Evaluating Classifier Combination Using Simulated Classifiers

J.R. Parker
Laboratory for Computer Vision
Department of Computer Science
University of Calgary
Calgary, Alberta, CANADA

Abstract

The use of standard data sets with known properties is standard practice for the evaluation of composite classifiers. However, because the properties of the original test data cannot be specified in advance, it is difficult to conduct controlled tests on classifier combination methods. In these cases it is common to use simulated data, but a standard means for doing this has not evolved. Confusion matrices can also be used to create simulated classifiers having similar properties to the original and, by generating specific confusion matrices and basing a set of simulated classifiers on these, test suites can be designed that have pre-determined properties. This simulation allows controlled testing of classifier combination techniques. Such a scheme is described and evaluated using known combination methods and standard data sets, both to confirm the simulation and to demonstrate its flexibility.

1. Introduction

In the process of designing vision systems and pattern analysis algorithms, it is natural that a variety of techniques be tried, with the overall goal of improving the rate at which patterns can be correctly classified. Many diverse algorithms each have strengths and weaknesses, good ideas and bad. One way to take advantage of this variety is to apply many methods to the same recognition task, and have a scheme to merge the results [1,5]; this should be successful over a wider range of inputs than would any individual method. Indeed, it is what will be demanded of such a multiple classifier system: it must have a higher overall recognition rate than any of the individual classifiers that make it up, or there is no point to the extra effort involved.

Composite classifiers can be made up of similar classifier types (e.g. nearest K neighbor, geometric features) or of quite diverse types (non-homogeneous). In the latter case it is hoped to take advantage of the diversity in the classifier set to achieve an increase in robustness. Classifiers can differ in both the nature of the measurements used in the classification process and in the type of classification produced. A Type I classification is a simple statement of the class, a Type II classification is a ranked list of probable classes, and a Type III classification assigns probabilities to classes [9].

A problem with the evaluation of composite classifiers is the availability of the needed large number of classifications for a valid characterization, along with the associated ground truth. A large data set is not enough; a correct implementation of a collection of classifiers is needed. Even then the nature the classifications is pre-determined, in that there are many variables that simply cannot be controlled. Thus it is that many published combination algorithms have been tried only on relatively few data sets and in relatively few variations. How does a composite classifier behave when more than half of the component classifiers have high correlations? When correlations are small? When one or more classifiers fail on a few classes? It is desirable to promise a set of predictable behaviors under a wide variety of circumstances.

A goal of this work is to provide a flexible, controlled, and reliable way to evaluate and experiment with classifier combination techniques; this will be based on simulated classifiers. Section 2 gives a brief discussion of schemes for generating simulated classifications, focusing on the use of measured properties of classifiers. Section 3 discusses the procedure for simulating and evaluating the multiple classifier combinations using a statistical model of the classifier. Section 4 discusses feature-based models, and section 5 discusses data-based models.

2. Classifier Simulation

One of the problems with the empirical evaluation of classifier combination algorithms is the need for vast amounts of data. In this case, the data consists of classifier output from many different classifiers and a great many individual classification tasks. This means that the input data, with ground truth, must exist, and the classifiers must be correctly implemented. Even if sufficient data can be found, it has certain pre-defined properties that can't be altered. A classifier has a given recognition rate for a given type of data, and it may not be possible to provide all desired combinations of characteristics. For example, a classifier combination algorithm may work well when all individual classifiers operate at over 90% recognition, or when the classifiers have nearly the same rates. It is important to be able to determine benefits and limitations, either from a theoretical basis or empirically.

A classifier simulation will produce any number of classifications for use in evaluating combination methods. These classifications may be based on existing classifiers and data sets, or may have completely user defined properties. There are essentially three ways to produce simulated classifications: by using the measured properties of a real classifier acting on a real data set (classifier model), by producing sets of features that are typical of class instances and then using a real classifier on these (feature model), or by creating virtual class instances that are typical of the classes involved and feeding these through a feature extractor and classifier (data model). All three of these ideas will be briefly examined.

