Classify a Sparse Photo Dataset: Implementation a Neural Network with Threshold Technique and Evolutionary Algorithm

Xinhui Bian

Research School of Computer Science, Australian national University, U7096593@anu.edu.au

Abstract. Facial recognition is an outstanding research field of artificial intelligence. When a person has multiple photos of himself, AI can be used to determine the person's identity accurately. But when there are few relevant face photos, identification will become very difficult. In order to better identify this "sparse" situation, a naive binary classification networks have been implemented. At the same time, threshold and cross-validation techniques are applied here to observe whether the prediction accuracy can be improved. Also, an evolutionary algorithm is implemented as an attempt to help fine tune network architectures and thresholds.

Keywords: neural network · threshold · cross-validation · face matching · evolutionary algorithm

1 Introduction

Artificial neural network is a huge field in computer science. The application of neural network covers classification, prediction and regression, face detection and recognition, etc. Facial recognition is a very important part of it. However, when recognizing "sparse" photos, the result is still not clear enough. For example, in a series of unidentified facial photos, we took out one of them, it is difficult to confirm or accurately find the other photos of this person. The reason is that there are many features that affect a photo, such as the angle of head up, shadows, etc.

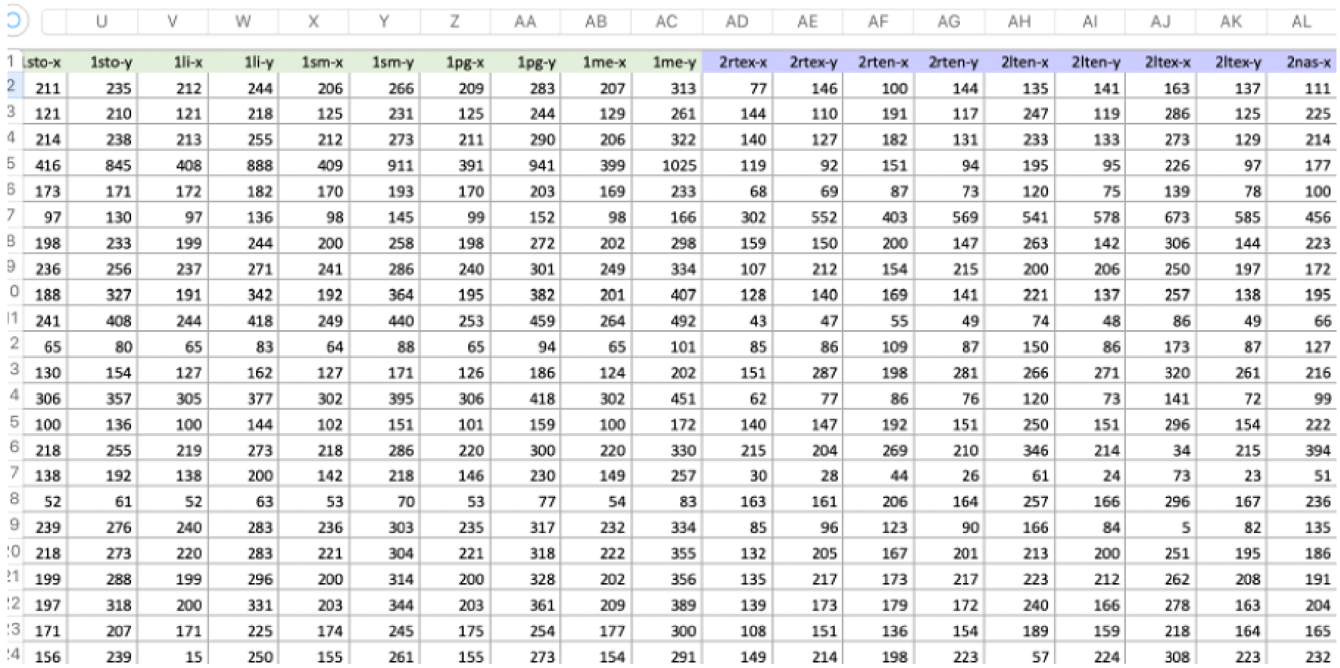
This paper establishes a neural network with 57 input neural and 1 output neural to deal with face matching problem. The source of the dataset used in this paper is a subset of the National Archives of Australia "Bonds of Sacrifice exhibition" (Australia, n.d.) [1]. We could study the facial information of people whose names are unknown in this dataset, finding their identities which is very helpful for us to restore history and find important historical information. Caldwell [2] and others have done some classification research on this dataset. In their paper, the researchers used a two-layer neural network and obtained 75% accuracy by classifying the distance. In addition, they also used Adaboost, SVM, KNN and random forest, etc., but the results were not satisfactory. The best one SVM only had an average accuracy of 58%.

