Machine Learning Based Fault Prediction for Real-time Scheduling on Shop-floor

WENDA WU

Master in Science
Date: November 19, 2017
Supervisor: Wei Ji
Examiner: Lihui Wang
Swedish title: Detta är den svenska översättningen av titeln
School of Electrical Engineering

Abstract

English abstract goes here. Need result from algorithms evaluation to finish this part.

Sammanfattning

Träutensilierna i ett tryckeri äro ingalunda en oviktig faktor, för trevnadens, ordningens och ekonomiens upprätthållande, och dock är det icke sällan som sorgliga erfarenheter göras på grund af det oförstånd med hvilket kaster, formbräden och regaler tillverkas och försäljas Kaster som äro dåligt hopkomna och af otillräckligt.

Contents

Chapter 1

Introduction

We use the *biblatex* package to handle our references. We therefore use the command `parencite` to get a reference in parenthesis, like this [**heisenberg2015**]. It is also possible to include the author as part of the sentence using `textcite`, like talking about the work of **einstein2016**.

1.1 Research Question

The main goal of this project research, in general, is to find a relative best algorithm within the scope of machine learning that has the best performance regarding fault prediction of real-time scheduling on shop floor. Further, the question can be described in minor parts.

First of all, what the formation of data is like. Lacking real data from factory directly, a potential formation of data which is close to real shop floor data form is created and presented in this project. This first question is answered in detail in Chapter 2 with a separate section.

Second, with given dataset, various algorithms within the sub-field of machine learning are chosen. The motivation of choosing algorithms is that we try to cover as many fields as possible, including regression analysis, Bayesian classifier, artificial neural network, and the recently uprising field of deep learning, in order to find the relative best algorithms possible, as described in Chapter 2.

With data and algorithms settled, the remaining question is to how to evaluate the performance of each algorithm. This question is an-

swered in Chapter 3 with the discussion of project result.

1.2 Background

Chapter 2

Methods

To solve the problem of fault prediction for real-time scheduling on shop-floor, 10 commonly used machine learning algorithms are selected. 9 of the selected algorithms are used to train models for fault prediction including Logistic Regression, Naive Bayes Classifier, Back-propagation Neural Network(BPNN), Support Vector Machine(SVM), Convolutional Neural Network(CNN), Deep Belief Network(DBN), Ensemble Learning, Radial Based Function Neural Network(RBFNN) and Stacked Auto-encoders(SAEs). The last one of selected algorithms is Hopfield Network, which is used to counter noise and cleanse the data. The 10 algorithms are described in detail in separate sections of this chapter.

2.1 Logistic Regression

Logistic regression, developed by David Cox in 1958, is a regression model that describes data and explains relationship between binary categorical dependent value and one or more nominal, ordinal, interval or ratio-level independent variables. Like all regression analysis, logistic regression performs predictive analysis, leading to its broad applicability ranging from investigating changes in birthweight for term singleton infants in Scotland (Sandra.R.Bonellie, Journal of Clinical Nursing[1]) to serious injuries associated with motor vehicle crashes(Douglas W.Kononen, Carol A.C.Flannagan and Stewart C.Wang, Accident Analysis and Prevention[2]) and assessing groundwater vulnerability to contamination in Hawaii(Alan Mair, Aly I.El-Kadi, Journal of Contaminate Hydrology[3]). In this project, we use logistic regression to

analyse shop-floor data for real-time fault prediction. Since the dependent data has only two categories, 0 for good and 1 for fault, logistic regression is fit for the purpose. Other alternative machine learning algorithms are evaluated in this project as well, similar to the work of Daniel Westreich, Justin Lessler and Michael Jonsson Funk on propensity score estimation, Journal of Clinical Epidemiology[4]

After simplifying the research problem to a binary classification problem, which is predicting binary valued labels $y_i \in \{0, 1\}$ as fault and normal based on the i'th example x_i, we try to learn a function of the form:

$$P(y = 1|x) = h_\theta(x) = \frac{1}{1 + exp(-\theta^{\mathrm{T}} x)} \equiv \sigma(\theta^{\mathrm{T}} x) \tag{2.1}$$

$$P(y = 0|x) = 1 - P(y = 1|x) = 1 - h_\theta(x) \tag{2.2}$$

where $\sigma(z) \equiv \frac{1}{1+exp(-z)}$ is the logistic function that push the value of $\theta^{\mathrm{T}} x$ into range [0,1] in order to achieve probabilistic interpretation. The cost function evaluating h_θ can be described as:

$$J(\theta) = -\sum_i (y_i log(h_\theta(x_i)) + (1 - y_i log(1 - h_\theta(x_i))) \tag{2.3}$$

