PERFORMANCE OF
BAYESIAN CLASSIFIER BASED ON GAUSSIAN GENERATIVE MODEL

Tony Liu

(C number: C12205888)

ECE 730 STATISTICAL LEARNING

Project Report

Contents

1. Introduction

The main purpose of this project is to design experiments in order to investigate the performance of Bayesian classifier based on Gaussian generative model. To this end, simulated data generated from multi-variate Gaussian distribution with different generating parameters was used to evaluate the performance of the classifier in various data environments. The project also applied the classifier to zip code data set in order to explore the behavior of it in real-world problem, specifically, multi-class classification problem.

2. Bayesian Classifier Based on Gaussian Generative Model

Bayesian classifier classifies a new example $\boldsymbol{x}$ with the label that maximizes the class-conditional possibility, $p(C_k|\boldsymbol{x})$. According to Bayes' theorem,

$$p(C_k|\boldsymbol{x}) \propto p(\boldsymbol{x}|C_k)p(C_k)$$

thus, to determine the value of the posterior probability, we need to first estimate the prior probability, $p(C_k)$ and the class-conditional density, $p(\boldsymbol{x}|C_k)$. Gaussian generative model holds the assumption that the class-conditional density is Gaussian, so the parameters needed are the mean vector, $\boldsymbol{\mu}$, and the covariance matrix, $\boldsymbol{\Sigma}$.

Suppose we have the data set $\{\boldsymbol{x}_n, \boldsymbol{t}_n\}$, using MLE (Maximum Likelihood Estimation), the estimated value of the parameters above are:
Class prior: $\hat{\pi}_k = \frac{N_k}{N}$
Class conditional density: $p(\boldsymbol{x}|C_k) = N(\boldsymbol{x}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma}_k)$
where:

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k}\sum_{i=1}^{N_k} \boldsymbol{x}_i \ (\boldsymbol{y}_i = k)$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k}\sum_{i=1}^{N_k}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T$$

Therefore, the predicted class (k) for a new example is:

$$\underset{k}{argmax}\ \hat{\pi}_k|\hat{\boldsymbol{\Sigma}}_k|^{-\frac{1}{2}} exp[-\frac{1}{2}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T\hat{\boldsymbol{\Sigma}}_k^{-1}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)]$$

3. Experiments on Simulated Data set

In this chapter, the performance of Bayesian classifier based on Gaussian generative model was measured in different kinds of simulated data sets. The data sets were generated from multi-variate Gaussian distribution, but they varied in the number of relevant attributes, the number of classes, noise level, etc. Additionally, the project also shows the effect of whether employing CoV (Coefficient of Variation) as the feature selection method.

General Setting of Experiment

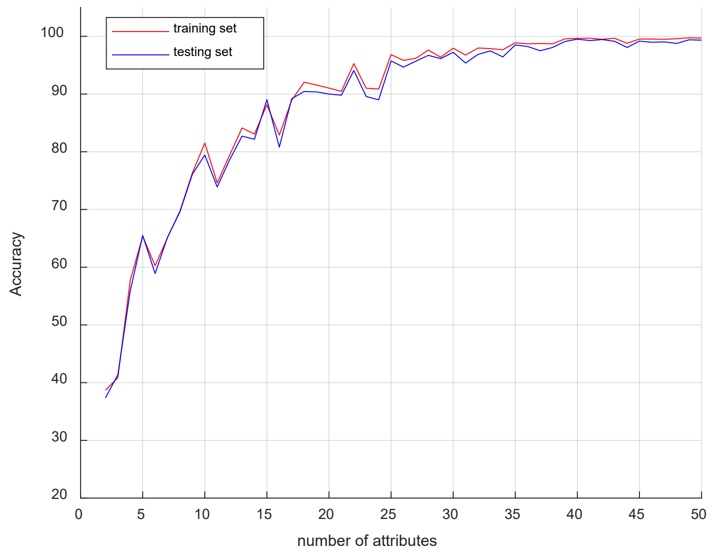
Unless otherwise specified, the parameters used to generate data are listed as follow:

Number of attributes (p)	20	Number of all data (Σn_k)	10,000
Number of relevant attributes (p_1)	20	Proportion of training and testing data	7:3
Number of classes (K)	5	Imbalanced data	False
Noise level (σ^2):	5	The range of mean of each attribute	0 to 5

The five classes share the same covariance matrix, $\sigma^2\mathbf{I}$.

3.1 Accuracy against the Number of Attributes

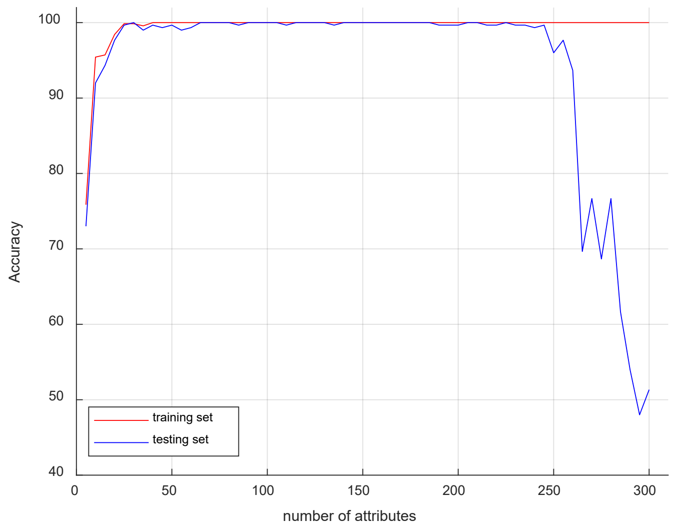
The number of attributes varied in this experiment, which were all relevant.

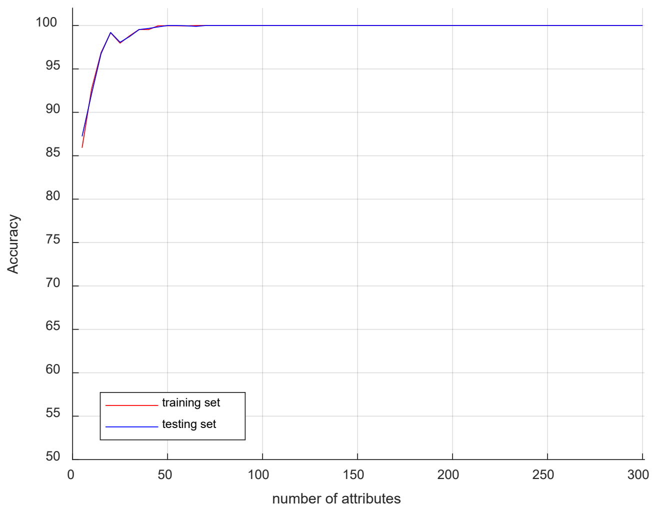


Intuitively, as the number of attributes grows, the classes would be different in more “aspects”, which makes it easier for the classifier to distinguish them from each other. The result above is indeed consistent with this intuition. Therefore, to improve the accuracy of the classifier, adding more attributes could be a potentially fruitful method.

