ChiMerge: Discretization of Numeric Attributes

Randy Kerber
Lockheed Artificial Intelligence Center
O/96-20, B/254F, 3251 Hanover Street
Palo Alto, California 94304
kerber@titan.rdd.lmsc.lockheed.com

Abstract

Many classification algorithms require that the training data contain only discrete attributes. To use such an algorithm when there are numeric attributes, all numeric values must first be converted into discrete values—a process called discretization. This paper describes ChiMerge, a general, robust algorithm that uses the χ^2 statistic to discretize (quantize) numeric attributes.

Introduction

Discretization is performed by dividing the values of a numeric (continuous) attribute into a small number of intervals, where each interval is mapped to a discrete (categorical, nominal, symbolic) symbol. For example, if the attribute in question is a person's age, one possible discretization is: [0...11] → *child*, [12...17] → *adolescent*, [18...44] → *adult*, [45...69] → *middle age*, [70...∞] → *elderly*.

Few classification algorithms perform discretization automatically; rather, it is the user's responsibility to define a discretization and construct a data file containing only discrete values. While the extra effort of manual discretization is a hardship, of much greater importance is that the classification algorithm might not be able to overcome the handicap of poorly chosen intervals. For example, consider the discretization of the attribute *age*, defined above. It is impossible to inductively learn the concept *legal to drink alcohol in California* (where the drinking age is 21) because all ages between 18 and 45 have been mapped to the same discrete value. In general, unless users are knowledgeable about the problem domain and understand the behavior of the classification algorithm they won't know which discretization is best (something they probably expected the classification system to tell them).

Related Work

Although it can significantly influence the effectiveness of a classification algorithm, discretization is usually considered a peripheral issue and virtually ignored in the machine learning literature. Typically, authors of papers describing discrete classification algorithms either apply their systems to purely discrete problems, discretize manually (using expert advice, intuition, or experimentation), or employ a simple method.

The most obvious simple method, called *equal-width-intervals*, is to divide the number line between the minimum and maximum values into N intervals of equal size (N being a user-supplied parameter). Thus, if A and B are the low and high values, respectively, then the intervals will have width $W = (B - A)/N$ and the interval boundaries will be at $A + W, A + 2W, \ldots, A + (N-1)W$. In a similar method, called *equal-frequency-intervals*, the interval boundaries are chosen so that each interval contains approximately the same number of training examples; thus, if $N = 10$, each interval would contain approximately 10% of the examples.

These algorithms are easy to implement and in most cases produce a reasonable abstraction of the data. However, there are many situations where they perform poorly. For example, if the attribute *salary* is divided up into 5 equal-width intervals when the highest salary is $500,000, then all people with salary less than $100,000 would wind up in the same interval. On the other hand, if the *equal-frequency-intervals* method is used the opposite problem can occur: everyone making over $50,000 per year might be put in the same category as the person with the $500,000 salary (depending on the distribution of salaries). With both of these discretizations it would be difficult or impossible to learn certain concepts. The primary reason that these methods fail is that they ignore the class of the training examples, making it very unlikely that the interval boundaries will just happen to occur in the places that best facilitate accurate classification.

Classification algorithms such as C4 [Quinlan *et al.*, 1987], CART [Breiman *et al.*, 1984], and PVM [Weiss *et al.*, 1990] do consider the class information when constructing intervals, but differ in that discretization is performed not as a pre-processing step, but dynamically as the algorithm runs. For example, in C4 (a member of the ID3 [Quinlan, 1986] family of decision tree algorithms) the same measure used to choose the best attribute to branch on at each node of the decision tree (usually some variant of *information gain*) is used to determine the best value for splitting a numeric attribute into two intervals. This value, called a cutpoint, is found by exhaustively checking all possible binary splits of the current interval and choosing the splitting value that maximizes the information gain measure.

However, it is not obvious how such a technique should be used or adapted to perform static (non-dynamic) discretization when more then two intervals

per attribute are desired. [Catlett, 1991] describes one possible extension, called D-2, which applies the above binary method recursively, splitting intervals as long as the information gain of each split exceeds some threshold and a limit on the maximum number of intervals has not been exceeded.