The ensuing discussion will use the well known Iris data set [5] as an example. This set is small, only 150 items, but the point is not to produce new results, but to explain the techniques involved in classifier simulation. There are four attributes in the Iris set, sepal length, sepal width, petal length, and petal width. These are numerical, and represent measurements of physical characteristics of three types of iris, a flower.

3. Classifier Simulation Using Classifier Models

A confusion matrix for a classifier is a matrix in which the actual class or a datum under test is represented by the matrix row, and the classification of that particular datum is represented by the confusion matrix column. The diagonal elements indicate correct classifications and, if the matrix is not normalized, the sum of row I is the total number of elements of class I that actually appeared in the data set. The element M[i][j] gives the number of times that a class i object was assigned to class j.

Consider a classifier that produces only a single output class (Type I) and that has been trained and tested on many thousands of data elements. During the this process, the following confusion matrix was generated (assume there are four classes):

Table 1
Example confusion matrix

	Classified as Class A	Classified as Class B	Classified as Class C	Classified as Class D
Actual class A	5210	159	101	530
Actual class B	320	5090	111	479
Actual class C	110	28	5813	148
Actual class D	210	12	7	5771

Each row sums to 6000, which was the number of elements of each class in the data set. Now as an example, presume that this classifier issues a classification of A for a given input datum. From the first column of the confusion matrix, it can be seen that the most likely actual class is A, the second most likely class is B, followed by D, and finally C. This is a fair ranking of the possible classes based on the past history of the classifier. In other words, given a classification of A:

5210/5850 will be correct (class A)
320/5850 will actually be class B
210/5850 will actually be class D
110/5850 will actually be class C

The confusion matrix will be accurate to the extent that the classifier has been thoroughly trained. Xu [9] used a collection of four classifiers of handprinted digits, and explored the use of Bayesian methodology, Dempster-Shafer theory, and voting to implement a combined classifier that improved the overall recognition rate to 98.9% from 93.9%. He first suggested that a confusion matrix (*a posteriori* probabilities of each classification) of a Type I classifier could be used to create response vector, thus giving numerical values to the degree to which the each possible classification corresponds to the current instance being classified. This procedure can clearly be reversed; given a confusion matrix, random classifications an be generated in the same proportions as those seen by the classifier before. We are, in essence, mining the confusion matrix for information. The assumptions are that, first, the behavior of the classifier is known and is characterized in a confusion matrix and, second, that the prior behavior or the classifier is representative of its future behavior. The larger the data set on which the classifier has been tested, the more thoroughly will the second assumption be true.

3. 1 Simple Classifiers

Let there be **m** possible classes. A real classifier is given an input pattern and produces either a classification, which is an integer in the range 1..**m**, or is unable to classify the input pattern, a situation represented by the classification value -1. A simulated classifier is given the *a posteriori* probabilities of each possible classification, as represented by a confusion matrix, and the actual class of the input; it also produces a classification as before, but based solely on the probabilities in the confusion matrix.

Of course, the confusion matrix does not tell all that can be told about the nature of a classifier. For example, the misclassification of a class A object as class B may result from two different processes, for two quite distinct reasons. The joint probability distribution associated with these two processes may well vary as a function of time, as the nature of the input data changes. The complexity associated with this situation may well be manageable, but as a first approximation the view will be taken that a classifier can be considered to be a single, possibly composite, process having relatively simple measurable properties.

3. 2 Detailed Method

Generating a classification from a confusion matrix is a simple task. Given a confusion matrix M as defined above, first decide what the true class X of the object should be. This can be done completely at random, or by considering the relative frequencies of classes in a population. Then construct a cumulative frequency histogram from the row of the confusion matrix corresponding to X:

```
hist[0] = M[x][0];
for (i=1; i<Nclasses; i++)
    hist[i] = hist[i-1] + M[x][i];
```

The histogram must be normalized by dividing by the sum of the elements in the row (the number of observed instances of the class X). Now draw a random number Z, 0<=Z<1, and scan the histogram for the first (smallest index) value of hist[j] satisfying hist[j] >= Z. The classification generated is j.