However, simply adding more attributes would sometimes lead to a worse performance. One possible reason is that as the number of attributes increases linearly, the number of data points needed may grow exponentially, which is known as “curse of dimensionality”

Suppose we have only 1000 data points, we would expect the accuracy of the classifier improves at the beginning but then falls down, which is shown as follow.





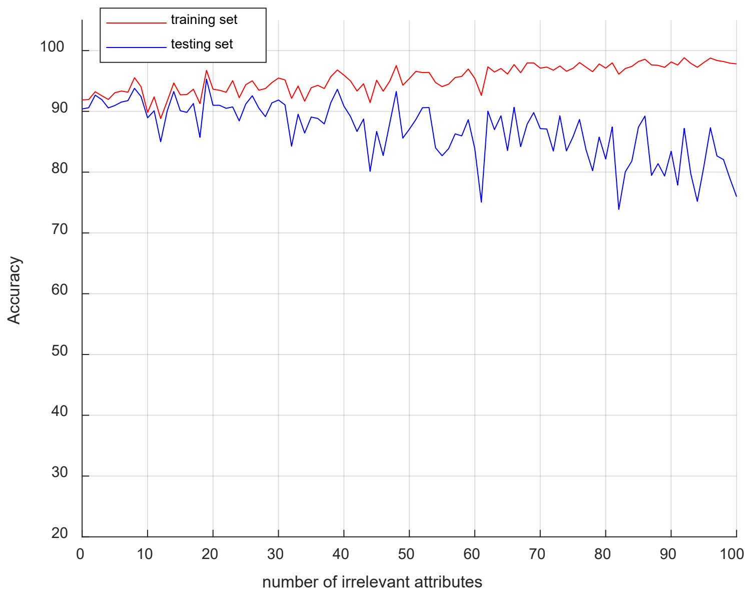
The first graph was made with only 1,000 data points available while there were 10,000 for the second.

The result of the first graph is exactly what we expected. The accuracy goes up at the beginning but then start to drop significantly when the number of attributes is approximately 250 as the result of "curse of dimensionality". The curse has been overcome in the second graph as there are enough data points, but it may appear again when there are 3000 attributes (which could not be confirmed due to limited computing power)

Another possible reason that accounts for the worse performance by adding more attributes is that the attributes added are irrelevant to the classification problem, as we would see in 3.2.

3.2 Accuracy against the Number of Irrelevant Attributes

In this experiment, the number of relevant attributes is fixed to 20, and the number of irrelevant attributes grows from 0 to 100.



As was discussed in 3.1, simply adding more attributes is not guaranteed for a better performance. The accuracy on training set gradually goes up while it goes down on testing set. If the number of irrelevant attributes keeps increasing, we would expect to observe that the classifier has high accuracy on training set whereas perform unacceptably on the testing set.

Mathematically speaking, those irrelevant attributes come to disturb when the classifier is making the decision by computing $exp[-\frac{1}{2}(\boldsymbol{x}_i-\widehat{\boldsymbol{\mu}}_k)^T\widehat{\boldsymbol{\Sigma}}_k^{-1}(\boldsymbol{x}_i-\widehat{\boldsymbol{\mu}}_k)]$. They make the value slightly greater or less than it is supposed to be. The more the number of irrelevant attributes is, the stronger this value will be influenced.

In this section, the mean of relevant attributes is independently randomly generated from the range of 0 to 5, while the mean of the irrelevant attributes is set to 0. The difference between them is relatively big enough, which accounts for the classifier still having good performance when the number of irrelevant attributes is 40, twice more than relevant attributes. Actually, if the mean of relevant attributes is set to generated from 0 to 100 and the mean of irrelevant attributes remains 0, the accuracy curve would be a straight line at 100.

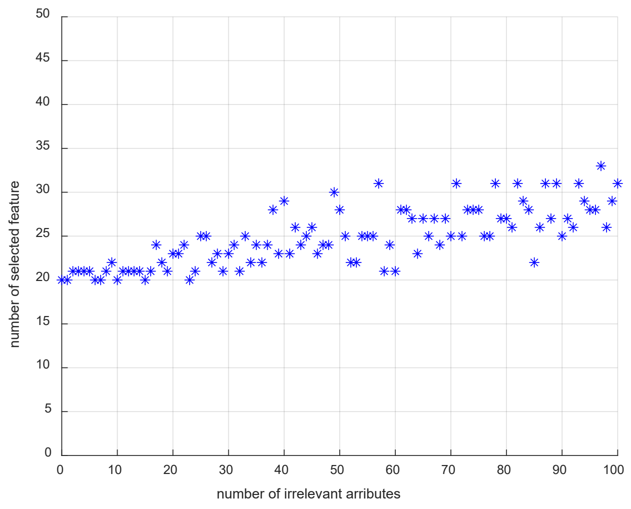
Therefore, to avoid the problem brought by irrelevant attributes, one optional method is to increase the mean of relevant attributes, but this could be impractical because we may not know which attributes are relevant to the problem in advance.

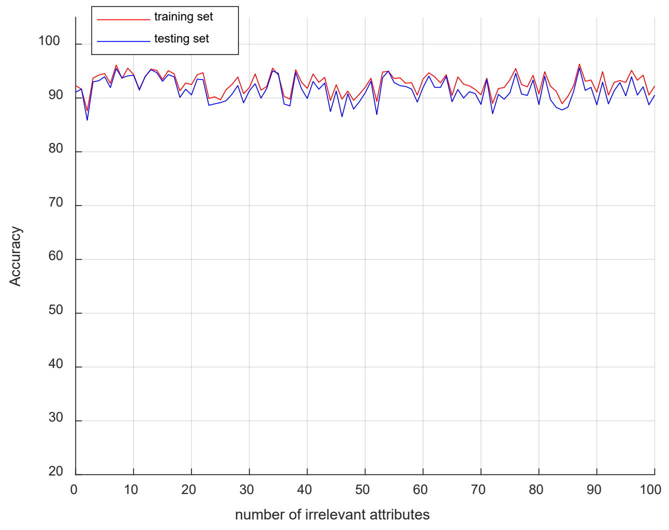
Another practical method is designing an appropriate method to rule out these irrelevant attributes, making only relevant ones left, which will be discussed in 3.3

3.3 Accuracy with Applying CoV Method

The experiment setting of this section is similar with 3.2. The only difference is that CoV method had been introduced to select features before the classifier was trained.

NOTE: the CoV method was used slightly different from our project instruction. In the project instruction, the attribute whose CoV value is greater than a threshold is to be used for training. However, in this experiment, since CoV value is computed by σ / μ, and the irrelevant attributes has the mean of 0, CoV value of irrelevant attributes is expected to be relatively large. Thus, it is those attributes whose CoV value is less than a threshold that are kept for classification. The threshold was set to 50 in this experiment.



