1. In the lecture, we discussed that the prediction error decomposes usually into three basic components: model variance, model bias and noise. The following exercise illustrates how these error components emerge when we consider a simple one-dimensional regression task. For clarity, we fix locations of $x$ coordinates in the training and test sets, as the latter makes it easier to visualise the results. Analogous treatment can be carried out when the training set is sampled according to some distribution.

Let training set $S$ consist of pairs $(x_i, y_i)$ where the values of $x_i$ form the grid $(-2.0, -1.8, \ldots, 1.8, 2.0)$. Let the test set $T$ consist of pairs $(x_i, y_i)$ where the values of $x_i$ form the grid $(-1.9, -1.7, \ldots, 1.7, 1.9)$. For both datasets, the values of $y_i$ are computed according to the model

$$y_i = x_i^3 - x_i^2 + \varepsilon_i$$

where $\varepsilon_i$ are drawn independently form $\mathcal{N}(0, \sigma)$. We recommend the to set $\sigma = 1$ although you could also try different values to see what is the impact of noise in linear and polynomial regression. Note that, for this particular exercise, the set of future observations is finite—a sample from the test set—and thus the risk $R(f)$ can be expressed as an average

$$R(f) = \frac{1}{|T|} \cdot \sum_{(x, y) \in T} \mathbb{E}[(f(x) - y)^2]$$

where the expectation is taken over all possible error values. In the standard conditions the risk $R(f)$ would be a two-dimensional integral to factor in also the probability of different $x$ values.

(a) A priori it is not clear that a function with the minimal risk is the actual function $f(x) = x^3 - x^2$ behind the model. Verify empirically that $f_0(x) = x^3 - x^2$ does achieve the minimal risk compared to all other functions. For that generate 100 $y_i$ values for each $x_i$ in the test set range. Let $T_*$ be the corresponding set of data points. Next find the values $y_{\text{median}}$ for each $x_i$ value that minimise the approximation of the risk function

$$R_n(f) = \frac{1}{|T_*|} \cdot \sum_{(x, y) \in T_*} (f(x) - y)^2.$$ 

Plot the graph $(x_i, y_{\text{median}})$, the graph of the function $f_0(x)$ and visualise variation of $y_i$ values by computing 90% quantiles for each $x_i$. (1p) 

\textbf{Hint:} Recall that the mean value is the best predictor if we want to minimise mean square error between predictor and measured value.
(b) Let $f$ be a some predictor and let $f_o$ be the true solution. Then
\[ E[(f(x_i) - y_i)^2] = (f(x_i) - f_o(x_i))^2 + E[(f_o(x_i) - y_i)^2] \]
holds since
\[ E[(f(x_i) - f_o(x_i))(f_o(x_i) - y_i)] = (f(x_i) - f_o(x_i))E[\varepsilon_i] = 0 . \]
As a result, we can express
\[ R(f) = \frac{1}{|T|} \cdot \sum_{(x_i, y_i) \in T} (f(x_i) - f_o(x_i))^2 + R(f_o) \]
where the first term is known as the bias of $f$
\[ \text{Bias}(f) = \frac{1}{|T|} \cdot \sum_{(x_i, y_i) \in T} (f(x_i) - f_o(x_i))^2 \]
Given that information find a linear function $f_1(x)$ that achieves minimal risk and cubic function $f_2$ that achieves minimal risk. Verify the correctness of the formula by approximating the risk. Compute the confidence intervals and verify that the value is in the range $(1/p)$.

(c) In practice, we derive predictors by observing the training set. Since the sampling procedure is noisy, each time we get different predictors. To visualise this concept. Generate 100 training sets on the grid points specified above. Use `lm` to fit linear and cubic models. Plot the first 10 linear and 10 cubic prediction functions. Next compute the predictions of all functions on the test set points and visualise the results. Draw mean values and 90% quantiles. Draw also functions $f_1$ and $f_2$ from the previous sub-task. Interpret results. $(1p)$

**Comment:** With a reasonable amount of luck, the averaged prediction is really close to the optimal functions $f_1$ and $f_2$.

