CS410 AI Project Report

Qingrong Chen
5140309203
chenqingrong@sjtu.edu.cn
Zhengtian Xu
5140309178
xztlook@sjtu.edu.cn

ABSTRACT

In this project, we firstly give a detailed analysis into our dataset. Then we have tried different classical machine learning methods and deep learning methods on training/testing this dataset. After we train the models, we have done a deep analysis or comparison among all the experiment results. And finally we have come to some conclusions about different machine learning methods, especially between classical machine learning methods and deep learning.

1 INTRODUCTION

In this part, I will give a brief introduction on the dataset and what we have done in general. In this project, we need to deal with the Gene data. There are 5896 cases in the file and each of them has 22283 dimension features. So, we can see this is a large p and small n problem. Also, in another file, there are many labels defined. So in our experiment, we have decided to choose disease state as the corresponding label. More details will be discussed in the following parts.

Based on above observation and some basic principles in machine learning, we can see that reducing dimension is a very important step in this work. As there are too many features, it is easy to cause overfitting problem whether in classical model or deep learning model. There are many ways to solve this problem like PCA [1] and LDA [5]. What's more, regularization like L1 norm and L2 norm can also be used to reduce complexities.

After reducing the dimensions, we just use classical ways and deep learning methods to do training work and compare the results. More details about these will be discussed in the following parts.

2 METHODS

In this part, I will give a description of what machine learning methods have been used in our experiment and how they are realized. In general, it is divided into three parts–dimension reduction, classical methods and deep learning methods.

2.1 Dimension Reduction

In our experiment, we have chosen to use the mainstream method PCA.

Here I will only give a brief introduction of what concepts are behind this method and its functionalities will be discussed in detail in the experiment part.

2.1.1 *PCA Method*.

PCA is a statistical procedure to convert a set of observations of variables into a set of values of linearly uncorrelated variables called principal components. A graph presentation is as follows:

The key idea is that it tried to find the principal components with

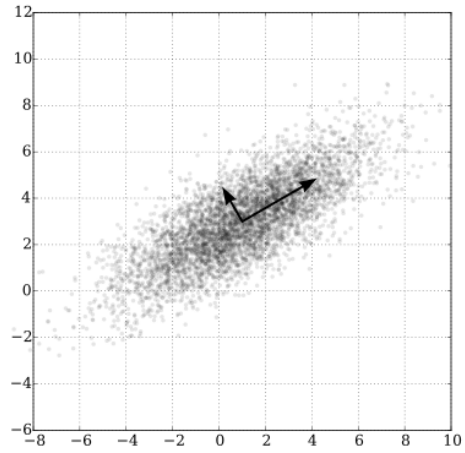


Figure 1: PCA Example

the largest possible variances. In this way, although dimension may be reduced, there is still lot of information(variability) in the remaining data.

In practical realization, it is done by hand. The code has two inputs. One is the feature matrix and the other is the variance percentage. The percentage is defined as:

$$Percen = \frac{\sum_{i=1}^{k} \Lambda_i}{\sum_{i=1}^{n} \Lambda_i} (\Lambda \textit{ means the eigenvalue})$$

So we can see here, the bigger the percentage, the more eigenvalues we use. In our experiment, different percentages are used. More information will be introduced in the experiment part.

2.2 Classical Methods

In total, we have used K Nearest Neighbor, Decision Tree, Linear Classifier, Multiclass Support Vector Machine , and Softmax Classifier classification methods.

2.2.1 ***KNN***.

K-Nearest Neighbor classifier [6] is a supervised learning method. For a test exmaple, it will try to find the K neighbors which are nearest to it. What's more, these K neighbors are already labeled. Then it will pick out the class which accounts for most of the K neighbors and assigns this class to the unlabeled test example. So we can see here, it also needs some method to define distance between data. Generally, there are two formulas used most frequently.
One is called L1 distance and the other is called L2 distance.
L1 distance is defined as:

$$d_{ij} = \sum_{k=1}^{n} \|x_{ik} - x_{jk}\|$$

It simply sums up the absolute differences in all dimensions of the vector.
L2 distance is defined as:

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \|x_{ik} - x_{jk}\|^2}$$

Compared with L1, L2 is more smooth and from mathematical side, it has analytical solution when searching for extremity.
I will talk about these two methods in detail in the experiment part.

