12-2020

# Nearest Centroid: A bridge between statistics and machine learning

Manoj THULASIDAS
*Singapore Management University*, manojt@smu.edu.sg

# Nearest Centroid: A Bridge between Statistics and Machine Learning

Manoj Thulasidas
School of Information Systems
*Singapore Management University*
Singapore
manojt@smu.edu.sg

*Abstract*—**In order to guide our students of machine learning in their statistical thinking, we need conceptually simple and mathematically defensible algorithms. In this paper, we present the Nearest Centroid algorithm (NC) algorithm as a pedagogical tool, combining the key concepts behind two foundational algorithms: K-Means clustering and K Nearest Neighbors (k-NN). In NC, we use the centroid (as defined in the K-Means algorithm) of the observations belonging to each class in our training data set and its distance from a new observation (similar to k-NN) for class prediction. Using this obvious extension, we will illustrate how the concepts of probability and statistics are applied in machine learning algorithms. Furthermore, we will describe how the practical aspects of validation and performance measurements are carried out. The algorithm and the work presented here can be easily converted to labs and reading assignments to cement the students' understanding of applied statistics and its connection to machine learning algorithms, as described toward the end of this paper.**

***Keywords—statistical thinking, applied statistics, machine learning, nearest centroid, k-means clustering, k nearest neighbor***

## I. Introduction

In teaching the foundations of data analytics, we usually introduce K-Means clustering [1] as the first unsupervised algorithm and K Nearest Neighbors (k-NN) [2] as the introductory supervised one because they are both conceptually simple, and easy to explain and understand. However, while teaching it, we do not get to illustrate the interconnections between them and the statistical reasoning underpinning it. For instance, K-means clustering is usually taught in terms of the simplified heuristic approach [3], which makes it much more computationally efficient, but hides its mathematical foundations.

In K Nearest Neighbor (k-NN) classification [2], the class prediction of a new observation is performed based on how close it is to the existing observations, using a majority voting scheme. In the regression mode, k-NN is used to predict a numeric value as a simple (or a distance-weighted [4]) average of a predefined number of existing observations. This simple and intuitively obvious voting scheme as taught in the classrooms again masks the statistical concepts that we could be reinforcing. The importance of highlighting the statistical thinking aspects with real-world experience [5] and of taking a wholistic approach [6] has been underscored in recent studies.

If the assumptions behind K-Means clustering hold true, the centroids the observations belonging to a class can be viewed as a representative proxy for them. We can therefore use the centroids instead of the nearest neighbors for the purpose of classification [7]. Such an approach was reported [8] for a specific text-mining application, as well as for the diagnosis of cancer types [9]. The nearest centroid approach also figures in the vector space models in text analytics classification, where it is referred to as the Rocchio classifier.

In this article, we treat the Nearest Centroid (NC) algorithm as a pedagogical tool, going into a theoretical exploration of the nature of the underlying probability distributions, and considering the correlations between the variables to compute the expected probabilities of classification. Such an exploration brings into focus a variety of concepts from probability and statistics [10], perfectly illustrating how they influence and shape the algorithm. For completeness, we also describe our implementation of the NC algorithm and study its performance on several data sets. We then establish that the probability estimate is accurate and reproducible, providing an alternative explanation of Quantile-Quantile plots and their verification.

Targeted at computer science and information systems students, we believe this topic will reinforce their understanding and kindle their interest in connecting the concepts in applied statistics to practical algorithms in machine learning and real-world data analysis [11]. Therefore, after a comparative study with k-NN, we will list the statistical and analytics concepts covered in this pedagogical topic. These concepts will translate directly to the teaching objectives of the courses incorporating the NC algorithm as a topic. We will also highlight some formative assessment ideas for such courses.

## II. Background and Notations

In describing the NC algorithm, we will use notations similar to that of K-Means and k-NN algorithms. For this reason and to provide a sound basis for further discussion, we formally describe both these algorithms in some detail.

### *A. K-Means Clustering*

In K-Means clustering, we start with $n$ observations along $p$ variables, $\vec{x_i} \in \mathbb{R}^p$. We would like to make $K$ clusters such that each observation belongs to one and only one cluster. We denote the Euclidean distance between two observations $\vec{x_i}$, $\vec{x_j}$ as $D(\vec{x_i}, \vec{x_j}) = \sqrt{(\vec{x_i} - \vec{x_i})^T(\vec{x_j} - \vec{x_j})}$.

Each observation is assigned to one of $K$ clusters. Therefore, in principle, all possible combinations (of observations to cluster assignments) should be considered, and the centroids and distances recomputed. Because of this combinatorial explosion, finding the global minimum in K-Means clustering algorithm is an NP-Hard problem [12], and slow even when optimized [13]. There have been some attempts in simplifying the algorithm by balancing efficiency vs. interpretability [14], which is further mathematically expanded in [15]. The simplified heuristic approach [3] makes it much more computationally efficient, albeit with the limitation that it may not converge to the global minimum. For our purposes in this work, what is more important than the actual workings of K-Means algorithm are the notions of centroids and their errors, as described below.

*Centroids and Error:* After forming the $K$ clusters, we can define the $j^{\text{th}}$ component of the $k^{th}$ centroid as the mean of the $j^{\text{th}}$ component of all the data points in the $k^{th}$ cluster: $\mu_{k_j} = \mathbf{Mean}\{x_{ij} : \hat{g}_i = k\}$, where $\hat{g}_i$ represents the cluster membership of the $i^{\text{th}}$ observation. Similarly, we can define the standard deviations for each cluster centroid for each variable: $\sigma_{k_j} = \mathbf{StdDev}\{x_{ij} : \hat{g}_i = k\}$. The summations in the definitions (of $\overrightarrow{\boldsymbol{\mu}_k}$, $\overrightarrow{\boldsymbol{\sigma}_k} \in \mathbb{R}^p$) are over the $n_k$ members of each cluster. We will be using these basic concepts of centroids and their standard deviations (or, more generally, their covariance matrices) in order to develop the NC algorithm.

