}^{(3)} = 0, \mathbf{x}^{(4)} = 1$ represents `Red`. This yields

$$\begin{aligned}
\mathbf{x}_1 &= [0.3\ 6.5\ 0\ 0]^{\mathsf{T}}, & y_1 &= 0,\\
\mathbf{x}_2 &= [0.2\ 5.2\ 1\ 0]^{\mathsf{T}}, & y_2 &= 1,\\
\mathbf{x}_3 &= [0.3\ 5.8\ 1\ 0]^{\mathsf{T}}, & y_3 &= 0,\\
\mathbf{x}_4 &= [0.3\ 6.1\ 0\ 1]^{\mathsf{T}}, & y_4 &= 1,\\
\mathbf{x}_5 &= [0.4\ 6.9\ 0\ 1]^{\mathsf{T}}, & y_5 &= 0.
\end{aligned}$$

   Note that in the text above, $\mathbf{x}^{(1)}$ is meant to be the fist dimension of the vector $\mathbf{x}$ while in the block above, $\mathbf{x}_1$ denotes the first (out of five) vectors. Other alternatives are also possible. For example, a one-hot encoding of `Color` would increase the dimension of each $\mathbf{x}$ vector by one. (2p)

   (c) We start by using the product rule for logarithms which yields

$$\log \ell(\beta) = \log \prod_{i=1}^{5} \Pr(y = y_i \,|\, \mathbf{x}_i; \beta) = \sum_{i=1}^{5} \log \Pr(y = y_i \,|\, \mathbf{x}_i; \beta).$$

   Then, we can distinguish between $y = 1$ and $y = 0$. They yield

$$\log \Pr(y = 1 \,|\, \mathbf{x}_i; \beta) = \log \frac{e^{\beta^{\mathsf{T}}\mathbf{x}}}{1 + e^{\beta^{\mathsf{T}}\mathbf{x}}} = \beta^{\mathsf{T}}\mathbf{x} - \log(1 + e^{\beta^{\mathsf{T}}\mathbf{x}}), \quad \text{and}$$
$$\log \Pr(y = 0 \,|\, \mathbf{x}_i; \beta) = \log \left(1 - \frac{e^{\beta^{\mathsf{T}}\mathbf{x}}}{1 + e^{\beta^{\mathsf{T}}\mathbf{x}}}\right) = -\log(1 + e^{\beta^{\mathsf{T}}\mathbf{x}}),$$

   respectively. Plugging them into the sum above and expanding the sum yields

$$\begin{aligned}
&-\log(1 + e^{\beta^{\mathsf{T}}\mathbf{x}_1}) + \beta^{\mathsf{T}}\mathbf{x}_2 - \log(1 + e^{\beta^{\mathsf{T}}\mathbf{x}_2}) - \log(1 + e^{\beta^{\mathsf{T}}\mathbf{x}_3})\\
&\quad + \beta^{\mathsf{T}}\mathbf{x}_4 - \log(1 + e^{\beta^{\mathsf{T}}\mathbf{x}_4}) - \log(1 + e^{\beta^{\mathsf{T}}\mathbf{x}_5}).
\end{aligned}$$

   (3p)

(d) As a model we consider $y_i \approx f(\mathbf{x}_i) = \theta_0 + \theta_1 \mathbf{x}_i^{(1)} + \theta_2 \mathbf{x}_i^{(2)}$. Here, $\mathbf{x}^{(1)}$ denotes `Domestic`, $\mathbf{x}^{(2)}$ denotes `Diameter`, and y denotes `Weight`. The objective function that we want to minimize is

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i),$$

where we use the squared loss, i.e., $\ell(f(\mathbf{x}_i), y_i) = (f(\mathbf{x}_i) - y_i)^2$. This can be equivalently phrased as

$$J(\theta) = \frac{1}{n} \|\mathbf{X}\theta - \mathbf{y}\|_2^2$$

using

$$\mathbf{X} = \begin{pmatrix} 1 & 0 & 6.5 \\ 1 & 1 & 5.2 \\ 1 & 0 & 5.8 \\ 1 & 1 & 6.1 \\ 1 & 0 & 6.9 \end{pmatrix} \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} 0.3 \\ 0.2 \\ 0.3 \\ 0.3 \\ 0.4 \end{pmatrix}.$$

