Performance evaluation, Bayes, and Naive Bayes with PYTHON

Objective: The objective of today's exercise is to estimate confidence intervals for the performance of both regression and classification models, to compare two regression and two classification models against each other, and finally to introduce the Naïve Bayes classification method.

Material: Lecture notes *"Introduction to Machine Learning and Data Mining"* as well as the files in the exercise 7 folder available from Campusnet.

Piazza discussion forum: You can get help by asking questions on Piazza: https://piazza.com/dtu.dk/fall2019/02450

Software installation: Extract the Python toolbox from DTU Inside. Start Spyder and add the toolbox directory (`<base-dir>/02450Toolbox_Python/Tools/`) to `PYTHONPATH` (Tools/PYTHONPATH manager in Spyder). Remember the purpose of the exercises is not to re-write the code from scratch but to work with the scripts provided in the directory `<base-dir>/02450Toolbox_Python/Scripts/` Representation of data in Python:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Regression	`y`	numpy.array	$N \times 1$	Dependent variable (output): For each data object, `y` contains an output value that we wish to predict.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\mathtt{y}_n \in \{0, 1, \dots, C-1\}$, where C is the total number of classes.
	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
	`C`	integer	Scalar	Number of classes.
Cross-validation				All variables mentioned above appended with `_train` or `_test` represent the corresponding variable for the training or test set.
	`⋆_train`	—	—	Training data.
	`⋆_test`	—	—	Test data.

7.1 Statistical evaluation of classifiers

In this part of the exercise, we will examine how to evaluate and compare classification methods. The tasks we will be concerned with solving is, firstly, how to determine a reasonable interval $[\theta_L, \theta_U]$ for the accuracy of a single classifier based on it's prediction on a training set and secondly, how to compare two classifiers by estimating reasonable bounds for the difference of their accuracy $\theta = \theta_A - \theta_L$.

For simplicity, we will focus on comparing two k-nearest neighbor methods on the Iris dataset. Since this dataset consist of only $N = 150$ observations, we will use leave-one-out cross validation to construct the $n = N$ model predictions.

7.1.1 In this problem, we will evaluate the performance of a single model. The models we will consider are k-nearest neighbor classification models for different values of k. Specifically, we will consider $k = 1$, $k = 20$ and $k = 80$ to obtain three models, $\mathcal{M}_A$, $\mathcal{M}_B$ and $\mathcal{M}_C$, with a reasonably different performance. As a first step, inspect and run `ex7_1_1.py`. Note it computes and saves the predictions of all three models as `yhat`. Modify the script to compute the accuracy of model $\mathcal{M}_A, \mathcal{M}_B$ and $\mathcal{M}_C$. What are your conclusions?

Script details:

- *This is simply a matter of comparing the predictions* `yhat` *with* `y_true` *and counting the fraction of times the model makes the right prediction.*
- *You should find that model $\mathcal{M}_B$ seems to perform relatively worse than $\mathcal{M}_A$ and $\mathcal{M}_C$.*

7.1.2 Inspect and run `ex7_1_2.py`. The script computes the Jeffrey interval of model $\mathcal{M}_A$ as defined in Box 11.3.1. Modify the script to compute the Jeffrey interval of all three models. What is your conclusion?

7.1.3 Try to change α to $\alpha = 0.1$ and $\alpha = 0.01$. What effect does this have on the Jeffrey interval? Explain how α influences $P(\theta \in [\theta_L, \theta_U])$.

Script details:

- *This is connected to the definition of the CDF. Under the assumption in the book leading to the definition of the Jeffrey interval it is easy to show*

$$P(\theta \in [\theta_L, \theta_U]) = 1 - P(\theta < \theta_L) - P(\theta > \theta_U)$$

and you should be able to relate the two quantities on the right-hand side to α.

7.1.4 Inspect and run `ex7_1_4.py`. The script attempts to estimate the difference in performance

$$\theta = \theta_A - \theta_B$$

between model $\mathcal{M}_A$ and $\mathcal{M}_B$ using McNemar's test (see lecture notes Box 11.3.2 for more details). Relate the quantities the script output to

Box 11.3.2 in the lecture notes. You should find that $\hat{\theta} < 0$. Does this signify $\mathcal{M}_A$ is better than $\mathcal{M}_B$? How does the result relate to your conclusions in 7.1.2?

Script details:

- *You should find the confidence interval barely doesn't contain* 0*, which is weak evidence towards* $\mathcal{M}_B$ *having a relatively higher accuracy than* $\mathcal{M}_A$*. Meanwhile, the p-value is relatively high, indicating the result is likely due to chance. All in all the result is inconclusive and we should not conclude* $\mathcal{M}_B$ *is better than* $\mathcal{M}_A$*.*

7.1.5 Modify the script to compare $\mathcal{M}_A$ against $\mathcal{M}_C$. Based on these results, are there statistically good reasons to believe $\mathcal{M}_A$ is better than $\mathcal{M}_B$? Are there statistically good reasons to believe $\mathcal{M}_A$ is better than $\mathcal{M}_C$?