## B. K Nearest Neighbors

We will follow notations similar to the one we used for K-Means clustering to formally state the k-NN algorithm. We have $n$ data points (or observations) along $p$ variables for the *training* set ($\overrightarrow{\boldsymbol{x}_i} \in \mathbb{R}^p$). For the $n$ observations, we also have the labels, or their true class memberships, $g_i$. The training phase of the k-NN algorithm merely stores the $n$ points and their labels in memory. Only when it is asked to classify a new observation, does k-NN perform the necessary computations. In other words, it is a lazy algorithm.

To classify a new observation $\overrightarrow{\boldsymbol{x}}'$, the original k-NN algorithm [2] computes its Euclidean distances between $\overrightarrow{\boldsymbol{x}}'$ and each of the $n$ training observations. The algorithm then sorts these $n$ values and considers the smallest $\kappa$ distances and the associated training vectors and labels. The estimated class membership of $\overrightarrow{\boldsymbol{x}}'$ is the most frequent label among the $\kappa$ observations with the smallest distances $\delta_i^{(s)} : \hat{g}' = \mathbf{Mode}(g_1^{(s)}, g_2^{(s)}, \ldots, g_\kappa^{(s)})$ where the superscript $(s)$ indicates sorted entities. $\kappa$ is usually chosen as an odd number, typically small. (We are using the symbol $\kappa$ instead of the traditional $k$ or $K$, which we already used in this article to describe K-Means clustering.)

Modified versions of the algorithm may use other distance measures [16], or distance-weighted count [4], [17], [18]. Changing the weightage essentially counts the training observations that are close to $\overrightarrow{\boldsymbol{x}}'$ more than the ones that are farther away. The k-NN algorithm does not assume any boundary or even separation among the training data points, while K-Means algorithm implicitly assumes that the observations are cleanly separated into spherical clusters in the data space and the separation boundaries are perpendicular bisectors, which are subspaces of dimension $p - 1$.

## III. Nearest Centroid Algorithm

In k-NN, as the number of rows in the training data ($n$) increases, all those rows will have to be stored in memory during the training phase because the k-NN model *is* the aggregate of the training data set. During the testing or production phase, k-NN has to compute $n$ distances, and sort them to get the top $\kappa$ candidates with the smallest distances.

In the Nearest Centroid (NC) algorithm for classification described in this paper, instead of storing all $n$ rows of the training data, we will store only as many rows as there are distinct classes ($K$) in the data. The $K$ rows stored are the centroids of the observations belonging to each class.

During the classification phase, we will only need to compute the distances for $K$ (a much smaller number compared to $n$) and find the smallest among them, thereby tremendously improving the memory requirement and performance of the classifier. In addition, the "regression" part of the NC algorithm computes the probability of the right classification.

### A. Training Phase

In our training data set, we have $n$ observations ($\overrightarrow{\boldsymbol{x}_i} \in \mathbb{R}^p$) and their labels, or their true class memberships, $g_i$. Training the NC classifier involves going through the $n$ rows of the training data set and computing the centroids grouped by the class. Exactly as in K-Means clustering, we have the class centroid $\mu_{k_j} = \mathbf{Mean}\{x_{ij} : g_i = k\}$ and their standard deviations $\sigma_{k_j} = \mathbf{StdDev}\{x_{ij} : g_i = k\}$. The only difference is that in NC, we are using the true class label $g_i$ rather than the cluster affiliation $\hat{g}_i$ and that the implied summations in the definitions of **Mean** and **StdDev** run over the $n_k$ observations with the true class label $k$.

In addition, we also compute and store the full covariance matrix $\boldsymbol{\Sigma}_k$ (again, on a per-class basis). For the observations with the class label $k$, the covariance between the variables $x_j$ and $x_m$ is

$$\begin{aligned}\Sigma_{k_{jm}} &= \mathbf{COV}\{(x_{ij}, x_{im}) : g_i = k\} \\ &= \frac{1}{n_k - 1} \sum_{i=1}^{n; g_i = k} (x_{ij} - \mu_{k_j})(x_{im} - \mu_{k_m})\end{aligned} \quad (1)$$

which defines the $p \times p$ covariance matrix $\boldsymbol{\Sigma}_k$.

To summarize, the training phase involves computing and storing three entities per class. With $K$ classes in the data set of $n$ observations along $p$ variables, these entities are:

1. $K$ means of the $p$ variables, $\overrightarrow{\boldsymbol{\mu}_k} \in \mathbb{R}^p$,
2. $K$ standard deviations of the $p$ variables, $\overrightarrow{\boldsymbol{\sigma}_k} \in \mathbb{R}^p$,
3. $K$ covariance matrices, $\boldsymbol{\Sigma}_k \in \mathbb{R}^{p \times p}$ as defined in Eq. (1)

### B. Testing/Production Phase

Using the $3K$ entities defined in the previous section (and stored during the training phase of the NC classifier), we can classify a new observations based on how distant they are from the class centroids.

*1) Euclidean Distance:* The simplest approach would be to assign a new observation $\overrightarrow{\boldsymbol{x}}'$ to the class of the nearest centroid. For this, the NC algorithm computes the $K$ Euclidean distances between $\overrightarrow{\boldsymbol{x}}'$ and $\overrightarrow{\boldsymbol{\mu}_k}$, $\delta_k = D(\overrightarrow{\boldsymbol{x}}', \overrightarrow{\boldsymbol{\mu}_k})$ (instead of the $n$ computations that would be necessary in the k-NN algorithm).

It then finds the class corresponding the smallest of the $K$ distances, and assigns it to the new observation.

$$\hat{g}' = \mathbf{argmin}_k \delta_k \quad (2)$$

which says that the estimated class membership of the new observation is the value of $k$ associated with the shortest distance. This naive approach works reasonably well, but it suffers from its sensitivity to the scale of the variables, as well as the differences in their spreads.