We know that the gradient is given by

$$\nabla_\theta J(\theta) = \frac{2}{n} \mathbf{X}^\mathsf{T} \mathbf{X} \theta - \frac{2}{n} \mathbf{X}^\mathsf{T} \mathbf{y}.$$

Note that this is how we derived the normal equations. (We set this gradient to zero, multiply by $\frac{2}{n}$, and add $\mathbf{X}^\mathsf{T}\mathbf{y}$.) Using actual numbers as stated above yields a gradient of

$$\nabla_\theta J(\theta) = \begin{pmatrix} 14.4 \\ 5.92 \\ 88.092 \end{pmatrix}.$$

(4p)

3. (a) The misclassification loss is given by $I(y \neq \widehat{y}(x))$ and the exponential loss is given by $\exp(-yC(x))$, where $C(x) = x - 2.625$. Hence we have:

x_i	1.75	2	2.5	2.75	3
y_i	1	-1	-1	1	1
$C(x_i)$	-0.875	-0.625	-0.125	0.125	0.375
$\widehat{y}(x_i)$	-1	-1	-1	1	1
$y_iC(x_i)$	-0.875	0.625	0.125	0.125	0.375
$I(y_i \neq \widehat{y}(x_i))$	1	0	0	0	0
$\exp(-y_iC(x_i))$	2.40	0.54	0.88	0.88	0.69

The misclassification rate (average misclassification loss) is thus

$$\frac{1+0+0+0+0}{5} = 0.2$$

and the average exponential loss is

$$\frac{2.40+0.54+0.88+0.88+0.69}{5} = 1.08.$$

(2p)

(b) To achieve zero misclassification, the classifier needs to flip the sign at least two times: (i) from 1 to -1 between 1.75 and 2 and (ii) from -1 to 1 between 2.5 and 2.75.

For a LDA classifier the decision boundary will be linear of the form $bx + c = 0$, which is only fulfilled at maximum one point on the real line. Hence, it is not possible to achieve zero misclassification for a LDA classifier.

For a QDA classifier the decision boundary will be quadratic on the form $ax^2 + bx + c = 0$, which is fulfilled at maximum two points on the real line. Hence, a QDA classifier could obtain zero misclasification on this data. (2p)

(c) For the LDA classifier, we obtain $\widehat{\pi}_1 = 0.6$, $\widehat{\pi}_{-1} = 0.4$, $\widehat{\mu}_1 = 2.5$, $\widehat{\mu}_{-1} = 2.25$ and $\widehat{\sigma}^2 = 0.33$. The decision boundary is defined by x such that $p(y = 1 \,|\, x) = p(y = -1 \,|\, x)$. That is

$$\begin{aligned} & p(y = 1 \,|\, x) = p(y = -1 \,|\, x) \\ \iff & \frac{p(x \,|\, y = 1)p(y = 1)}{p(x)} = \frac{p(x \,|\, y = -1)p(y = -1)}{p(x)} \end{aligned}$$

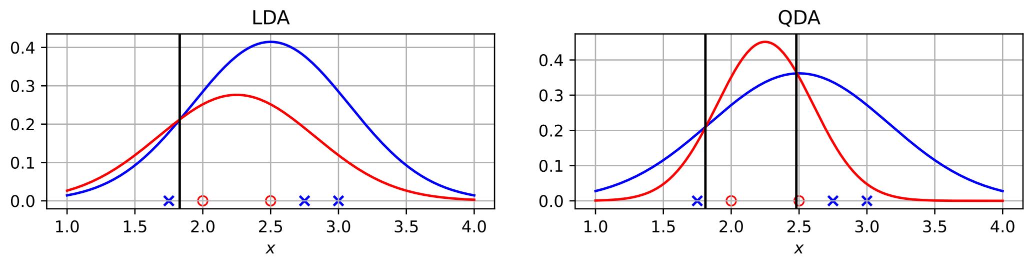

Figure 2: The data and the two classifiers.

