05_accuracy_standalone

December 22, 2019

1 Accuracy measurements in machine learning

The most common measurement in machine learning is 'accuracy, that is the proportion of cases where the classification is correct. This may at times not be the best measurement. For example, imagine you wish to detect who is carrying the back death among people coming through an airport, by measuring a range of features (such as body temperature, or the presence of a particular skin rash pattern). Now imagine that one in a thousand people are carriers of black death, who if allowed through will start off a major epidemic. We may then optimise a machine learning algorithm and it reports 99.9% accuracy. Impressive! Or is it? The algorithm may have simply classed everybody as not having black death. It will be right 999 times out of 1,000. But at the same time it will miss every single case of black death that you intended to detect. This is an extreme example of where we need more sophisticated measurements of accuracy. In this particular case, we probably want to measure sensitivity and specificity. Sensitivity measures the proportion of positive cases (black death) correctly classified, while specificity measures the proportion of negative cases (no black death) correctly classified. We will later see how we might change machine learning thresholds to adjust the balance between sensitivity and specificity.

Sensitivity and specificity are not the only measures (though they are common in health diagnosis problems). Below are a range of measures that our function will calculated (given the observed and predicted class values of a range of cases). In addition to sensitivity and specificity, common measures used in machine learning are precision, recall and f1 (a combination of precision and recall).

The full list of measures that our function will return (in the form of a dictionary) is:

```
 1) observed positive rate: proportion of observed cases that are +ve
 2) Predicted positive rate: proportion of predicted cases that are +ve
 3) observed negative rate: proportion of observed cases that are -ve
 4) predicted negative rate: proportion of predicted cases that are -ve
 5) accuracy: proportion of predicted results that are correct
 6) precision: proportion of predicted +ve that are correct
 7) recall: proportion of true +ve correctly identified
 8) f1: harmonic mean of precision and recall
 9) sensitivity: Same as recall
10) specificity: Proportion of true -ve identified:
11) positive likelihood: increased probability of true +ve if test +ve
12) negative likelihood: reduced probability of true +ve if test -ve
13) false positive rate: proportion of false +ves in true -ve patients
14) false negative rate: proportion of false -ves in true +ve patients
```

```
15) positive predictive value: chance of true +ve if test +ve
16) negative predictive value: chance of true -ve if test -ve
```

Please note that these measures are for a 'binomial' classification problem. That is where there are two alternatives for each case (e.g. survived or not-survived on the Titanic). Where there are more than two possible classes, a confusion matrix is most commonly used (e.g. see https://pythonhealthcare.org/2018/04/21/77-machine-learning-visualising-accuracy-and-error-in-a-classification-model-with-a-confusion-matrix/)

1.1 Define function

[1]:
```python
import numpy as np

def calculate_accuracy(observed, predicted):

    """
    Calculates a range of acuuracy scores from observed and predicted classes.

    Takes two list or NumPy arrays (observed class values, and predicted class
    values), and returns a dictionary of results.

     1) observed positive rate: proportion of observed cases that are +ve
     2) Predicted positive rate: proportion of predicted cases that are +ve
     3) observed negative rate: proportion of observed cases that are -ve
     4) Predicted neagtive rate: proportion of predicted cases that are -ve
     5) accuracy: proportion of predicted results that are correct
     6) precision: proportion of predicted +ve that are correct
     7) recall: proportion of true +ve correctly identified
     8) f1: harmonic mean of precision and recall
     9) sensitivity: Same as recall
    10) specificity: Proportion of true -ve identified:
    11) positive likelihood: increased probability of true +ve if test +ve
    12) negative likelihood: reduced probability of true +ve if test -ve
    13) false positive rate: proportion of false +ves in true -ve patients
    14) false negative rate: proportion of false -ves in true +ve patients
    15) positive predictive value: chance of true +ve if test +ve
    16) negative predictive value: chance of true -ve if test -ve
    """

    # Converts list to NumPy arrays
    if type(observed) == list:
        observed = np.array(observed)
    if type(predicted) == list:
        predicted = np.array(predicted)

    # Calculate accuracy scores
    observed_positives = observed == 1
```

```
    observed_negatives = observed == 0
    predicted_positives = predicted == 1
    predicted_negatives = predicted == 0

    true_positives = (predicted_positives == 1) & (observed_positives == 1)

    false_positives = (predicted_positives == 1) & (observed_positives == 0)

    true_negatives = (predicted_negatives == 1) & (observed_negatives == 1)

    accuracy = np.mean(predicted == observed)

    precision = (np.sum(true_positives) /
                 (np.sum(true_positives) + np.sum(false_positives)))

    recall = np.sum(true_positives) / np.sum(observed_positives)

    sensitivity = recall

    f1 = 2 * ((precision * recall) / (precision + recall))

    specificity = np.sum(true_negatives) / np.sum(observed_negatives)

    positive_likelihood = sensitivity / (1 - specificity)

    negative_likelihood = (1 - sensitivity) / specificity

    false_postive_rate = 1 - specificity

    false_negative_rate = 1 - sensitivity

    positive_predictive_value = (np.sum(true_positives) /
                                 np.sum(observed_positives))

    negative_predicitive_value = (np.sum(true_negatives) /
                                  np.sum(observed_positives))

    # Create dictionary for results, and add results
    results = dict()

    results['observed_positive_rate'] = np.mean(observed_positives)
    results['observed_negative_rate'] = np.mean(observed_negatives)
    results['predicted_positive_rate'] = np.mean(predicted_positives)
    results['predicted_negative_rate'] = np.mean(predicted_negatives)
    results['accuracy'] = accuracy
    results['precision'] = precision
    results['recall'] = recall
```

```
    results['f1'] = f1
    results['sensivity'] = sensitivity
    results['specificity'] = specificity
    results['positive_likelihood'] = positive_likelihood
    results['negative_likelihood'] = negative_likelihood
    results['false_postive_rate'] = false_postive_rate
    results['false_negative_rate'] = false_negative_rate
    results['positive_predictive_value'] = positive_predictive_value
    results['negative_predicitive_value'] = negative_predicitive_value

    return results
```

1.2 Demonstrate method

[2]:
```
# Create two lists to test function
observed =  [0, 0, 1, 0, 1, 0, 1, 0, 1, 0]
predicted = [0, 0, 1, 0, 1, 0, 1, 0, 0, 1]

# Call calculate_accuracy function
accuracy = calculate_accuracy(observed, predicted)

# Print results up to three decimal places
for key, value in accuracy.items():
    print (key, "{0:0.3}".format(value))
```

```
observed_positive_rate 0.4
observed_negative_rate 0.6
predicted_positive_rate 0.4
predicted_negative_rate 0.6
accuracy 0.8
precision 0.75
recall 0.75
f1 0.75
sensivity 0.75
specificity 0.833
positive_likelihood 4.5
negative_likelihood 0.3
false_postive_rate 0.167
false_negative_rate 0.25
positive_predictive_value 0.75
negative_predicitive_value 1.25
```