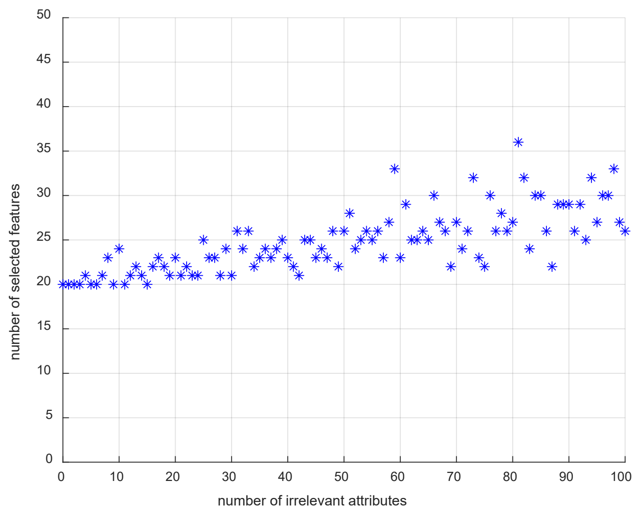


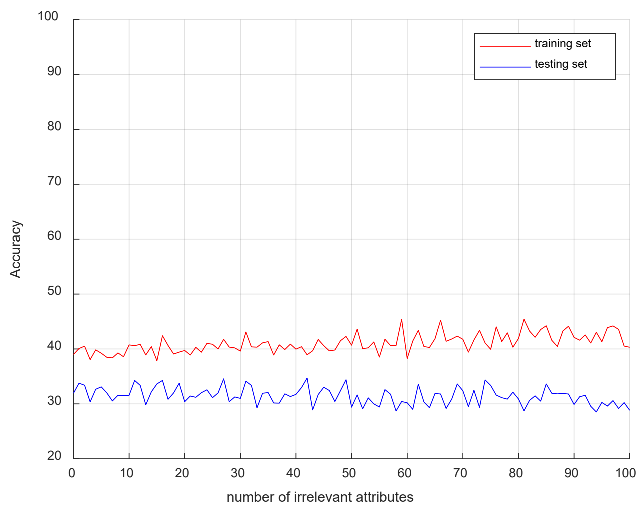
The first graph is the number of attributes remained against the number of irrelevant attributes added. The second graph is the accuracy curve over the number of irrelevant attributes.

By applying CoV method to select attribute, the number of attributes is dramatically lessened when there are fairly many irrelevant attributes.

The curves of accuracy on testing set and training set keep the same pace all the time and slightly fluctuate around 90, which is the same accuracy achieved in section 3.1 when the number of attributes is 20 and in section 3.2 when the number of irrelevant attributes is 0. Since there is enough gap between the mean of relevant and irrelevant attributes, CoV method performs quite well even there are fairly many irrelevant attributes.

However, if the mean of relevant attributes was generated from 0 to 1, CoV method would fail to give a satisfying result.





These two graphs are the results when the mean of relevant attributes is generated from 0 to 1 (the mean of irrelevant attributes is still 0)

In this case, CoV method fails to distinguish relevant and irrelevant attributes although the number of attributes remained is similar with previous result. This indicates that CoV method works only when the CoV value of relevant and irrelevant attributes are distinctive, otherwise there is great possibility that it would mix up relevant and irrelevant attributes and thus gives the result even worse than without it.

3.4 Accuracy against Mean Vector

Since the data points of the five classes share the same covariance matrix in this project, the mean vector of each class becomes the key factor for the classifier to predict the class of a new example. If two classes have "similar" mean vector, they may become indistinctive to the classifier.

In this experiment, the accuracy of classification over the "similarity" of mean vector is explored. The "similarity" between mean vectors is defined by the Euclidean distance. The shorter the distance, the more similar they are. Several Gaussian distributions with similar mean vectors are close to each other, having larger overlapping area and thus are harder to be distinguished.

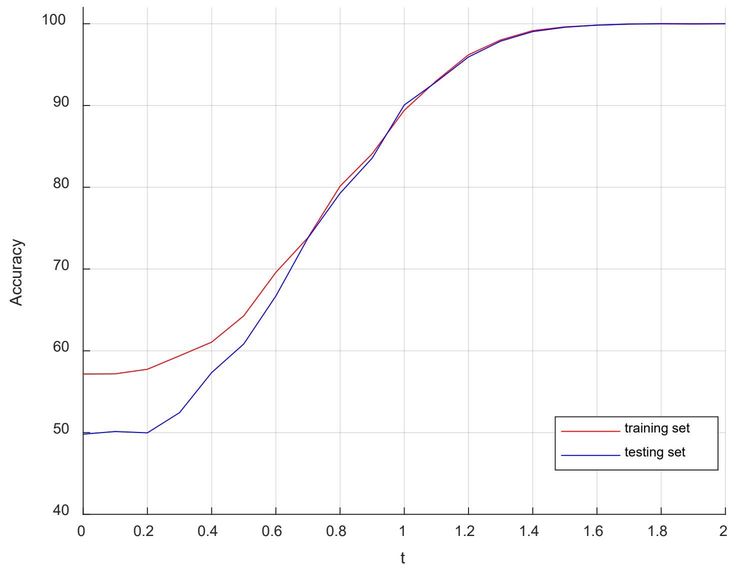
There are only two classes, class1 and class2, in this experiment, which simplifies the experiment with no harm to the idea behind, and it could be easily extended to multi-classes situation.

The mean vector of class1 was first randomly generated and then fixed all the time. The mean vector of class2 was generated by:

$$\boldsymbol{\mu_2} = \boldsymbol{\mu_1} + t \times \frac{\vec{\mathbf{1}}}{\sqrt{p}}$$

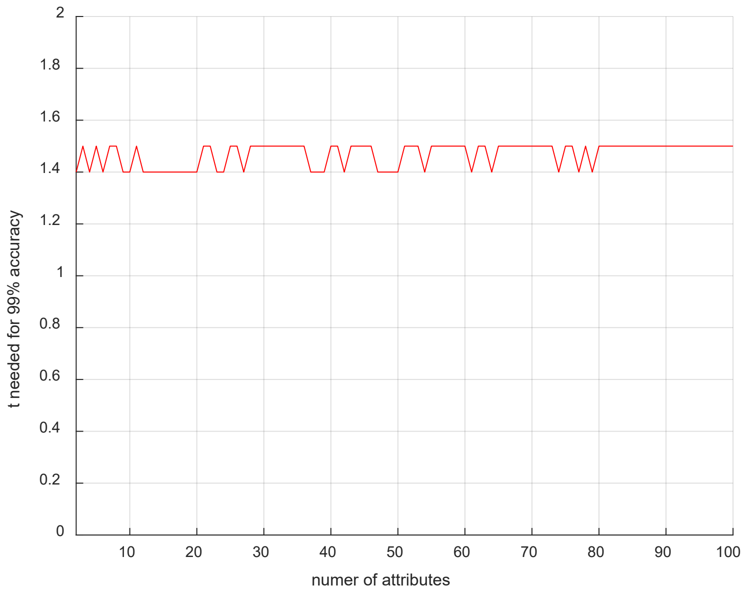
where t is the coefficient used to control the Euclidean distance with $\boldsymbol{\mu_1}$. $\vec{\mathbf{1}}$ is a vector with the same dimension as $\boldsymbol{\mu_1}$ and all components as 1. p is the number of attributes (no irrelevant attributes in this experiment) and $\sqrt{p}$ is a normalization term so that the Euclidean distance between the two vectors is exactly t.

The result of the experiment is shown as follow.