2.2.2 ***Decision Tree***.

A decision tree [9]is a flowchart-like structure in which each internal node represents a "test" on an attribute , each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. Its loss function is defined as follows:

$$e = \sum_{x,y} D(x,y) * l(y, f(x))$$

$$l(y, f(x)) = \begin{cases} 1 & y \neq f(x) \\ 0 & y = f(x) \end{cases}$$

where y is the real label ,f(x) is the predicted one and D(x,y) means the distribution between x and y.

2.2.3 ***Linear Classifier***.

Since the data given is a large p and small n problem, we start out with arguably the simplest possible function, a linear mapping:

$$f(x_i, W, b) = W * x_i + b$$

In this function, we let x_i flatten out to a single column vector of shape[D X 1]. The size of matrix W is [K X D], and the size of vector b is [K X 1]. Both W and b are the parameters of the function. In linear classifier, we employ two different loss functions to compare the differences. More information will be discussed in the experiment part.

2.2.4 ***Multiclass Support Vector Machine***.

As SVM [10] is inherently two-class classifiers, we have referred to the One versus rest method which is currently used most widely. Its idea is very simple. If we have k categories, then it will train k different classifiers. Each classifier will be responsible for discriminating category i from the rest, so this method has low complexity and also runs fast.
The loss function is defined as same as in the 2-class SVM:

$$L_i = \sum_{j \neq y_i} max\{0, (w_j^T x_i - w_i^T x_i + \delta)\}$$

where w_j is the j-th row of W reshaped as a column. However, we extend the loss function with a regularization penalty R(w) to avoid overfitting problem. We use the regularization penalty L2 norm which will discourage large weight through square sum of all the elements in matrix:

$$R(W) = \sum_i \sum_j W_{ij}^2$$

Then the Hinge loss can be represented by L_i and R(W):

$$L = \frac{1}{N} \sum_i L_i + \Lambda R(W)$$

I will discuss how it performs in detail in the experiment part.

2.2.5 ***Softmax Classifier***.

Softmax Classifier [3] is essentially a logistic regression model. It can be used to do multi-class classifications. The Softmax regression model is useful for questions such as MNIST [8], handwriting, digital classification, and so on. The loss function is defined as follows:

$$L_i = -f_{yi} + log \sum_j e^{f_j}$$

where the notation f_j meas the j-th element of the vector of class scores computed in the Linear Classifier.

2.3 Deep Learning based method

In deep learning [7] part, we have combined the classical machine learning method and deep neural network. More specifically, we have used the dimension reduction method like PCA to reduce the data dimensions at first, as we know in this dataset, we have large p and small n, so direct training may not get enough test examples. Then we use deep neural network to train the model.

2.3.1 *Deep Neural network.*

Based on the experiment setting of using deep learning, we use neural network to train a classifier. Since the features do not have any locality or direct relationship, we don't use convolutions. We employ a method based on encoding-coding with a dully-connected multilayer structure. We use L2 normalization in the input layer to ensure the data smoothing and use L1 to control the model complexity.

- **Weight Initialization**
  Although it is easy to do all zero initialization, it is not good to do that. As if we set all weights to zero at first, they will also all compute the same gradients during backpropagation and undergo the exact same parameter updates.In other words, there is no source of asymmetry between neurons if their weights are initialized to be the same.
  So in our experiment, we have initialized each neuron's weight vector as $\frac{w=np.random.randn(n)}{\sqrt{n}}$ where n is the number of its inputs. In this way, we can ensure that all neurons in the network initially have approximately the

same output distribution and empirically improves the rate of convergence.

- **Regularization**
  There are several ways of controlling the capacity of Neural Networks to prevent overfitting: L2 regularization and L1 regularization. We have chosen L2 because it is more smooth than L1 and using the L2 regularization ultimately means that every weight is decayed linearly, while in L1, it tends to make the vectors more sparse. As we already do the dimension reduction work by PCA at first, we prefer more smooth way which is just L2 regularization. To be specific, for each weight w in the network, we add the term $\frac{1}{2}\Lambda w^2$ to the objective function.

