TÉCNICO LISBOA

Homework 2 - Group 076

Aprendizagem 2021/2022

1 Pen and Paper

1) Applying the linear basis function $\phi(\mathbf{x}) = (1, \|\mathbf{x}\|_2, \|\mathbf{x}\|_2^2, \|\mathbf{x}\|_2^3)$ (with $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2}$) to each instance $\mathbf{x}^{(i)}$ ($i \in \{1, ..., 8\}$) in the training set, we get a new design matrix:

$$\mathbf{\Phi} = \begin{bmatrix} — & (\phi(\mathbf{x}^{(1)}))^T & — \\ — & (\phi(\mathbf{x}^{(2)}))^T & — \\ & \vdots & \\ — & (\phi(\mathbf{x}^{(8)}))^T & — \end{bmatrix} = \begin{bmatrix} 1.0 & 1.4142 & 2.0 & 2.8284 \\ 1.0 & 5.1962 & 27.0 & 140.2961 \\ 1.0 & 4.4721 & 20.0 & 89.4427 \\ 1.0 & 3.7417 & 14.0 & 52.3832 \\ 1.0 & 7.2801 & 53.0 & 385.8458 \\ 1.0 & 1.7321 & 3.0 & 5.1962 \\ 1.0 & 2.8284 & 8.0 & 22.6274 \\ 1.0 & 9.2195 & 85.0 & 783.6613 \end{bmatrix}$$

To learn the regression model, we must compute the weight vector $\mathbf{w}$ that minimizes the Sum of Squares error between the outputs $\mathbf{z} = \begin{bmatrix} 1 & 3 & 2 & 0 & 6 & 4 & 5 & 7 \end{bmatrix}^T$ and predictions $\hat{\mathbf{z}} = \mathbf{\Phi w}$ (i.e., $\mathbf{w} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{z}$):

$$\mathbf{\Phi}^T\mathbf{\Phi} = \begin{bmatrix} 8.0 & 35.8843 & 212.0 & 1482.2811 \\ 35.8843 & 212.0 & 1482.2811 & 11436.0 \\ 212.0 & 1482.2811 & 11436.0 & 93573.5164 \\ 1482.2811 & 11436.0 & 93573.5164 & 793976.0 \end{bmatrix}$$

$$(\mathbf{\Phi}^T\mathbf{\Phi})^{-1} = \begin{bmatrix} 8.1955 & -6.2313 & 1.3049 & -0.0793 \\ -6.2313 & 5.0781 & -1.1044 & 0.0686 \\ 1.3049 & -1.1044 & 0.2472 & -0.0157 \\ -0.0793 & 0.0686 & -0.0157 & 0.001 \end{bmatrix}$$

$$(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T = \begin{bmatrix} 1.7686 & -0.0811 & -0.6694 & -1.0069 & 1.3794 & 0.9051 & -0.785 & -0.5107 \\ -1.0644 & -0.0319 & 0.5312 & 0.904 & -1.307 & -0.3922 & 0.8501 & 0.5101 \\ 0.1933 & 0.0436 & -0.0907 & -0.1868 & 0.3232 & 0.0524 & -0.1954 & -0.1395 \\ -0.0107 & -0.0043 & 0.0044 & 0.0109 & -0.0214 & -0.0022 & 0.0123 & 0.011 \end{bmatrix}$$

$$\mathbf{w} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{z} = \begin{bmatrix} 4.5835 & -1.6872 & 0.3377 & -0.0133 \end{bmatrix}^T$$

2) Similarly to the previous question, we compute the image of each instance $\mathbf{x}^{(i)}$ ($i \in \{1, 2\}$) from the testing set by ϕ and place it in each row of the matrix $\mathbf{\Phi}$. Using the obtained weight vector $\mathbf{w}$, we have the following estimates vector:

$$\hat{\mathbf{z}} = \mathbf{\Phi w} = \begin{bmatrix} 1.0 & 2.0 & 4.0 & 8.0 \\ 1.0 & 2.4495 & 6.0 & 14.6969 \end{bmatrix} \begin{bmatrix} 4.5835 \\ -1.6872 \\ 0.3377 \\ -0.0133 \end{bmatrix} = \begin{bmatrix} 2.4536 \\ 2.2816 \end{bmatrix}$$

Computing the root mean square error, we have:

$$\text{RMSE}(\hat{\mathbf{z}}, \mathbf{z}) = \sqrt{\frac{1}{2}\sum_{i=1}^{2}(\hat{z}_i - z_i)^2} = 1.2567$$

