CS441 Artificial Intelligence (Fall 2019)

Writeup--HW 2: Heart Anomalies

Weiwei Chen

weiwchen@pdx.edu

Task:

The task for the program is to build machine learners to diagnose heart anomalies from radiology data. We will run the learners on some real heart anomaly data, sliced and diced in various ways and get the models which will be used to run on the test data and get the predictions.

Learners used:

In this program, basically we just need to get the model on the train data, and then run the model on the test data.

The first learner I used is **Naïve Bayesian**. The learning approach involves counting feature occurrences in the learning phase, and then using these counts in the classification phase. We can get the Naïve Bayes classifier as follows:

$$Class(x) = \text{argmax } P(class)\prod P(Xi|class)$$

My steps are as follows:

Step 1: Create a probabilistic model, that is, calculate P(class) and P(Xi|class) for each class. The two classes are 1(normal) and 0(abnormal)

a. For P(class), use the count of normal/abnormal instances in the training data divided by the total number of instances.
b. For P(Xi|class), split the training data into two different kinds of data, one for all the normal instances, and another for the abnormal instance.
c. In the normal data, count the occurrences of normal signal of each feature and calculate the probabilities. Create one list to hold the probabilities of normal signal probabilities of each feature in the whole normal instances, create another list to hold the probabilities of abnormal signal probabilities of each feature in the whole normal instances
d. For the abnormal data, do the same thing as what we did in Step 1(c).

Step 2: According to the instances in the test data, calculate P(class) ∏P(Xi|class) for each class, make sure to choose the right list(from Step 1) to use based on Xi is 1 or 0. To avoid underflow, we use logs. To avoid the log of 0, we add an arbitrary 0.5 to the numerator and denominator counts of each probability

Step 3: Compare P(class) for each class, and the higher one determines the property of The predicted class. Store the predicted classes in a list

Step 4: Compare the predicted classes with the actual classes in the test data, we can Get the accuracy, true negative rate and true positive rate

The second learner I used is **Gaussian Naïve Bayes**. When Naïve Bayes is extended to real-valued attributes, we can use Gaussian Naïve Bayes. All we need to do is to calculate the mean and the standard deviation of the features for each class. We can use the Gaussian Naïve Bayes algorithm as follows:

$$P(x_i \mid c_j) = N(x_i; \mu_{i,c_j}, \sigma_{i,c_j})$$

where

$$N(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

My steps are as follows;

Step 1: Create probabilistic model. Compute the prior probability for each class, normal and abnormal.

Step 2: For each feature, compute the mean and the standard deviation in the training set of the values given each class. To avoid any standard deviation equal to 0, we add a small value 0.0001 to each standard deviation we compute

Step 3: Use the Gaussian Naïve Bayes algorithm we stated above to classify the Instances in the test set by using the means and standard deviations we Computed in Step 1. Since a product of the probabilities will be very small, we will use the log of the product

$$class_{NB}(\mathbf{x}) = \underset{class}{\operatorname{argmax}} \left[P(class) \prod_i P(x_i \mid class) \right]$$

Since

$$\underset{z}{\operatorname{argmax}}\, f(z) = \underset{z}{\operatorname{argmax}} \log f(z)$$

we have:

$$class_{NB}(\mathbf{x}) = \underset{class}{\operatorname{argmax}} \log \left[P(class) \prod_i P(x_i \mid class) \right]$$
$$= \underset{class}{\operatorname{argmax}} \left[\log P(class) + \log P(x_1 \mid class) + \ldots + \log P(x_n \mid class) \right]$$

to get predicted classes on the test data

Step 4: Compare the predicted classes with the actual classes in the test data, we can get the accuracy, true negative rate and true positive rate

Results from The Two Learners:

Note: The first number is the fraction of instances that are classified correctly("accuracy"). The second number is the fraction of class-0 (abnormal) instances that are classified correctly ("true negative rate"). The third number is the fraction of class-1 (normal) instances that are classified correctly ("true positive rate").

Naive Bayesian Learner:

orig 142/187(0.76) 10/15(0.67) 132/172(0.77)

resplit 78/90(0.87) 17/19(0.89) 61/71(0.86)

itg 145/187(0.78) 15/15(1.00) 130/172(0.76)

resplit-itg 63/90(0.70) 17/19(0.89) 46/71(0.65)

Gaussian Naive Bayes Learner:

orig 112/187(0.60) 12/15(0.80) 100/172(0.58)

resplit 54/90(0.60) 19/19(1.00) 35/71(0.49)

itg 132/187(0.71) 14/15(0.93) 118/172(0.69)

resplit-itg 61/90(0.68) 16/19(0.84) 45/71(0.63)

Thoughts:

In this program, we know that the accuracy on the abnormal instances is more important than that on the normal instances. Because we are facing the patients' heart anomaly data, trying to find the correct abnormal instances and remind the patients about their condition in order to avoid delayed treatment. The accuracy on the normal instances is not as import as that on the abnormal ones. For example, if we predict that a patient's condition is abnormal but actually the condition is normal, we remind the patient, this situation doesn't hurt much on the patient and will not delay any treatment since the real situation is normal.

For Naïve Bayes learner, the results from it indicate that itg dataset gave the best accuracy on the abnormal instances, which is 15/15(1.00). resplit and resplit-itg datasets resulted in the same accuracy on the abnormal instances, 17/19(0.89), which is a little lower than that from itg dataset. resplit gave the highest accuracy based on the total instances, which is 78/90(0.87). itg dataset gave the accuracy on all the instances 145/187(0.78), which ranked the second place. Based on those results we discussed above, I think itg dataset gave the best results since it gave the highest accuracy on the abnormal instances and its accuracy on the all instances is also high.

For Gaussian Bayes learner, resplit dataset gave a highest accuracy on the abnormal instances, which is 19/19(1.00) but gave the lowest accuracy on all the instances, itg dataset provided a little lower accuracy on the abnormal ones, which is 14/15(0.93). itg dataset gave a highest accuracy on all the instances which is 132/187(0.71), resplit-itg dataset gave 61/90(0.68) on all the instance which is a little lower than that from itg. But the accuracy on the abnormal instance from resplit-itg(16/19(0.84)) is much lower than the highest one. Combine all the accuracies, I think itg dataset gave the best results since it gave the highest accuracy on all the instance and the accuracy on the abnormal instances are not low which is 0.93.

Comparing the two learners from the accuracy results, I can say Naïve Bayes learner works better than Gaussian Naïve Bayes learner on these datasets. Because Naïve Bayes learner gave better accuracy on the all instances, the highest accuracy on the abnormal instances from both learners are the same, and the Naïve Bayes learner gave much better accuracy on the normal instances too. Why Naïve Bayes is better on those datasets? I think the reason is all the data given are the binary ones, Gaussian Naïve Bayer is better used on real-valued data.

Setup and Build:

1) Install Python
2) $ git clone https://github.com/waiwaixiaochen/heart-anomaly.git
3) Run the program: $ python heart_anomaly.py