2006

Grinding wheel condition monitoring with boosted classifiers

Fengming Tang

Louisiana State University and Agricultural and Mechanical College, ftang1@lsu.edu

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_theses

Part of the Operations Research, Systems Engineering and Industrial Engineering Commons

Recommended Citation


https://digitalcommons.lsu.edu/gradschool_theses/3092

This Thesis is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Master’s Theses by an authorized graduate school editor of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.
GRINDING WHEEL CONDITION MONITORING WITH
BOOSTED CLASSIFIERS

A Thesis
Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Industrial Engineering

in

The Department of Industrial and Manufacturing Systems Engineering

By
Fengming Tang
B.S. In M.E., Northwest Univ. of Light Industry, China,1999
M.S. In M.E., Xian Jiaotong University, China, 2002
December, 2006
Acknowledgements

I would like to express my sincere appreciation to my advisor, Dr. Liao, for his kind help, guidance, support and encouragement throughout all my study.

I would also like to thank Dr. Chen and Dr. Jiang for their encouragements and being the committee members.

I would like to thank Dr. Wells for providing an assistantship position to me so that I can complete my study.

Also I would thank my husband for his unbelievable support and understanding through my entire project.
Table of Contents

Acknowledgements ...........................................................................................................ii

Abstract................................................................................................................................iv

Chapter 1 Introduction and Literature Review.................................................................1

Chapter 2 Grinding Experiments and Data Description.......................................................5
  2.1 The First Set of Grinding Experiments.................................................................5
  2.2 The Second Set of Grinding Experiments...............................................................6

Chapter 3 Feature Extraction...............................................................................................8
  3.1 Autoregressive Time Series Model.........................................................................8
  3.2 Regression Model with Auto Correlated Errors .....................................................9
  3.3 Model Order Decision ...........................................................................................9
  3.4 Model Coefficients Estimating...............................................................................11

Chapter 4 Classification ....................................................................................................13
  4.1 AdaBoost ...............................................................................................................14
  4.2 Averaged Boosting.................................................................................................14
  4.3 AdaBoost.M1 ..........................................................................................................15
  4.4 Weak Learner .........................................................................................................19
    4.4.1 Prototype Algorithm .......................................................................................19
    4.4.2 Knn Algorithm ..............................................................................................19
  4.5 AdaBoost-M and A-boosting-M ............................................................................20

Chapter 5 Results..............................................................................................................22
  5.1 Results for the Data from the First Set of Experiments .........................................22
    5.1.1 The Comparisons of Boosted Methods against the Corresponded Weak Learners ........................................................................................................23
  5.1.2 The Comparisons of Adaboost-M and A-boosting-M against Adaboost.M1 ..............................................................................................................26
  5.2 Results for the Data from the Second Set of Experiments ....................................29

Chapter 6 Conclusion .......................................................................................................35

References .........................................................................................................................36

Vita .....................................................................................................................................39
Abstract

In this thesis, two data sets collected in grinding process under different cutting and wheel conditions were studied. One is the cutting forces in three directions, i.e. X, Y and Z, collected under two different cutting conditions. The other one is the acoustic emission (AE) signals collected under different wheel conditions (sharp and dull). For the goal of grinding wheel condition monitoring, the regression model with autocorrelated errors was proved to be effective and was used to extract features from signals in this study. The coefficients of the models served as the features used in the classification step that employed boosting method. Based on the AdaBoost and A-boosting algorithms which can only be used in two classes situation, two improved boosting methods called Adaboost-M and A-boosting-M, which can be used to classify multiple classes, are proposed. With the forces data set, we compared Adaboost-M and A-boosting-M against the traditional AdaBoost.M1 and the corresponding weak learners (KNN and Prototype). The accuracies of Adaboost-M and A-boosting-M are higher than that of AdaBoost.M1 and the weak learners in our application. With the AE data set, our focus is to recognize the signals collected when the wheels were dull from the signals collected when the wheels were sharp. The AdaBoost, A-boosting and the corresponding weak learners (KNN and Proto) were used. The results indicate that (i) boosting does not improve the effectiveness of $k$-nearest neighbor but greatly improve the effectives of the prototype classifier, (ii) depending upon the data, AdaBoost or A-
Boosting might produce higher classification accuracy, (iii) the error of false positive is higher than the error of false negative for the better classifiers.