Script details:

- *As we saw in 7.1.1, there is a relatively higher difference in performance between* $\mathcal{M}_A$ *and* $\mathcal{M}_C$*, which is confirmed by McNemar's test. The performance difference* θ *is estimated to be between (approximately)* 0.05 *and* 0.1*. The confidence interval is therefore well clear of* 0 *and the low p-value* ($p < 0.01$) *indicates the result is not likely to be due to chance.*

7.1.6 Modify the script `ex7_1_1.py` to replace $\mathcal{M}_B$ with a classification tree model. Use McNemar's test to compare the decision tree against a KNN classifier with $k = 1$. Comment on the result of the evaluation. Is one model better than the other?

7.2 Statistical evaluation of a regression model

In this problem, we will statistically evaluate a regression model and compare two regression models. The problem will be based on the `wine` dataset, where the goal is to predict the alcohol content (in percent) from the other features. Since we do not have many candidate regression models to choose from at this point, we will compare the linear regression model against a regression tree. Please note regression trees are starred material in the course notes, and they can be treated as a black-box method for the purpose of this exercise.

7.2.1 Inspect and run the script `ex7_2_1.py`. Notice how the model use the hold-out method to generate a training/test split and then use the t-test to obtain a confidence interval for the regression model as in Box 11.3.3. Modify the script to obtain a confidence interval for the regression tree.

7.2.2 Continuing the above exercise, notice how the model compare the linear regression model and regression tree as described in Box 11.3.4. What is the confidence interval and p-value, and what would you conclude based on the results?

7.2.3 Modify the script to use $K = 10$-fold cross validation and compare the p-value and confidence interval to that obtained with the hold-out method. What changes do you expect to occur with the confidence interval when going from hold-out to K-fold to leave-one-out cross-validation? Re-do the exercise with L_1 loss as performance measure.

7.3 Statistical evaluation in **setup II** (Optional)

The past two problems have considered **setup I**, in which we are asking the narrow question which model perform better when trained on a particular data set denoted. That is, our conclusions only hold for the particular dataset set, and if we are therefore not justified in claiming our results hold for a new dataset drawn from the same statistical source. We refer the reader to section 11.1 which elaborates on this distinction. In this section, we will therefore consider a statistical evaluation which based on the correlation heuristic compares estimates of the true generalization error as described in section 11.4.

7.3.1 In this problem, we will once more consider the `wine` dataset, and attempt to compare a linear regression model and a regression tree. Inspect and run the script `ex7_3_1.py` which implements the method described in Box 11.4.1. Carefully examine how it computes the quantities that enter into the test. What is the p-value and confidence interval?

7.3.2 The script `ex7_3_1.py` can also compute the p-value and confidence interval using the methods in `ex7_2_1.py` by using K-fold cross-validation. Compare the p-values and confidence intervals with those you just obtained above, i.e. $J = K$. How do they differ and how do you explain this difference?

7.3.3 Try to alter the script to compare two classification models (you can use the KNN example above as inspiration).

7.4 Bayes and Naïve Bayes

In this part of the exercise we will classify names as female or male names `http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/corpora/names/`, for further details see also the readme_male_female.txt file in the Data folder. We have a database with 2943 male names and 5001 female names. We will only consider names that contain at least four letters resulting in a total of 2785 male and 4866 female names. As feature for the classification we will use the first and second letter as well as the second last and last letter of the names denoted respectively x_1, x_2, x_3 and x_4. Thus the name "Richard" will have $x_1 = r$, $x_2 = i$, $x_3 = r$, $x_4 = d$. Therefore, $x_i \in \{a, b, c, d, \ldots, z\}$ such that each feature can take the value of any of the 26 letters of the alphabet (from a to z) hence the attributes used are discrete/categorical. In Python, we code each letter as a numbers between 1 and 26 where 1 corresponds to a and 26 to z.

Suppose we let $y = 0$ corresponds to Male and $y = 1$ correspond to Female. In this case, according to Bayes rule, we get (see section 13.1.1):

$$p(y = c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = c)p(y = c)}{p(\boldsymbol{x}|y = 0)p(y = 0) + p(\boldsymbol{x}|y = 1)p(y = 1)}. \tag{1}$$

We will classify a name as a female name if $P(y = 1|x_1, x_2, x_3, x_4) > P(y = 0|x_1, x_2, x_3, x_4)$ and as a male name otherwise. We will split the data into a training and a test set and build our classifier using the training data, i.e.

$$\begin{aligned} P(x_1, x_2, x_3, x_4|y = 0) &= \tfrac{N^{\text{Female}}_{\text{train}}(x_1,x_2,x_3,x_4)}{N^{\text{Female}}_{\text{train}}} \\ P(x_1, x_2, x_3, x_4|y = 1) &= \tfrac{N^{\text{Male}}_{\text{train}}(x_1,x_2,x_3,x_4)}{N^{\text{Male}}_{\text{train}}}, \end{aligned}$$

where $N^{\text{Female}}_{\text{train}}(x_1, x_2, x_3, x_4)$ and $N^{\text{Male}}_{\text{train}}(x_1, x_2, x_3, x_4)$ denotes respectively the number of female and male names with first letter x_1, second letter x_2, second last letter x_3 and last letter x_4 in the training data. The prior terms $p(y)$ are estimated as described in section 13.1.1.