*2) Standard Distance:* In order to address the scale and spread issues, we can consider standard distances (z-scores) between $\vec{x}'$ and $\vec{\mu}_k$, as defined in the equations below.

$$\delta_{k_j} = x'_j - \mu_{kj} \quad z_{k_j} = \frac{\delta_{k_j}}{\sigma_{k_j}} = \frac{x'_j - \mu_{kj}}{\sigma_{k_j}}$$
$$z_k = \sqrt{\sum_{j=1}^{p} (z_{k_j})^2} = \sqrt{\sum_{j=1}^{p} \left(\frac{x'_j - \mu_{kj}}{\sigma_{k_j}}\right)^2} \tag{3}$$

The classification will then proceed as in the case of Euclidean distance, by finding the smallest standard distance.

$$\hat{g}' = \mathbf{argmin}_k z_k \tag{4}$$

While using the standard distance is a definite improvement, it still doesn't fare well when the variables used for classification are highly correlated with each other.

*3) Mahalanobis Distance:* The generalized version of the distance to be used in the presence of correlations is the Mahalanobis Distance[19], which is defined using the inverse of the covariance matrix $\boldsymbol{\Sigma}_k$ as defined in Eq. (1). The Mahalanobis Distance $D_M(\vec{x}', \vec{\mu}_k)$ between $\vec{x}'$ and $\vec{\mu}_k$ can be written as

$$D_M(\vec{x}', \vec{\mu}_k) = \sqrt{(\vec{x}' - \vec{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\vec{x}' - \vec{\mu}_k)} \tag{5}$$

Here, $\boldsymbol{\Sigma}_k^{-1}$ is the inverse of the covariance matrix, encapsulating the correlations among the clustering variables belonging to the $k^{\text{th}}$ cluster. (In our implementation, in the rare cases where the covariance matrix was singular, we used the Moore-Penrose [20] pseudoinverse.)

Under the reasonable assumption that the variables are multivariate normal, $D_M^2$ is a random variable which follows a $\chi^2$ distribution with a parameter (or degrees of freedom, **DoF**) $p$, the number of variables in the data set.

$$\chi_k^2 = \left(D_M(\vec{x}', \vec{\mu}_k)\right)^2 = (\vec{x}' - \vec{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\vec{x}' - \vec{\mu}_k) \tag{6}$$

Note that if the covariance matrix $\boldsymbol{\Sigma}_k$ is diagonal (implying no correlation among the variables), $\chi_k^2$ defined in Eq. (6) reduces to the square of $z_k$ defined in Eq. (3). We have $K$ measures $\chi_k^2$ corresponding to the $K$ class centroids. As before, the classification is performed by locating the minimum among the $K$ Mahalanobis distances for $\vec{x}_i$ and assigning its class as the estimated classification.

$$\hat{g}' = \mathbf{argmin}_k \chi_k^2 \tag{7}$$

## C. NC Classification and Regression

In the rest of this paper, we will be using the minimum Mahalanobis distance for classification. In other words, the class label of a new observation is that of the nearest centroid in terms of (the square of) the Mahalanobis distance, as defined in Eq. (7). We will propose this method as the Nearest Centroid classification algorithm and study its performance on four data sets using certain metrics, such as accuracy and Cohen's Kappa [21].

As mentioned earlier, the square of the Mahalanobis distance of an observation from the centroid of all the observations belonging to the same class has a $\chi_k^2$ distribution of degrees of freedom $\mathbf{DoF} = p$, the number of variables used for classification. Knowing that $\chi_k^2$ has a well-defined probability distribution, we can compute the probability of the *right* classification. We propose the Nearest Centroid "regression" algorithm as the estimate of this probability. If we classify new observations based on this regressed probability, the accuracy of the classifier can be readily predicted. Note, however, that both the probability distribution and indeed the definition of the Mahalanobis distance itself are contingent on the underlying variables following a multivariate normal distribution. From our studies, this assumption seems to be well-supported, as we might expect from the Central Limit theorem.

## IV. Experiments

Now that we have defined our NC algorithm and hinted at the methods to verify the validity of the underlying mathematical assumptions, we proceed to see how they perform in four data sets. We will study the classification accuracy (and other quality metrics) of the algorithm as well as the validity of the assumption of the $\chi_k^2$ probability distribution in the regression mode.

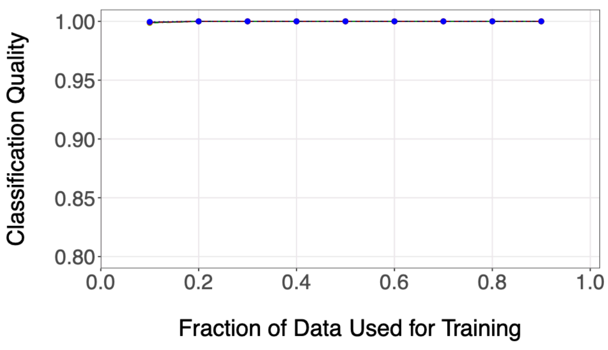
We will pay particular attention to the statistical error on our accuracy measurements and the distribution comparisons. In all the plots that follow, we will consistently display the error bars or bands corresponding to one standard deviation (or, equivalently, a more conservative 68% confidence level) as opposed to the 95% CL commonly found in the literature.

### A. Data Set

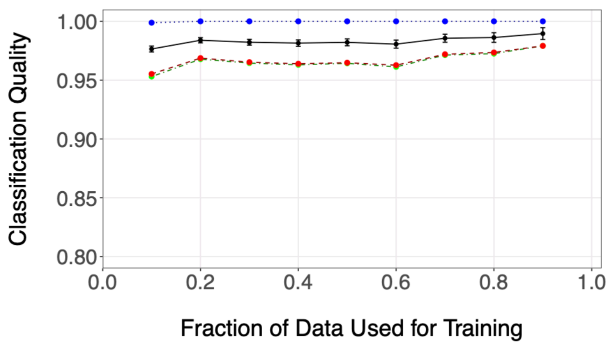
We will use four data sets as described below. For each data set, we will show a plot of the accuracy and other metrics like Cohen's Kappa [21] as we vary the training/testing split. In all the data sets, we have selected the "best" subset of variables to use by directly computing the purity when using various combinations of variables in K-Means clustering.