3) To make an equal depth binarization of y_3, we compute the median of the values of y_3 of all the instances in the dataset:

$$\text{median}(\{y_3^{(i)}\}_{i=1}^{10}) = 2.5$$

and make the following transformation to y_3:

$$y_3 := \begin{cases} 0, & \text{if } y_3 < \text{median} \\ 1, & \text{otherwise} \end{cases}$$

Considering the given class targets, we have the following (training) data table:

y_1	y_2	y_3	t
1	1	0	N
1	1	1	N
0	2	1	N
1	2	1	N
2	0	1	P
1	1	0	P
2	0	0	P
0	2	1	P

- Starting entropy:

$$H(t) = -\sum_{i=0}^{1} \frac{\#\{k|t^{(k)}=i\}}{8} \log_2\left(\frac{\#\{k|t^{(k)}=i\}}{8}\right) = -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = 1\ bit$$

- Variable conditional entropies and corresponding information gains ($\text{IG}(y_i) = H(t) - H(t|y_i)$):

$$\begin{aligned} H(t|y_1) &= \sum_{i=0}^{2} \frac{\#\{k|y_1^{(k)}=i\}}{8} H(t|y_1=i) \\ &= -\sum_{i=0}^{2} \frac{\#\{k|y_1^{(k)}=i\}}{8} \sum_{j=0}^{1} \frac{\#\{k|y_1^{(k)}=i, t^{(k)}=j\}}{\#\{k|y_1^{(k)}=i\}} \log_2\left(\frac{\#\{k|y_1^{(k)}=i, t^{(k)}=j\}}{\#\{k|y_1^{(k)}=i\}}\right) \\ &= -\frac{1}{4}\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) - \frac{1}{2}\left(\frac{3}{4}\log_2\left(\frac{3}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right) - \frac{1}{4}\left(0\log_2(0) + 1\log_2(1)\right) \\ &= 0.65556\ bit \\ H(t|y_2) &= -\frac{1}{4}\left(0\log_2(0) + 1\log_2(1)\right) - \frac{3}{8}\left(\frac{2}{3}\log_2\left(\frac{2}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) - \frac{3}{8}\left(\frac{2}{3}\log_2\left(\frac{2}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) \\ &= 0.68872\ bit \\ H(t|y_3) &= -\frac{3}{8}\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right)\right) - \frac{5}{8}\left(\frac{3}{5}\log_2\left(\frac{3}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) = 0.95121\ bit \end{aligned}$$

$$\text{IG}(y_1) = 1 - 0.65556 = 0.34436\ bit \quad \text{IG}(y_2) = 0.31128\ bit \quad \text{IG}(y_3) = 0.04879\ bit$$

Since y_1 yields the biggest information gain, it will take part in the first node in the decision tree and the following division in the dataset is obtained:

$y_1 = 0$			$y_1 = 1$			$y_1 = 2$		
y_2	y_3	t	y_2	y_3	t	y_2	y_3	t
2	1	N	1	0	N	0	1	P
2	1	P	1	1	N	0	0	P
			2	1	N			
			1	0	P			

Partition ($y_1 = 2$) has no uncertainty and we cannot reduce uncertainty in partition ($y_1 = 0$) (note that there is only one value of y_2 (resp., y_3), so $H(t|y_2, y_1 = 1) = H(t|y_1 = 1)$ (resp., $H(t|y_3, y_1 = 1) = H(t|y_1 = 1)$), and there is no possible information gain). We thus proceed to divide partition ($y_1 = 1$) even further in an analogous way as before:

- Starting entropy:

$$H(t|y_1 = 1) = -\sum_{i=0}^{1} \frac{\#\{k|t^{(k)} = i \wedge y_1^{(k)} = 1\}}{\#\{k|y_1^{(k)} = 1\}} \log_2 \left(\frac{\#\{k|t^{(k)} = i \wedge y_1^{(k)} = 1\}}{\#\{k|y_1^{(k)} = 1\}} \right)$$
$$= -\left(\frac{3}{4} \log_2 \left(\frac{3}{4} \right) + \frac{1}{4} \log_2 \left(\frac{1}{4} \right) \right) = 0.81127 \; bit$$

- Variable conditional entropies and corresponding information gains (where $\text{IG}(y_i|y_1 = 1) = H(t|y_1 = 1) - H(t|y_i, y_1 = 1)$; also note that $\#\{k|y_2^{(k)} = 0 \wedge y_1^{(k)} = 1\} = 0$, so $H(t|y_2 = 0, y_1 = 1)$ is not defined and we discard that term from the first sum):