- **Loss Function**
  The loss is divided into two parts. One is called loss part of the objective which is already discussed above. The other is called data loss which measures the compatibility between a prediction and the ground truth. In our model, we have chosen to use the cross-entropy loss as the loss function:

$$L_i = -log(\frac{e_{f_{y_i}}}{\sum_j e^{f_j}})$$

  It is the same as the loss function in softmax classifier.

- **Parameter Updates**
  Once the gradient is computed with backpropagation, the gradients are used to perform a parameter update.
  In our experiment, we have chosen RMSprop [11]as our updating strategy. It is a very effective adaptive learning rate method and adjusts the Adagrad method in a very simple way in an attempt to reduce it aggressively, monotonically decreasing learning rate.
- **Batch Normalization**
  Nowadays, it has become a very common practice to use Batch Normalization [4] in neural networks. Batch Normalization method has many advantages. One is that, by using this method, we can accelerate the initial learning rate and this learning process can converge quickly. Another is that, this method has increased the ability of the model for generalization which can increase its accuracy. Additionally, batch normalization can be interpreted as doing preprocessing at every layer of the network, but integrated into the network in a differentiable manner.
  More details about this method will be discussed in the experiment part.

3 EXPERIMENT

In this experiment, we have firstly done the data preprocessing work. This work is mainly to reduce the dimensions of original data. We have adopted the mainstream method PCA and tried different percentage to which the eigenvalue is preserved. To be specific, we test variance percentage of 95% 90% and 85%. We have compared them with some classical methods.
Then we have copied with the label file. By analyzing data, we find that among 5896 samples, 1875 of them have no labels. Aside from the samples with no labels, there are classes which contain little samples like only 1.This small number is not suitable for training. So in the experiment, we have tested classes whose data number is no less than 10 times and split them into two sets: training set and testing set. Also, the ratio is about 8:2. To solve the imbalance data distribution problem, the ratio is calculated for each class but not for the whole data. At last we have 2470 training data and 1059 testing data.
Furthermore, for those samples which contain no labels, we have used the already trained methods to predict their labels. If there are certain number of models giving out the same result, we will assume it is right(may not be really right, since we have no ground truth).

3.1 Data Prepocessing

In this part, we mainly do two works. One is to sort out the samples whose labels occur not less than 10 times. The other is to do dimension reduction work on the original dataset. The dimension reduction work is done by PCA. We have set different parameters for PCA, which preserves variance percentage of 95%, 90% and 85% for the data. For 95%, the feature is reduce to 1271 dimensions. For 90%, the feature is reduced to 453 dimensions. For 85%, the feature is reduced to 187 dimensions. We have used SVM to compare these three data and the better one will be used for other models.

3.2 Classical Method

For KNN method, we have tried different distance functions and see how they influence the results. We also adjust the parameters in linear classifier, softmax classifier and SVM to make for better performance. In the end, we have compared the different results among these models.

3.3 Deep Learning Method

We have implemented deep learning method with tensorflow [2]. Totally, there are four hidden layers and the neural units are 256,512, 512,256 respectively. To classify, we have connected a layer consisting of 79 neural units to the model. For the loss function, cross-entropy loss is used. We also oversample the minority and undersample the majority to solve data unbalanced distribution problem(some labels have over 100 samples while some only have 10).

3.4 Oversampling and Undersampling Method

Oversampling and undersampling are techniques used to adjust the class distribution of a data set. They both use a bias to select samples. Oversampling can be done by adding copies of instances from the under-represented class(small class). Undersampling can be done by dropping when selecting training samples. Both of this add little burden to whole process and is easy to implement.

3.5 Evaluation Metric

We have used the confusion matrix [12] to evaluate our system. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class. This allows more detailed analysis than mere proportion of correct guesses (accuracy). Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced (that is, when the number of samples in different classes vary greatly, just the case we face). For example, if there were 95 cats and only 5 dogs in the data set, the classifier could easily be biased into classifying all the samples as cats and it will still have an accuracy of around 95%.