3. 3 Experimental Test Bed

As a means of experimenting and evaluating classifier combination techniques, a testing system based on two simulations and an evaluation module was constructed. The first, and simplest, simulation is that of a classifier. This consists of essentially a single function:

simc (C, U, i)

where **C** is the confusion matrix for the classifier, **U** is the set of probabilities that each class is not successfully classified, and **i** is the actual class of the input object. The function returns a classification that is correct, incorrect, or not classified with the probabilities specified in **C** and **U**.

To avoid the necessity of having a large amount of actual classified data, the second portion of the simulation consists of a mechanism for generating confusion matrices with specific properties. The most obvious property to be specified is the recognition rate, but other possibilities might reflect the way in which misclassifications occur. Considering this latter property, and recognizing that there are a great many possible misclassification processes, the classifier generator (**gen*** for future reference) can create three basic types of confusion matrix:

1. A *deterministic* matrix, in which the diagonal elements are each equal to the specified recognition rate and the probability of every misclassification is identical.
2. A *uniform* matrix, in which the diagonal elements combine to give the specified recognition rate, but individually are drawn from a normal distribution with a known mean and standard deviation. The other elements in the matrix are identical by column, summing to 1.0.

3. A *stochastic* matrix, in which the diagonal elements are constructed as in 2 above, and the elements in each column are drawn randomly from a normal distribution with known properties and summing to 1.0.

These types of matrix cause quite different classifier behavior, representing as they do different processes. It is anticipated that the majority confusion matrices that represent actual classification processes would most similar to those of the third type, where the correct classification would have the highest probability and a few misclassifications would be much more likely than the remainder. An example of each type is shown in Figure 1.

The evaluation module implements the classifier combination algorithms and measures the properties of the composite classifiers. The combination techniques under consideration were: a standard Borda count, the APborda method, and the wBorda method, for a variety of weights; a simple majority vote, the sum rule and the median rule. Each evaluation consists of 1000 individual classifications from each of five simulated classifiers merged and compared with the known result. For each set of five simulated classifiers, 1000 evaluations were performed and the results accumulated. Thus, five million simulated classifications were performed for each set of generated classification matrices. This was repeated for each matrix type and over a specified range of recognition rates.

FIGURE 1. The three types of confusion matrix. (a) Deterministic, in this case with a recognition rate of 90%. (b) Uniform, where the recognition rate is the mean value of the non-identical diagonal elements. (c) Stochastic, all elements are random but the columns sum to 100% and the mean of the diagonal is the recognition rate. In all cases, the last row indicates unclassified items and entries are in percent.

90	1	1	1	1	1	1	1	1	1
1	90	1	1	1	1	1	1	1	1
1	1	90	1	1	1	1	1	1	1
1	1	1	90	1	1	1	1	1	1
1	1	1	1	90	1	1	1	1	1
1	1	1	1	1	90	1	1	1	1
1	1	1	1	1	1	90	1	1	1
1	1	1	1	1	1	1	90	1	1
1	1	1	1	1	1	1	1	90	1
1	1	1	1	1	1	1	1	1	90
1	1	1	1	1	1	1	1	1	1

(a)

95	2	.8	1	1.5	.4	.6	1.4	1	0
.5	80	.8	1	1.5	.4	.6	1.4	1	0
.5	2	92	1	1.5	.4	.6	1.4	1	0
.5	2	.8	90	1.5	.4	.6	1.4	1	0
.5	2	.8	1	85	.4	.6	1.4	1	0
.5	2	.8	1	1.5	88	.6	1.4	1	0
.5	2	.8	1	1.5	.4	94	1.4	1	0
.5	2	.8	1	1.5	.4	.6	86	1	0
.5	2	.8	1	1.5	.4	.6	1.4	90	0
.5	2	.8	1	1.5	.4	.6	1.4	1	100
.5	2	.8	1	1.5	.4	.6	1.4	1	0

