Due date: Wednesday, November 17, 2021

!!Please make sure that you properly comment your code in the jupyter notebook!!

1 Non-linear regression (10 pt)

Using support vector machines (SVM) and neural networks (NN), the following function is to be approximated over the interval $x \in [-1.0, 2.0]$

$$f(x) = x + 1.5x^2 + 0.5x^3 - 0.7x^4 + \sin(5x) + \cos(10x) \quad . \tag{1}$$

1.1 Task 1 – Data acquisition

Visualize the function $f(x)$ using at least 100 equally distributed points over $[-1.0, 2.0]$ in an array `xall` (to obtain a smooth representation of the function), e.g.:

```
xall = np.linspace(-1, 2, 100).reshape(-1, 1)
yall = f(xall).flatten()
```

Create a training dataset with 30 and a test dataset with 10 *randomly* selected (not equally distributed, so not from the array `xall`) points without any overlap between the training and test set. You can, e.g., do this by randomly picking 40 points in the interval $[-1.0, 2.0]$ and store them in an array `x` and then use `ShuffleSplit` to divide these data into a training and test set. Visualize the training and test sets together with the function $f(x)$.

1.2 Task2 – Support vector machine + linear kernel

To perform regression with SVM, the class `SVR` (support vector regression) should be used which can be imported from `sklearn.svm`, see https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html. Create a variable `svr_lin` of class `SVR` with a linear kernel (`kernel = 'linear'`) and a regularisation parameter `C = 100`. Write down the definition of the linear kernel.

Fit the linear model and predict the y-value for all x-values in `xall` as well as for all your training and test data:

```
yall_pred = svr_lin.predict(xall)
y_pred = svr_lin.predict(x)
```

Visualize the original function $f(x)$ together with the training/test data and the predicted values `yall_pred`. Report the values of w_0 (intercept) and w_1 (slope) of the linear fit. Compute and report the coefficient of determination (R^2) for the test dataset and the mean squared error (MSE) and mean absolute error (MAE) for the training and test set. How well can the function be approximated using a linear kernel?

1.3 Task 3 – Support vector machine + Gaussian kernel

Next, we use `SVR` with a Gaussian kernel (`kernel = 'rbf'`). Write down the definition of the Gaussian kernel. The selection of the adjustable parameters is not as simple as for the linear kernel. The class `GridSearchCV` imported from `sklearn.model_selection` (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) can perform an exhaustive search over a specified set of parameters. Use this class to optimize the regularisation parameter C and the parameter γ over the sets $C = \{1, 10, 100, 1000\}$ and $\gamma = \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$. As the scoring method, use `neg_mean_squared_log_error`. Report the optimized values for C and γ. What does γ control?

Fit the model with the training data and visualize the original function $f(x)$ together with the training/test data and the predicted values over the entire range. Compute and report the coefficient of determination (R^2) for the test dataset and the MSE and MAE for the training and test set. How well can the function be approximated using a Gaussian kernel?

1.4 Task 4 – Neural network

To perform regression with a NN, use the class `MLPRegressor` imported from `sklearn.neural_network` (https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html).

Use the logistic activation function (`activation`) and an lbfgs solver (`solver`) for the weight optimization. You will also need to increase the maximum number of iterations (`max_iter`) to 10 000. We want to find the number of hidden layers and nodes that represents a suitable NN architecture to fit the function. For this:

(a) fit a set of neural networks with a single hidden layer and $1 - 30$ nodes in the hidden layer (parameter `hidden_layer_sizes`). For each NN, determine the MSE for the test and training datasets and plot the MSE as a function of the number of nodes (use a logarithmic scale for the y-axis). Describe how the MSE in both the test and training dataset depends on the number of nodes. How many nodes would you choose to properly represent the function? For your optimal NN architecture (number of nodes), visualize the original function $f(x)$ together with the training/test data and the predicted values over the entire range. Compute and report the coefficient of determination (R^2) for the test dataset and the MSE and MAE for the training and test set.

(b) fit NNs with two hidden layers and $1 - 15$ nodes/hidden layer (use the same number of nodes in the two hidden layers). For each NN, determine the MSE for the test and training datasets and plot the MSE as a function of the number of nodes/hidden layer. Describe how the MSE in both the test and training dataset depends on the number of nodes. How many nodes would you choose to properly represent the function? For your optimal NN architecture, visualize the original function $f(x)$ together with the training/test data and the predicted values over the entire range. Compute and report the coefficient of determination (R^2) for the test dataset and the MSE and MAE for the training and test set.

Compare the performance of the NN with one and two hidden layers. Which NN architecture would you ultimately choose and why?

2 Kernel function vs. feature map (2 pt)

The elements of the Kernel matrix $\mathbf{K}$ are related to the feature map $\mathbf{\Phi}(\mathbf{x})$ by the kernel (or kernel function) k:

$$K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \mathbf{\Phi}(\mathbf{x}^{(i)}) \cdot \mathbf{\Phi}(\mathbf{x}^{(j)}) \tag{2}$$

For a polynomial kernel

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z} + c)^d \tag{3}$$

with $d = 2$ and $\mathbf{x}, \mathbf{z} \in \mathbb{R}^2$, derive the feature map $\mathbf{\Phi}(\mathbf{x})$.

3 Classification (8 pt)

We use a NN to solve a non-linear classification problem in 2D, $\mathbf{x} \in \mathbb{R}^2$.

3.1 Task 1 – Generating the dataset

Generate 100 data points $(\mathbf{x}, y)$, where $x_1, x_2 \in [-1, 1]$ and

$$y = \begin{cases} +1.0 & \text{if} \quad x_1 \cdot x_2 > 0 \\ -1.0 & \text{otherwise} \end{cases} . \tag{4}$$

Use `SchuffleSplit` to divide the dataset into a training (80%) and test (20%) set. Make a 2D scatter plot of the data where points with $y = 1.0$ are coloured in blue and $y = -1.0$ in red and using different markers for the training and test set.

3.2 Task 2 – Initial fitting

To perform classification with a NN, use the class `MLPClassifier` imported from `sklearn.neural_network` (https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html). Use the default setting for all parameters, except increase `max_iter` to 5000 and fit the NN with the training data. Visualise the decision boundary in a filled contour plot (with a red-blue colour map, `RdBu`) by creating a grid in x_1 and x_2 between $[-1.1, 1.1]$ and a distance between grid points of $h = 0.02$ (using e.g. `numpy.meshgrid`) and evaluating the prediction probability (`predict_proba(X)`) for the second class ($y = +1.0$) for all grid points. Together with the decision boundary, plot the training and test dataset coloured according to their y-value. Compute and report the coefficient of determination (R^2) for the test dataset and the MSE and MAE for the training and test set.

3.3 Task 3 – Activation functions

There are four possible choices for the activation function: identity, logistic, tanh, and relu. Write down the definition for each of the activation functions. Set the `solver` to lbfgs and fit a separate NN using each of the four activation functions. As in Task 2, plot the decision boundary for each case together with the training and test data. Compute and report the coefficient of determination (R^2) for the test dataset and the MSE and MAE for the training and test set. Compare the results for the four activation functions, which ones appear suitable for the current problem and why?

3.4 Task 4 – Regularisation

The parameter α determines the strength of the L2 regularisation. Use an NN with a logistic activation function and an lbfgs solver and fit six NNs with $\alpha = \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. As in Task 2, plot the decision boundary for each case together with the training and test data. Compute and report the coefficient of determination (R^2) for the test dataset and the MSE and MAE for the training and test set. Compare the results for the different values of α. How do the fits change and why?