*1) Iris Data Set:* The famous Iris data set [22] contains 150 flower measurements along four variables (**Sepal Length**, **Sepal Width**, **Petal Length** and **Petal Width**) from three different iris species (**Setosa**, **Versicolor** and **Virgnica**). There are 50 data points for each species. Although the data set has four variables, we use only two of them (**Petal Length** and **Petal Width**) for the studies here because we have identified them as the ones contributing most in separating the three species in terms of purity when clustering using the K-Means algorithm. This historically significant data set is known to be easy to classify, and our NC also algorithm works very well on it, with near perfect accuracy.

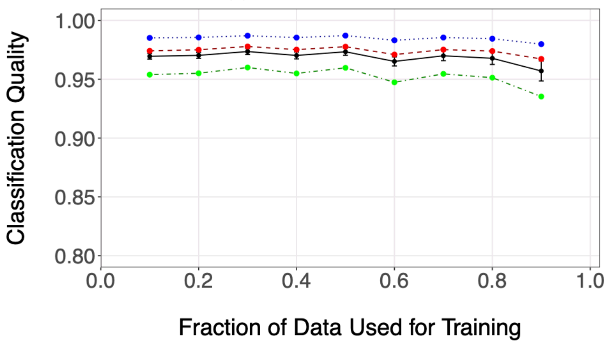
*2) Young Adults Data Set:* This data set is from the anonymous data collected from our students, and contains with 127 observations along four numeric variables (**Height**, **Weight**, **Age** and **HairLength**) and a label (M or F for male or female). Note that in Singapore, male university students are expected to be about two to three years older than their female classmates because of their military service obligation. Therefore, we may expect the Age variable to have some differentiating power while clustering the data.



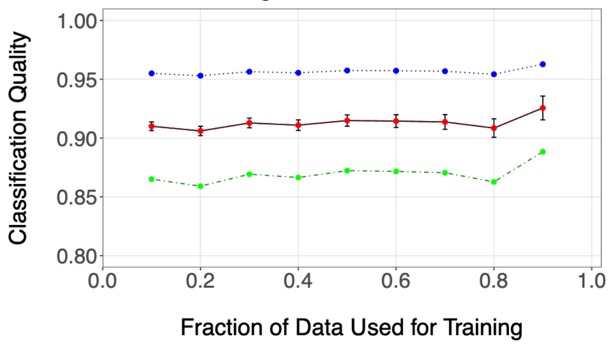
(a) Iris Data Set



(b) Young Adults Data Set



(c) Wine Data Set



(d) Seeds Data Set

Fig. 1. Accuracy (solid line with error bars) and Cohen's Kappa (green dots) of the NC classifier when running on the various data sets, with hold-out data. The x-axis is the fraction of the data used for training. Also shown are the one-vs-all sensitivity (red dashed line) and specificity (blue dotted line).

Again, from our studies, we have identified **Weight** and **HairLength** as the only two variables contributing to the differentiation between the two classes, and used them for the rest of this study.

*3) Wine Data Set:* The Wine data set [23], from the UCI Machine Learning Repository [24] is publicly available and has 12 attributes in three classes. We select four of them (Alcohol, Ash, Flavanoids and OD280_OD315) as the best combination to use in our studies. Note that the number of variables $p = 4$, which also is the DoF for the $\chi^2$ distribution as defined in Eq. (6). Later on, we will intentionally introduce an error by specifying $\boldsymbol{DoF} = 3$ and 5 in order to demonstrate that our theoretical treatment is sound using a negative test.

*4) Seeds Data Set:* Another publicly available resource from the UCI Machine Learning Repository, the Seeds data set [25] contains three classes of wheat seeds with 70 observations each and has seven attributes, of which we select **Area**, **Perimeter**, **Compactness** and **Asymmetry** for our studies.

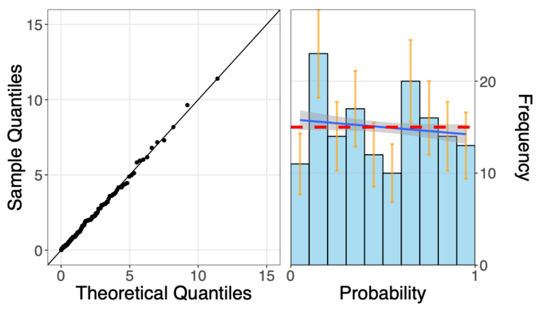
## *B. Accuracy and Other Metrics*

As discussed earlier, during the training phase, the NC algorithm computes the means, standard deviations and the covariance matrices of the variables, grouped by the classes. We do the training using a fraction of the data, and compute the accuracy by testing on the rest of the data. We redo the training-testing cycle using different splits, varying the training fraction from 10% to 90% in steps of 10% as shown in Fig. 1, where we have plotted the accuracy and Cohen's Kappa. In order to reduce the statistical error in the reported numbers, we repeat the measurement 32 times for each split and take the average.