With a cost function measuring the performance of a given hypothesis h_θ the training process is finding the best choice of θ by minimizing cost function $J(\theta)$. In order to minimize the cost function, the gradient of $J(\theta)$ can be expressed as:

$$\nabla_\theta J(\theta) = \sum_i x_i (h_\theta(x_i) - y_i) \tag{2.4}$$

2.2 Naive Bayes Classifier

Speaking of probabilistic classifier in machine learning, it is natural to think of naive Bayes classifiers, a group of simple classifiers developed by using Bayes' theorem under assumptions of strong independence between the features. Since early 1950s, naive Bayes has been extensively studied in various fields including text categorization, automatic diagnostics ect. For example, Satyendr Singh et al. developed a naive Bayes classifier for Hindi word disambiguation (Proceedings

of the 7th ACM India Computing Conference)[1], Rong Zhen et al. combined vessel trajectory clustering technique with naive Bayes classifier for maritime anomaly detection(Journal of Navigation)[2] and even developmental toxicity assessment, performed by Hui Zhang et al. (Reproductive Toxicology)[3]. Other than using the naive Bayes method to solve different problems, researches were conducted to improve the method itself as well. For example, Kalyan Netti and Y Radhika proposed a novel method to minimizing accuracy loss brought by assumption of independence in naive Bayes classifier(2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC))[4]. Zhiyong Yan, Congfu Xu and Yunhe Pan from Zhejiang University improved naive bayes classifier by dividing its decision regions. Moreover, Paraskevas Tsangaratos and Ioanna Ilia performed comparison logistic regression and naive bayes classifier in landslide susceptibility assessment and formed the conclusion that naive bayes classifier outperforms logistic regression(Catena)[5].

A naive Bayes classifier is built according to Bayes' theorem. Let vecotr $x = (x_1, \cdots, x_n)$ represent n independent features for each k classes C_k and in our case $C_k \in \{0, 1\}, k \in \{1, 2\}$. With Bayes' theorem, the conditional probability can be described as:

$$p(C_k|x) = \frac{p(C_k p(x|C_k))}{p(x)} \tag{2.5}$$

Using the chain rule, the joint probability can be derived into:

$$P(C_k, x_1, \cdots, x_n) = p(x_1, \cdot, x_n, C_k) \tag{2.6}$$

$$= p(x_1|x_2, \cdots, x_n, C_k)p(x_2|x_3, \cdots, x_n, C_k) \cdots p(x_{n-1}|x_n, C_k)p(x_n|C_k)p(C_k) \tag{2.7}$$

Assume each feature x_i is conditionally independent of every other feature x_j where $i \neq j$ given class C, the conditional distribution over C is:

$$p(C_k|x_1, \cdots, x_n) = \frac{1}{p(x)} p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{2.8}$$

Thus the corresponding naive Bayes classifier can be expressed as follows:

$$\hat{y} = \underset{k \in \{1, \cdots, K\}}{\arg\max}\, p(C_k) \prod_{i=1}^{n} p(x_i | C_k) \tag{2.9}$$

where the class label is predicted as $\hat{y} = C_k$

2.3 BPNN

The back propagation (BP) neural network algorithm is a multi-layer feedforward network trained according to error back propagation algorithm and is one of the most widely applied neural network models. BP network can be used to learn and store a great deal of mapping relations of input-output model, and no need to disclose in advance the mathematical equation that describes these mapping relations. Its learning rule is to adopt the steepest descent method in which the back propagation is used to regulate the weight value and threshold value of the network to achieve the minimum error sum of square. BP neural network has been studied in many different fields including image recognition: An Optimal Backpropagation Network for Face Identification and Localization-Goutam Sarker(International Journal of Computers and Application), electrical engineering: Flashover forecasting on high-voltage insulators with a back propagation neural net-Manuel Mejia-Lavalle and Guillermo Rodriguez-Ortiz(Canadian Journal of Electrical and Computer Engineering) as well as on-line recommendation: Ubiquitous Hotel Recommendation Using aFuzzy-Weighted-Average and Back propagation-Network Approach-Toly Chen(International Journal of Intelligent Systems).

For a general feed-forward network with single real input, the training process using back propagation can be divided into three phases. First a forward pass is performed where activities of the nodes are computed layer after layer. Then the backward pass is performed where an error signal δ is computed for each node. Because the value of δ depends on the values of δ in the following layer, this second step is performed backward, from the output layer to the input layer. The final step is weight update. The computations of three phases are described in detail as following.

