Name: Isha Aggarwal

Bayes and Naïve Bayes Classifier Implementation

Abstract: The report is about the implementation of Bayes and Naïve Bayes Classifier from scratch. IRIS dataset is used for training and testing the classifiers. For both the problems, there are 3 sections: the first one describes the theoretical/mathematical details which are translated to the implementation, the second section describes the methods in the real implementation/code and the last one presents the accuracy results.

Implementation of Bayes Classifier

Theoretical and Implementation Details:

Section 1: Theory behind Implementation

Given a feature vector X = (a_1, a_2 , ... , a_d)^T (d= 4 acc. to the given dataset) , which is the mathematical representation of the object to be classified, Bayes Classifier outputs a class label y , corresponding to the maximum posterior probability.(posterior probabilities are determined following the Bayes Theorem)

3 classes w_1 ,w_2 w_3 are there in the given dataset. (Notations w_1 ,w_2 w_3 are used here for class labels instead of full class names for convenience)

Probability structure of the problem:

The priori probabilities P(w_1), P(w_2),P(w_3) are assumed to be equal as instructed in the question. P(w_1) = P(w_2) = P(w_3) = 1/3

Class conditional probability density function i.e. p(X|w_i) is assumed to be joint Normal.

X|w_i ~ N(u_i, Σ_i)

p(X|w_i) = 1/((2π)$^{d/2}$|Σ_i|$^{1/2}$) exp[-1/2 * $(X - u_i)^T\, \Sigma_i^{-1}(X - u_i)$]

d = 4 acc. to the given dataset

u_i and Σ_i (parameters of p(X|w_i)) are approximately taken as the sample mean and the sample covariance matrix as instructed.

Therefore,

$u_i \cong \frac{1}{n}\sum_{j=1}^{n} X_j$ (where n = # of training examples with class labels : w_i)

$\Sigma_i \cong \frac{1}{n}\sum_{j=1}^{n}(X_j - u_i)(X_j - u_i)^T$ (n = # of training examples with class labels : w_i)

Now, $P(w_i|X) = (p(X|w_i).P(w_i))/p(X)$ $(p(X) \neq 0)$ (Bayes Theorem)

$g_i(X) \equiv -1/2 * (X_j - u_i)^T \Sigma_i^{-1}(X - u_i) - 1/2 \ln|\Sigma_i| + \ln P(w)$ (Discriminant function for class w_i)

Output class label = argmax{ $P(w_1|X)$, $P(w_2|X)$, $P(w_3|X)$ }

= argmax{ $g_1(X)$, $g_2(X)$, $g_3(X)$ }

Section-2: Implementation

→Implemented from scratch without using any external library except numpy for matrix operations such as inverse, matrix product and determinant…

In the BayesClassifier class, several methods are there which to deal with:

priori probabilities: Priori(self,class_label)

estimation of parameters u_i and Σ_i for the class conditional probability density function : MeanVec_CovMatrix_of_class_conditional_distibution(self,class_name)

discriminant function: Discriminant_function_for_given_class(self,class_label , feature_vec)

Given an input feature vector, the class label corresponding to the maximum discriminant function value is returned: Classify_input_feature_vector(self,feature_vec)

Methods for measuring accuracy and accepting user feature vector input are also there.

Section-3 : Accuracy of the classifier

Accuracy of the Bayes Classifier over the test set (= 100* # of correct decisions/total # of test set examples) is calculated practically(since theoretical error calculation is tedious), which comes out to be **100%.**

This Bayes Classifier would be the best classifier with minimum average error if the priori probabilities and $p(X|w_i)$ values are correct/exact.

E_X[P (error|X)] = $\int_{R^d} P(\text{error}|X)\, p(X)\, dX$; P(errox|X) = 1 - max{ $P(w_1|X)$, $P(w_2|X)$, $P(w_3|X)$ }

<u>**Implementation of Naïve Bayes Classifier(discrete features)**</u>

Theoretical and Implementation Details:

Section-1: Theory behind Implementation

Given a feature vector X = (a_1, a_2 , ... , a_d)^T (d= 4 acc. to the given dataset) , which is the mathematical representation of the object to be classified, Naïve Bayes Classifier outputs a class label y , corresponding to the maximum posterior probability.(posterior probabilities are determined following the Bayes Theorem) It assumes that individual features are mutually independent, given a class.

3 classes w_1 ,w_2 w_3 are there in the given dataset. (Notations w_1 ,w_2 w_3 are used here for class labels instead of full class names for convenience)

Probability structure of the problem:

The priori probabilities $P(w_1)$, $P(w_2)$,$P(w_3)$ are estimated using the training set.

$P(w_i)$ = # of training examples with class label w_i / total # of training examples

$P(X|w_i) = \prod_{j=1}^{d} P(a_j|\ w_i)$, where $X = (a_1, a_2 , \ldots , a_d)^T$ (because of conditional independence assumption)

Feature vectors are taken to be discrete, if not then they are discretized first.

$P(a_j|\ w_i)$ is also estimated using the training set.

$P(a_j|\ w_i)$ = # of training examples with jth feature equal to a_j and class label: w_i / total # of training examples having class label w_i

Now, $P(w_i|X) = (P(X|w_i).P(w_i))/P(X)$ $(P(X) \neq 0)$ (Bayes Theorem)

$g_i(X) \equiv P(X|w_i).P(w_i)$ (Discriminant function for class w_i)

Output class label = argmax{ $P(w_1|X)$, $P(w_2|X)$, $P(w_3|X)$ }

= argmax{ $g_1(X)$, $g_2(X)$, $g_3(X)$ }

Section-2: Implementation

→Implemented from scratch without using any external library except numpy for matrix operations such as inverse, matrix product and determinant…

In the NaiveBayesClassifier class, several methods are there which to deal with:

priori probabilities: Priori(self,class_label)

Class conditional probability: ClassConditional(self,feature_vec,class_label,discretization_level)

discriminant function:
Discriminant_function_for_given_class(self,class_label,feature_vec,discretization_level)

Given an input feature vector, the class label corresponding to the maximum discriminant function value is returned:
Classify_input_feature_vector(self,feature_vec,discretization_level)

Methods for measuring accuracy and accepting user feature vector input are also there

NOTE: Discretization:- Feature vectors are discretized to different levels depending upon the user arguments. Features may be rounded off to the first decimal place or the nearest integer.

Section-3 : Accuracy of the classifier

Accuracy of the Naïve Bayes Classifier over the test set (= 100* # of correct decisions/total # of test set examples) is calculated practically(since theoretical error calculation is tedious), which comes out to be **around 97%, when the features are rounded off to the nearest integer.**

The accuracy varies from 65% to 80% over the test set (in different executions of the program) when the features are rounded off to the first decimal place. The reason for this variation is that a random label is outputted in case the discriminant function values are equal for all classes given an X. And for many feature vectors in the test set, the discriminant function values are all equal, so the output class label (decided randomly) may not match the output label produced before, hence causing variation in accuracy for the same test data.

Theoretically the expected error is:

E_X[P (error|X)] = ∑ P(error|X) P(X) (Summation over all the discrete feature vectors) ;

P(errox|X) = 1 - max{ $P(w_1|X)$, $P(w_2|X)$, $P(w_3|X)$ }