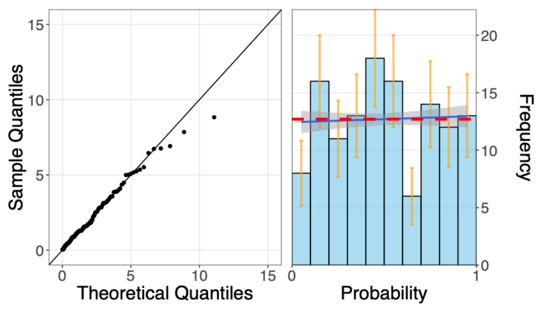
As is well-known [26], accuracy is an incomplete measure of the performance of a classifier. Cohen's Kappa has been shown [27] to be as complete a metric as the area under the curve (AUC) of the Receiver Operating Characteristic (ROC) for binary classifiers, and generalizes well for multi-class problems. The concepts of sensitivity and specificity cannot be consistently defined for multi-class classifiers, but need to be generalized [28] from a binary classifier as one-vs-all. We have included the one-vs-all sensitivity and specificity as well in Fig. 1 for the sake of completeness, while reiterating that Cohen's Kappa is probably a better metric to use. We note, however, that it is not universally accepted [29] as a perfect measure, especially when a class imbalance is expected, which is not the case in our experiments. Both by Cohen's Kappa and by one-vs-all metrics, NC classification gives remarkably high and stable quality metrics even with very small training fractions.

In order to estimate the error in the accuracy measurements, we consider the classification process to be a Bernoulli trial (with the single trial probability equal to the classification accuracy $a$, and number of trials $N$). The standard error on the estimate of the accuracy $\hat{a}$ is

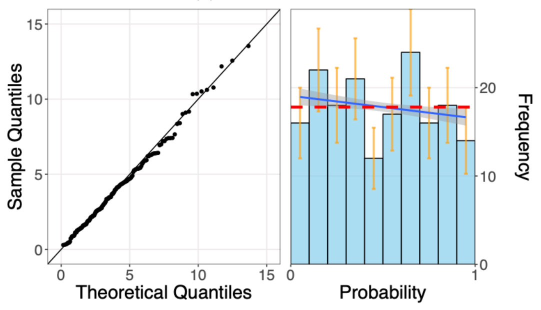
$$\sigma_{\hat{a}} = \sqrt{\frac{\hat{a}(1-\hat{a})}{N}} \tag{8}$$



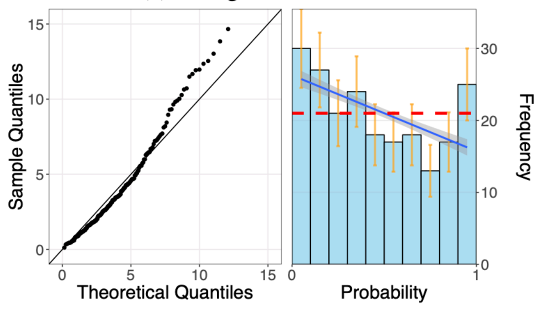
(a) Iris Data Set



(b) Young Adults Data Set



(c) Wine Data Set



(d) Seeds Data Set

Fig. 2. Verifying the distributional assumption ($\chi 2$ of $\mathbf{DoF} = p$) in various data sets. The left pane in each subfigure shows the agreement between the theoretical quantiles and the observed ones, showing remarkable agreement. The right pane shows the "probability histogram" (with one-$\sigma$ error bars), which is expected to be flat. The horizontal dashed line is the flat (average) line expected. The solid line with the error band (one-$\sigma$) is the best fitted line. The flatness is well within the expected limits.

For each point in the accuracy curves in Fig. 1, the number of trials $N$ would be equal to the fraction of the data used for *testing* times the data set size multiplied by the number of folds, which is 32. Note that Eq. (8) holds true only when $\hat{a}$ is away from 0 or 1. Since our estimated accuracies are close to 1, we have verified that the error estimates are correct by repeating the process multiple times and studying the spread of each point in the plots.

## C. Mathematical Verifications

The square of the Mahalanobis distance, as defined in Eq. (6), being the sum of squares of multivariate normal variables, should follow a $\chi^2$ probability distribution. We verify this assertion on our data sets directly, as described below.

1. For each class in each data set, compute the centroids ($\overrightarrow{\boldsymbol{\mu}_k}$) and covariance matrix $\boldsymbol{\Sigma}_k$ (as specified in Eq. (1)). This calculation needs to be performed only once.
2. For each observation in the data set, compute its Mahalanobis distance from the class centroid, as shown in Eq. (5).
3. Compare the distribution of $\chi_k^2$ with the theoretically predicted one using a quantile-quantile plot.

Although quantile-quantile plot is the standard way of comparing two distributions, we also use a "probability histogram" method to redo the comparison as described below.

4. Compute the probability of the Mahalanobis distance (based on a $\chi^2$ distribution of $\mathbf{DoF} = p$).
5. Plot the distribution of the probability for the *right* classification as a histogram.
6. If the assumed theoretical distribution is correct, the probability histogram should be flat between 0 and 1.

Statement (6) above is trivially true because the frequency distribution of any random variable is expected to have the same shape as its probability density function (PDF). For example, if we generate $n$ samples of a normally distributed random variable ($Z_j, j = 1 \dots n$) with a PDF: $N_{(\sigma,\mu)}(z)$, then compute the $n$ probability values ($p_j = N_{(\sigma,\mu)}(Z_j)$), the $p_j$s are expected to have a flat distribution from 0 to 1.

We use this "probability histogram" method also to ensure the rightness of our assumption about the normality of the variables, because we can calculate the statistical errors in the bin count assuming that it comes from a Poisson distribution, where the variance equals the mean. Considering the bin counts to be unbiased estimates of the means, their standard deviations are merely the square-root of the frequency (when non-zero) in the bin. In the probability histograms, we will also overlay a linear regression line, again with a 68% confidence level (or one-$\sigma$) error band.

As shown in Fig. 2, the distributional behavior of the Mahalanobis distance (leading to the $\chi^2$ distribution of $p$ degrees of freedom) is very well-supported in all the data sets we study. For the Seeds data set (Fig. 2(d)), there seems to be a disagreement in the quantile-quantile plot (and lack of flatness in the probability histogram), but the difference is statistically insignificant. The flatness in the probability histogram, for instance, is within the one-$\sigma$ error bar for seven out of ten bins, and well within two-$\sigma$ for all ten.