When t is 0, the two Gaussian distributions could be regarded as the same. The performance of the classifier on testing set has nothing better than guessing. As t increases, the accuracy improves extraordinarily. When t is 1.4, the classifier has 99% accuracy.

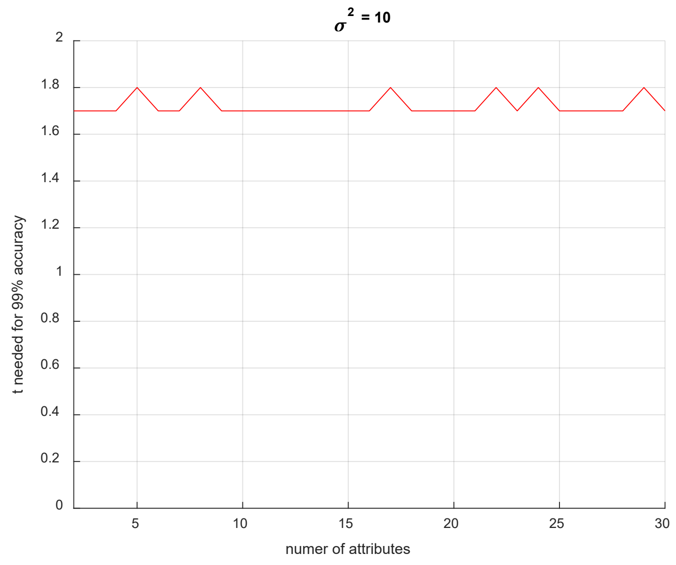
The minimum t needed for the classifier to achieve 99% accuracy in other spaces of different dimension is fixed to 1.45 on average.

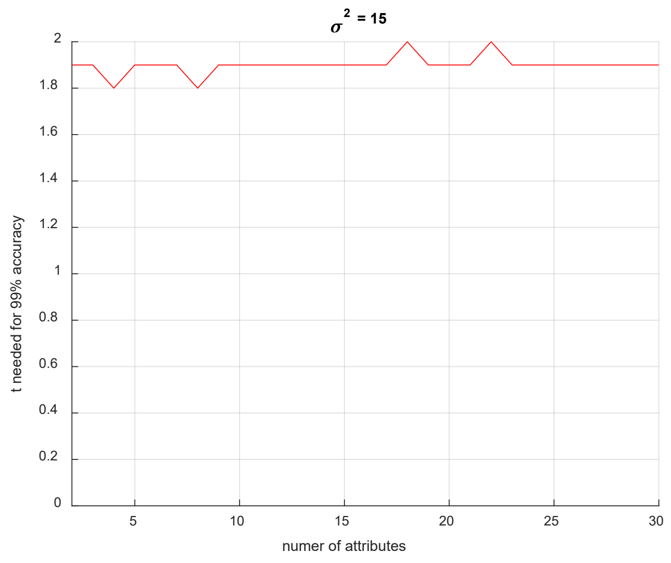


This graph shows the result that for different number of attributes, the minimum t needed for the classifier to achieve 99% accuracy. We could learn from the graph that as the dimension increases, the minimum t is fixed around 1.45. This result could be interesting as it shows that Bayesian classifier based on Gaussian generative model has this "dimensionality-proof" property that the classifier can perfectly distinguish two Gaussian distributions as long as the Euclidean distance between them is more than 1.45 no matter how many dimensions there are.

The distance such as the 1.45 in this case shall be referred as "minimum separation", meaning that the classifier could achieve 99% accuracy if the Euclidean distance between two mean vectors of two Gaussian distributions is greater than or equal to it.

The minimum t was also measured with different value of noise level (σ^2).



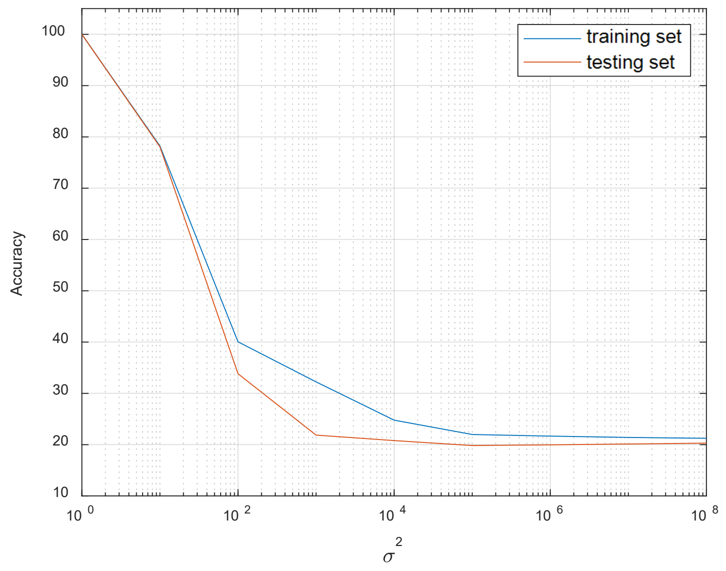


These two graphs show the minimum t when noise level is 10 and 15. Based on this, we could learn that as the noise level strengthens, the minimum separation increases, but it does not depend on the number attributes.

In a nutshell, fix some value of noise level, minimum separation is not related to the number of attributes.

3.5 Accuracy against Noise Level

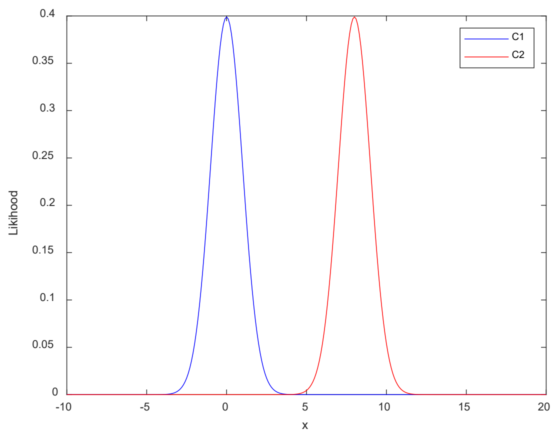
The noise level grows exponentially in this experiment. Other parameters remain unchanged as in the general setting. The result is plotted in a semilogarithmic manner.



The accuracy of the classifier on testing set deteriorate when σ^2 increases and ultimately has the same performance as pure guessing.

As a matter of fact, the classifier has perfect estimated value of mean vector and covariance matrix for each class even when σ^2 is 10^8. In other words, the classifier knows exactly how data points of each class are generated but still fails to make a fair prediction. The reason for this disappointing behavior is that when σ^2 is extremely large, the data are not Gaussian distributed but uniform distributed. There is no obvious difference if class-conditional probability is calculated.

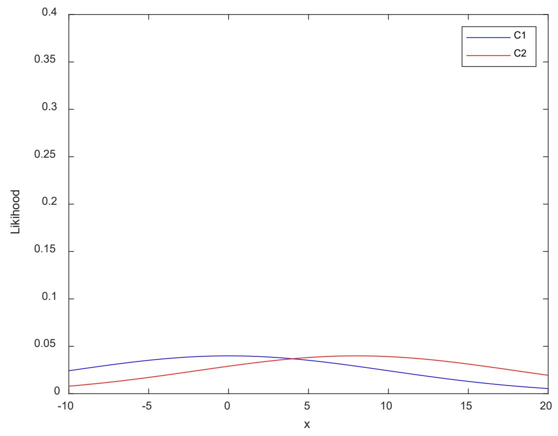
I would like to take the example of univariate Gaussian distribution for further explanation. Suppose there are two classes in the data set where data points of each class are Gaussian distributed, and the two distributions have different mean but share the same variance.