$$
\begin{aligned}
&\iff p(x \,|\, y=1)p(y=1) = p(x \,|\, y=-1)p(y=-1)\\
&\iff \log p(x \,|\, y=1) + \log p(y=1) = \log p(x \,|\, y=-1) + \log p(y=-1)\\
&\iff \log \mathcal{N}(x; \widehat{\mu}_1, \widehat{\sigma}^2) + \log \pi_1 = \log \mathcal{N}(x; \widehat{\mu}_{-1}, \widehat{\sigma}^2) + \log \pi_{-1}\\
&\iff -\frac{1}{2}\log 2\pi\widehat{\sigma}^2 - \frac{1}{2\widehat{\sigma}^2}(x-\widehat{\mu}_1)^2 + \log \pi_1 = -\frac{1}{2}\log 2\pi\widehat{\sigma}^2 - \frac{1}{2\widehat{\sigma}^2}(x-\widehat{\mu}_{-1})^2 + \log \pi_{-1}\\
&\iff (x-\widehat{\mu}_{-1})^2 - (x-\widehat{\mu}_1)^2 = 2\widehat{\sigma}^2(\log \pi_{-1} - \log \pi_1)\\
&\iff 2x(\widehat{\mu}_1 - \widehat{\mu}_{-1}) - (\widehat{\mu}_1^2 - \widehat{\mu}_{-1}^2) = 2\widehat{\sigma}^2(\log \pi_{-1} - \log \pi_1)\\
&\iff x = \frac{2\widehat{\sigma}^2(\log \pi_{-1} - \log \pi_1) + (\widehat{\mu}_1^2 - \widehat{\mu}_{-1}^2)}{2(\widehat{\mu}_1 - \widehat{\mu}_{-1})} = 1.83.
\end{aligned}
$$

In summary, we get

$$
\widehat{y}(x) = \begin{cases} -1, & \text{if } x \leq 1.83,\\ 1, & \text{otherwise,} \end{cases}
$$

which gives a misclassification rate 3/5 for the training data. (3p)

(d) For the QDA classifier, we obtain $\widehat{\pi}_1 = 0.6$, $\widehat{\pi}_{-1} = 0.4$, $\widehat{\mu}_1 = 2.5$, $\widehat{\mu}_{-1} = 2.25$, $\widehat{\sigma}_1^2 = 0.44$ and $\widehat{\sigma}_{-1}^2 = 0.13$. Similarly to LDA, we find the decision boundary as

$$
\begin{aligned}
&p(y=1 \,|\, x) = p(y=-1 \,|\, x)\\
\iff& -\frac{1}{2}\log \widehat{\sigma}_1^2 - \frac{1}{2\widehat{\sigma}_1^2}(x-\widehat{\mu}_1)^2 + \log \pi_1 = -\frac{1}{2}\log \widehat{\sigma}_{-1}^2 - \frac{1}{2\widehat{\sigma}_{-1}^2}(x-\widehat{\mu}_{-1})^2 + \log \pi_{-1},
\end{aligned}
$$

which is a quadratic equation with solutions $x = 1.81$ and $x = 2.48$, and hence

$$
\widehat{y}(x) = \begin{cases} -1, & \text{if } 1.81 \leq x \leq 2.48,\\ 1, & \text{otherwise,} \end{cases}
$$

which gives a misclassification rate 1/5 for the training data. (3p)

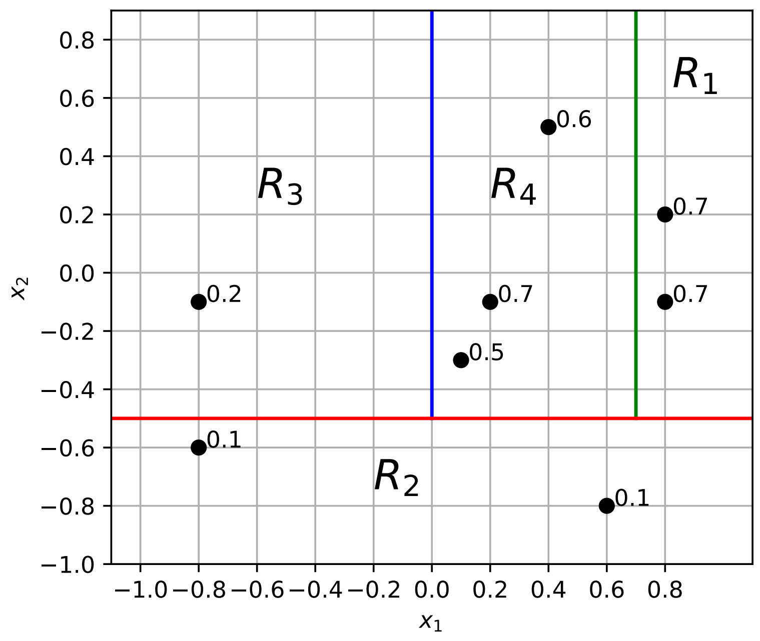

Figure 3: Partitioning of the input space for Problem 4a.