(b)

86	1	2	2	1	0	1	5	5	4
0	85	0	2	3	0	0	2	1	1
1	1	86	1	1	5	0	0	2	0
0	2	0	80	0	4	0	1	0	0
4	2	0	0	84	0	0	1	0	5
1	1	1	1	1	89	4	0	2	1
1	1	4	3	3	0	74	1	1	0
1	4	0	0	2	0	0	84	0	0
0	0	3	7	1	0	10	2	86	2
3	3	0	0	7	1	5	3	1	87
3	1	4	4	2	1	6	1	3	0

(c)

3. 4 Test Bed Verification

In any complex computer program the issue of correctness can be raised. Simulations, being stochastic by nature, can be more difficult to confirm. One way to do so is to compare some measured results of the simulation against the initial assumptions and verify that there is a high degree of correspondence. Verification of this particular simulation was done in this way also, including:

- **simc** was presented with a series of deterministic, uniform, and stochastic matrices, and the resulting classifications in each case collected into a measured confusion matrix. These agreed with the type and properties of the specifications.
- **gen***, when presented with a type, mean, and standard deviation, yielded confusion matrices that agreed with the specification.
- **gen*** was used to create matrices of given type and properties which were given to simc; the resulting classifications yielded a measured confusion matrix having type and properties that correlated sufficiently well with the specifications.

3. 5 Correlated Classifiers

In practice it is common to find that some of the classifiers in a collection make the same mistakes, or the same sort of mistakes. The behavior of a composite classifier containing correlated classifiers is of practical interest, and so it must be possible to create classifications that have varying degrees of correlation, either overall agreement or for specified classes.

Consider that it is important to have two classifiers agree with each other 95% of the time. This means that the errors they make, as well as their correct classifications, will agree to that amount. The agreement can be achieved in the following way:

1. Create the first classifier, A, in the manner already described.
2. Generate a classification from A, in the manner already described. Call this C0.
3. When generating classifications, classifier B (95% agreement with A) starts with A's classification. Now draw a uniform random number x between 0-1, and emit A's classification of C0 if $x < 0.95$. Otherwise, remove C0 from consideration, and generate a new classification using confusion matrix A and the remaining, re-normalized, probabilities.

It is possible to allow some classes between the two classifiers to be correlated, while having the remaining classes independent, by generating a confusion matrix for B also, and special casing the correlated columns. The result is the ability to generated classifications having all degrees of correlation, for any set of classes, for any recognition rate, number of classes, and number of classifiers. To simulate correlated classifiers, a new module **csimc** (correlated-simc) was devised and tested.

4. Classifier Simulation Using Feature Models

A feature is a measurement of a property of a class instance. Classifiers use these to produce a classification, and the same set of features should always give the same classification. The Iris data set has four features (also called attributes in this context) which are all numeric. It should be possible to characterize the typical values of each feature for each class based on the actual data at hand. Then new feature sets can be created at random, following the characterization.

For example, the Iris data set, mentioned above and seen in Appendix I, has the following statistics:

	Setosa		Versicolor		Virginica	
Feature	Mean	SD	Mean	SD	Mean	SD
1	5.006	0.352	5.936	0.516	6.588	0.636
2	3.418	0.381	2.770	0.314	2.974	0.322
3	1.464	0.174	4.260	0.470	5.552	0.552
4	0.244	0.107	1.326	0.198	2.026	0.275

The simplest way to create a feature set typical of an instance of Iris Setosa is to generate normally distributed random numbers in quads, the first having mean 5.006 and standard deviation 0.352, the second having mean 3.418 and standard deviation 0.381, and so on, using the mean and standard deviation columns from the Setosa portion of the table above. This is a rather common way to create artificial data, and it is considered so simple that little has been published on this subject.