The two "bells" are well separated from each other. After training, the classifier would have perfect estimated parameters of these two distributions. Suppose the prior of both classes is 0.5, thus if a new example with x = 0, label = 1 would have the posterior probability $p(C_1|x = 0) =$

$0.4 \times 0.5 = 0.2$ and $p(C_2|x = 0) \approx 0 \times 0.5 = 0$. The classifier would perform quite well in this case.

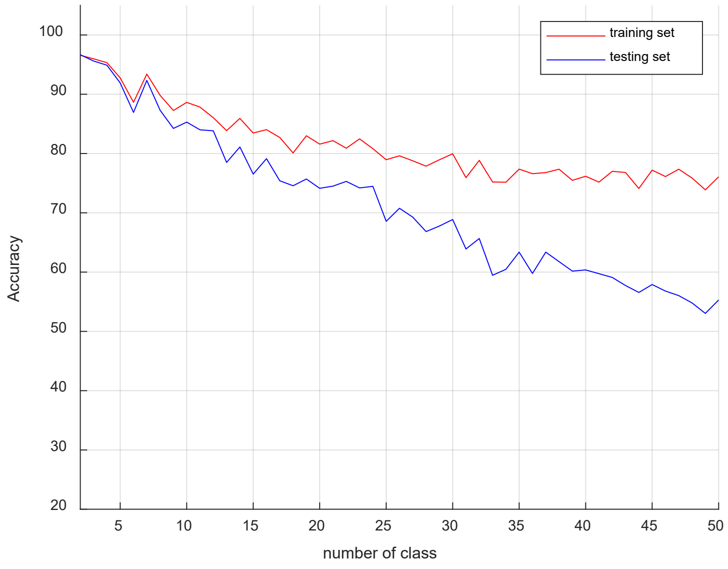
However, if we fix the mean of each distribution and increase the variance, the curves will tend to be more and more flatter and even almost parallel in extreme cases. The posterior probability of the example with x = 0 for C_1 and C_2 could be very close, which results in the classifier being unable to make the correct decision.



When σ^2 is getting even larger, Gaussian distribution would degenerate to uniform distribution. Therefore, it is not surprising to expect the failure of Bayesian classifier based on Gaussian generative model as it has an inappropriate assumption on the process where the data points are generated.

3.6 Accuracy against the Number of Classes

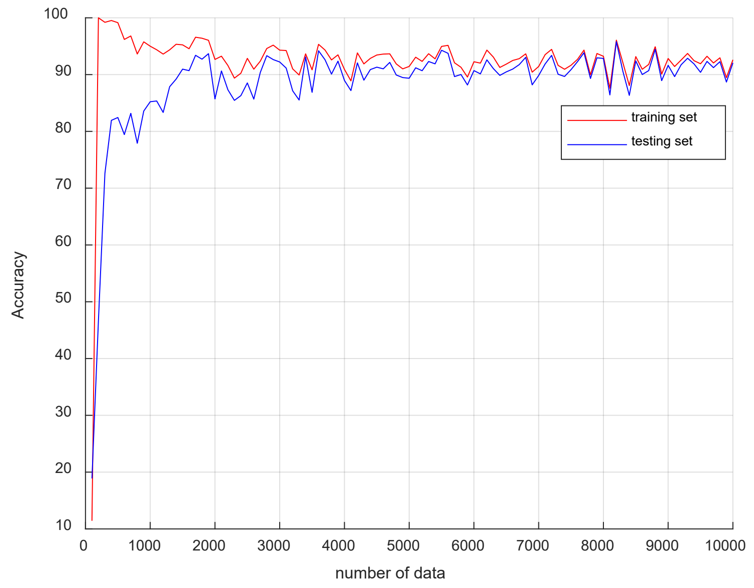
Fix some number of attributes, it would be more and more difficult for the classifier to achieve great performance as the number of possible classes grows.



The result of the experiment is the same as our expectation. With the increasement of the number of classes, the average number of attributes that could be used to distinguish classes decreases. Thus, the classifier will naturally fail to make great prediction and even overfit when the number of classes is significantly more than available features.

3.7 Accuracy against the Number of Data

The number of available data plays an important role in the whole machine learning framework. In the case of Gaussian generative model, data points are used to estimate the mean vector and covariance matrix. The more data we have, the more accurate they will be. The number of available training data grows from 100 to 10,000 in this experiment.



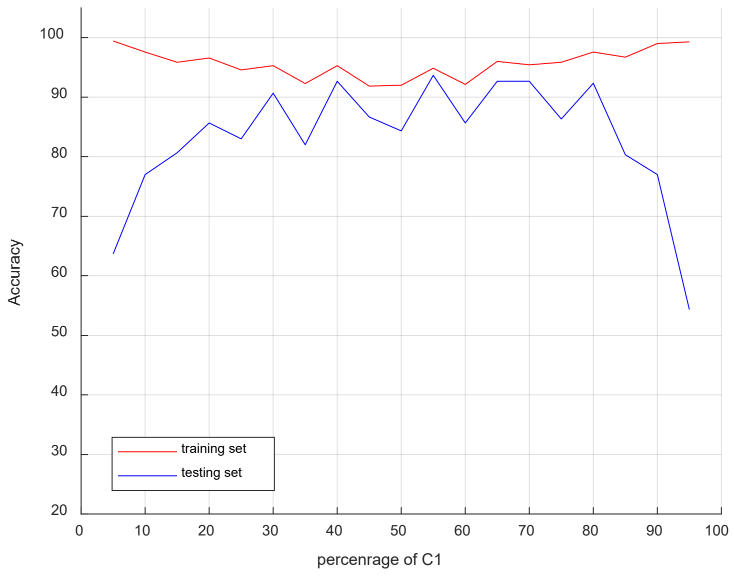
At the beginning when there are only 100 data points, the classifier was even worse than pure guessing because the estimated covariance matrix of each class is non-invertible. The two accuracy curves converged at 2000 data points with the accuracy around 90. There is no improvement when more data points were added.

This result could be good news for those practitioners who expect an accuracy around 90 because only 2000 data points are needed for a five-class classification problem with 20 attributes. However, it may disappoint those with higher desired performance as accuracy gets no improvement with even more data.