The main point of this article is to create a very simple artificial neural network. After applying some common technologies which we think could improve performance. I could judge whether these technologies are effective by observed results. This article uses techniques similar to [1]: cross-validation, thresholding and hidden layer specification. At the same time, in order to better adjust the parameters of the neural network, an evolutionary algorithm will be implemented. Evolutionary algorithm have shown that they are widely used in fields such as parameter optimization, industrial scheduling, resource allocation, and complex network analysis [7]. The results of the paper show that the application of these techniques has resulted in some improvements.

2 Method

2.1 Data

The data set contains 57 columns, the first 56 columns are the abscissa and ordinate of facial feature makers which is marked through the photos. The selection of the facial maker depends on many factors. It is necessary to ensure that the marked points can represent the characteristics of the image face well (such as the position of the bones etc.). Furthermore, it should be less affected by the change of the face orientation (yaw, pitch, roll)[2]. The last column is the target. When the two pictures are the same person, the target is 1, otherwise, the target is 0. Each row of the data set contains information about two pictures. The first half is the FFMs coordinates of the first person, and the second half is the FFMs coordinates of the other person. Before putting the data into the neural network, the data set needs to be preprocessed. First I delete the first row and first column of the table, which is the explanation of the data. These attributes cannot describe the distribution law of the sample itself, so simply delete these attributes. Then through observation, it was found that some data values can reach 1000. So we need to do normalize to ensure that each feature is treated equally by the classifier. With this in mind, We apply min-max



U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL
1sto-x	1sto-y	1li-x	1li-y	1sm-x	1sm-y	1pg-x	1pg-y	1me-x	1me-y	2rtex-x	2rtex-y	2rten-x	2rten-y	2lten-x	2lten-y	2ltex-x	2ltex-y	2nas-x
211	235	212	244	206	266	209	283	207	313	77	146	100	144	135	141	163	137	111
121	210	121	218	125	231	125	244	129	261	144	110	191	117	247	119	286	125	225
214	238	213	255	212	273	211	290	206	322	140	127	182	131	233	133	273	129	214
416	845	408	888	409	911	391	941	399	1025	119	92	151	94	195	95	226	97	177
173	171	172	182	170	193	170	203	169	233	68	69	87	73	120	75	139	78	100
97	130	97	136	98	145	99	152	98	166	302	552	403	569	541	578	673	585	456
198	233	199	244	200	258	198	272	202	298	159	150	200	147	263	142	306	144	223
236	256	237	271	241	286	240	301	249	334	107	212	154	215	200	206	250	197	172
188	327	191	342	192	364	195	382	201	407	128	140	169	141	221	137	257	138	195
241	408	244	418	249	440	253	459	264	492	43	47	55	49	74	48	86	49	66
65	80	65	83	64	88	65	94	65	101	85	86	109	87	150	86	173	87	127
130	154	127	162	127	171	126	186	124	202	151	287	198	281	266	271	320	261	216
306	357	305	377	302	395	306	418	302	451	62	77	86	76	120	73	141	72	99
100	136	100	144	102	151	101	159	100	172	140	147	192	151	250	151	296	154	222
218	255	219	273	218	286	220	300	220	330	215	204	269	210	346	214	34	215	394
138	192	138	200	142	218	146	230	149	257	30	28	44	26	61	24	73	23	51
52	61	52	63	53	70	53	77	54	83	163	161	206	164	257	166	296	167	236
239	276	240	283	236	303	235	317	232	334	85	96	123	90	166	84	5	82	135
218	273	220	283	221	304	221	318	222	355	132	205	167	201	213	200	251	195	186
199	288	199	296	200	314	200	328	202	356	135	217	173	217	223	212	262	208	191
197	318	200	331	203	344	203	361	209	389	139	173	179	172	240	166	278	163	204
171	207	171	225	174	245	175	254	177	300	108	151	136	154	189	159	218	164	165
156	239	15	250	155	261	155	273	154	291	149	214	198	223	57	224	308	223	232