(d) Note that there are two ways how to compare machine learning methods. First, we can try to decide which method works better in terms of risk for this particular training data set. It is the best we can do if we want to solve a specific task. Second, we can try to decide which method works better for this data distribution (read problem class) in general. In this case, we have to take account the variability of the predictor function due to the variability of training sets. More formally, let $f_S$ denote the predictor function computed from the training set $S$. In the previous sub-task, we generated 100 samples of such predictors both for the class of linear and cubic predictors. Formally, we need to compute average risk
\[ E_S[R(f_S)] = \frac{1}{|T|} \cdot \sum_{(x_i, y_i) \in T} E_S[(f_S(x_i) - f_o(x_i))^2] + R(f_o) . \]
Let \( \overline{f} \) be the average of all prediction functions, i.e., \( \overline{f}(x) = E_S(f(x)) \). Then
\[
E_S[(f_S(x) - f_o(x))^2] = E_S[(f_S(x) - \overline{f}(x))^2] + E_S[(\overline{f}(x) - f_o(x))^2]
\]
and thus
\[
E_S[R(f_S)] = \frac{1}{|T|} \sum_{(x_i,y_i) \in T} E_S[f_S(x_i) - \overline{f}(x_i))^2 + \text{Bias}(\overline{f}) + R(f_o) .
\]
where the first term will be called model variance.

In the previous sub-task, you computed 100 predictor functions for linear and cubic regression. Use these predictor functions to estimate bias and model variance for both model classes. Interpret results. Which regression method gives better results on average (1p).

2. The classification accuracy can be also split into three components: noise, model bias and model variance. However, the decomposition is not so easy to obtain. Let \( f_o \) be the optimal classifier that minimises the risk
\[
R(f) = E[f(x) \neq y] = \Pr [f(x) \neq y]
\]
and let \( f \) be some other classifier. Then \( f(x) \neq f_o(x) \) means actually a correct classification if \( f_o(x) \neq y \). Thus, the derivation of the bias-variance decomposition is not so straightforward. In the following, you have to derive the corresponding decomposition and measure it in practice.

As an example consider a checker board pattern in the two-dimensional square \([-1,1] \times [-1,1]\). Assume that the training set is uniformly sampled from the square and that each class label is flipped with a probability \( \delta \).

(a) Derive an expected classification error \( R(f|x) \) for each point \( x \) and find the optimal output \( f_o(x) \) that minimises \( R(f|x) \). Compute \( R(f|x) \) for the linear classifier and classifier with second-order terms for the example case. (1p)

(b) Express the expected classification error \( R(f|x) \) as a linear combination of zero-one indicator value \( [f(x) \neq f_o(x)] \) and \( R(f_o|x) \). Note that the coefficients can depend on \( x \). Find a closed form solution to our example. Interpret the resulting formula. (1p)

(c) Next note that the prediction value \( f(x) \) for a particular point depends on the training sample \( S \). Hence, we need to compute the expectation \( E_S[R(f_S|x)] \) to get the prediction risk for the particular point. Express this expectation as a linear combination of terms \( R(f_o|x) \), \( [f_o(x) \neq f_o(x)] \) and \( E_S[f_S(x) \neq f_o(x)] \) where \( f_o(x) \) is the function with minimal risk among the functions discoverable by the classifying algorithm. Estimate corresponding risk and all three components of the decomposition for the corner points of \([-1,1] \times [-1,1]\).
if we use linear and second order classifier. (2p)

**Comment:** Note that the decomposition is not so clean in the case of classification, since there is a weird interactions between misclassifiation and optimal classifier errors.

3. This exercise illustrates the main concepts of statistical learning theory. As in the previous exercise, consider uniform distribution over the square $[-1, 1] \times [-1, 1]$. Secondly, consider linear classifiers and linear classifiers over second order terms, i.e., you fit $y \sim x_1 x_2 + x_1 + x_2 + 1$.

