FIT5186 Intelligent Systems

Assignment

Solving a Neural Network Problem

Steel Plate Fault Classification

HU Ying (2819****)

SOUTHEAST UNIVERSITY - MONASH UNIVERSITY JOINT

GRADUATE SCHOOL

Submission date: 29/5/2017

Abstract

Fault detection plays an important role in enhancing the manufacturing quality to reduce the testing cost. Hence, an effective and efficiency detection method is desired to be proposed instead of artificial cognition. Fault detection can be considered as a pattern recognition problem. In this paper, we build neural networks to classify the steel plate fault into different types, applying three architectures (Ward Networks, General Regression Neural Networks, and Probabilistic Neural Networks) provided by NeuroShell 2. Multiple training parameters are attempted in our experiments to find the most suitable model for this fault classification. The limitation and conclusion are stated for future works.

1 Introduction

In industrial production, a fault refers to an unacceptable property or attribute of one product distinguished from the acceptable typical product. The fault detection and classification, which aims to describe the characteristics of a fault and classify the fault into a proper type, is a critical component to ensure the quality of products and facilitates precautionary maintenance. Therefore, an intelligent system is desired to enhance the efficiency of fault detection instead of manual observation. Isermann (2005) generated several symptoms to indicate the difference between nominal and faulty status so as to determine the fault by classification or inference. A fault detection model was developed by using feed forward network with back propagation algorithm and binary adaptive resonance network (Rajakarunakaran et al., 2008). A review also introduces the selection of proper indices for existing conditions (Faiz and Ojaghi, 2009).

During the steel plate production process, the snapshots for all plates are recorded, which are about to be pre-processed so as to extract attributes of plate faults for each plate. By analyzing those attributes, the type of fault for each steel plate could be decided. In this paper, we use NeuroShell2 to train a neural network for the steel plates faults classification. The neural network models are built based on three architectures, Ward Networks (backpropagation network with hidden slabs with different activation functions), General Regression Neural Networks (GRNN), and Probabilistic Neural Networks (PNN). For Ward Networks, the number of hidden neurons are tunable for a higher classification accuracy.

2 Data Sets

The dataset for this study was obtained from UCI Machine Learning Repository Steel Plates Faults Dataset. This dataset consists of 1941 instances of steel plates. Each instance contains 27 attributes (Table 1) describing the image features, and is labeled by one of 7 fault types (Pastry, Z_Scratch, K_Scatch, Stains, Dirtiness, Bumps and Other_faults). The fault types are already represented by using 1-out-of-N encoding where the variable is replaced with N variables each containing either a zero or a

one. For all architectures, we apply this type representation as shown in Table 2. Among these 1941 instances, the distribution of fault types is shown in Figure 1. Instances belonging to different fault types show a quite uneven distribution