Fig. 1. screenshot of FFMs dataset

standardization here.[3] This method records Max and Min by go through each data in the feature vector, then uses Max-Min as the base to normalize the data (a flaw of this method is that when new data is added, it may lead to changes in max and min).

$$X* = (X - min)/(max - min)$$

After processing the data, the program will read the new processed data and create tensors for input and output before putting it into the nervous system training. Then the data is randomly divided into a test set and a training set. Baseline randomly split data into training set (80%) and testing set (20%). In the subsequent steps, cross-validation will be used to divide the test set and training set, which will be analyzed later.

2.2 implement a baseline

After reading technical paper and data paper, we will first build a three-layer neural network (contain one hidden layer), which contains 60 hidden neurons, 56 input neurons and 1 output neurons. The reason why we decided select a small neural network is that we shouldn't create a bulky model for a small task. In order to get better hyper-parameters, we organize tests of different learning rates and epochs for 50 times. The average accuracy was used to observe the performance of the model. By observing the table below, I choose the learning rate and epoch as 0.001 and 500. After epoch exceeds 500, it will lead to overfitting. So there's a poor performance in the test data set. When the epoch is less than 500, the model is under-fitting, which decrease the performance of the neural network as well. Similarly, when the learning rate is higher than 0.001, learning will overshoot the minimum. However, when the learning rate is lower than 0.001, on the contrary, the neural network falls into a local minimum. Classification

Table 1. Average accuracy for different threshold epoch and learning rate

Learning rate / Epoch	0.01	0.007	0.005	0.001	0.0005
50	68.9%	68.6%	69.1%	67.1%	66.2%
100	69.9%	70.9%	70.1%	66.6%	67.6%
300	64.2%	61.3%	64.4%	70.1%	68.6%
500	62.3%	64.6%	65.3%	71.5%	70.9%
1000	62.4%	65.2%	62.1%	69.8%	64.2%

process is non-linear, and the output of the neural network should be between 0 and 1. Thus sigmoid function is applied for hide layers and output layers(We don'e choose softmax in the output layer because this task is a two-classification problem, so sigmoid is sufficient). This output could do threshold technique easy after running through the network. The optimization algorithm used to find the weights of neural networks is gradient descent. It uses the loss for backpropagation, then iterates continuously, updates the weights, finds the lowest point of loss, finally makes our model achieve better results. The network uses binary cross entropy[5] as the loss function. Cross entropy can measure the degree of difference between two variables. The smaller the value of cross entropy, the better the prediction effect of the model. The network uses Adam as the optimizer. Adam is a highly efficient stochastic gradient decent optimization function.It was chosen for its efficiency.

2.3 Techniques

Apply cross-validation In the method adopted by the above baseline, the selection of the final model and parameters will greatly depend on the division method of the training set and the test set. Under different division methods, test loss changes greatly. So if our division method is not good enough, we will not be able to choose the best model and parameters. However, the division of training set and test set means that we cannot make full use of the data we have.Therefore, we use LOOCV cross-validation to overcome this problem. The idea of LOOCV is shown in the figure below.

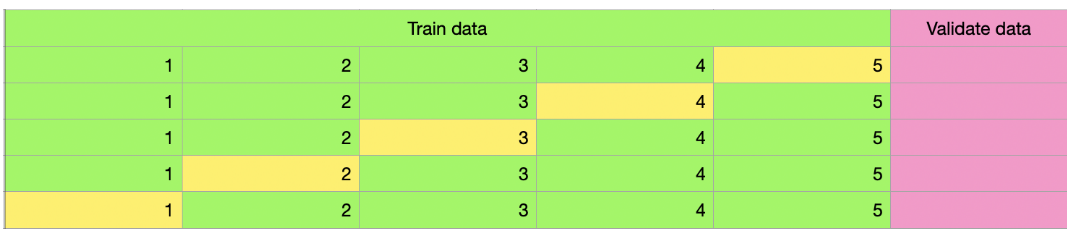


Fig. 2. LOOCV idea

Green part is used for train model and pink part don't participate the training. Data determines the upper limit of the process performance, especially for the small dataset. The common cross-validation methods are hand-out cross validation, K-fold cross validation and leave one out.I apply leave one out here.The reason why LOOCV is used instead of normal k-fold cross validation is that the value of k is difficult to determine. A small value of K will result small amount of data available for modeling. The results of the leave-one-out method are more uniform and trustworthy.