First let x_i denote the activity level in node i in the output layer,

h_j be the activity in node j in the hidden layer $\phi(x)$ be the non-linear transfer function of the network. The output signal h_j becomes

$$h_j = \phi(h_j^*) \tag{2.10}$$

where h_j^* denotes the summed input signal to node j as $h_j^* = \sum_i w_{j,i} x_i$ here $w_{j,i}$ is the weight between node j and i.
After same operation in the nest layer with k nodes, the final output can be written as

$$o_k = \phi(o_k^*) o_k^* \qquad = \sum_j v_{k,j} h_j \tag{2.11}$$

The second phase is the backward pass computation. In this phase the error δ is computed as

$$\delta_k^o = (o_k - t_k \cdot \phi'(o_k^*)) \tag{2.12}$$

The error δ in the next layer is

$$\delta_j^h = \left(\sum_k v_{k,j} \delta_k^{(o)} \right) \cdot \phi'(h_j^*) \tag{2.13}$$

In the final step, the weight update then becomes

$$\Delta w_{j,i} = -\eta x_i \delta_j^{(h)} \tag{2.14}$$

$$\Delta v_{k,j} = -\eta h_j \delta_k^{(o)} \tag{2.15}$$

2.4 SVM

Support vector machines(SVM), introduced byVapnik and coworkers in the 1990s, are a family of supervised learning models widely used in classification and regression analysis. Mapping the training data into two separate categories by constructing a hyperplane in a high- or infinite-dimensional space, SVM algorithm creates a non-probabilistic binary classification model. To solve the problem of non-linear separable data, SVM introduced kernel function for reducing dimensions and increasing computational efficiency.
SVM has been widely used in research in recent years, like other machine learning algorithms. Pao-Shan Yu et al. compared SVM and random forest based on real-time radar-derived rainfall forecasting and

concluded that SVM outperforms random forest(Journal of Hydrology)[1]. Verena Klass et al. used a SVM model to capture lithium-ion battery dynamics(Journal of Power Sources)[2]. Ni Dong et al. conducted crash prediction at the level of traffic analysis zones with SVM model(Accident Analysis and Prevention)[3].

In short, SVM is an algorithm which aims at finding a $p-1$ dimensional hyperplane that best separate given data points as p dimensional vectors.

Suppose the given training data has the form of $(\vec{x_1}, y_1), ..., (\vec{x_n}, y_n)$ where y_i are transformed from $0, 1$ to $-1, 1$ a hyperplane can be written as $\vec{w} \cdot \vec{x} - b = 0$. Therefore, finding the hyperplane with the largest margin to both sides of data becomes the problem of maximizing the distance between the two hyperplanes of data labeled as $y = -1$ and $y = 1$. These two hyperplanes can be described as

$$\vec{w} \cdot \vec{x} - b = 1 \tag{2.16}$$

$$\vec{w} \cdot \vec{x} - b = 1 \tag{2.17}$$

The distance between two hyperplanes is $\frac{2}{\|\vec{w}\|}$ and to maximize this distance we minimize $\| \vec{w} \|$. To prevent data from falling into the margin the following constraint is added: for each i either

$$\vec{w} \cdot \vec{x_i} - b \geq 1, if y_i = 1 \tag{2.18}$$

or

$$\vec{w} \cdot \vec{x_i} - b \leq -1, if y_i = -1 \tag{2.19}$$

And the above can be summarised as

$$y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1, \text{for all } 1 \leq i \leq n \tag{2.20}$$

Since in our case, the data are not linearly separable, the hinge loss function is introduced with the form

$$max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b)) \tag{2.21}$$

so that for the data on the wrong side of the margin, the value of hinge loss function and the distance from the hyperplane is proportional.

The problem then becomes minimizing

$$\left[\frac{1}{n} \sum_{i=1}^{n} max(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b)) \right] + \lambda \| \vec{w} \|^2 \tag{2.22}$$

The process of reducing the above expression to a quadratic programming problem is discussed below.

First for each $i \in 1, ..., n$ introduce a variable $\zeta_i = max(0, 1 - y_i(w \cdot x_i - b))$ where ζ_i is the smallest non-negative number that satisfies $y_i(w \cdot x_i - b) \leq 1 - \zeta_i$.