The precise mathematical formulas are defined in the following. There are four conditions–false positives(FP), false negatives(FN), true positives(TP), and true negatives(TN). The accuracy(ACC) is defined as $\frac{TP+TN}{TP+TN+FP+EN}$, meaning in all the results returned by the model how many of them are right. The recall rate(TPR) is defined as $\frac{TP}{TP+FN}$, meaning in all the correct labels(ground truth) how many of them are found. To fully evaluate the system, we can use the overall metric F1-score which is defined as $\frac{2ACC}{2ACC+TPR}$.

4 RESULT

In this part, I will just show the experiment result and discussion about different models will be presented in the final part.

4.1 PCA

We test variance percentage of 95% , 90% and 85% for the original data. The first has 1271 dimensions, the second has 453 features and the last has 187 dimensions. We use the SVM model to test their performance. The result is that for the 90% PCA, the F1-score is 0.869, for the 95% PCA, the F1-score is 0.838885 and for the 85% PCA, the F1-score is 0.853. So after we compare their performances, we choose to use the result from 90% PCA.

4.2 Decision Tree

As the Decision Tree code is too complex, we have referred to the standard library sklearn for realization. So, we cannot change parameters in this model. Through the experiment, we get the F1-score of 0.723534.

4.3 K Nearest Neighbor

We have chosen L1 distance function and L2 distance function for evaluating KNN. The result is presented in Table 1. We can see

Table 1: KNN Performance

	L1 distance	L2 distance
F1-score	0.864	0.839

that L1 distance performs better than L2. Maybe it is because only several genes(among all the features) determine the disease, so L1 sparsity performs better in selecting those features out.

4.4 Linear Classifier with SVM

To figure out the best performance for SVM, we have adjusted different parameters for this model such as learning rate, regularization strength. We have tried the learning rate from 0.001 to 0.16 by step of 0.002 and the regularization strength from 0.001 to 0.01 with step of 0.001. The result is presented in graph figure 2 and figure 3. From the results, we finally get F1-score scores best of 0.869 when

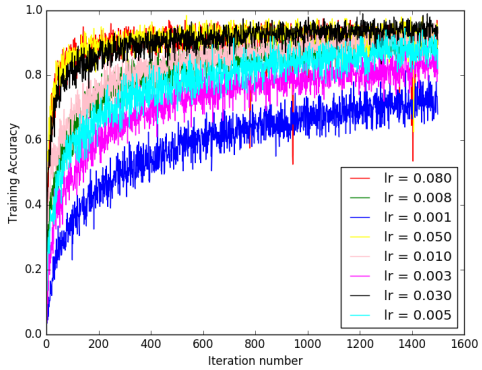


Figure 2: Accuracy Value for SVM

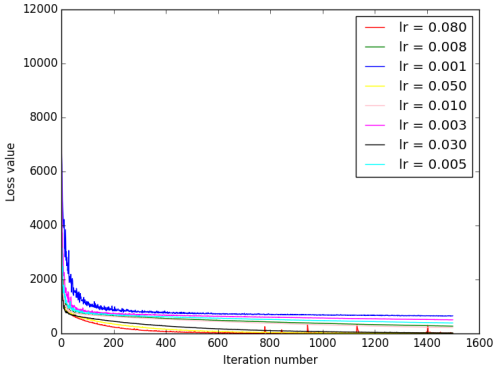


Figure 3: Loss Value for SVM

the learning rate is 0.05. In average, the running time is about 4.67 seconds.

4.5 Linear Model with Softmax

In this model, we also try different learning rates and different regularization strength. These parameters are the same as them in the SVM model. The final result is in figure 4 and figure 5. From the results, we see this algorithm can achieve a best F1-score of 0.888 when learning rate is 0.05 and regularization strength is 0.07. In average it costs about 13.42 seconds. We can see that although this method performs a little better than SVM but it takes a much longer time which may not be desirable when data size is large.