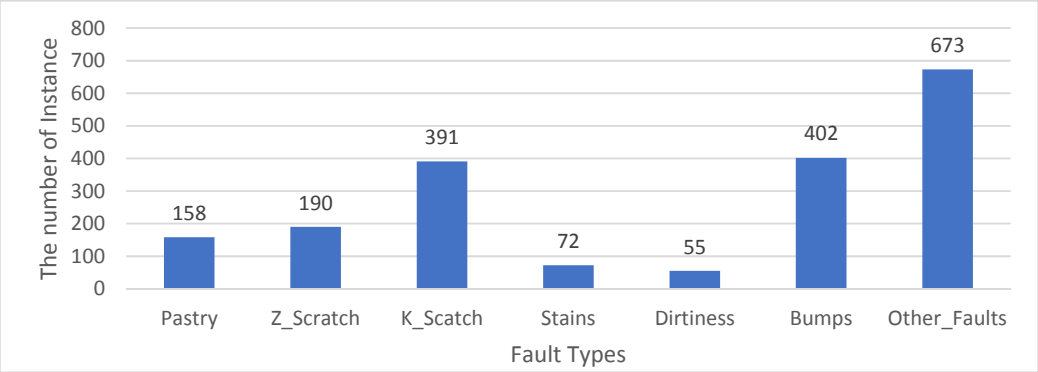

Figure 1 Distribution of Fault Types

Table 1 Attributes

Attributes	Type	Attributes	Type
X_Minimum	numerical	Edges_Index	numerical
X_Maximum	numerical	Empty_Index	numerical
Y_Minimum	numerical	Square_Index	numerical
Y_Maximum	numerical	Outside_X_Index	numerical
Pixels_Areas	numerical	Edges_X_Index	numerical
X_Perimeter	numerical	Edges_Y_Index	numerical
Y_Perimeter	numerical	Outside_Global_Index	numerical
Sum_of_Luminosity	numerical	LogOfAreas	numerical
Minimum_of_Luminosity	numerical	Log_X_Index	numerical
Maximum_of_Luminosity	numerical	Log_Y_Index	numerical
Length_of_Conveyer	numerical	Orientation_Index	numerical
TypeOfSteel_A300	numerical	Luminosity_Index	numerical
TypeOfSteel_A400	numerical	SigmoidOfAreas	numerical
Steel_Plate_Thickness	numerical		

Table 2 Fault Types

Fault Types	Encoding
Pastry	(1, 0, 0, 0, 0, 0, 0)
Z_Scratch	(0, 1, 0, 0, 0, 0, 0)
K_Scatch	(0, 0, 1, 0, 0, 0, 0)
Stains	(0, 0, 0, 1, 0, 0, 0)
Dirtiness	(0, 0, 0, 0, 1, 0, 0)
Bumps	(0, 0, 0, 0, 0, 1, 0)
Other_Faults	(0, 0, 0, 0, 0, 0, 1)

3 Training Issues

Ward Networks with 3 hidden slabs and GRNN are two architectures recommended for prediction in NeuroShell2, while PNN is recommended for classification. So, we build neural network based on these three architectures for the steel plate faults classification. We just maintain the default activation functions in all architectures.

In the experiments, 25 percent of the dataset (485 in 1,941 instances) is extracted as the test set. In order to avoiding the accuracy difference resulting from different test sets, the same test set is used in all experiments. The trained neural networks are applied to the whole dataset to evaluate the training results.

We conduct a series of experiments dividing into 3 rounds based on different architectures, 6 of which are shown in this paper, to find the most suitable architecture and training parameters for the fault classification. Round 1 experiments apply the Ward Networks using 27 inputs and 7 outputs in the original dataset file. Experiments 1 to 3 change the number of hidden neurons. In Round 2, experiments 4 apply GRNN with 27 inputs and 7 outputs. In Round 3, experiment 5 apply PNN with 27 inputs and 7 output. While in experiment 6, we reduce the 27 inputs to 16 inputs (Table 3) according to the contribution factors in Round 1

For all training models, we use 1-out-of N encoding to represent the output. the calibration is set to 200, so the smoothing factor is automatically computed in GRNN and PNN. In Ward Networks, if the training set has no improvement within 2000 epochs, or the learning events exceeds 20000, or the minimum average error is smaller than 0.002, the training will be stopped. In GRNN and PNN, if the results have no improvement for 20 generations, the training will be stopped.

As for the result processing, for Ward Network and GRNN with continuous result values, the largest value of the result in one instance is processed to 1 while others is processed to 0. For PNN, the results are already expressed by 0 or 1.

Table 3 16 Input attributes in Round 3

X_Minimum	X_Maximum	Y_Minimum	Y_Maximum
Pixels_Areas	Minimum_of_Luminosity	Maximum_of_Luminosity	Length_of_Conveyer
TypeOfSteel_A300	TypeOfSteel_A400	Steel_Plate_Thickness	Edges_Index
Empty_Index	Square_Index	Edges_X_Index	Luminosity_Index

4 Results

4.1 Round 1: Ward Networks with different hidden neurons

In Ward Networks with 3 Hidden Slabs, each slab has a different activation function offering three ways of viewing the data.

As an initial experiment, we use a default number of hidden neurons, decided by 1/2(Inputs + Outputs) + $N(P)^{1/2}$. N(P) is the number of patterns in the dataset. So experiment 1 implements 3×19 hidden neurons ($1/2\times(27 + 7) + 1941^{1/2} \approx 19\times3$). We respectively decrease and increase the hidden neurons in experiment 2 and experiment 3. These 3 experiments share the same learning rate and momentum.