## D. Monte Carlo Simulation

In addition to establishing the validity of the distributional assumptions in the data, we can also use Monte Carlo simulation techniques to verify it. Although there is no extra value in using the simulation (compared to direct mathematical verification), we do it so that we can run through the whole chain of training and testing, as we describe below. Moreover, using simulation, it is possible to predict the (one-vs-all) sensitivity and specificity when using the classifier with a specified **p-value** criterion.

Starting from $p$ independent standard normal variables, we can generate $p$ multivariate normal random variables with a specified means $\overrightarrow{\boldsymbol{\mu}_k}$and covariance matrix $\boldsymbol{\Sigma}_k$. It involves Cholesky or eigenvalue decomposition of the covariance matrix and a few matrix operations, which are all encapsulated in a straightforward function call in most statistical tools. In this study, we use the `mvrnorm` from the `MASS` package [30] in R. We generate 3000 rows of data for all four data sets we study, ensuring that we have the same proportion of observations in each class in the Monte Carlo as we have in the real data. We will simulate the data and use it for further verifications as described in the steps below.

1. For each class in the data set, compute the centroids ($\overrightarrow{\boldsymbol{\mu}_k}$) and covariance matrix $\boldsymbol{\Sigma}_k$ (as specified in Eq. (1)).
2. Generate simulated observations distributed according to the means $\overrightarrow{\boldsymbol{\mu}_k}$ and covariance matrices $\boldsymbol{\Sigma}_k$ (multivariate normal distributions) with class labels in the same proportion as found in the data set.
3. Train and build the Nearest Centroid (NC) model on the simulated data set, which involves storing the centroids and covariance matrices of all $K$ classes as described in **Training Phase** (Section III-A). Note that the centroids and covariance matrices are expected to be the same as the ones computed in step 1, but subject to statistical fluctuations.
4. Test the NC model on the data set, which involves computing the Mahalanobis distance ($\chi_k^2$) as defined in Eq. (6) for each observation in the simulated data, from the centroid of all the observations with its *true* class label.
5. Compare the distribution of $\chi_k^2$ with the theoretically predicted one using a quantile-quantile plot, and as probability histogram.

We have created such plots and compared to the ones in Fig. 2 and found them to be consistent. This verification is left as a suggested assignment when the NC algorithm is used as a pedagogical topic in a course.

## E. Verification by Negative Test

Since our results in all four data sets look exceptionally good (in the quantile-quantile plots and probability histograms in Fig. 2), one might be justified in suspecting that there is some systematic or method-related reason for the positive findings. To show that the results are not an accident, we perform a negative test with an intentional error. In the Wine data set, where the number of variables used, $p = 4$, the **DoF** for the $\chi^2$ distribution is expected to be 4. We redo our quantile-quantile plots and probability histograms with $\mathbf{DoF} = 3$ and 5.

As can be seen in Fig. 3, both these plots show the characteristic skewing expected of a wrong assumption of the underlying probability distribution. When compared to the "right" plots (Fig. 2), we can conclude that our theoretical foundation in assuming that the probability distribution is a $\chi^2$ of degrees of freedom $\mathbf{DoF} = p = 4$ is sound.

## F. Accuracy: Measured vs. Predicted

The knowledge of the probability distribution of the classifying distance gives as a direct handle on the accuracy of the NC classifier. We can, for instance, apply a 95% confidence level criterion on the probability and be certain of obtaining 95% of the new observations correctly classified. In this section, we verify this assertion by measuring the accuracy and comparing it against the prediction as shown in Table I.

TABLE I. ACCURACY VS. PROBABILITY

| p-value | Data Set | | | |
|---|---|---|---|---|
| | *Iris* | *YA* | *Wine* | *Seeds* |
| 0.1 | 0.08 | 0.8 | 0.9 | 0.9 |
| 0.2 | 0.19 | 0.18 | 0.20 | 0.18 |
| 0.3 | 0.27 | 0.26 | 0.31 | 0.29 |
| 0.4 | 0.46 | 0.61 | 0.59 | 0.59 |
| 0.5 | 0.46 | 0.49 | 0.50 | 0.47 |
| 0.6 | 0.53 | 0.61 | 0.59 | 0.59 |
| 0.7 | 0.67 | 0.68 | 0.68 | 0.67 |
| 0.8 | 0.78 | 0.77 | 0.76 | 0.75 |
| 0.9 | 0.88 | 0.87 | 0.87 | 0.87 |

Predicted vs. measured accuracy of the NC classifier in the four data sets when using classifying criterion $p <$ **p-value**, showing good agreement between the prediction and the measurement.

We define our NC classifier as giving a positive result when the computed probability is less than the **p-value** specified. The predicted accuracy is therefore the **p-value**, and the table shows that the measured ones closely track the prediction, although we can detect a weak trend that the measurement is slightly lower than the prediction. The statistical error on the accuracy as defined in Eq. (8) is about 0.01 for each measurement. This agreement between the predicted (**p-value**) accuracy and the measured one is not a surprise, given the flatness of the probability histograms in Fig. 2.

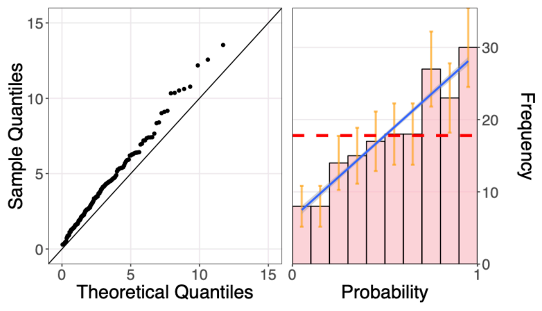
## G. Nearest Centroid vs. k-NN

One of the main motivations behind the NC algorithm for classification is the promise of performance improvement in terms of computing time and memory usage. In this section, we present the results using the Iris data set for the timing studies, with the number of observations replicated to increase the load. We compare the training and classifying times taken by NC and k-NN for 300, 3000 and 6000 observations in the data set, and report them in Table II.