It is likely that some of the features are correlated with one another. The procedure for generating correlated random features is based on determining the covariance matrix of the features in the training set. In the N dimensional case, let this matrix be P and imagine that a vector C of normal random numbers exists. We want a set of correlated random values Y, and can find this by computing a rotation

$$\overline{Y} = A\overline{C}$$

for some matrix A. It turns out that A is the square root (Cholesky decomposition) of P. The process is simply:

1. Compute the covariance (or correlation) matrix P.
2. Compute A such that P = A'A (Cholesky decomposition)
3. Draw an N dimensional vector C of normal random numbers.
4. Multiply C by A, giving the N correlated values Y.

This is a standard mechanism for simulating multivariate normal observations.

5. Classifier Simulation Using Data Models

It is also possible to base a simulation on a model of the data, which is after all the source of the feature set. This is a rather difficult task - the idea is to construct an accurate model of a class instance, whatever it might be. In the case of the Iris data set, we're talking about generating a flower, a rather tall order. However, there are cases where a data model might reasonably be used. Historically, when training a neural network, it is not unusual to add noise to training data to create a new training instance. This is use of a data model.

Data models are quite specific to the type of data concerned, and generalization is pretty much impossible. However, as one example of this idea consider a set of digitized handprinted digits; these are bi-level images of small dimension. Variation can be introduced in a number of ways: random rotation, slant, and scale (within small limits), adding pixels to boundaries or removing them, and combining two digit images after alignment to form a new one. Examples of this appear in Figure 2. Each of the images are then passed to the feature extraction stage, and thence to the classifier.

There are many ways in which a data model can go wrong, and it is not recommended that this technique be used for any but the simplest data sets.

6. Conclusions

We have discussed, in a general way, the types of classifier simulation that might be constructed, and have made some measurements concerning the accuracy of simulated data and simulated classifications. Simulations can be run using as much simulated data as desired. The simulated classifications can be tailored to have specific properties, or can be modeled after real sets of classifications.

The use of simulated classifiers requires further energy and study. The degree to which simulation can be used to evaluate composite classifiers depends on the degree to which the simulated classifiers can be shown to be representative of the real thing. The degree of control available to the experimenter using the simulation makes it an extremely attractive solution.

7. Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

FIGURE 2. Introducing variation into image data, in this case a handprinted digit '3'. (a) Original glyph (b) Random margin offsets (slants). (c) Increasing black boundary at random. (d) Merging two similar glyphs into one.

(a) (b) (c) (d)

8. References

[1] Baraghimian, G.A., Klinger, A., "Preference Voting for Sensor Fusion", *SPIE Sensor Fusion III*, Orlando, Fl. April 19-20, 1990.

[2] Jean-Charles de Borda, "Memoire sur les Elections au Scrutin", *Histoire de l'Academie Royale des Sciences*, Paris, 1781.

[3] Marquis de Condorcet, "Essai sur l'application de l'analyse a la probabilite des decisions rendues a la pluralite des voix" (Essay on the Application of Analysis to the Probability of Majority Decisions), Paris, 1785.

[4] P. Fishburn, "Preference Structures and Their Numerical Representations", ORDAL'96, Ottawa, Aug 5-9, 1996.

[5] Fisher, R.A., "The Use of Multiple Measurements in Taxonomic Problems", Annual Eugenics 7(Part II): 179-188. 1936.

[5] F. Kimura and M. Shridhar, "Handwritten Numeral Recognition Based on Multiple Algorithms", *Pattern Recognition*, Vol. 24, No. 10, 1991. pp 969-983.

[6] J. Kittler, M. Hatef, and R.P. Duin, "Combining Classifiers", Proceedings of the ICPR, 1996. pp 897-901.

[7] J.R. Parker, "Voting Methods for Multiple Autonomous Agents", Proc. ANZIIS'96, Perth, Australia, 1996.

[8] P.D. Straffin, Jr., "Topics in the Theory of Voting", Birkhauser, Boston, 1980.

[9] L. Xu, A. Krzyzak, and C.Y. Suen, "Methods of Combining Multiple Classifiers and their Applications to Handwriting Recognition", *IEEE Trans. SMC*, vol. 22, No. 3, 1992. pp 418-435.