Some classification algorithms can be easily extended to discretize dynamically, but many cannot. Even for algorithms that could use a dynamic method, it might still be preferable to use static discretization. [Catlett, 1991] reports over a 10-fold speed-up (50-fold in one domain) for ID3/C4 when the data is discretized initially using D-2 rather than the standard dynamic approach, with little or no loss of accuracy (and sometimes increased accuracy). The dramatic increase in efficiency is because the dynamic ID3/C4 algorithm must re-discretize all numeric attributes at every node in the decision tree, whereas when D-2 is used each attribute is discretized only once.

Objectives

The search for an improved discretization algorithm was undertaken in order to extend the OTIS classification system [Kerber, 1988][Kerber, 1991] to handle numeric data automatically (OTIS constructs symbolic classification rules from relational, noisy, and multi-class training examples). Initially, the *equal-width-intervals* and *equal-frequency-intervals* methods were implemented and found to be generally effective, but fragile—occasionally producing discretizations with obvious and serious deficiencies, such as with the *age* and *salary* examples described earlier. As a result, it was seldom possible to feel confident that a given discretization was reasonable; a classification algorithm cannot distinguish a non-predictive from a poorly discretized attribute and the user cannot do so without examining the raw data.

Evaluating the quality of a discretization or a discretization algorithm is difficult because it is seldom possible to know what the *correct* or *optimal* discretization is (it would be necessary to know the true distribution of the model from which the data was generated, generally possible only with artificially generated data). Further complicating evaluation is that discretization quality depends on the classification algorithm that will use the discretization; for instance, classification algorithms will differ according to whether they prefer many or few intervals.

While it is not possible to have an optimal discretization with which to compare results, some notion of quality is needed in order to design and evaluate a discretization algorithm. The primary purpose of discretization, besides eliminating numeric values from the training data, is to produce a concise summarization of a numeric attribute. An interval is essentially a summary of the relative frequency of classes within that interval (e.g., if an interval contains 28 positive and 12 negative examples, the interval would be described as 70% positive and 30% negative). Therefore, in an accurate discretization, the relative class frequencies should be fairly consistent within an interval (otherwise the interval should be split to express this difference) but two adjacent intervals should not have similar relative class frequencies (otherwise the intervals should be combined to make the discretization more concise). Thus, the defining characteristic of a high quality discretization can be summarized as: intra-interval uniformity and inter-interval difference.

ChiMerge operationalizes this notion of quality by using the χ^2 statistic to determine if the relative class frequencies of adjacent intervals are distinctly different or if they are similar enough to justify merging them into a single interval. χ^2 is a statistical measure used to test the hypothesis that two discrete attributes are statistically independent. Applied to the discretization problem, it tests the hypothesis that the *class* attribute is independent of which of two adjacent intervals an example belongs. If the conclusion of the χ^2 test is that the class is independent of the intervals, then the intervals should be merged. On the other hand, if the χ^2 test concludes that they are not independent, it indicates that the difference in relative class frequencies is statistically significant and therefore the intervals should remain separate.

ChiMerge Algorithm

The ChiMerge algorithm consists of an initialization step and a bottom-up merging process, where intervals are continuously merged until a termination condition is met. Chimerge is initialized by first sorting the training examples according to their value for the attribute being discretized and then constructing the initial discretization, in which each example is put into its own interval (i.e., place an interval boundary before and after each example). The interval merging process contains two steps, repeated continuously: (1) compute the χ^2 value for each pair of adjacent intervals, (2) merge (combine) the pair of adjacent intervals with the lowest χ^2 value. Merging continues until all pairs of intervals have χ^2 values exceeding the parameter χ^2-*threshold* (described below); that is, all adjacent intervals are considered significantly different by the χ^2 independence test.

The formula for computing the χ^2 value is:

$$\chi^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$$

Where:

m = 2 (the 2 intervals being compared)
k = number of classes
A_{ij} = number of examples in ith interval, jth class.
R_i = number of examples in ith interval = $\sum_{j=1}^{k} A_{ij}$
C_j = number of examples in jth class = $\sum_{i=1}^{m} A_{ij}$
N = total number of examples = $\sum_{j=1}^{k} C_j$
E_{ij} = expected frequency of A_{ij} = $\frac{R_i * C_j}{N}$