(a) Determine VC-dimension empirically. Draw $n$ data points from the square $[-1, 1] \times [-1, 1]$ and test which label configurations are feasible. Build corresponding tables for both classifiers where $n \in \{1, \ldots, 10\}$. Find the largest $n$ for which all configurations are possible. Formally, this will be the lower bound on VC-dimension but in the context of current exercise we take it as a precise value. (1p)

(b) Recall that in order to derive bounds on the risk of a classifier the statistical learning theory considers the distribution of optimism—difference between training and test errors over the sets of same size. The former allows to reduce the infinite class of potential classification rules into a finite set of evaluations. To illustrate this concept, consider four data points $x_1, \ldots, x_4$ that are either sampled uniformly from the square or placed at corners. Next, draw randomly coefficients for the model $y \sim x_1 x_2 + x_1 + x_2 + 1$ and classify the model into eight classes depending which points are put into same class. Continue with sampling until you have three models for each class. Visualise the result by drawing a plot with $3 \times 3$ subplot where each subplot contains decision borders for each class. (2p)

**Hint:** Use `contour(zmatrix, levels=0, add=TRUE)` to draw the decision borders. If it takes too much time to get representatives for all eight classes you can seek them directly with `glm`.

(c) As a next task, let us estimate how many different classification patterns we could potentially get. Let $x_1, \ldots, x_{2n}$ be the set of training and test samples. Choose $n$ small, say 4, such that we can go through all $2^{2n}$ potential labellings of all data points. Now try all possible labellings of the data points and keep only those, for which there exist a classifier without training error. Let $G(2n)$ denote the corresponding number of valid labellings. Note that not all of them do occur in the training. To get the number of labellings that could emerge form the training, split the data set into equal parts in all possible ways and train the classifier on the training set using all possible labellings for it. Let $H(2n)$ be the actual number of different labellings. By
construction $H(2n) \leq G(2n)$ but it is important to measure the difference, since it is one factor that determines how loose are the SLT bounds which use $G(2n)$ instead of $H(2n)$. (1p)

(d) Let $y_1, \ldots, y_{2n}$ be a valid labelling and let $y_1^0, \ldots, y_{2n}^0$ be the actual labelling determined by the data distribution we want to learn and let $k$ be the number of errors, i.e., differences in the class labels. Let $k_1$ denote the number of errors in the training set. Then the optimism can be computed as

$$ \Delta = \frac{k_1 - k_2}{n} = \frac{2k_1 - k}{n}.$$ 

Tabulate the distribution of optimism under the assumption that the $2n$ element data set is randomly split into two equal parts. Note that there are only

$$ \binom{k}{k_1} \binom{2n - k}{n - k_1}$$

ways to choose split so that the training set contains $k_1$ errors and there are in total

$$ \binom{2n}{n}$$

ways to split the set into two halves. As a consequence, you can compute the largest value of $\Delta(k)$ such that

$$ \Pr[\Delta \geq \Delta(k)] \leq 5\%.$$

Report the corresponding values of $\Delta(k)$. Explain how to interpret this confidence interval for the optimism. (1p).

(e) The bound on the optimism established in the previous sub-task holds only if the classification algorithm returns the same labelling for any test-training set split. In reality, the labelling can depend on the split. However, it is rather straightforward to establish that

$$ \Pr[\Delta \geq \Delta(k)] \leq \frac{5\%}{|H(2n)|}$$

assures that the optimism is above $\Delta(k)$ under 5% cases when the function is chosen according to split. Compare this theoretical bound with the bound rising from actual observations, i.e., sample enough training and test set splits for some fixed true labelling. (1p)

4. The number of accessible labellings are is bounded by VC-dimension $d$. More precisely Sauer’s Lemma states

$$ G(n) \leq \sum_{i=0}^{d} \binom{n}{i} \leq n^d$$
where \( d \) is the VC-dimension of the classifier. Use that fact to estimate VC dimension of a neural network with 20 hidden sigmoid units. For that sample random configurations and labellings for each \( n \) until the classifier fails 20 times. Next use the Sauer’s lemma to bound the probability of individual failure and find such \( d \) value that the procedure would return \( d \) that is smaller than the actual VC-dimension with probability 5%. (3p)

5. The next exercise shows why in practice bounds from statistical learning theory do not imply the results we would like. It is well known fact that VC-dimension of linear classifiers is \( n + 1 \). The most basic SLT bounds states

\[
\Pr \left[ \exists f \in \mathcal{H}: R(f) > R_m(f) + \sqrt{\frac{\text{VCdim}(\mathcal{H}) (\ln \frac{2m}{\text{VCdim}(\mathcal{H})} + 1) + \ln \frac{1}{\delta}}{m}} \right] \leq \delta
\]

and thus training error is not far from the true risk.