The optimization problem can be written as

$$\text{minimize } \frac{1}{n}\zeta_i + \lambda \parallel w \parallel^2$$
$$\text{subject to } y_i(w \cdot x_i - b) \leq 1 - \zeta_i \text{ and } \zeta_i \leq 0, \text{ for all } i.$$

After solving the Lagrangian dual of the above, we get the simplified problem

$$\text{maximize } f(c_1...c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j$$
$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, and 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.$$

where the variables c_i are defined such that $\vec{w} = \sum_{i=1}^{n} c_i y_i \vec{x_i}$
The offset b can be computed by solving $y_i(\vec{w} \cdot x_i - b = 1) \longleftrightarrow b = \vec{w} \cdot \vec{x_i} - y_i$
To learn a non linear classification rule for transformed data points $\phi(\vec{x_i})$ a kernel function k is needed, which satisfies $k(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$.
The vector $\vec{w}$ in the transformed space satisfies

$$\vec{w} = sum_{i=1}^{n} c_i y_i \phi(\vec{x_i}) \tag{2.23}$$

where c_i can be obtained by solving

$$\text{maximize } f(c_1...c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i k(\vec{x_i}, \vec{x_j}) y_j c_j$$
$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i.$$

and then

$$-b = \vec{w} \cdot \phi(\vec{x_i}) - y_i = \left[\sum_{i=1}^{n} c_k y_k \phi(\vec{x_k}) \cdot \phi(\vec{x_i})\right] - y_i$$
$$= \left[\sum_{k=1}^{n} c_k y_k k(\vec{x_k}, \vec{x_i})\right] - y_i$$

Eventually the new data points can be classified after computing

$$\vec{z} \mapsto sgn(\vec{w} \cdot \phi(\vec{z}) - b) = sgn\left(\left[\sum_{i=1}^{n} c_i y_i k(\vec{x_i}, \vec{z})\right] - b\right). \quad (2.24)$$

2.5 CNN

Convolutional neural network(CNN) is a class of deep feed-forward artificial neural network. A convolutional neural network is comprised of one or more convolutional layers (often with a sub-sampling step) and then followed by one or more fully connected layers as in a standard artificial neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.
Along with the development of GPU computation and other supporting techniques, CNN has achieved significant result on image recognition, video analysis, natural language processing and other fields. More and more researchers are focusing on CNNs now. Wenqin Sun et al. developed an enhanced deep CNN for breast cancer diagnosis(Computerized Medical Imaging and Graphics)[1]. Mehdi Hajinoroozi et al. used deep CNN to predict driver cognitive performance with EEG data(Signal Processing: Image Communication)[2]. Yuanyuan Zhang et al. developed an adaptive CNN and explored its performance on face recognition(Neural Processing Letters)[3]. Hugo Alberto Perlin et al. used CNN as an approach to extract multiple human attributes from image(Pattern Recognition Letters)[4].Oliver Janssens et al. used CNN for fault detection on rotating machinery(Journal of Sound and Vibration)[5].

2.6 DBN

Deep belief nets(DBNs) are probabilistic generative models that are composed of multiple layers of stochastic, latent variables. The latent variables typically have binary values and are often called hidden units or feature detectors. The top two layers have undirected, symmetric connections between them and form an associative memory. The lower layers receive top-down, directed connections from the layer above. The states of the units in the lowest layer represent a data vector.(Geoffrey E. Hinton (2009) Deep belief networks. Scholarpedia, 4(5):5947.)[1] DBN can be viewed as a composition of simple, unsupervised networks like restricted Boltzmann machines(RBMs) or auto-encoders. In the structure of DBN, each sub-network's hidden layer serves as the visible layer for the next. This structure leads to a layer-by-layer fast training process.
Deep belief nets have been widely used for image and video sequence recognition, as well as fault diagnosis and prediction in other fields. For example, Yan Liu et al. developed a discriminative DBN for visual data classification which outperforms both representative semi-supervised classifiers and existing deep learning techniques(Pattern Recognition)[2]. Hai B. Huang et al. combined regression-based DBN with SVM and performed sound quality prediction of vehicle interior noise with their model where the result shows the combined model outperforms four conventional machine learning methods multiple linear regression(MLR), back-propagation neural network(BPNN), general regression neural network(GRNN) and SVM(Applied Acoustic)[3]. Furao Shen et al. used DBN for exchange rate forecasting in finance and found their model better than typical forecasting methods such as feed forward neural network (FFNN)(Neurocomputing)[4]. De-long Feng et al. developed a DBN model for fault-diagnosis simulation study of gas turbine engine(Frontiers of Information Technology and Electronic Engineering)[5].