The value for χ^2-*threshold* is determined by selecting a desired significance level and then using a table or formula to obtain the corresponding χ^2 value (obtaining the χ^2 value also requires specifying the number of *degrees of freedom*, which will be 1 less than the number of classes). For example, when there are 3 classes (thus 2 *degrees of freedom*) the χ^2 value at the .90 percentile level is 4.6. The meaning of this threshold is that among cases where the class and attribute are independent, there is a 90% probability that the computed χ^2 value will be less than 4.6; thus, χ^2 values in excess of the threshold imply that the attribute and class are not independent. As a result, choosing higher values for χ^2-*threshold* causes the merging process to continue longer, resulting in discretizations with fewer and larger intervals. The user can also override χ^2-*threshold*, if desired, through use of two parameters—*min-intervals* and *max-intervals*—which specify a lower and upper limit on the number of intervals to create (the defaults, 1 and ∞, have no effect). The standard recommended procedure for using ChiMerge would be to set the χ^2-*threshold* at the .90, .95, or .99 significance level and set the *max-intervals* parameter to a value of around 10 or 15 to prevent an excessive number of intervals from being created.

Example

The behavior of ChiMerge will be demonstrated using the well known *iris* classification problem [Fisher, 1936]. The iris database contains 50 examples each of the classes *Iris setosa*, *Iris versicolor*, and *Iris virginica* (species of iris). Each example is described using four numeric attributes: *petal-length, petal-width, sepal-length,* and *sepal-width*. Figure 1 shows the distribution of classes with respect to the *sepal-length* attribute. The numbers on the left are the values of *sepal-length* and the symbols to their right each represent a single instance with that *sepal-length* value, coded as follows: "⋆" = *setosa*, "o" = *versicolor*, and "•" = *virginica*. Blanks lines show the location of interval boundaries chosen by ChiMerge. Similar plots for the other three iris attributes are included as an appendix.

Figure 2 shows both an intermediate and the final discretization when ChiMerge is applied to the *sepal-length* attribute. Each row of the figure represents one interval of the current discretization. The number on the left is the lower bound of that interval; thus, the extent of an interval is from its lower bound up to (but not including) the lower bound of the next interval. The next three numbers in each row comprise the *class frequency vector*, which represents how many examples of each class are in that interval (the order is *setosa*, *versicolor*, *virginica*). The numbers on the right are the result of applying the χ^2 statistic to measure the difference between adjacent intervals; the higher the χ^2 value the greater the belief that the difference between the two intervals is statistically significant. For example, in the interval [5.0→5.5) there are 25 instances

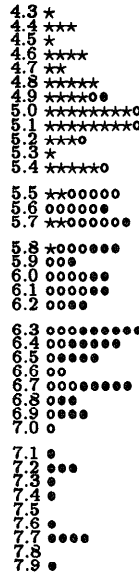


Figure 1: Class histogram for *sepal-length*

of *setosa*, 5 instances of *versicolor*, 0 instances of *virginica*, and a χ^2 difference of 8.6 between it and the following interval. The intermediate discretization (on the left) shows the situation after running ChiMerge with the threshold set at 1.4 (the .50 significance percentile level, which is extremely low). If ChiMerge is now resumed with a higher threshold, on the next iteration the intervals [6.7→7.0) and [7.0→7.1) will be merged since they are the pair of adjacent intervals with the lowest χ^2 score. The second discretization in Figure 2 shows the final result produced by ChiMerge at the .90 significance level ($\chi^2 = 4.6$). The appendix includes the discretizations produced by ChiMerge for the other three iris attributes.

Int	Class frequency			χ^2
4.3	16	0	0	4.1
4.9	4	1	1	2.4
5.0	25	5	0	8.6
5.5	2	5	0	2.9
5.6	0	5	1	1.7
5.7	2	5	1	1.8
5.8	1	3	3	2.2
5.9	0	12	7	4.8
6.3	0	6	15	4.1
6.6	0	2	0	3.2
6.7	0	5	10	1.5
7.0	0	1	0	3.6
7.1	0	0	12	

Int	Class frequency			χ^2
4.3	45	6	1	30.9
5.5	4	15	2	6.7
5.8	1	15	10	4.9
6.3	0	14	25	5.9
7.1	0	0	12	

Figure 2: ChiMerge discretizations for *sepal-length* at the .50 and .90 significance levels ($\chi^2 = 1.4$ and 4.6)