4. (a) The first split divides x_2 two half-spaces, the region $x_2 < -0.5$ correspond to leaf node R_2. The second split divides the region $x_2 \geq -0.5$ at $x_1 = 0.7$, where the region $x_1 \geq 0.7$ corresponds to node R_1. Finally, the third split divides the region $x_1 < 0.7$ and $x_2 > -0.5$ at $x_1 = 0.0$ resulting in two regions where R_3 corresponds to $x_1 \leq 0.0$ and R_4 corresponds to $x_1 > 0.0$. The partitioning of the input space is thus as shown in Figure 3. (2p)

   (b) Since $x_2^\star = 0.2 \geq -0.5$, $x_1^\star = 0.5 < 0.7$ and $x_1^\star = 0.5 > 0.0$, the test point belongs to region R_4. To compute the predicted output we also need to know which regions the training data points fall into. We do this for all eight data points and get

x_1	0.4	-0.8	0.8	0.2	0.6	0.8	0.1	-0.8
x_2	0.5	-0.1	0.2	-0.1	-0.8	-0.1	-0.3	-0.6
y	0.6	0.2	0.7	0.7	0.1	0.7	0.5	0.1
Region	R_4	R_3	R_1	R_4	R_2	R_1	R_4	R_2

   In region R_4 we have three data points. Thus, we take the mean of these three data points to compute the predicted output, which is (0.6+0.7+0.5)/3 = 0.6. (3p)

(c) All regions already have two or less data points, except for region R_4 which has three. Therefore, we need to make one additional split in that region, where one of the resulting regions will have two data points and the other region one data point.

In Figure 3 the three data points are depicted. The two possible splits are (i) to put 0.5 in one region and 0.7 and 0.6 in the other region, or (ii) to put 0.6 in one region and 0.5 and 0.7 in the other region. The MSE for these two options will be

(i) $\text{MSE} = (0.5 - 0.5)^2 + (0.7 - 0.65)^2 + (0.6 - 0.65)^2 = 2 \cdot 0.05^2$,

(ii) $\text{MSE} = (0.6 - 0.6)^2 + (0.5 - 0.6)^2 + (0.7 - 0.6)^2 = 2 \cdot 0.1^2$.

Clearly, option (i) will give a smaller MSE. This split could be realized, for example with the split $x_2 \leq -0.2$. (3p)

(d) The disadvantage of growing a decision tree too deep is overfitting. The performance on an unseen test data set and no generalization is achieved if there are too few data points in each node. (2p)

5. (a) In a dense layer each input unit is connected with *all* output units of this layer and each input-output-unit-pair has its *unique parameter*. In contrast, in a convolutional layer, each input unit is only connected with a *region of units* and all input units share the *same set of parameters*. Thus, convolutional layers have a weight-sharing property and usually have less parameters to learn. (2p)

   (b) The first layer has 784 input neurons and 256 output neurons. Thus, there are $784 \cdot 256$ individual weights to learn. In addition, we have 256 bias terms. The second layer has $256 \cdot 32$ individual weights and 32 bias terms. The third layer has $32 \cdot 10$ individual weights and 10 bias terms.

   Thus, we have $784 \cdot 256 + 256 + 256 \cdot 32 + 32 + 32 \cdot 10 + 10 = 209514$ parameters to learn. (3p)

   (c) With $n = 100\,000$ and $b = 100$, one epoch consists of 1000 iterations. Running the algorithm 10 epochs means the parameters are updated in total 10 000 times. (2p)

   (d) The computational load of each iteration is much smaller. (1p)

   (e) With a too low learning rate, the parameter updates will be very small in each iteration and the time needed for training becomes very long. With a too high learning rate, the change in the parameter in each update will be large, which might cause the training do diverge (instead of converging). (2p)