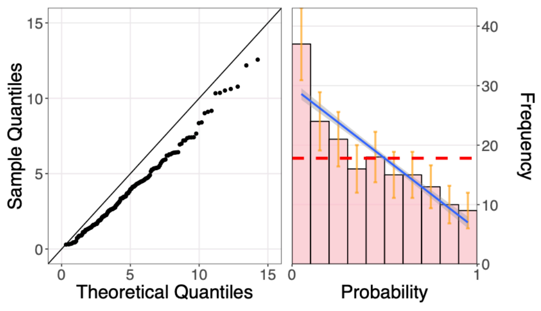
TABLE II. TIMING COMPARISON

| $n$ | Nearest Centroid | | k-NN | |
|---|---|---|---|---|
| | *Train* | *Classify* | *Train* | *Classify* |
| 300 | 4.9 | 3.1 | 1.2 | 4.0 |
| 3000 | 7.0 | 21.2 | 1.5 | 87.7 |
| 6000 | 8.9 | 43.2 | 1.9 | 312.2 |

Median training and classifying computing time usage (in milliseconds) by the NC algorithm and k-NN for various $n$.



(a) Negative Test with $p = 3$



(b) Negative Test with $p = 5$

Fig. 3. Verifying the distributional assumption ($\chi 2$ with **DoF** = 4) using a negative test, with an intentional error in **DoF** in the Wine data set. (a) **DoF** = 3, instead of 4. (b) **DoF** = 5, instead of 4. Left pane: quantile-quantile plot. Right pane: "probability histogram." The agreement in the q-q plot and the flatness in the probably histogram are gone, as expected.

As expected, k-NN classification time explodes as $n$ increases, while NC shows more moderate increase. However, it takes longer to build an NC model because of the extra calculations needed. We do not report the improvements in the amount of memory used, which seem trivially obvious, but hard to measure accurately.

We also compare the quality of the classification for NC and k-NN in Table III. We can see that in data sets where the classes are not well-separated such as Wine or Seeds, k-NN will give better results than the NC algorithm. Our focus in this work is on the viability of the NC algorithm as a pedagogical tool on data sets that cluster well under the K-Means algorithm, not its supremacy in all possible data sets or over other classification algorithms.

TABLE III. COMPARISON OF ACCURACY AND COHEN'S KAPPA

| Data | Nearest Centroid | | k-NN | |
|---|---|---|---|---|
| | *Accuracy* | *Kappa* | *Accuracy* | *Kappa* |
| Iris | 100% | 1.000 | 100% | 1.000 |
| YA | 98.8% | 0.976 | 98.1% | 0.961 |
| Wine | 96.9% | 0.953 | 99.8% | 0.996 |
| Seeds | 91.2% | 0.868 | 98.2% | 0.972 |

Comparison of the accuracy and Cohen's Kappa of NC and the four data sets for 70:30 split.

Although the NC classifier does not outperform k-NN in terms of accuracy, its conceptual simplicity is likely to inspire the students to explore and develop the ideas further.

## V. CONCEPTS AND ASSIGNMENTS

### A. Key Concepts

In addition to the formal descriptions of K-Means and k-NN algorithms, the NC algorithm presented in this article brings a host of concepts and practices that can be taught in classrooms in an experiential learning framework.

**Distance Measures**: In defining the NC algorithm, we move from the Euclidean distance to Standard (z-scores) and then to Mohalanobis (using the data covariance matrix) distance measures.

**Quantification of Classifier Performance**: We use multiple measures (accuracy, Cohen's Kappa, one-vs-all sensitivity and specificity) in assessing the NC classifier. Although we do not describe the underlying confusion matrix, it may be discussed before or while teaching this subtopic.

**n-Fold Cross Validation**: In quantifying the performance of the NC classifier (Fig. 1 and Table I), we use multiple folds and training fractions. The understanding of this crucial technique of improving the accuracy of validation using multiple folds can be further reinforced by class exercises and assignments.

**Standard Error Estimates**: We describe how the standard errors on the accuracy are estimated (based on Binomial distribution) in Fig. 1 and on the bin frequencies (based on Poisson distribution) in the "probability histograms" in Fig. 2.

**Probability Distributions**: In addition to the Binomial and Poisson distributions for error estimates, we touch upon the $\chi^2$ distribution for the Mohalanobis distance.

**Quantile-Quantile Plots**: We use QQ plots to compare the theoretical and observed distributions in Fig. 2.

**Verifying Distributional Assumptions**: A new "probability histogram" approach is developed in Section IV-C, which further reinforces the concept of probability distributions and error estimation.

**Monte Carlo Simulation**: In Section IV-D, we provide step-by-step instructions on how to generate and use synthetic data from a multivariate normal distribution with a specified covariance matrix and means.

**Degree of Freedom**: In Section IV-E, Fig. 3 shows what happens to the QQ plots and the probability histograms when we make an intentional error in the degrees of freedom of the underlying $\chi^2$ distribution, illustrating its importance and estimation.

**p-value**: Table II illustrates the foundational notion of p-value by comparing it to the relative frequency in the data for various values.

**Performance Comparisons**: Tables II and II are tools to teach how two algorithms can be compared.

All these concepts have their origins in applied statistics and are common to a vast array of machine learning algorithms, making the Nearest Neighbor algorithm an ideal starting topic for introductory courses in these two subjects.

## B. *Assignment Ideas*

Based on the key concepts and the discussions in this article, heare are some ideas for formative assessments when the NC algorithm is used as a topic in a course.

- Perform a Monte Carlo simulation to generate synthetic data and reproduce the plots given in Figs. 1 and 2.
- Use the same simulated synthetic data (from the previous item) to verify the error estimates of classification accuracies as defined in Eq. (8).
- What are the assumptions in the NC algorithm? As an amalgamation of K-Means and k-NN algorithms, we expect the NC algorithm to have a superset of the underlying assumptions. How do the assumptions affect its efficiency and applicability?
- What are the limitations of the NC algorithm? These limitations may originate from its assumptions, or from the deviations from the algorithms on which it is based.
- Compare the NC algorithm with other classifiers, such as Decision Trees, Naïve Bayes etc. This assignment may provide a chance for the students to either learn or review other foundational algorithms in an experiential way.