7.4.1 In order to classify names as male or female according to the above we need to count the number of times given letter combinations occurred in the male and female names in the training data, i.e. how many times each of the letter combinations $(x_1, x_2, x_3, x_4) = (a, a, a, a), (x_1, x_2, x_3, x_4) = (a, a, a, b), \ldots, (x_1, x_2, x_3, x_4) = (z, z, z, z)$ occurred. How many different letter combinations do we have to evaluate?

7.4.2 How well do you think we can identify the probabilities $P(x_1, x_2, x_3, x_4|\text{Male})$ and $P(x_1, x_2, x_3, x_4|\text{Female})$ from the data at hand? Consider how likely it is that a given dataset will contain enough training samples to allow us to accurately estimate these probabilities.

Rather than finding the full joint distribution $P(\boldsymbol{X}|Y)$ we will use a Naïve Bayes classifier instead that assumes that each attribute (letter in a name) is independent, and thus we arrive at the expression:

$$p(y|x_1, x_2, \ldots, x_M) = \frac{p(x_1|y) \times \cdots \times p(x_M|y)p(y)}{\sum_{c=1}^{C} p(x_1|y = c) \times \cdots \times p(x_M|y = c)p(y = c)}.$$

From the training data we can obtain

$$\begin{aligned} P(X_i = x_t|y = 0) &= \frac{N^{\text{Male}}_{\text{train}}(X_i = x_t)}{N^{\text{Male}}_{\text{train}}} \\ P(X_i = x_t|y = 1) &= \frac{N^{\text{Female}}_{\text{train}}(X_i = x_t)}{N^{\text{Female}}_{\text{train}}} \end{aligned}$$

where $N^{\text{Male}}(X_i = x_t)$ is the number of times the letter x_t occurred in the i^{th} considered letter of the name. Using Naïve Bayes we only need to estimate the probability of observing each of the 26 letters for each of the four considered position in the spelling of the name separately. From more than one thousand male and female names we can quite accurately estimate these probabilities.

7.4.3 Inspect and run the script `ex7_4_3.py`. The script loads the male and female names, filters the text data and extracts the first two and last two letters of each name as feature.

7.4.4 Inspect and run the script `ex7_4_4.py`. The script is used to classify the names using a naïve Bayes classifier. Use a uniform prior, assuming male and female names are equally likely (i.e. $P(y = 0) = P(y = 1) = 0.5$). Compute the classification error using 10-fold crossvalidation.

Script details:

- *Type* `help(sklearn.naive_bayes.MultinomialNB)` *to learn how to use naive bayes framework in Python.*
- *To specify that the data is categorical we use the* `sklearn.preprocessing.OneHotEncoder`*. Otherwise* `MultinomialNB` *assumes that the input numbers are e.g. discrete counts of words or tokens. Without the encoding, the value 26 for a given element would signify 26 counts of a given tokens, whereas the encoding ensures that the value 26 is interpreted a level of a categorical variable corresponding to the letter "z".*
- *The multinomial distribution is used for describing discrete/categorical features, as we do with this name dataset. However, if we need to model continous data, we need to use a different distribution, such as a Gaussian, see* `GaussianNB` *or* ***https://scikit-learn.org/stable/modules/naive_bayes.html*** *for more information. An alternative to Gaussian Naïve Bayes is to discretize the input data (e.g. by binarizing based on median).*
- *As usual, use* `sklearn.cross_validation` *module to set up the crossvalidation partitions.*

Try classifying names based on only two of the letters in the name. You can do this by writing, e.g., `X=X[:,0:2]` to choose the first two letters. Are the first or last letters more useful for classifying names? Can you explain why?

7.4.5 Change the prior to `empirical` such that $P(y=1) = \frac{N_{\text{train}}^{\text{Female}}}{N_{\text{train}}^{\text{Female}}+N_{\text{train}}^{\text{Male}}}$, $P(y=0) = \frac{N_{\text{train}}^{\text{Male}}}{N_{\text{train}}^{\text{Female}}+N_{\text{train}}^{\text{Male}}} = 1 - P(\text{Female})$. Does this setting improve the classification and if so why?

7.5 Tasks for the report

Statistical evaluation will be important for the report, but we still need to introduce regularization and artificial neural networks to state the tasks. See report description for more information.

References