Apply evolutionary algorithm for threshold the hidden neurons The neural network outputs a value between [0, 1]. The baseline uses 0.5 as the threshold value for judging match or not. Through the technique paper[2], it is found that sometimes setting the threshold to middle does not necessarily bring good results. Therefore, it is necessary to set different thresholds to test the neural network and find the most suitable value. Here we use evolutionary algorithm to help us find parameters. This is a common technique in current papers. Our paper implements a very simple evolutionary algorithm. We assumes that each individual has only two elements, threshold and hidden neuron. The algorithm can be simply divided into 5 steps.

1. Initialise the population and the coding method we use is binary coding.
2. Judge the fitness of the individual according to the accuracy of the network. Judge whether it meets the optimization criteria. If so, output the best individual and its optimal solution,. Otherwise, proceed to the next step.
3. Choose parents based on fitness. Here we use the strategy that individuals with high fitness have a high probability of being selected, and individuals with low fitness are eliminated.
4. Use parental chromosomes to cross over according to a certain method to produce offspring.
5. Make mutations to the offspring's chromosomes.

Because the network is evaluated every time the hyperparameter changes. This means that the algorithm is quite slow, therefore, a small number of individuals and a small number of generations are used for testing.

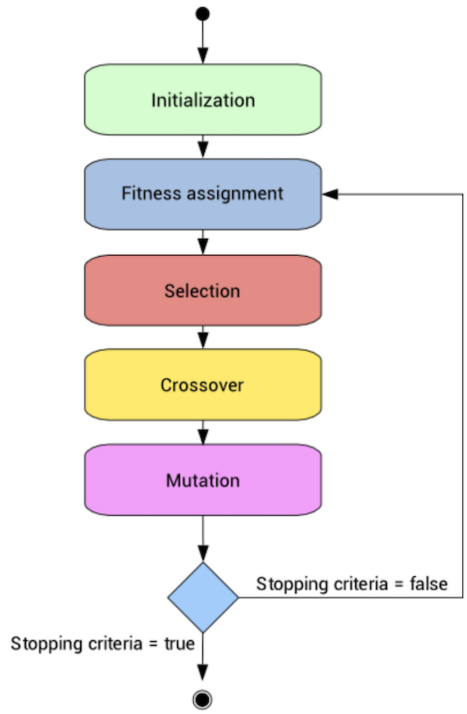


Fig. 3. EA steps [8]

2.4 Evaluate the module

There are two commonly evaluation methods, which are the accuracy of the test set and confusion matrix.[4] The confusion matrix is composed of false positive (true is 1, prediction is 0), false negative (true is 0, prediction is 1), true positive (truth is 1, prediction is 1), and the true negative (the truth is 0, the prediction is 0). These four variables form a table with two rows and two columns. The following is the structure of the confusion matrix: It

Table 2. Example of confusion matrix

	positive	negative
positive	TP(true positive)	FN(false negative)
negative	FP(false positive)	TN(true negative)

can be seen from the figure that the accuracy can be derived from the confusion matrix

$$Accuracy = (TP + TN)/(TP + FN + FP + TN) \tag{1}$$

3 Results and discussion

Due to the fluctuation of the performance, each result is calculated by taking the average value after 50 runs.

3.1 Results of Threshold Technique

This paper builds a table similar to the technical paper (similar to the confusion matrix as well) to help find a better threshold. The threshold varies from 0.2 to 0.8. The table on the left shows the impact of theta in the test set. It can be seen that as the threshold increases, the number of correct results also increases. The right side shows that the number of correct results slowly increased, peaking at 0.5 and 0.6. Then gradually decreased. Note that when the threshold is 0.5, the number of correct in training sets and test set is maximized, and the number of false positives and negatives is roughly similar. This also shows that for the dataset selected in this papert. We don't need to change the threshold to improve performance. But this does not mean that the threshold is useless. Changing the threshold to make all errors FP or FN can help us identify boundaries better.