## VI. Conclusion

In this paper, we have described a classification technique termed the Nearest Centroid algorithm, which combines the centroid concept from K-Means clustering and the instance-based, non-parametric learning from the k-NN classifier. It also synthesizes and applies several statistical and theoretical ideas in a theoretically rigorous manner to deepen the students' understanding and appreciation of the mathematical underpinnings of machine learning algorithms.

Developed as a pedagogical topic, appropriate for an advanced undergraduate course on the foundations of machine learning, this algorithm has mathematical properties that are extremely useful in assessing its accuracy, one-vs-all sensitivity, and specificity. Despite its origin in pedagogical reason, the Nearest Centroid algorithm has impressive accuracies as a multi-class classifier, which remain stable even with very small training fractions. These properties may further heighten the students' interest in the algorithm as a viable alternative to more established ones, provided the underlying assumptions hold true in the dataset under study. In the near future, we plan to explore the inclusion of the NC algorithm in our introductory machine learning course as an assignment or lab.

The R code and the datasets described in this article are available from the authors, along with instructions on how to use it. The solutions to the suggested assignments and discussions of the underlying ideas also can be obtained from the authors upon request.

## References

[1] J. A. Hartigan, *Clustering algorithms*. Wiley, 1975.

[2] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, vol. 13, pp. 21–27, 1967.

[3] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. 28, pp. 129–137, 2006.

[4] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Trans. Syst. Man, Cybern.*, vol. SMC-6, pp. 325–327, 1976.

[5] S. C. Hicks and R. A. Irizarry, "A guide to teaching data science," *The Amer. Statistician*, vol. 72, no. 4, pp. 382–391, 2018.

[6] R. J. Brunner and E. J. Kim, "Teaching datas science," *Procedia Comput. Sci.*, vol. 80, pp. 1947–1956, 2016.

[7] C. Manning, P. Raghavan, and H. Schu ̈tze, *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

[8] P. W. Buana, S. D. R. M. Jannet, and K. J. D. Putra, "Combination of k-nearest neighbor and k-means based on term re-weighting for classify indonesian news," *Int. J. Comput. Appl.*, vol. 50, pp. 37–42, 2012.

[9] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Diagnosis of multiple cancer types by shrunken centroids of gene expression," *Proc. Natl. Acad. Sci. United States Amer.*, vol. 99, pp. 6567–6572, 2002.

[10] B. Chance, "Components of statistical thinking and implications for instruction and assessment," *J. Stat. Educ.*, vol. 10, 11 2002.

[11] J. Singer and J. Willett, "Improving the teaching of applied statistics: Putting the data back into data analysis," *Amer. Statistician*, vol. 44, pp. 223–230, 08 1990.

[12] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "NP-hardness of Euclidean sum-of-squares clustering," *Mach. Learning*, vol. 75, pp. 245– 248, 2009.

[13] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *J. Royal Stat. Soc. Series C (Applied Stat.)*, vol. 28, pp. 100–108, 1979.

[14] E. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classification," *Biometrics*, vol. 21, pp. 768–769, 1965.

[15] J. MacQueen, "Some methods for classification and analysis of multi-variate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pp. 281– 297, 1967.

[16] L. Hu, M. Huang, S. Ke, and C. Tsai, "The distance function effect on k-nearest neighbor classification for medical datasets," *SpringerPlus*, vol. 5, 2016.

[17] T. Bailey and A. K. Jain, "A note on distance-weighted k-nearest neighbor rules," *IEEE Trans. Syst. Man, Cybern.*, vol. 8, pp. 311–313, 1978.

[18] J. Gou, L. Du, Y. Zhang, and T. Xiong, "A new distance-weighted k-nearest neighbor classifier," *J. Inform. Comput. Sci.*, vol. 9, 2011.

[19] P. C. Mahalanobis, "On the generalized distance in statistics," *Proc. Natl. Inst. Sci. India*, vol. 2, pp. 49–55, 1936.

[20] R. Penrose, "On best approximate solutions of linear matrix equations," *Math. Proc. Camb. Philosph. Soc.*, vol. 52, pp. 17–19, 1956.

[21] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Meas.*, vol. 20, pp. 37–46, 1960.

[22] R. A. Fisher, "The use of multiple measurements in taxonomic prob-lems," *Ann. Eugenics*, vol. 7, pp. 179–188, 1936.

[23] S. Aeberhard, D. Coomans, and O. de Vel, "Comparison of classifiers in high dimensional settings," Tech. Rep. 92-02, Dept. Comput. Sci. and Dept. Math. and Stat., James Cook Univ. North Queensland, 1992.

[24] D. Dheeru and E. K. Taniskidou, "UCI machine learning repository," Univ. California, Irvine, School Inform. Comput. Sci., 2017.

[25] M. Charytanowicz, J. Niewczas, P. Kulczycki, P. A. Kowalski, S. Łukasik, and S. Zak, "Complete gradient clustering algorithm for features analysis of x-ray images," *Inform. Technol. Biomedicine*, vol. 69, pp. 15–24, 2010.

[26] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inform. Process. Manage.*, vol. 45, pp. 427–437, 2009.

[27] A. Ben-David,"About the relationship between ROC curves and Cohen's kappa.," *Eng. Appl. AI*, vol. 21, pp. 874–882, 2008.

[28] P. Machart and L. Ralaivola, "Confusion matrix stability bounds for multiclass classification," *CoRR*, 2012.

[29] R. Delgado and X. A. Tibau, "Why Cohen's kappa should be avoided as performance measure in classification," *PLOS ONE*, vol. 14, pp. 1–26, 09 2019.

[30] B. D. Ripley, *Stochastic Simulation*. Wiley, 1987.