Empirical Results

Because ChiMerge is not itself a classification algorithm it cannot be tested directly for classification ac-

curacy, but must be evaluated indirectly in the context of a classification algorithm. Therefore, ChiMerge, D-2, and the *equal-width-intervals* and *equal-frequency-intervals* algorithms were used to create intervals for the Back Propagation neural network classifier [Rumelhart and McClelland, 1986]. Back Propagation was chosen primarily because it is widely known, thus requiring no further description. The results shown were obtained using 5-fold cross validation testing in the *glass fragment* classification domain and 2-fold cross validation with the numeric attributes of the *thyroid* domain[1]. The glass data consists of 214 examples divided into 6 classes and described in terms of 9 numeric attributes. The thyroid data contains 3772 examples, 7 classes, and 6 attributes (the discrete attributes were ignored). For the *equal-width-intervals*, *equal-frequency-intervals*, and D-2 algorithms, the value of N (the number of intervals) was set to 7. For ChiMerge, the χ^2-threshold was set at the .90 significance level and constrained to create no more than 7 intervals. The results have a standard deviation of 0.5% for the glass data and 0.2% for the thyroid data; thus, the improvement of ChiMerge and D-2 over the simple methods is statistically significant.

Algorithm	Glass ($\sigma = 0.5\%$)	Thyroid ($\sigma = 0.2\%$)
ChiMerge	29.8 %	9.8 %
D-2	30.2 %	10.2 %
Equal-width-intervals	33.3 %	18.3 %
Equal-frequency-intervals	35.7 %	16.4 %

Figure 3: Error rates

Discussion

A very important characteristic of ChiMerge is robustness. While it will sometimes do slightly worse than other algorithms, the user can be fairly confident that ChiMerge will seldom miss important intervals or choose an interval boundary when there is obviously a better choice. In contrast, the *equal-width-intervals* and *equal-frequency-intervals* methods can produce extremely poor discretizations for certain attributes, as discussed earlier. Another feature is ease of use; while discretization quality is affected by parameter settings, choosing a χ^2-*threshold* between the .90 and .99 significance level and *max-intervals* to a moderate value (e.g., 5 to 15) will generally produce a good discretization (some qualifications are discussed later). A major source of robustness is that unlike the simple methods, ChiMerge takes the class of the examples into consideration when constructing intervals and adjusts the number of intervals created according to the characteristics of the data. In addition, ChiMerge is applicable to multi-class learning (i.e., domains with more than two classes—not just positive and negative examples). Another benefit of ChiMerge is that it provides a concise summarization of numeric attributes, an aid to increasing human understanding of the relationship between numeric features and the class attribute.

One problem with ChiMerge is a tendency to construct too many intervals due to the difficultly in distinguishing between true correlations and coincidence. The role of the χ^2 statistic is to help determine whether the difference in relative frequencies between adjacent intervals reflects a real relationship between the numeric attribute and the class attribute or is the result of chance. The χ^2-*threshold* parameter presents a trade-off. Setting higher values reduces the likelihood of false intervals; however, if the threshold is set too high, intervals representing real phenomena will also be eliminated. In general, it's probably not very harmful to have a few unnecessary interval boundaries; the penalty for excluding an interval is usually worse, because the classification algorithm has no way of making a distinction that is not in the data presented to it (such as occurs in the drinking age example in the introduction). To illustrate the difficulty of avoiding spurious intervals, a simple test was conducted in which randomly generated data was given to ChiMerge. Therefore, ideally, ChiMerge should not produce any interval boundaries, since any found are known to be false. However, the χ^2-*threshold* parameter generally had to be set to a very high value (above the .99 significance level) to force it to eliminate all intervals.

While the χ^2 statistic is general and should have nearly the same meaning regardless of the number of classes or examples, ChiMerge does tend to produce more intervals when there are more examples. One reason is that when there are more examples there are simply more opportunities for coincidences to occur. Another important factor is that real phenomena will be more likely to pass the significance test as the number of examples increases. For example, if the true distributions of one region is 80% positive instances and that of a neighboring region is 60% positive, this difference is unlikely to be detected when there are only 10 examples per region, put probably would pass the χ^2 test when there are more than 100 examples per region. This problem is controlled by using the *max-intervals* parameter to place an upper limit on the number of intervals ChiMerge is allowed to create.