Let us now consider 256 different classification problems. In all of them zero-one variables \( x_1, x_2 \) and \( x_3 \) take all possible values. In the first task the response \( y \equiv 0 \) and in the last task \( y \equiv 1 \). The remaining 254 cases correspond to the remaining labellings. For each classification problem, the training data is sampled uniformly with replacement.

(a) Choose 4 data points for training and use linear classifier to find a predictor. Tabulate the training error and risk for each classification problem. Draw the corresponding graph, where the \( x \)-axis represents the problem instance and \( y \)-axis the error. (1p)

**Hint:** You can compute the risk directly by computing the error rate on the entire data set.

(b) Since the SLT bound is probably too loose for 8 element data set consider 300 different values of \( x_1, \ldots, x_8 \) instead. Generate 200 problem sets by sampling \( y \) values randomly for the half of the cases and the remaining 100 cases so that linear classifier makes errors at most 10% cases. For that choose labels randomly, train a classifier on the entire data and then flip wrongly classified labels so that overall error is at most 10%. Next, choose training set of size 150 and compute the SLT correction by setting \( \delta = 5\% \). Draw the corresponding graph, where the \( x \)-axis represents the problem instance and \( y \)-axis the error. Put also the true risk on the graph. Verify that the number of cases where the SLT bound underestimates the true risk is below 5%. (1p)

(c) Draw the same graph as in previous sub-task but sort the cases according to the SLT bound. Do you see a clear pattern? Can you interpret the result? To make the effect more pronounced, split the classification problems into two classes: simple and hard by the following way. Draw 20 training sets and estimate the average risk. If
the median is above 40% then the classification problem is hard. If the median is below 10% then the classification problem is easy.

Choose one easy and one hard problem set. Draw 1000 training sets for both problem sets. Compute SLT bounds and the true risk on the same graph, where x-axis represents training and y-axis the error. Sort training samples according to the SLT bound. How are the approximation error cases clustered for different problem sets? Interpret the result. (2p)

6. Although the last exercise implicitly used No Free Lunch Theorem to construct hard and easy instances for a machine learning algorithm, it was still implicit. The following exercise explains NFL construction in more explicit manner. Again consider these 256 problem sets with inputs \(x_1, x_2, x_3\) as in the previous exercise. Let \(R^c_S(f)\) denote the empirical error that is computed outside training set \(S\), i.e.,

\[
R^c_S(f) = \frac{1}{|S^c|} \sum_{(x,y) \in S^c} [f(x) \neq y]
\]

where \(S^c\) is the complement of \(S\).

(a) Fix a set of input \(X\) consisting of 4 elements and choose some classification algorithm. Now for each problem set fix a training set \(S\) such that \((x, y) \in S \iff x \in X\), i.e., we consider same inputs for training. Compute the corresponding out of training set error and draw the corresponding graph. Verify that the average out of training set error is \(\frac{1}{2}\). Explain why this result holds. (1p)

(b) Consider a zero-one variables \(x_1, \ldots, x_8\). Let \(X\) be some 300 element set of inputs. Consider the set of classification problems consisting of all \(2^{300}\) possible output labellings. Let \(S\) be the training set show that for each labelling \(y\) there exist another labelling \(\overline{y}\) such that the sum of out of training errors is 1. Conclude that the average out of training error is 0.5 and that there is equal number of problem instances where one algorithm is better than the other. (1p)

(c) Let us consider a learning algorithm with fixed VC-dimension. Let the training set be so large that SLT bound is meaningful and let the set of all samples be so large that out of training set error is rather close to the risk, i.e., the number of untouched elements is much larger. These are mild restrictions, when the inputs are continuous. So that under these conditions the fraction of problem instances, where the learning algorithm achieves good performance in term of risk is negligible. What does this say about existence of universal artificial intelligence? Can you learn from data alone? (2p)