Based on the study, the combined use of AR models for feature extraction and boosted algorithms for classification are proved to be a viable approach for grinding wheel condition monitoring.
Chapter 1 Introduction and Literature Review

Tool wear monitoring is very important in metal cutting process. Tool wear is a primary factor affecting the quality of production. A worn tool can increase rejects of production and can cause problems to both machine and personnel at the same time[1,2]. A lot of works has been done on the signal of tool wear monitoring for different cutting processes.

There are various methods for tool wear monitoring in the literature. According to the tool monitoring sensing, these methods have been classified into direct and indirect measurement techniques by Micheletti et al[3]. Direct methods are those that measure the actual tool wear, such as optical scanning of the tool tip, electrical measurement of contact resistance between the tool and work pieces, and radioactive analysis of the chip[4]. This kind of methods hardly be used online. However, the advanced metal cutting processes require accurate in-process tool wear monitoring and fast failure detection. Consequently, most research have been done with indirect methods in the literature. Indirect methods concerned with measuring some process born features from tool wear, such as cutting forces, torques, vibrations, roughness of machined surface, temperature and thermoelectric effects, and acoustic emissions.

For indirect methods, most of them have similar structure, regardless of the cutting processes being turning, milling, or grinding. Normally, the basic structure includes four parts: signal collection, pre-processing, feature extraction and decision making.
The first part is signal collection, using single or multi sensor to collect signal produced in the cutting process. The cutting forces, vibrations and acoustic emissions are used in the literature mostly.

The second part is pre-processing. In this part, the signal are processed to get rid of noise and to separate cutting signals from non-cutting signals.

The third part is feature extraction. Algorithms of feature extraction are diversified. Different algorithms deal with different situation. Some of them can only tell the worn tools from fresh ones, yet some of them can give a fuzzy concept of how much the tool is worn. Some of them can only be used in turning processing and some of them can only be used in milling processing. Typically, there are two approaches to extract the features [1]. One is the mechanics method, in which the mechanics of cutting process are analyzed and the models are built based on the properties of the cutting processes. The other approach is statistical method, in which the signals are treated as time series. For the later method, ARIMA models or AR models are proved to be very effective in milling or turning processes and a lot of works have been done. The first and second differences of the average cutting force synchronized with cutter teeth were used by Altintas et al.[5] to detect cutter breakage and changes of the cutting condition in a milling operation. A 28th-order autoregressive model with Kalman filtering for the cutting torques was employed by Takata et al.[6]. A 15th order autoregressive model with adaptive signal processing technique was proposed by Lan and Naerheim[7].
which extended Takata’s approach.

The fourth part is decision making. Normally, it’s much easier for people to build several low accuracy learning algorithms than to build a very high accuracy learning algorithm. Boosting is a general method for improving the accuracy of any given learning algorithm. Schapire[8] gave a very good overview about boosting methods in 2001. The most original ideal of boosting, deriving from Valiant’s PAC (probably approximately correct) learning model[9], was first proposed by Kearns and Valiant[10]. The first provable polynomial-time boosting algorithm was present by Schapire[11] in 1989. In 1990, Freund [12] developed an efficient boosting algorithm and Drucker, Schapire and Simard[13] carried out the first experiment with the boosting algorithms on an OCR task. The AdaBoost algorithm solved many of the practical difficulties of the earlier boosting algorithms was introduced in 1995 by Freund and Schapire[14]. The AdaBoost algorithm can only be used as a binary class classification. In order to deal with the multi-class situation, several methods have been proposed by extending the AdaBoost algorithm. The AdaBoost.M1 is the most straightforward generalization of AdaBoost, which require the base learner to achieve at least 50% accuracy. Some more sophisticated methods have been developed to overcome the shortcoming, such as AdaBoost.MH developed by Schapire and Singer [15], and AdaBoost.M2 developed by Freund and Schapire[16]. Boosting is a data-driven machine learning method which depends on the abundant data. In the case of data being
severely limited, human knowledge might be used to compensate for the lack of data.