One difficulty with using the χ^2 statistic is that it can be unreliable (misleading) when the expected frequency (denoted E_{ij} in the formula) of any of the terms is less than about 1.0. When this occurs, there is a tendency for the resultant χ^2 value to over-estimate the degree of difference. This bias has been partially alleviated by altering the χ^2 formula so that the denominator of every term of the χ^2 formula has a value of at least 0.5 (to avoid dividing by a very small number, which produces an artificially large χ^2 value). This modification, despite its aesthetic shortcomings, seems

[1]Obtained from the University of California-Irvine induction database repository: ml-repository@ics.uci.edu.

to work quite well. In any case, this is usually not an important problem since intervals containing few examples, where this bias is most prevalent, will still result in χ^2 values below the threshold; thus they tend to be absorbed early in the merging process anyway.

Another shortcoming of ChiMerge is its lack of global evaluation. When deciding which intervals to merge, the algorithm only examines adjacent intervals, ignoring other surrounding intervals. Because of this restricted local analysis, it is possible that the formation of a large, relatively uniform interval could be prevented by an unlikely run of examples within it. One possible fix is to apply the χ^2 test to three or more intervals at a time. The χ^2 formula is easily extended, by adjusting the value of the parameter m in the χ^2 calculation.

Computational Complexity

The version of ChiMerge described here has a computational complexity, in terms of the number of times that the χ^2 function is called, of $O(n^2)$, where n is the number of examples[2]. However, implementation of some simple optimizations can reduce the complexity to $O(n \log n)$. One source of increased speed is to be more aggressive about merging when constructing the initial list of intervals. In addition, several pairs of intervals, not near each other, could be merged on each iteration. The χ^2 values could also be cached rather than re-computed each iteration for those intervals not affected. However, the lower bound of ChiMerge is $O(n \log n)$—the complexity of sorting the examples in the initial step of the algorithm—unless some means can be devised to eliminate this step.

Limitations

ChiMerge cannot be used to discretize data for unsupervised learning (clustering) tasks (i.e., where the examples are not divided into classes), and there does not appear to be any reasonable way to extend it to do so. Also, ChiMerge is only attempting to discover first-order (single attribute) correlations, thus might not perform correctly when there is a second-order correlation without a corresponding first-order correlation, which might happen if an attribute only correlates in the presence of some other condition.

Future Work

A variation of ChiMerge that might be interesting to investigate, would be to modify it to create a generalization hierarchy of intervals rather than a fixed partition of the number line. This hierarchy would be the binary tree corresponding to the evolutionary history of the intervals as constructed by the algorithm: the root of the tree would be the entire number line, and for every non-leaf node in the tree its children would be the two intervals that were merged to create it. Such an interval hierarchy could be used by symbolic induction methods (such as INDUCE[Michalski, 1983] and OTIS[Kerber, 1988]) that handle hierarchically defined attributes. The induction algorithm could climb or descend the generalization tree as needed to obtain an interval of the desired specificity. Another advantage is that the χ^2-*threshold*, *min-intervals*, and *max-intervals* parameters would become irrelevant.

[2]a non-optimized version of ChiMerge was defined for reasons of clarity

Summary

ChiMerge is a general, robust discretization algorithm that uses the χ^2 statistic to determine interval similarity/difference as it constructs intervals in a bottom-up merging process. ChiMerge provides a useful, reliable summarization of numeric attributes, determines the number of intervals needed according to the characteristics of the data, and empirical testing indicates significant improvement over simple methods that do not consider the class information when forming intervals.

References

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees.* Wadsworth, Belmont, CA.

Catlett, J. 1991. On changing continuous attributes into ordered discrete attributes. In *European Working Session on Learning.*

Fisher, R. A. 1936. The use of multiple measurements in taxonomic problems. *Ann. Eugenics* 7(2):179–188.

Kerber, R. 1988. Using a generalization hierarchy to learn from examples. In *Fifth International Conference on Machine Learning*, Ann Arbor, MI. Morgan Kaufmann. 1–7.

Kerber, R. 1991. Learning classification rules from examples. In *Workshop Notes of the AAAI-91 Workshop on Knowledge Discovery in Databases.* 160–164.

Michalski, R. S. 1983. A theory and methodology of inductive learning. In Michalski, R. S.; Carbonell, J. G.; and Mitchell, T. M., editors 1983, *Machine learning: An artificial intelligence approach.* Morgan Kaufmann, Los Altos, CA.