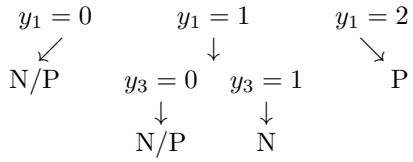
$$H(t|y_2, y_1 = 1) = \sum_{i=1}^{2} \frac{\#\{k|y_2^{(k)} = i \wedge y_1^{(k)} = 1\}}{\#\{k|y_1^{(k)} = 1)\}} H(t|y_2 = i, y_1 = 1)$$
$$= -\sum_{i=1}^{2} \frac{\#\{k|y_2^{(k)} = i \wedge y_1^{(k)} = 1)\}}{\#\{k|y_1^{(k)} = 1\}} \sum_{j=0}^{1} \frac{\#\{k|t^{(k)} = j \wedge y_2^{(k)} = i \wedge y_1^{(k)} = 1)\}}{\#\{k|y_2^{(k)} = i \wedge y_1^{(k)} = 1\}} \log_2 \left(\frac{\#\{k|t^{(k)} = j \wedge y_2^{(k)} = i \wedge y_1^{(k)} = 1)\}}{\#\{k|y_2^{(k)} = i \wedge y_1^{(k)} = 1\}} \right)$$
$$= -\frac{3}{4} \left(\frac{2}{3} \log_2 \left(\frac{2}{3} \right) + \frac{1}{3} \log_2 \left(\frac{1}{3} \right) \right) - \frac{1}{4} \left(1 \log_2(1) + 0 \log_2(0) \right) = 0.688721 \; bit$$
$$H(t|y_3, y_1 = 1) = -\frac{1}{2} \left(\frac{1}{2} \log_2 \left(\frac{1}{2} \right) + \frac{1}{2} \log_2 \left(\frac{1}{2} \right) \right) = 0.5 \; bit$$

$$\text{IG}(y_2|y_1 = 1) = 0.81127 - 0.688721 = 0.1225979 \; bit \quad \text{IG}(y_3|y_1 = 1) = 0.81127 - 0.5 = 0.31127 \; bit$$

As y_3 provides the biggest information gain, it will divide the partition ($y_1 = 1$). The resulting dataset cannot be further partitioned (($y_1 = 1, y_3 = 1$) has no uncertainty and we can't obtain information gain from dividing ($y_1 = 1, y_3 = 0$) for the same argument as the one applied to ($y_1 = 0$)):

$y_1 = 0$			$y_1 = 1$				$y_1 = 2$		
y_2	y_3	t	$y_3 = 0$		$y_3 = 1$		y_2	y_3	t
2	1	N	y_2	t	y_2	t	0	1	P
2	1	P	1	N	1	N	0	0	P
			1	P	2	N			

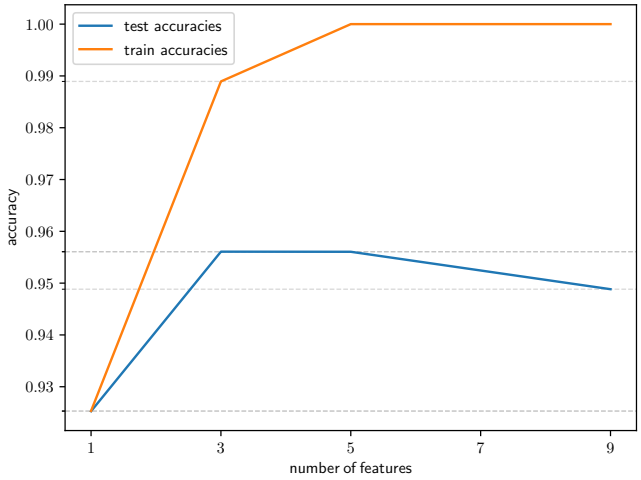
Thus, we have the desired decision tree:



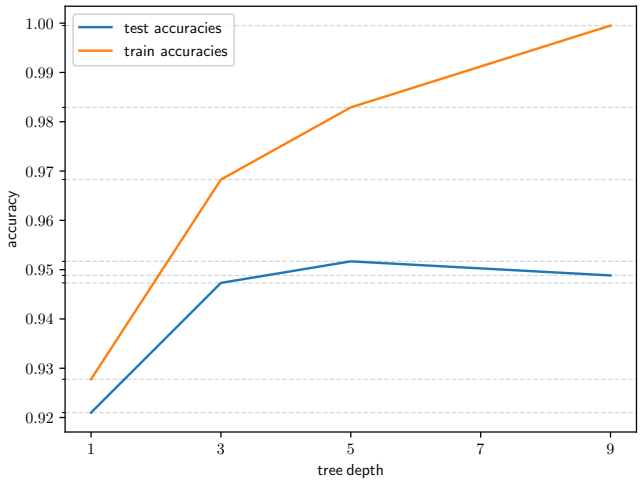
4) Considering the computed decision tree, for $\mathbf{x}^{(9)} = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^T$ we have that $\hat{\mathbf{t}}(\mathbf{x}^{(9)}) = \text{P}$, and, for $\mathbf{x}^{(10)} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}^T$, we have that $\hat{\mathbf{t}}(\mathbf{x}^{(10)}) = \text{N}$. Since $\mathbf{t}(\mathbf{x}^{(9)}) = \text{N}$ and $\mathbf{t}(\mathbf{x}^{(10)}) = \text{P}$, there are no true positives and no true negatives in the classification of these instances. Hence, accuracy on the testing data is 0 according to the following formula: Accuracy $= \frac{TP+TN}{P+N}$.

2 Programming and critical analysis

5) (i)



(ii)



6) The correlation between tendencies shown by both plots is due to the following reasons:

- depth and number of considered features are positively correlated: if there are more features then it is expected that rules consider a bigger number of said features, making the decision tree deeper; if we loosen the depth threshold, then it is more likely for the learning algorithm to consider more features (in particular, features which provide a smaller information gain start to be taken into account at some point during the expansion of the tree). Therefore, interactions between depth/no. of features and training/testing accuracy can become highly similar;
- both an increased number of considered features (according to decreasing mutual information) and tree depth make splits more granular and the rules more specific to the regularities of the training data, making it harder for the model to generalize to unseen data (i.e., there is overfitting). Hence the always-rising tendency in training accuracy and a decay in testing accuracy from a certain tree depth/no. of features onwards (5, to be exact).

7) According to the plot in question 5) (ii), the most suitable tree depth is 5, since it is the one that maximizes mean testing accuracy across train/test folds. Consequently, it is the one that is less expected to overfit, following the last argument in the previous question.

3 Appendix

```
import numpy as np
import pandas as pd
from scipy.io import arff
from sklearn import feature_selection, model_selection, tree
import matplotlib.pyplot as plt
plt.rcParams["text.usetex"] = True

def load_data(filename):
    dataset = arff.loadarff(filename)
    dataset = pd.DataFrame(dataset[0])
    dataset[dataset.columns[-1]] = dataset[dataset.columns[-1]].str.decode('utf8')
    dataset = dataset.dropna()
    return dataset

def compare_accuracy(inputs, outputs, folds, mode):
    test_accuracies = []
    train_accuracies = []
    for param in (1, 3, 5 ,9):
        if mode == "features":
            inputs_new = feature_selection.SelectKBest(feature_selection.mutual_info_classif, \
                                                       k = param).fit_transform(inputs, outputs)
            estimator = tree.DecisionTreeClassifier(criterion = "entropy", random_state = 76)
            label = "number\:of\:features"
        elif mode == "tree_depths":
            inputs_new = inputs
            estimator = tree.DecisionTreeClassifier(criterion = "entropy", \
                                                    max_depth = param, random_state = 76)
            label = "tree\:depth"
        results = model_selection.cross_validate(estimator = estimator, \
                                                 X = inputs_new, y = outputs, \
                                                 scoring = "accuracy", cv = folds, \
                                                 return_train_score = True)
        test_accuracies.append(np.mean(results["test_score"]))
        train_accuracies.append(np.mean(results["train_score"]))
    fig, ax = plt.subplots()
    ax.plot([1, 3, 5, 9], test_accuracies, label = "test accuracies")
    ax.plot([1, 3, 5, 9], train_accuracies, label = "train accuracies")
    ax.set_yticks(test_accuracies + train_accuracies, minor=True)
    ax.set_xticks([1,3,5,7,9])
    ax.yaxis.grid(True, which='minor', alpha = 0.5, linestyle = "dashed")
    ax.legend(loc = "best")
    ax.set_xlabel(label)
    ax.set_ylabel("accuracy")
    plt.savefig(f"output/accuracy_n_{mode}.pdf")

dataset = load_data("../data/breast.w.arff")
inputs = dataset.iloc[:, :-1].to_numpy()
outputs = dataset.iloc[:, [-1]].to_numpy().T.flatten()
kf = model_selection.KFold(n_splits = 10, shuffle = True, random_state = 76)
compare_accuracy(inputs, outputs, kf, "features")
compare_accuracy(inputs, outputs, kf, "tree_depths")
```