4.6 Deep Learning Method

In deep neural network, we set four hidden layers and one additional layer L2 regularization for smoothing. Also, we use batch

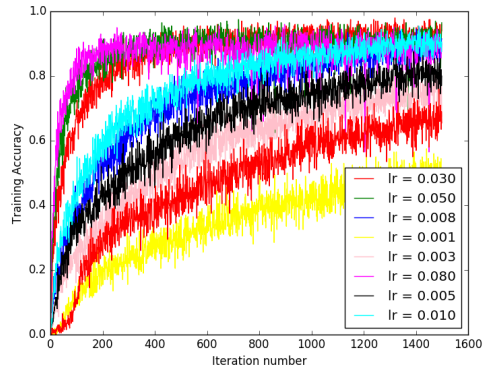


Figure 4: Accuracy Value for Softmax

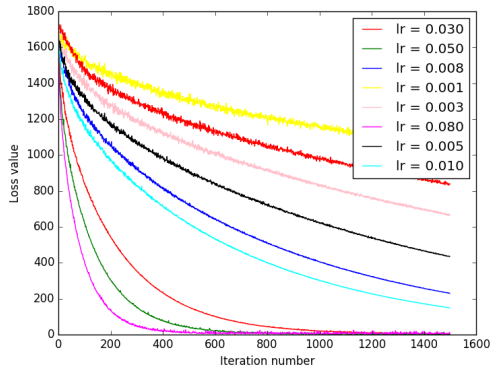
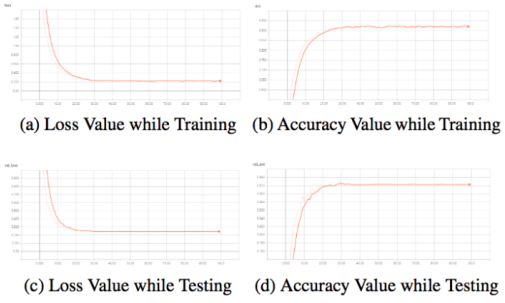


Figure 5: Loss Value for Softmax

normalization in each layer which greatly increases the performance. Through adjusting, we find when learning rate is $2.5x10^{-3}$, we can achieve highest F1-score of 0.931 and the final results are presented in figure 6. The lines in these figures just show the learning process in deep neural network.



(a) Loss Value while Training (b) Accuracy Value while Training

(c) Loss Value while Testing (d) Accuracy Value while Testing

Figure 6: Deep Learning Process

5 DISCUSSION PART

In this part, I will talk about many comparisons such PCA parameters choosing , Decision Tree and SVM , SVM and Softmax Comparison, KNN and SVM/Softmax Comparison, Deep Learning and Classical Method Comparison and so on.

5.1 PCA Parameter Choosing

We both know PCA is a very good algorithm for reducing dimensions of features but how much we should reduce the features is really a big issue. For example, in our experiment, we have used three percentages 95%, 90% ,85% and the results show a difference, displaying percentage 90% performs better. As the reduction result from PCA will be used for all training models in the later phase, we should be more careful in choosing one. From my side, I think dichotomy is very good. Maybe we can firstly try 50% and from the result, we can try 75 or 25 and etc.

5.2 Decision Tree and SVM

From the experiment result, we can see Decision Tree falls behind SVM. Maybe it is partially related to some parameters settings. Also, it may be related to the model itself. After we do the PCA work, the data may become linearly separable which is better for SVM while decision tree is more suitable for some non-linear data.

5.3 SVM and Softmax

They both derive from the linear model, but they choose different loss functions. SVM chooses hinge loss as its loss function while softmax chooses cross-entrophy fucntion. Softmax method will compute the possibility for all the labels while SVM will only try to make one label stand out from the others. So, we can understand why SVM performs faster as shown in our experiment. As they both derive from the same model, we can also know from the experiment result that they don't differ much in performance.