3.8 Performance on Imbalanced Data set

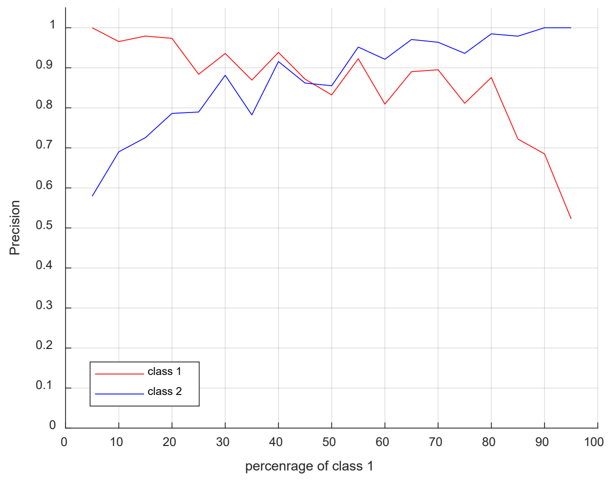
Up to now, all the experiments were conducted under uniform training set where every class has similar number of data points. However, this is not always the case in practice. Therefore, the performance of the classifier on imbalanced data set is tested in this section.

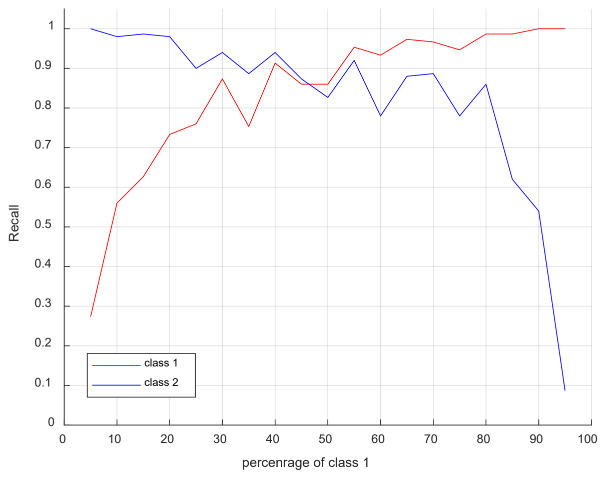
Again, there are only two classes in this experiment. The training set generated is imbalanced, where the proportion of class1varies from 5% to 95% while the testing set remain uniform. The evaluation of the performance is based on accuracy, precision, recall and F_1 score. Except accuracy, all these criterions were calculated in testing set only.



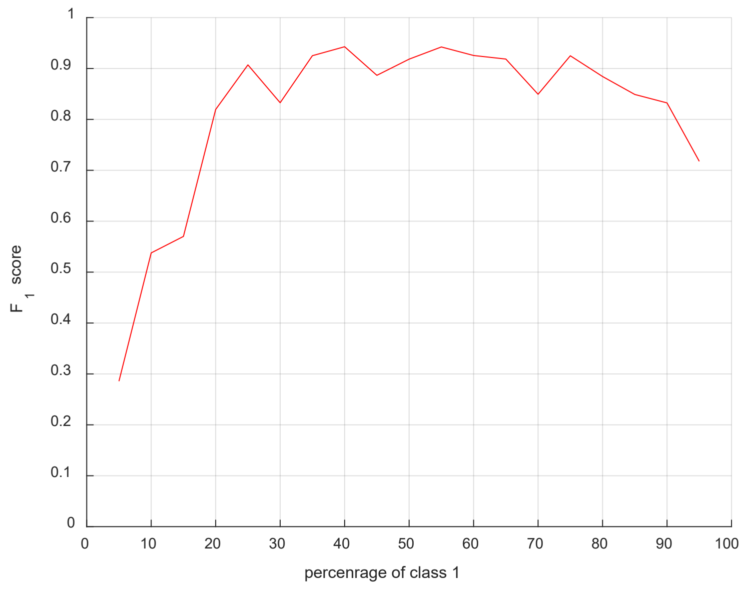
The result of accuracy is shown as above. The accuracy on training set remains high level but it is low on testing set whenever there are too few or too many C1 examples.

In the case of few C1 examples, the classifier fits C2 better, which lead to an imbalanced performance on the two classes, resulting in the overall accuracy being worse than on the training set. Similar things happened when there are too many C1 examples.





Concretely, the imbalanced performance of the classifier on the two classes could be observed by computing precision and recall. Trained from an imbalanced data set, the classifier could not have both high precision and recall at the same time, that is, when the precision on one class is high, the recall on that class is doomed to be low and vice versa.



F_1 score is calculated as the general indicator of the performance where both precision and recall are considered. We could learn from the graph that F_1 score remains 0.9 when the percentage of class1 examples is between 30% to 80%. If the result of accuracy, precision and recall is observed in this interval, the two curves are closer to each other than if observed outside of it. In other words, the classifier's performance in terms of accuracy, precision and recall is relatively balanced in this interval. Therefore, the classifier's performance could be acceptable when the proportion of the examples of one class is at most 30% more or fewer than the other. In concise language, the "tolerance of imbalance" of Bayesian classifier based on Gaussian generative model is 30%.

4. Experiments on Real Data set

In this chapter, the performance of Bayesian classifier based on Gaussian generative model is measured on real data set, namely, zip code data set.

In the zip code data set, each example consists of the digit id (0-9) followed by the 256 grayscale values (-1 to 1). There are 7291 training observations and 2007 test observations, distributed as follows:

	0	1	2	3	4	5	6	7	8	9
Train	1194	1005	731	658	652	556	664	645	542	644
Test	359	264	198	166	200	160	170	147	166	177

Or as proportions:

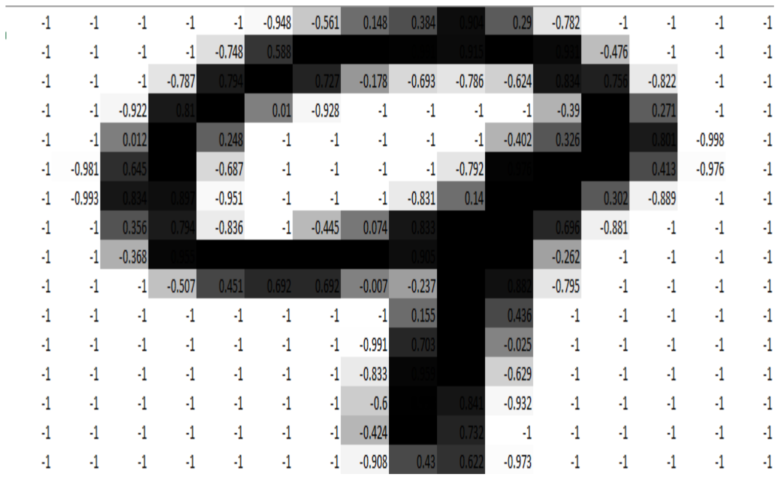
	0	1	2	3	4	5	6	7	8	9
Train	0.16	0.14	0.1	0.09	0.09	0.08	0.09	0.09	0.07	0.09
Test	0.18	0.13	0.1	0.08	0.1	0.08	0.08	0.07	0.08	0.09

The data set is uniform, making the test result depend on only the classifier itself.

4.1 Feature selection

Since there are 256 features, the covariance matrix of the attribute could be singular, making it impossible to compute the inverse of the covariance matrix. Therefore, ruling out irrelevant attributes to reduce the total number of attributes has to be done before training.

Reshape the 256 features to form a 16-by-16 pixel matrix, we could visualize an example as follows.