2.7 Ensemble Learning

Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve

a particular computational intelligence problem. Ensemble learning is primarily used to improve the (classification, prediction, function approximation, etc.) performance of a model, or reduce the likelihood of an unfortunate selection of a poor one. Other applications of ensemble learning include assigning a confidence to the decision made by the model, selecting optimal (or near optimal) features, data fusion, incremental learning, nonstationary learning and error-correcting. (Robi Polikar (2009) Ensemble learning. Scholarpedia, 4(1):2776.)[1]
Kunwar P. Singh et al. developed tree ensemble models for seasonal discrimination and air quality prediction and found that their models outperforms SVMs(Atmospheric Environment)[2]. Anders Elowsson and Anders Friberg used ensemble learning model to predict performed dynamics of music audio and found the result well above that of individual human listeners(The Journal of the Acoustical Society of America 141, 2224 (2017))[3]. Kunwar P. Sing and Shikha Gupta used a few simple non-quantum mechanical molecular descriptors as an ensemble classifier to discriminate toxic and non-toxic chemicals and predict toxicity of chemicals in multi-species(Toxicology and Applied Pharmacology)[4]. Jinrong Bai and Junfeng Wang developed a multi-view ensemble learning model to improve malware detection and successfully reduced false alarm rate(Security and Communication Networks)[5].

2.8 RBFNN

Radio basis function network(RBF network), first formulated in a 1988 paper by Broomhead and Lowe, is a class of artificial neural networks that uses radial basis functions as activation functions The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. Till now, RBF networks are widely used in function approximation, time series prediction, classification, and system control.
Barnali Dey et al. introduced computational intelligence to spectrum sensing in Cognitive Radio by using RBF network as a model and found their model better than conventional ones(Intelligent Automation and Soft Computing)[1]. Hitoshi Nishikawa and Seiichi Ozawa proposed a novel type of RBF network for multitask pattern recogni-

tion(Neural Processing Letters)[2]. H. Z. Dai et al. developed an improved RBF network for structural reliability analysis(Journal of Mechanical Science and Technology 25 (9) (2011) 2151 2159)[3]. Mohammad Reza Sabour and Saman Moftakhari Anasori Movahed applied RBF neural network to predict soil sorption partition coefficient(Chemosphere)[4].

2.9 SAEs

A stacked auto-encoder is a neural network consisting of multiple layers of sparse auto-encoders in which the outputs of each layer is wired to the inputs of the successive layer. Stacked autoencoders take advantage of the greedy layerwise approach for pretraining a deep network by training each layer in turn. To do this, first train the first layer on raw input to obtain parameters of weights and biases. Use the first layer to transform the raw input into a vector consisting of activation of the hidden units. Train the second layer on this vector to obtain parameters of weights and biases.Repeat for subsequent layers, using the output of each layer as input for the subsequent layer.
This method trains the parameters of each layer individually while freezing parameters for the remainder of the model. To produce better results, after this phase of training is complete, fine-tuning using back propagation can be used to improve the results by tuning the parameters of all layers are changed at the same time.
Dal Xi Wu et al. constructed a stacked denoising auto-encoder architecture with adaptive learning rate for action recognition based on skeleton features and found their results with better robustness and accuracy than that of classic machine learning models including SVM, REFTrees, Linear Regression, RBF Network and Deep Belief Network(Applied Mechanics and Materials)[1]. Heung-Il Suk et al. used stacked auto-encoders for diagnosis of Alzheimer's disease and its prodromal stage mild cognitive impairment(Brain Structure and Function)[2]. Earnest Paul Ijjina and Krishna Mohan C built a stacked auto-encoder for human actions classification using pose based features(Pattern Recognition Letters)[3].

2.10 Hopfield Network

Hopefield network is a form of recurrent artificial neural network described by Little in 1974 and popularized by John Hopfield in 1982. Despite its recurrent structure, a main feature of Hopfield network is the use of binary threshold units, meaning the units only take on two different values for their states and the value is determined by whether or not the units' input exceeds their threshold. Since Hopefield network has the ability of association, in machine learning meaning the ability to retrieve distorted patterns, it provides a way to understand human memory as well.
Many researches have been conducted on Hopfield network in recent years. For example Dayal Pyari Srivastava et al. used Hopefield network to model the microtubules in the brain(International Journal of General Systems)[1]. Yu.A.Basistov and Yu,G.Yanovskii performed comparison between Bayes, correlation algorithm and Hopfield network in the field of image recognition finding Bayes and Hopfield as equals, outperforming correlation algorithm in general(Pattern Recognition and Image Analysis)[2]. Furthermore, Hopfield network was also used in biochemistry by Quan Zou et al. for RNA secondary structure prediction(Computers in Biology and Medicine)[3]. Jiakai Li and Gursel Serpen equipped wireless sensor network with computational intelligence and adaptation capability using Hopfield network model(Applied Intelligence)[4].

Chapter 3

Results

Appendix A

Unnecessary Appended Material