5.4 KNN and SVM/Softmax Comparison

KNN has some nice properties: it is automatically non-linear, it can detect linear or non-linear distributed data, it tends to perform very well with a lot of data points as it leverages the similarities between different data points. Compared with KNN, SVM doesn't rely on much data. Instead, when you have a limited set of points in many dimensions, SVM tends to be very good because it should be able to find the linear separation that should exist. In our experiment, our data is not that much, so we can see that SVM performs both more quickly and better.

5.5 L2 and L1 Regularization Comparison

The difference between the L1 and L2 is just that L2 is the sum of the square of the weights, while L1 is just the sum of the weights, so we can see L1 is more sparse while L2 is more smooth. We cannot say which one is better, but it depends on the data we use. For example, in our experiment, L1 performs better. It may be because in disease, actually only some certain genes will function. So the features tends to be more effective when becoming more sparse. So this also shows if we can know a little about the background information of the data, we can choose a more wise model.

5.6 Batch Normalization and no Batch Normalization

Before we use Batch Normalization, the F1-score of deep learning can only achieve 0.857 while after using it we can achieve F1-score of 0.924 which is a great improvement. Also, the latter result has

ranked the best among all the other methods. This phenomenon on one side shows the effectiveness of batch normalization which can make the model more generalizable. On the other hand, it shows the necessity of pre-processing/pre-training for deep learning. Only when the data has been processed properly, can deep learning show its power.

5.7 Deep Learning and Classical Method Comparison

In our experiment, we have realized both deep learning methods and classical machine learning methods. They differ in many aspects. Firstly, in the design process, they already have many differences. In classical machine learning methods like KNN, PCA and SVM, we care more about the objective function, the loss function and the algorithm to solve the problem. For example, SVM tries to find a plane mostly separating two classes of data. However in deep learning, the work is not that specific, or in other words, we care more about the structure of our model, about how to build a reasonable structure. In deep learning, it specifically refers to its multi-layer structure. Secondly, we find that when data is not enough, deep learning may not be a good choice. In the first place, when we train the deep learning model, the experiment result is not satisfactory(with low F1-score). It may be because deep learning model needs to adjust too many inside parameters. So, we can only do a lot of additional work like doing L2 regularization, adding batch normalization, doing oversample and undersample and so on to improve its performance. However, for classical methods, most methods work so fine, especially SVM, as it requires the data to be simple because usually the simpler the data, the easier to find plane separating data. Thirdly, through the experiment, we find that deep learning can have so much improvement space while traditional methods don't. In traditional learning methods like SVM or Softmax, we have changed its learning rate and regularization strength, but that actually does not do much to the final result(only small changes) while in deep learning, after we have done the pre-processing work well, it really improves so much(from 0.4 to over 0.9 and rands 1st). So we suppose, with deep learning, we may be able to capture some essence or really important features of an object so the model can have good generalization ability and perform so well.

So in conclusion, we think both deep learning and classical methods have their disadvantages and advantages. To use them well, we don't have any precise formula for guidance. We can only get to know the fact after we do enough trials or attempts.

6 CONCLUSION

At last, we want to thank Professor Bo Yuan and two TAs for your patient guiding during our learning process. Your help really matters a lot to us! What's more, I should also thank my teammate Zhengtian Xu for his help during this process.

REFERENCES

[1] 2002. *Principal component analysis*. Wiley Online Library.
[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).
[3] Steven Gold, Anand Rangarajan, and others. 1996. Softmax to softassign: Neural network algorithms for combinatorial optimization. *Journal of Artificial Neural Networks* (1996).
[4] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).
[5] Alan Julian Izenman. 2013. *Linear discriminant analysis*. Springer.
[6] James M Keller, Michael R Gray, and James A Givens. 1985. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics* (1985).
[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* (2015).
[8] Yann LeCun, Corinna Cortes, and Christopher JC Burges. 2010. MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* (2010).
[9] S Rasoul Safavian and David Landgrebe. 1991. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics* (1991).
[10] Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters* (1999).
[11] T Tieleman and G Hinton. 2012. Leccture 6.5-rmsprop. *COURSERA: Lecture* (2012).
[12] James T Townsend. 1971. Theoretical analysis of an alphabetic confusion matrix. *Attention, Perception, & Psychophysics* (1971).