[10] K. Tumer and J. Ghosh, "Classifier Combining through Trimmed Means and Order Statistics", Proceedings of the international Joint Conference on Neural Networks , pp. 757-762, May 1998, Anchorage, AL.

[11] K. Tumer and J. Ghosh, "Order Statistics Combiners for Neural Classifiers," Proceedings of the World Congress on Neural Networks , pp. I:31-34, July 1995, Washington, DC.

12. Appendix I - The IRIS data set

Classes: Iris-setosa = 0 Iris-versicolor = 1 Iris-virginica = 2

Sepal length	Sepal width	Petal length	Petal width	Class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa

4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3.0	1.4	0.1	Iris-setosa
4.3	3.0	1.1	0.1	Iris-setosa
5.8	4.0	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa
5.1	3.5	1.4	0.3	Iris-setosa
5.7	3.8	1.7	0.3	Iris-setosa
5.1	3.8	1.5	0.3	Iris-setosa
5.4	3.4	1.7	0.2	Iris-setosa
5.1	3.7	1.5	0.4	Iris-setosa
4.6	3.6	1.0	0.2	Iris-setosa
5.1	3.3	1.7	0.5	Iris-setosa
4.8	3.4	1.9	0.2	Iris-setosa
5.0	3.0	1.6	0.2	Iris-setosa
5.0	3.4	1.6	0.4	Iris-setosa
5.2	3.5	1.5	0.2	Iris-setosa
5.2	3.4	1.4	0.2	Iris-setosa
4.7	3.2	1.6	0.2	Iris-setosa
4.8	3.1	1.6	0.2	Iris-setosa
5.4	3.4	1.5	0.4	Iris-setosa
5.2	4.1	1.5	0.1	Iris-setosa
5.5	4.2	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.0	3.2	1.2	0.2	Iris-setosa
5.5	3.5	1.3	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
4.4	3.0	1.3	0.2	Iris-setosa
5.1	3.4	1.5	0.2	Iris-setosa
5.0	3.5	1.3	0.3	Iris-setosa
4.5	2.3	1.3	0.3	Iris-setosa
4.4	3.2	1.3	0.2	Iris-setosa
5.0	3.5	1.6	0.6	Iris-setosa
5.1	3.8	1.9	0.4	Iris-setosa
4.8	3.0	1.4	0.3	Iris-setosa
5.1	3.8	1.6	0.2	Iris-setosa
4.6	3.2	1.4	0.2	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa

	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor
4.9	2.4	3.3	1.0	Iris-versicolor
6.6	2.9	4.6	1.3	Iris-versicolor
5.2	2.7	3.9	1.4	Iris-versicolor
5.0	2.0	3.5	1.0	Iris-versicolor
5.9	3.0	4.2	1.5	Iris-versicolor
6.0	2.2	4.0	1.0	Iris-versicolor
6.1	2.9	4.7	1.4	Iris-versicolor
5.6	2.9	3.6	1.3	Iris-versicolor
6.7	3.1	4.4	1.4	Iris-versicolor
5.6	3.0	4.5	1.5	Iris-versicolor
5.8	2.7	4.1	1.0	Iris-versicolor
6.2	2.2	4.5	1.5	Iris-versicolor
5.6	2.5	3.9	1.1	Iris-versicolor
5.9	3.2	4.8	1.8	Iris-versicolor
6.1	2.8	4.0	1.3	Iris-versicolor
6.3	2.5	4.9	1.5	Iris-versicolor
6.1	2.8	4.7	1.2	Iris-versicolor
6.4	2.9	4.3	1.3	Iris-versicolor
6.6	3.0	4.4	1.4	Iris-versicolor
6.8	2.8	4.8	1.4	Iris-versicolor
6.7	3.0	5.0	1.7	Iris-versicolor
6.0	2.9	4.5	1.5	Iris-versicolor
5.7	2.6	3.5	1.0	Iris-versicolor
5.5	2.4	3.8	1.1	Iris-versicolor
5.5	2.4	3.7	1.0	Iris-versicolor
5.8	2.7	3.9	1.2	Iris-versicolor
6.0	2.7	5.1	1.6	Iris-versicolor
5.4	3.0	4.5	1.5	Iris-versicolor
6.0	3.4	4.5	1.6	Iris-versicolor
6.7	3.1	4.7	1.5	Iris-versicolor
6.3	2.3	4.4	1.3	Iris-versicolor
5.6	3.0	4.1	1.3	Iris-versicolor
5.5	2.5	4.0	1.3	Iris-versicolor