Table 3. Neural net performance on the test and train set

theta(test)	correct	false +ve	false -ve
0.20	4	2	1
0.30	4	2	1
0.40	5	1	1
0.50	5	1	1
0.60	5	1	1
0.70	5	0	2
0.80	5	0	2

theta(train)	correct	false +ve	false -ve
0.20	18	9	2
0.30	22	5	2
0.40	24	3	2
0.50	25	1	3
0.60	25	0	4
0.70	23	0	6
0.80	22	0	7

3.2 Results of Evolutionary Algorithm

Here, the evolutionary algorithm try to find the combination of threshold and hidden layer that can achieve the best result. The displayed result is higher than the above result, but roughly the same, because the above result also performs well on the 0.6 threshold. The results are fluctuating up and down while iterate. As shown in Table 4 below.

Table 4. Evolutionary Algorithm results

iteration	threshold	hidden neuron	test loss
1	0.97	24	0.1108
2	0.65	80	0.1535
3	0.87	88	0.1782
4	0.59	72	0.0610
5	0.60	77	0.0533
6	0.16	88	0.0265
7	0.60	77	0.1041

Though the result is not clear, we agree that 0.6 as threshold and 77 as hidden neurons are the result we want. The paper then compare the baseline, the NN after using technologies, and the results of the dataset paper.

Table 5. Evolutionary Algorithm results

	Baseline	NN with techniques	dataset paper result
Accuracy	71.5%	77%	75%

3.3 Discussion

By observing the final results, we can see that the neural network we built did not perform well. Changing the classification threshold in the neural network can improve the classification accuracy. However, even under the optimal threshold, the overall accuracy achieved is about 77%. It is similar to the result obtained by using a neural network with two hidden layers in a technical paper. For this dataset, it seems that higher thresholds are better than lower thresholds. When trying to find the best parameters for the network model, the evolutionary algorithm will find different results each time it runs, and the trend of the results is not obvious. But it still seems to agree that a higher threshold will produce better results. One of the reasons is that because the algorithm needs to repeatedly train the model, the amount of calculation is too large. We did not reach a value that allows the result to converge enough. Another reason is that we randomly split the dataset, which will also affect the fitting function. The difference in the results of fitting function caused the evolutionary algorithm to sway. It is unable to find a suitable value. This is not to say that using evolutionary algorithm is bad, it is still very similar to manually adjusting network parameters. One potential reason for such low accuracy I considered is that the data used is difficult to analyze. Perhaps it would be a good choice to use CNN to extract features autonomously.

4 Conclusion and Future Work

The purpose of this paper is to determine whether two faces match based on the FFMs database. The binary classifier neural network splits the data set at a ratio of 4:1 for training and testing. This paper implements three technologies: cross-validation, threshold and evolutionary algorithm. When trying to use evolutionary algorithms to further adjust the parameters, the results given are similar to manually adjusting the network model. In the future, there are still many areas for improvement. we could use more complex learning models to improve learning, such as dropout. The implemented evolutionary algorithm can also be improved by improving its performance and fine-tuning options. In this paper, we use test loss as the evaluation standard, but the test loss is not stable due to the divide problem. In terms of data processing, we can use the original photos as data, extract more useful information on the photos. Better feature selection will be conducive to further research. In the future, more research on the classifier should be done to improve its performance in the future.

References

1. Milne L K, Gedeon T D, Skidmore A K. Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood[J]. training, 1995, 109(81): 0.
2. Caldwell, S., n.d. Human interpretability of AI-mediated comparisons of sparse natural person photos.
3. Loukas, S., 2021. Everything you need to know about Min-Max normalization in Python. [online] Medium. Available at: https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79
4. Lewis, H.G., Brown, M. 2001, "A generalized confusion matrix for assessing area estimates fromremotely sensed data", International Journal of Remote Sensing, vol. 22, no. 16, pp. 3223-3235.
5. Brink A D, Pendock N E. Minimum cross-entropy threshold selection[J]. Pattern recognition, 1996, 29(1): 179-188.
6. Shorten C, Khoshgoftaar T M. A survey on image data augmentation for deep learning[J]. Journal of Big Data, 2019, 6(1): 1-48.
7. Yao, L., Sethares, W.: Nonlinear parameter estimation via the evolutionary algorithm. IEEE Transactions on Signal Processing. 42, 927-935 (1994).
8. Neuraldesigner.com. 2021. Genetic algorithms for feature selection Neural Designer. [online] Available at: https://www.neuraldesigner.com/blog/genetic_algorithms_for_feature_selection.