Amin Fadaee

Machine Learning HW3

Python 3.5, Libraries: Numpy, Pandas, Random, Matplotlib, Math, Scipy Documentation; HTML, Bootstrap 4, jqMath, Graphs: Xrend

Problem 1

a. By inspecting the plot, it can be seen that based on the distributions that we would derive for each of the classes, the probability of $p(0 \mid c)$ is greater than $p(1 \mid c)$

b. The instance in the second plot is clearly an outlier whereby it belongs to a 1 as it. however would have higher probability to belong to 0 $p(0 \mid c)$ is $> p(p \mid c)$. This is due to the fact that the variance for the class .c is really small comparing to 0 and the probability of the specified deviation considering this little amount of variance is highly unlikely.

c. The decision boundary for this data is going to be a circle which inscribes the 0 points belonging to it. therefore all instances in the circle is going to be classified as 0 and all the ones out of the circle as a. So the label is 0.

Problem 2

Assuming we have m classes and each instance has n features if we were to classify a new instance based on a generative model .let's say a gaussian bayesian classifier) we would need to compute the covariance matrix of n by n ,C and the mean vector μ of length n for each of the m classes.

1. If all the features are correlated, then C would have no non zero elements and therefore finding C and μ would require finding the values of $n^2 + n$ parameters for each class. Like the plots should be obtained as well as priors), so in total we would need to compute the values of $mn^2 + m + m$.
2. If the features are independent then C would become a diagonal matrix and therefore we need to find 2n parameters for each class and m priors which add up to $2mn + m$ parameters.
3. Gaussian bayes classifiers are not constrained by the values of each attribute, so if instances have real values and features are independent, there we still need to derive $2mn + m$ parameters.

Problem 3

Here is the joint probability distribution of the network:

$$P(A|PC) : A|PA1 : A|PA2 : A|PA3 \mid B, C|PC0 : C0$$

Now if we like to obtain $P(B \mid D = T)$ we can do this using the conditional probability:

$$P(B \mid D = T) = \frac{P(A, B, C, D, T)}{P(D = T)} = \frac{P(A|PC) : A|PA1 : A|PA2 : A|PA3 \mid B, C|PC0 : D}{P(D = T, B, C)}$$

However, dealing with the posterior would be much easier:

$$P(B \mid D = T) = \frac{P(D = T : B|PB)}{P(D = T)}$$

Now we are going to compute the probability of each of the components in the above formula. first P(B):

$$P(B = T) = P(A = T|PB = T : A = F|PB = T) : P(A = T|PB = F : A = F|PB = F)$$
$$P(B = T) = 0.2 : 0.8 : 0.1 : 0.9$$
$$P(B = T) = 0.18$$

Now let's do $P(D = T \mid B)$:

$$P(D = T \mid B) = \frac{P(B, D)}{P(B)}$$

$$P(B, D = T) = \sum_{A} \sum_{C} P(A, B, C, D = T) = \sum_{A} \sum_{C} P(A|PB) : A|PC : A|PD = T \mid B, C)$$
$$= P(A = T|PB) : A = T|PC = T : A = T|PD = T \mid B = T) : P(A = T|PB) : A = T|PC = T : A = T|PD = T \mid B, C = F)$$
$$+ P(A = F|PB) : A = F|PC = T : A = F|PD = T \mid B, C = T) : P(A = F|PB) : A = F|PC = F : A = F|PD = T \mid B, C = F)$$
$$P(D = T, B = T) = 0.2 : 0.8 : 0.1 : 0.2 + 0.2 : 0.8 : 0.7 : 0.7 : 0.8 : 0.1 : 0.2 : 0.1 + 0.8 : 0.1 : 0.9 : 0.1 : 0.7 : 0.1$$
$$P(D = T, B = F) = 0.2 : 0.8 : 0.3 : 0.8 + 0.2 : 0.8 : 0.7 : 0.7 : 0.95 : 0.8 : 0.79 : 0.25 : 0.67 : 0.8 : 0.79 : 0.75 : 0.95$$
$$P(D = T, B = T) = 0.05577$$
$$P(D = T, B = F) = 0.665276$$

$$P(D = T \mid B = T) = \frac{P(B = T, D = T)}{P(B = T)} = \frac{0.05577}{0.247}$$
$$P(D = T \mid B = T) = 0.2342$$
$$P(D = T \mid B = F) = \frac{P(B = F, D = T)}{P(B = F)} = \frac{0.665276}{0.778}$$
$$\boldsymbol{P(B = T \mid D = T) = 0.242}$$
$$\boldsymbol{P(B = T \mid D = T) = 0.8776}$$