Rochery et al. presented a modification of boosting which combines and balances human expertise with available training data.
Chapter 2 Grinding Experiments and Data Description

2.1 The First Set of Grinding Experiments

The first set of grinding experiments were conducted using two different conditions (B1Aa and B2) at the High Temperature Materials Laboratory, Oak Ridge National Laboratory (TN, USA) in year 2000. Table 1 summarizes the grinding parameters used in each condition. For each grinding condition, the grinding wheel was started new and used continuously to grind a ceramic billet in a zigzag pattern. In other word, once set the billet is first ground from left to right, followed by feeding by the right amount, then ground from right to left. The whole sequence repeats till the entire billet is ground. The ceramic material is Coors AD995. The billet size is roughly 102 mm in length, 51 mm in width, and 4.5 mm in thickness. The grinding wheel used is made of synthetic diamonds supplied by the Norton Company.

A dynamometer was used to collect forces in the X, Y and Z directions. The force information collected at each path was recorded as a time series with three dimensions: x, y, and z. Totally, there are 18 records under condition B1Aa with 9 records collected when the table/work moved from left to right and 9 records when the table/work moved from right to left. In addition, there are 15 records under condition B2 with 9 records collected when the table/work moved from left to right and 6 records when the table/wheel moved from right to left. Our objective is to classify the test data to 4 classes which distinguish not only the manufacturing conditions but also the cutting
directions. The four classes are named as odd B1Aa, even B1Aa, odd B2, and even B2.

Table 1 Grinding conditions.

<table>
<thead>
<tr>
<th>Billet Number</th>
<th>B1Aa</th>
<th>B2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cleanup</td>
<td></td>
</tr>
<tr>
<td>Initial Thickness</td>
<td>4.40</td>
<td>3.02</td>
</tr>
<tr>
<td>Target Thickness</td>
<td>4.39</td>
<td>2.00</td>
</tr>
<tr>
<td>Stock to be Removed</td>
<td>0.01</td>
<td>1.02</td>
</tr>
<tr>
<td>Length mm</td>
<td>101.6</td>
<td>101.6</td>
</tr>
<tr>
<td>Width mm</td>
<td>50.8</td>
<td>50.8</td>
</tr>
<tr>
<td>Average Stepover Distance mm</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>Table Speed mm/s</td>
<td>16.7</td>
<td>10.0</td>
</tr>
<tr>
<td>Table Speed mm/min.</td>
<td>1000</td>
<td>600</td>
</tr>
<tr>
<td>Table Speed in/min.</td>
<td>39.38</td>
<td>23.62</td>
</tr>
<tr>
<td>Total Volume of Material to be Removed mm³</td>
<td>51.61</td>
<td>5264.51</td>
</tr>
<tr>
<td>Average Volume of Material Removed Per Linear Pass mm³</td>
<td>2.54</td>
<td>259.08</td>
</tr>
<tr>
<td>Average Material Removal Rate mm³/s.</td>
<td>0.42</td>
<td>25.50</td>
</tr>
<tr>
<td>Grinding Time Per Pass, seconds</td>
<td>6.1</td>
<td>10.2</td>
</tr>
<tr>
<td>Number of Linear Passes</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>Total Grinding Time, seconds</td>
<td>165</td>
<td>274</td>
</tr>
</tbody>
</table>

2.2 The Second Set of Grinding Experiments

In the first set of grinding experiment, we did not collect signals when the wheels were worn. In order to test our method using the real tool worn data, a new set of experiments were conducted at the High Temperature Materials Laboratory in 2005. Alumina (Coors AD995 CAP3) and silicon nitride (GS44) specimens were ground with a resin-bonded diamond wheel (Norton SD220 R75 B56 1/8 of 229 mm diameter) on a 10-horsepower K. O. Lee Vigor Creep Feed Grinder. Grinding conditions must be specified. Grinding procedure shall be given, too. The raw AE signals were collected at 1 MHz. Figure 1 show sample segments of AE raw signals acquired at wheel steady
state and wheel worn out state for the lower material removal rate grinding condition experimented in creep grinding of the alumina specimens. Each AE signal segment has 4000 data points, which equal to approximately a duration of 0.3 grinding wheel revolution. Note that for better visualization in each figure three was added to every point of the top series; two was added to every point of the second top series; and one was added to every point of the second bottom series. Of course original data values were processed in the subsequent analyses.