5.5	2.6	4.4	1.2	Iris-versicolor
6.1	3.0	4.6	1.4	Iris-versicolor
5.8	2.6	4.0	1.2	Iris-versicolor
5.0	2.3	3.3	1.0	Iris-versicolor
5.6	2.7	4.2	1.3	Iris-versicolor
5.7	3.0	4.2	1.2	Iris-versicolor
5.7	2.9	4.2	1.3	Iris-versicolor
6.2	2.9	4.3	1.3	Iris-versicolor
5.1	2.5	3.0	1.1	Iris-versicolor
5.7	2.8	4.1	1.3	Iris-versicolor
6.3	3.3	6.0	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.1	3.0	5.9	2.1	Iris-virginica
6.3	2.9	5.6	1.8	Iris-virginica
6.5	3.0	5.8	2.2	Iris-virginica
7.6	3.0	6.6	2.1	Iris-virginica
4.9	2.5	4.5	1.7	Iris-virginica
7.3	2.9	6.3	1.8	Iris-virginica
6.7	2.5	5.8	1.8	Iris-virginica
7.2	3.6	6.1	2.5	Iris-virginica
6.5	3.2	5.1	2.0	Iris-virginica
6.4	2.7	5.3	1.9	Iris-virginica
6.8	3.0	5.5	2.1	Iris-virginica
5.7	2.5	5.0	2.0	Iris-virginica
5.8	2.8	5.1	2.4	Iris-virginica
6.4	3.2	5.3	2.3	Iris-virginica
6.5	3.0	5.5	1.8	Iris-virginica
7.7	3.8	6.7	2.2	Iris-virginica
7.7	2.6	6.9	2.3	Iris-virginica
6.0	2.2	5.0	1.5	Iris-virginica
6.9	3.2	5.7	2.3	Iris-virginica
5.6	2.8	4.9	2.0	Iris-virginica
7.7	2.8	6.7	2.0	Iris-virginica
6.3	2.7	4.9	1.8	Iris-virginica
6.7	3.3	5.7	2.1	Iris-virginica
7.2	3.2	6.0	1.8	Iris-virginica
6.2	2.8	4.8	1.8	Iris-virginica
6.1	3.0	4.9	1.8	Iris-virginica
6.4	2.8	5.6	2.1	Iris-virginica
7.2	3.0	5.8	1.6	Iris-virginica

7.4	2.8	6.1	1.9	Iris-virginica
7.9	3.8	6.4	2.0	Iris-virginica
6.4	2.8	5.6	2.2	Iris-virginica
6.3	2.8	5.1	1.5	Iris-virginica
6.1	2.6	5.6	1.4	Iris-virginica
7.7	3.0	6.1	2.3	Iris-virginica
6.3	3.4	5.6	2.4	Iris-virginica
6.4	3.1	5.5	1.8	Iris-virginica
6.0	3.0	4.8	1.8	Iris-virginica
6.9	3.1	5.4	2.1	Iris-virginica
6.7	3.1	5.6	2.4	Iris-virginica
6.9	3.1	5.1	2.3	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
6.8	3.2	5.9	2.3	Iris-virginica
6.7	3.3	5.7	2.5	Iris-virginica
6.7	3.0	5.2	2.3	Iris-virginica
6.3	2.5	5.0	1.9	Iris-virginica
6.5	3.0	5.2	2.0	Iris-virginica
6.2	3.4	5.4	2.3	Iris-virginica
5.9	3.0	5.1	1.8	Iris-virginica