The results are shown in Tables 4, 5, and 6. In 3 experiments, the column accuracy rate of Z_Scratch is around 90%, while other type accuracy rates differ widely. It is easy to find that none of these experiment results are satisfactory. There are also other experiments done to different learning rates and momentum, but still with no much changes. So, Ward Networks may not suitable for this problem.

Table 4 Model 1-1: 27-19×3-7 architecture, learning rate: 0.1, momentum: 0.1

Actual / Output	Pastry	Z_Scratch	K_Scatch	Stains	Dirtiness	Bumps	Other_Faults	Row Accuracy
Pastry	87	0	162	1	10	26	28	27.71%
Z_Scratch	4	171	71	29	0	19	21	54.29%
K_Scatch	0	1	134	2	0	14	5	85.90%
Stains	0	0	0	32	0	5	4	78.05%
Dirtiness	0	0	0	0	22	2	3	81.48%
Bumps	21	0	0	1	8	230	96	64.61%
Other_Faults	46	18	24	7	15	106	516	70.49%
Column Accuracy	55.06%	90.00%	34.27%	44.44%	40.00%	57.21%	76.67%	

Table 5 Model 1-2: 27-15×3-7 architecture, learning rate: 0.1, momentum: 0.1

Actual / Output	Pastry	Z_Scratch	K_Scatch	Stains	Dirtiness	Bumps	Other_Faults	Row Accuracy
Pastry	112	2	35	29	11	50	36	40.73%
Z_Scratch	5	166	236	1	0	25	26	36.17%
K_Scatch	0	0	107	6	0	28	10	70.86%
Stains	1	0	0	32	0	19	4	57.14%

Dirtiness	0	0	0	0	31	4	5	77.50%
Bumps	7	1	1	1	2	192	76	68.57%
Other_Faults	33	21	12	3	11	84	516	75.88%
Column Accuracy	70.89%	87.37%	27.37%	44.44%	56.36%	47.76%	76.67%	

Table 6 Model 1-3: 27-23×3-7 architecture, learning rate: 0.1, momentum: 0.1

Actual / Output	Pastry	Z_Scratch	K_Scatch	Stains	Dirtiness	Bumps	Other_Faults	Row Accuracy
Pastry	83	0	82	10	5	37	24	34.44%
Z_Scratch	9	162	78	4	1	12	19	56.84%
K_Scatch	0	0	208	11	0	6	11	88.14%
Stains	2	0	0	40	1	12	4	67.80%
Dirtiness	6	0	0	0	38	18	17	48.10%
Bumps	8	1	4	2	1	200	63	71.68%
Other_Faults	50	27	19	5	9	117	535	70.21%
Column Accuracy	52.53%	85.26%	53.20%	55.56%	69.09%	49.75%	79.49%	

4.2 Round 2: GRNN with 27 Inputs

GRNN is usually used to train quickly on sparse data sets with continuous valued outputs. GRNN is a type of supervised network. It responds much better than backpropagation to most types of problems.

In this Experiment, we apply GRNN to train the model 2-1. In Table 7, the accuracy rates for all fault types are higher than 75%, far better than previous experiments using Ward Networks. The average column accuracy rate is 86.1% much higher than around 60% in experiments 1-3.

Table 7 Model 2-1: 27-1941-7 architecture

Actual / Output	Pastry	Z_Scratch	K_Scatch	Stains	Dirtiness	Bumps	Other_Faults	Row Accuracy
Pastry	122	0	1	0	1	19	8	80.79%
Z_Scratch	4	186	7	0	0	25	23	75.92%
K_Scatch	0	0	378	1	0	2	3	98.44%
Stains	0	0	0	69	0	4	2	92.00%
Dirtiness	2	0	0	0	51	5	4	82.26%
Bumps	10	0	1	1	1	332	69	80.19%

Other_Faults	20	4	4	1	2	15	564	92.46%
Column Accuracy	77.22%	97.89%	96.68%	95.83%	92.73%	82.59%	83.80%	

4.3 Round 3: PNN with Different inputs

Similar with GRNN, PNN is also suitable for training sparse data sets. Rather than creating continuous valued outputs, PNN separates data into a specified number of output categories. Thus, the results are ones or zeros which need no much later processing.