![Figure 1 Sample AE raw signal segments in creep feed grinding of alumina with high MRR.](image-url)
Chapter 3 Feature Extraction

Normally, a time series model includes two parts: the deterministic part and the stochastic part. The deterministic part consists of trends and seasonality. The stochastic part captures the information that can’t be captured by the deterministic part. For each force collected in the grinding process, linear regression is enough for modeling the deterministic part. After studying the autocorrelation and partial-autocorrelation of the residuals of the deterministic model, it is observed that the residuals have pretty similar autocorrelation and partial-autocorrelation pattern for signals collected from the same cutting condition. Furthermore, the coefficients of AR model built for the residuals are similar to each other. In this paper, regression model with auto correlated errors which consists of two parts: the deterministic part that is a linear regression model, and the stochastic part which is an AR model, is employed.

3.1 Autoregressive Time Series Model

The autoregressive process is a simple mathematical model in which the current value of a series is linearly related to its past values, plus an additive stochastic shock [21].

The $p$ order AR model has the following form,

\[ y_t = (\varphi_1 B + \varphi_2 B^2 + \varphi_3 B^3 + \ldots + \varphi_p B^p) y_t + \varepsilon_t \]

\[ = \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \varphi_3 y_{t-3} + \ldots + \varphi_p y_{t-p} + \varepsilon_t \quad (1) \]

Where, $y_t$ (t=1, 2, ..., $n$) is the signal at time t, $\varphi_1$, $\varphi_2$ ... $\varphi_p$ are the
coefficients of the model. \( B \) is the lag operator, i.e., \( y_t B^k = y_{t-k} \). \( \epsilon_t \) is stochastic shock and \( \epsilon_t \sim N(0, \sigma^2) \).

### 3.2 Regression Model with Auto Correlated Errors

As mentioned before, the model used in this paper is a model consists of two parts, called regression model with auto correlated errors[23]. The model has the following form:

\[
y_t = c + \beta t + \nu_t
\]

\[
\nu_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t
\]

which can be rewritten as

\[
y = c + \beta t + (1 - \phi_1 B^1 - \phi_2 B^2 - \ldots - \phi_p B^p)^{-1} \epsilon_t
\]

where \( y_t (t=1, 2, \ldots, n) \) is the measured force and \( \epsilon_t \sim N(0, \sigma^2) \)

### 3.3 Model Order Decision

Akaike information criterion (AIC) and Durbin-Watson are employed for the selection of appropriate model order [21].

AIC is an estimate of the out-of-sample forecast error variance and it penalizes degrees of freedom. It is used to select the model with the lowest AIC value among competing models. The formula is:

\[
AIC = \exp(2k / T) \sum_{t=1}^{T} e_t^2 / T
\]

where \( k \) is the number of parameters in the model, \( T \) is the number of data point, \( e_t \) is the residual at time \( t \).
Durbin-Watson statistic tests the correlation over time in the residuals. If the residuals made by a model are predictable, then we could improve the forecasts by forecasting the residuals. If the model is good, DW should be around 2.

\[
DW = \frac{\sum_{t=1}^{T} (e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2}
\]  

(5)

After analyzing typical time series along each cutting direction for each cutting condition, the results show that 20 is an acceptable order for most of the situations. Figure 2 and figure 3 plot the AIC and DW values of an x-directional force time series of B1Aa.

![Figure 2 AIC of x-force of B1Aa](image)
3.4 Model Coefficients Estimating

The Yule-Walker method is used to estimate the coefficients. For an AR model

\[ y_t = \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \varphi_3 y_{t-3} + \ldots + \varphi_p y_{t-p} + \epsilon_t \]

where \( \varphi_1, \varphi_2 \ldots \varphi_p \) are the coefficients of the model, \( \epsilon_t \) is stochastic shock. It can be shown that the autocorrelation function \( \rho(\tau) \) (\( \tau = 1 \ldots p \)) is related to the autoregressive parameters \( \varphi_i \) (i=1…p) through the Yule-Walker equation for the autoregressive process (Priestley, 1994):

\[
\begin{align*}
\rho(1) &= \varphi_1 \rho(0) + \varphi_2 \rho(1) + \varphi_3 \rho(3) + \ldots + \varphi_p \rho(p) \\
\rho(2) &= \varphi_1 \rho(1) + \varphi_2 \rho(0) + \varphi_3 \rho(2) + \ldots + \varphi_p \rho(p-1) \\
&\vdots \\
\rho(p) &= \varphi_1 \rho(p) + \varphi_2 \rho(p-1) + \varphi_3 \rho(p-2) + \ldots + \varphi_p \rho(0)
\end{align*}
\]

(6)
The estimation of autocorrelation $\rho(\tau)$ is

$$\hat{\rho}(\tau) = \frac{\sum_{\tau=1}^{N} [(y_{t}\bar{y})] \left[y_{t-\tau}\bar{y}\right]}{\sum_{t=1}^{N} (y_{t}\bar{y})^2}$$

$N$ is the size of the sample.