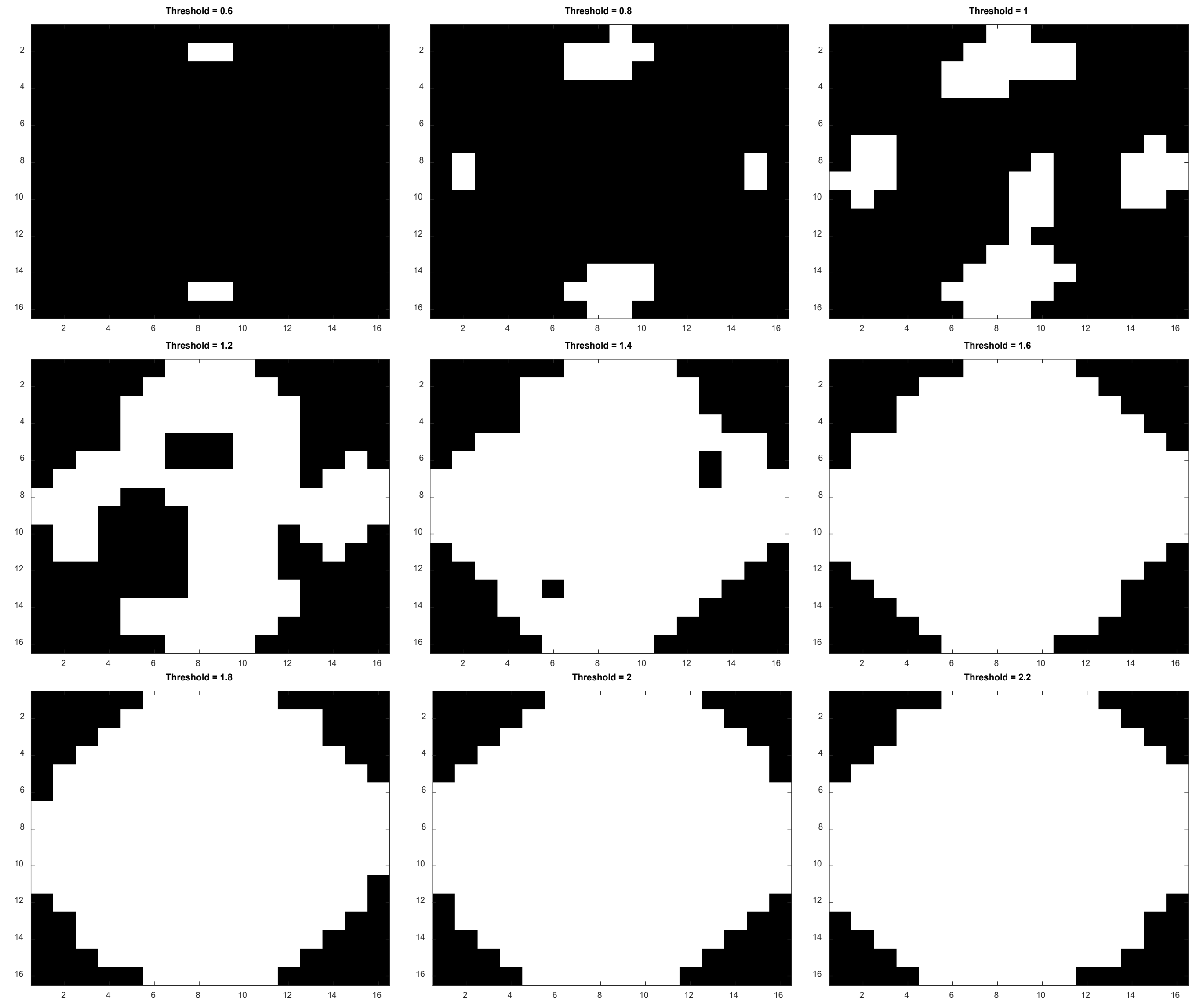


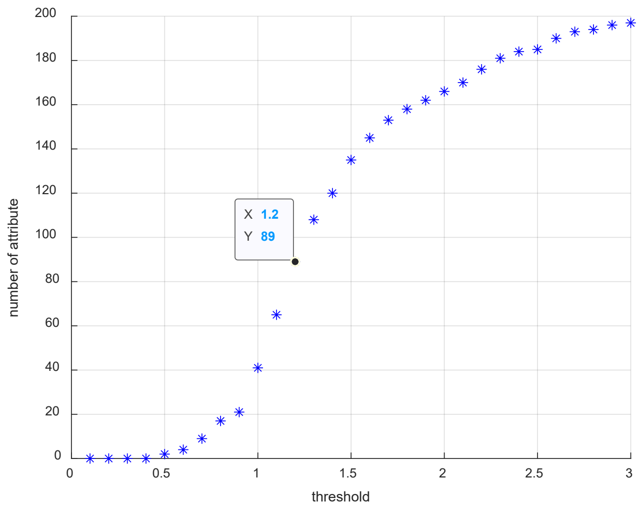
-1	-1	-1	-1	-1	-0.948	-0.561	0.148	0.384	0.904	0.29	-0.782	-1	-1	-1	-1
-1	-1	-1	-1	-0.748	0.588	[illegible]	[illegible]	[illegible]	0.915	[illegible]	0.931	-0.476	-1	-1	-1
-1	-1	-1	-0.787	0.794	[illegible]	0.727	-0.178	-0.693	-0.786	-0.624	0.834	0.756	-0.822	-1	-1
-1	-1	-0.922	0.81	[illegible]	0.01	-0.928	-1	-1	-1	-1	-0.39	[illegible]	0.271	-1	-1
-1	-1	0.012	[illegible]	0.248	-1	-1	-1	-1	-1	-0.402	0.326	[illegible]	0.801	-0.998	-1
-1	-0.981	0.645	[illegible]	-0.687	-1	-1	-1	-1	-0.792	0.976	[illegible]	[illegible]	0.413	-0.976	-1
-1	-0.993	0.834	0.897	-0.951	-1	-1	-1	-0.831	0.14	[illegible]	[illegible]	0.302	-0.889	-1	-1
-1	-1	0.356	0.794	-0.836	-1	-0.445	0.074	0.833	[illegible]	[illegible]	0.696	-0.881	-1	-1	-1
-1	-1	-0.368	0.955	[illegible]	[illegible]	[illegible]	[illegible]	0.905	[illegible]	[illegible]	-0.262	-1	-1	-1	-1
-1	-1	-1	-0.507	0.451	0.692	0.692	-0.007	-0.237	[illegible]	0.882	-0.795	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	0.155	[illegible]	0.436	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-0.991	0.703	[illegible]	-0.025	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-0.833	0.959	[illegible]	-0.629	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-0.6	[illegible]	0.841	-0.932	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-0.424	[illegible]	0.732	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-0.908	0.43	0.622	-0.973	-1	-1	-1	-1	-1

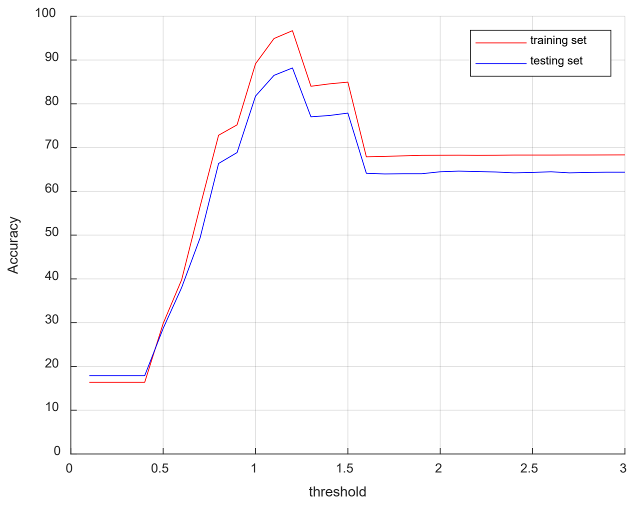
We would notice that irrelevant attributes like the four "corner features" would have the mean roughly equal to -1 and low variance roughly equal to 0. For relevant attributes, they would have different means and the variance greater than irrelevant features. Ruling out the features according to its CoV value is a potentially efficient method for feature selection.