Quinlan, J. R.; Compton, P. J.; Horn, K. A.; and Lazarus, L. 1987. Inductive knowledge acquisition: a case study. In Quinlan, J. R., editor 1987, *Applications of Expert Systems.* Addison-Wesley, Sydney. 157–173.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Rumelhart, D. E. and McClelland, J. L. 1986. *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, MA.

Schlimmer, J. 1987. Learning and representation change. In *Sixth National Conference on Artificial Intelligence*, Los Altos, CA. Morgan Kaufmann. 511–515.

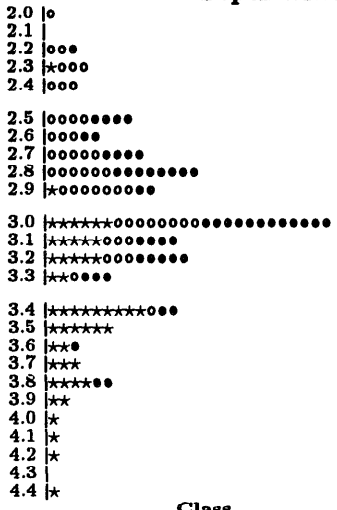
Weiss, S. M.; Galen, R. S.; and Tadepalli, P. V. 1990. Maximizing the predictive value of production rules. *Artificial Intelligence* 45:47–71.

Appendix

The appendix shows ChiMerge discretizations for the iris classification domain (with χ^2 threshold = 4.61, the .90 significance level). In the histogram figures, blank lines show where ChiMerge placed interval boundaries. The numbers on the left represent attribute values and the symbols to their right represent single instances with that particular attribute value, coded as follows: "⋆" = *setosa*, "o" = *versicolor*, and "•" = *virginica*.

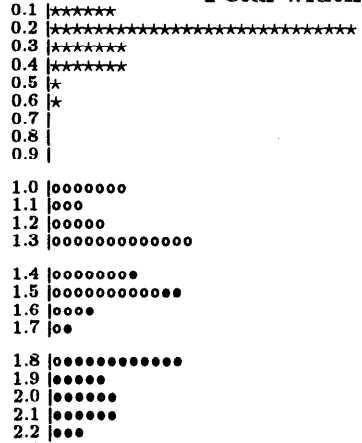
The tables summarize the information in the histograms. Each row represents one interval. The *Interval* column shows the lower bound of each interval. The *Class frequencies* show how many of each class are in that interval (the order is *setosa*, *versicolor*, *virginica*). The χ^2 column shows the result of computing the χ^2 value for each adjacent pair of intervals; the higher the χ^2 value the greater the belief that the difference between the two intervals is statistically significant.

Sepal width



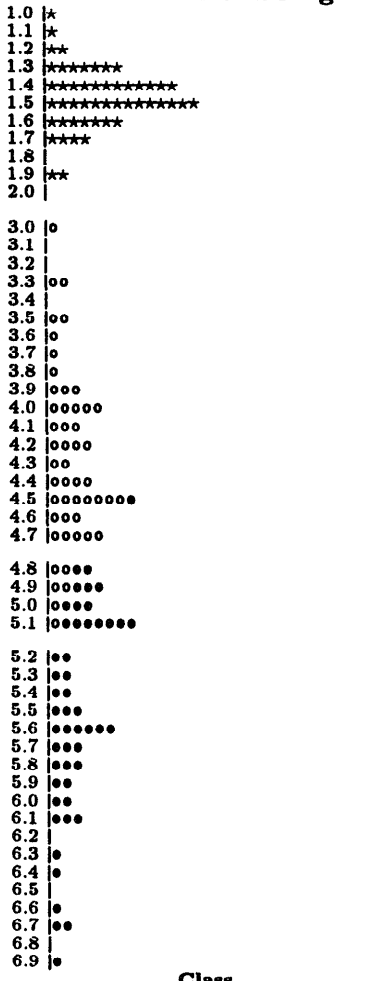
Interval	Class frequencies			χ^2
2.0	1	9	1	5.0
2.5	1	25	20	17.1
3.0	18	15	24	24.2
3.4	30	1	5	

Petal width



Interval	Class frequencies			χ^2
0.1	50	0	0	78.0
1.0	0	28	0	5.9
1.4	0	21	5	48.4
1.8	0	1	45	

Petal length



Interval	Class frequencies			χ^2
1.0	50	0	0	95.0
3.0	0	44	1	37.3
4.8	0	6	15	10.9
5.2	0	0	34	