In this Round, we firstly train a PNN model with 27 Inputs. After that, an extra experiment is conducted for comparison as clarified in Section 3.

4.3.1 PNN with 27 Inputs

In this Experiment, we apply PNN to train the model 3-1. In Table 8, the accuracy rates for all fault types are higher than 85%, and the average column accuracy rate is 92.04% higher than that of experiment 4 using GNN.

Table 8 Model 3-1: 27-1941-7 architecture

Actual / Output	Pastry	Z_Scratch	K_Scatch	Stains	Dirtiness	Bumps	Other_Faults	Row Accuracy
Pastry	137	0	0	0	1	6	18	84.57%
Z_Scratch	4	190	0	0	0	4	14	89.62%
K_Scatch	0	0	389	0	0	0	3	99.23%
Stains	0	0	0	71	0	0	2	97.26%
Dirtiness	3	0	0	0	52	2	4	85.25%
Bumps	1	0	0	0	1	377	26	93.09%
Other_Faults	13	0	2	1	1	13	606	95.28%
Column Accuracy	86.71%	100.00%	99.49%	98.61%	94.55%	93.78%	90.04%	

4.3.2 PNN with 16 Inputs

In this Experiment, we just select 16 out of 27 attributes as the inputs to train a PNN model 3-2. In Table 9, the accuracy rates for all fault types are also higher than 85%. The average column accuracy rate is 90..52% lower than that of experiment 5 with 27 inputs. But the accuracy rates for Pastry and

K_Scatch are little higher than those of experiment 5.

Table 9 Model 3-2: 16-1941-7 architecture

Actual / Output	Pastry	Z_Scratch	K_Scatch	Stains	Dirtiness	Bumps	Other_Faults	Row Accuracy
Pastry	139	1	0	0	0	5	16	86.34%
Z_Scratch	3	187	0	0	0	3	15	89.90%
K_Scatch	0	0	390	0	0	0	11	97.26%
Stains	0	0	0	71	0	2	9	86.59%
Dirtiness	1	0	0	0	52	3	4	86.67%
Bumps	9	2	1	0	2	374	24	90.78%
Other_Faults	6	0	0	1	1	15	594	96.27%
Column Accuracy	87.97%	98.42%	99.74%	98.61%	94.55%	93.03%	88.26%	

5 Limitations

From the experiment results, obviously, the Ward Network may not suitable for this classification problems of this steel plate fault dataset, even if a lot of parameter adjustments are made. But different activation functions for Ward Networks have not been considered in our experiments. PNN is the most suitable architecture for this problem. But it may become too slow to be feasible if there are more than 2000 patterns. On the other hand, the dataset has an uneven distribution so that the instances of some fault types are not enough, so the results may not be convincible. Besides, in future works, more architectures can be applied to pursue a better result of fault classification.

6 Conclusion

In this paper, we present the classification of steel plate fault by using NeuroShell2 to build Neural Networks. In our experiments, Ward Networks, GRNN and PNN are used to train the models. According to the 3 round experiments, in Ward Networks experiments, the accuracy rates of some fault types may be higher but with some other rates extremely low, so it is not a good architecture for this problem. GRNN is much better. PNN with 27-1941-7 architecture is considered to be the best one for this classification with higher average accuracy rates in this research. Experiment 6 in Round 3 aims to explore part of attributes possibly contributing more to the classification results. The unused attributes may somehow affect several type classifications, though showing a lower contribution factor in Ward Networks.

References

Faiz, J., & Ojaghi, M. (2009). Different indexes for eccentricity faults diagnosis in three-phase squirrel-cage induction motors: a review. Mechatronics, 19(1), 2-13.

Isermann, R. (2005). Model-based fault-detection and diagnosis – status and applications. Annual Reviews in Control, 29(1), 71-85.

Rajakarunakaran, S., Venkumar, P., Devaraj, D., & Rao, K. S. P. (2008). Artificial neural network approach for fault detection in rotary system. Applied Soft Computing, 8(1), 740-748.