Since the value of each feature lying in the range of -1 to 1 could bring inconvenience for comparing CoV value, the value of each attribute was manually plus one. This not only made the sign of the mean of each attribute positive but also made the irrelevant features like the "corner features" has the mean roughly equal to 0, leading their CoV value greater than relevant features' and could be ruled out if it is greater than a threshold. Thus, the attributes whose CoV value was less than the threshold were kept for model training.

The attribute survived as the threshold increased was plotted as a white square at the corresponding position for visualization.







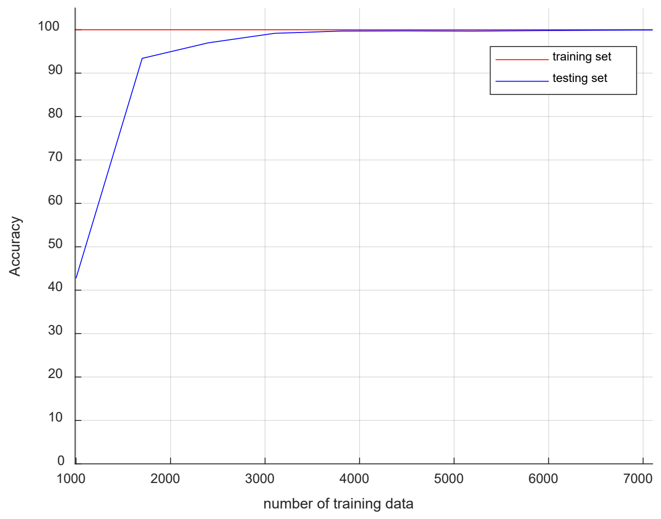
The first graph shows the number of attributes left as the threshold varies. The second graph shows the accuracy achieved when different threshold was applied.

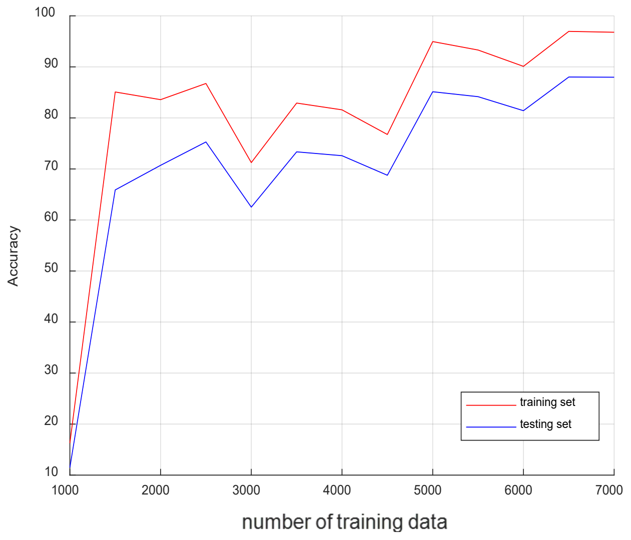
When the threshold increases from 0.1 to 1.2, the accuracy curve goes up because more relevant attributes survive. It reaches the best, 88.19, with 89 attributes left when threshold is 1.2.

We could observe that the accuracy does not change as the threshold is greater than 1.6. This could be interpreted as that the attributes added due to the increasement of threshold are those lying around the corners of the picture (This could be observed in the previous figure where the white area expanded toward the four corners as the threshold grows from 1.6 to 2.2). The mean of these attributes is roughly equal to 0, which is similar with the case when the mean of irrelevant attributes was set to 0 in section 3.3. "Since there is enough gap between the mean of relevant and irrelevant attributes, CoV method performs quite well even there are fairly many irrelevant attributes", as was concluded in that section.

4.2 Accuracy against the Number of Training Data

Indeed, the best accuracy achieved on testing set is 88.19, which is not that satisfying. One possible reason is that the number of training data is inadequate. The relation between accuracy and the number of training data is shown as follow.





The first graph is the experiment result on simulated data where there are 89 attributes and 10 classes. Other generating parameters are the same as general setting. The second graph is the result on real zip code data with CoV threshold being 1.2 with 89 attributes left.

If the data points are Gaussian distributed, the classifier would have already achieved 100 accuracy with only 3000 training examples needed, but in real zip code data set it achieved only roughly 63. There are three main reasons accounting for the difference between the two results. First, the data points in real data are not pure Gaussian distributed. Second, the attributes in real data are not independent as what we assumed in experiment. Third, the covariance matrix is invertible in simulated data, but it is close to singular in real data.

To improve the performance of the classifier in real data, using more training data could be potentially a good way to try as we observe that the accuracy goes up as the number of training data increases. We may also try introducing more sophisticated feature selection method to better decide which attributes should be used for model training.

5. Conclusion

In this project, the performance of Bayesian classifier based on Gaussian generative model is thoroughly investigated.

For simulated data generated from multi-variate Gaussian distribution, adding more features could improve the accuracy if there is enough data to afford the number of features. If there are irrelevant attribute in the data set, the impact of them is not significant when there is enough gap between the mean of relevant and irrelevant attributes. To address the problem brought by irrelevant attributes whose mean is not well separated, introducing CoV method for feature selection helps but only when relevant and irrelevant attributes are distinctive in terms of CoV value. Additionally, fix some value of noise level, the minimum separation is independent on the number of attributes increases. The classifier based on Gaussian generative model fails when there is relatively strong noise level because the assumption that data points are generated from Gaussian distribution become inappropriate. Large number of classes would lead to a worse performance as expected. Finally, if the training set is imbalanced, the classifier can still have great performance as long as the imbalance extent is within the classifier's tolerance.

The performance of the classifier was also measured in real data set, zip code data set. Feature selection in this case has to be done otherwise the covariance matrix could be non-invertible and the classifier is doomed to fail. The accuracy of the classifier on zip code data set could be unsatisfying because the data points are not Gaussian distributed, and it is rare that the attributes are absolutely mutually independent. To improve the performance of the classifier, we may try using more training data or introducing more sophisticated feature selection method.

6. Reference

[1] Bishop, C.M., 2006. *Pattern recognition and machine learning*. Springer Science+ Business Media.

[2] Kubat, M., 2017. *An introduction to machine learning* (Vol. 2). Cham, Switzerland: Springer International Publishing.