And finally $P(D = T)$:

$$P(D = T) = \sum_{B} P(B, D = T) = \sum_{B} P(B|PD = T \mid B) = P(B = T|PD = T|B = F)P(D = T \mid B = F)$$
$$= 0.242 : 0.242 + 0.778 : 0.8776$$
$$\boldsymbol{P(D = T) = 0.7278}$$

And now for the ultimate result:

$$P(B \mid D = T) = \frac{P(D = T : B|PB)}{P(D = T)}$$
$$P(B = T \mid D = T) = \frac{0.242 : 0.242}{0.7278}$$
$$P(B = F \mid D = T) = \frac{0.8776 : 0.778}{0.7278}$$
$$\boldsymbol{P(B = T \mid D = T) = 0.0081}$$
$$\boldsymbol{P(B = F \mid D = T) = 0.9191}$$

Problem 4

In finding the probability that i belongs to class c_i ($p(x_i \mid c)$ there are two general practices. The generative and discriminative methods. In Dicriminative methods the values of $p(x_i \mid c)$ are obtained directly from the data (logistic regression is an example of these models), whereas in the generative models they are derived with an additional step of generating the conditional probabilities ($p(c \mid x)$). Bayesian classifiers are examples of generative models.

Problem 5

Logistic Regression directly estimates the parameters of $P(Y \mid X)$ (discriminative), whereas Naive Bayes directly estimates parameters for $P(X)$ and $P(Y \mid X)$ (generative). Asymptotically (as the number of training examples grows toward infinity) the GNB and Logistic Regression converge toward identical classifiers, however, GNB and Logistic Regression converge toward their asymptotic accuracies at different rates and logistic Regression parameter estimates converge more slowly.

Logistic Regression

The following functions has been implement to build the logistic regression classifier for this problem:

- `predict_probability(x, w)`: This function takes in as argument the features matrix M and the weights w and computes the probability $P(y = 1 \mid x, w)$ for x being a row in M. It uses the sigmoid function $\frac{1}{1+e^{-w^T x}}$
- `feature_derivative(h, labels, w, l2_penalty)`: This function computes the derivative of log-likelihood and takes as input H, The feature matrix (also), list of labels of M rows, w, the list of coefficients and $l2\_penalty$, the parameter λ for L2 regularization. This function implements the following formula:

$$\sum_{i=1}^{n} x_{ij}(\mathbb{1}[y_i = +1] - P(y = +1 \mid x_i, w)) - 2\lambda w_j$$

  where $\mathbb{1}$ is the indicator function ($\mathbb{1}$ if $y_i = +1$, 0 if $y_i = -1$) and the $-2\lambda w_j$ will be omitted for w_0 and $\lambda = 0$ for no regularization.
- `compute_log_likelihood(h, labels, w)`: This function computes the log likelihood which is the quality measure of logistic regression and can be used to check the proper functioning of our algorithm and printing its output throughout the algorithm runtime in order to check whether our gradient ascent algorithm is seizing the log likelihood in each iteration.
- `logistic_regression(h, labels, w, step_size, max_iter, l2_penalty=0)`: This function implements the logistic regression model training via gradient ascent by updating w using the following formula:

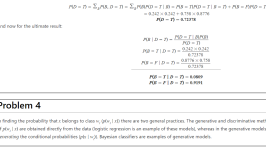
$$w_j^{t+1} = w_j^t + step\_size * \frac{\partial ll}{\partial w_j} - 2\lambda w_j^t$$

- `false_positive_rate(labels, predictions)`: Computes the false positive rate of the prediction
- `true_positive_rate(labels, predictions)`: Computes the true positive rate of the prediction

After training the model on the provided data with no regularization ($\lambda = 0$) in the training and test accuracy was as the following:

- Training Accuracy: 100%
- Test Accuracy: 84%

And here the ROC curve associated with this model (This curve is obtained using values between 0 to 1 with steps of 0.01 in size):

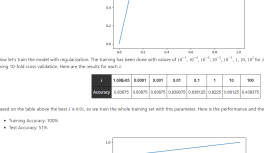


Now let's train the model with regularization. The training has been done with values of 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10, 10^2 for λ and each of the values has been assessed using 10-fold cross validation. Here are the results for each λ:

λ	1.00E-07	0.0001	0.001	0.01	0.1	1	10	100
accuracy	0.83075	0.83075	0.83075	0.83075	0.83075	0.8225	0.59125	0.43075

Based on the table above the best λ is 0.01, so we train the whole training set with this parameter. Here is the performance and the ROC curve of this model:

- Training Accuracy: 100%
- Test Accuracy: 91%



As can be seen the performance dropped tremendously with regularization.

General Naive Bayes

The following functions has been implement to build the Naive Bayes classifier for this problem:

- `stats(data, label)`: Finds μ and diagonal covariance matrix and the prior probability for each class (assuming 2, one of which being 1) of data
- `conditional(x, cov, u)`: Finds the conditional probability ($p(x \mid u)$ assuming a gaussian underlying distribution and using the multivariate normal distribution formula. For not dealing with the hassles of high precision data, the probability is returned in the logarithmic form.
- `get_classifymin(M, cond, prior0, mu0, cov0, prior1, u, threshold)`: Classifies x into 1 or zero using the stats provided via gaussian naive bayes procedure:

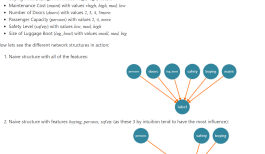
$$\begin{cases} \text{if } a_1 - a_0 > 0 & Class = 1 \\ else & Class = 0 \end{cases}$$
$$a_i = log(u \mid x_i) + log(prior)$$

- `false_positive_rate(labels, predictions)`
- `true_positive_rate(labels, predictions)`

After using the training data to obtain the stats, here is the accuracy for each set of data after they have been classified:

- Training Accuracy: 0.806679
- Test Accuracy: 0.7225

And here the ROC curve associated with this model (This curve is obtained using values between -1 to 1 with steps of 0.1 in size as threshold, after normalizing the posteriories by their sum):



Bayesian Network

Using the `pgmpy` python library the following function has been implemented:

- `bayes_network(train, test, columns, label, edges)`

This function creates a bayesian network using the edges provided as input, then test the model using the test data and returns the accuracy. Using 10-Fold cross validation and the Cars data set, different models has been tested. First let's take a look at the data.

The cars data set has 6 attributes in total:

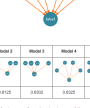
- Buying Price (buying) with values vhigh, high, med, low
- Maintenance Cost (maint) with values vhigh, high, med, low
- Number of Doors (doors) with values 2, 3, 4, 5more
- Passenger Capacity (persons) with values 2, 4, more
- Safety Level (safety) with values low, med, high
- Size of Luggage Boot (lug_boot) with values small, med, big

Now lets see the different network structures in action:

1. Naive structure with all of the features:

2. Naive structure with features *buying*, *persons*, *safety* (as these 3 by intuition tend to have the most influence):

3. Naive structure with features *maint*, *doors*, *safety*, *lug_boot*:



4. Complex structure with features *buying*, *maint*, *doors*, *persons*:

As the number of doors influences the number of persons able to use the car and the buying price influence the maintenance cost, we relate the two features

5. Complex structure with features *buying*, *maint*, *doors*, *persons*, *safety*:

6. Complex structure with features *buying*, *maint*, *doors*, *persons*, *safety*, *lug_boot*:

Here is the table of results:

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Network						
Accuracy	0.2312	0.8125	0.8302	0.8325	0.8287	0.2222

As can be seen the second model which we pick by intuition is the best among the naive structures and the accuracy even surpasses both models with all features (model 1 and model 6) and the best accuracy overall belongs to the 5th model where the logical assumption in the relations between 4 variables has been made added by safety.