Plug in $\hat{\rho}(\tau)$, solve equations (6) to get the estimations of coefficients of the AR model.
Chapter 4 Classification

Boosting is a well known method to construct an ensemble of classifiers to obtain higher accuracy. To the best of our knowledge the boosting idea has not been applied to condition monitoring of any cutting tools, including grinding wheels which are subject of our study here.

With the AE data set, two boosting algorithms capable of binary classification are employed in this study since our goal is to distinguish a sharp wheel from a dull wheel. They are AdaBoost [Freund and Schapire, 1997] and A-Boost [Kim, 2003]. Each boosting algorithm requires the use of a weak learner. Many different weak learners such as naïve Bayesian, decision trees, and neural networks have been used in a boosting algorithm. In this study, two weak learners, i.e., prototype classifier and \( k \)-nearest neighbors (KNN) are used in each boosting algorithm, resulting in four boosted classifiers. Their performances are also compared with single prototype classifier and KNN classifier.

With the forces data set, our objective is to classify the test data to 4 classes which distinguish not only the manufacturing conditions but also the cutting directions. Our data is very limited in the study. There are only 9 records, called B1AaO, from the odd path under condition B1Aa, 9 records, called B1AaE, from the even pass under condition B1Aa, 6 records, called B2O, from the odd pass under condition B2 and 9 records, called B2E, from the even pass under condition B2. In this situation, human
knowledge can be used to compensate the limitation of data. We do know that both B1AaO and B1AaE come from condition B1Aa, and B2O and B2E come from condition B2. To deal with this particular application of multi-class classification, a modified boosting method AdaBoost-M is developed based on AdaBoost, and A-boosting-M is developed based on A-boosting. At the same time, Adaboost.M1 is used in order to compare the new methods developed in the study. Two weak learners, prototype and KNN, are used in the boosting procedure.

4.1 AdaBoost

Figure 4 gives the pseudocode of AdaBoost algorithm [16]. The algorithm takes the training set \(\{(x_1, y_1), \ldots, (x_N, y_N)\}\) as the input. Each \(x_i\) belongs to some domain or instance space \(X\) and each \(y_i\) is in the label set \(Y = \{0, 1\}\). AdaBoost calls a given weak or base learning algorithm repeatedly in a series of rounds \(t=1, \ldots, T\). At each round, the weights \(w_i^t\) over the training set are updated according to the hypothesis obtained at this round and the old weights \(w_i^{t-1}\) \((t^1=1, \ldots, t)\). Initially, all the weights are set equally. But on each round, the weights of incorrectly classified examples are increased so that the base learner is forced to focus on the hard examples in the training set.

4.2 Averaged Boosting

‘Averaged boosting’ (A-Boosting) method was proposed by Yongdai Kim[26] in 2003 in order to overcome the over fitting drawback of AdaBoost. According to Kim, the A-Boosting is more resistant to noisy examples than AdaBoost. After large number
of iterations, AdaBoost tends to concentrate weights to a few frequently misclassified examples. Examples with incorrect class labels will persist in being misclassified, hence AdaBoost will mistakenly concentrate the weights on these noisy examples, eventually leading to overfitting. In contrast, A-Boosting gives significant weights to correctly specified cases even after large number of iterations, and reduces the size of noise effect.

Figure 5 is the pseudocode of A-Boosting. Compare with AdaBoost, there are two differences. The first is that the A-Boosting algorithm uses the average of the product of the base hypotheses and coefficients while AdaBoost uses the sum of it. The second difference is that the A-Boosting algorithm calculates the coefficient based on the error rate of the current hypothesis on the original training example while the AdaBoost algorithm uses the updated weights.

4.3 AdaBoost.M1

Figure 6 gives the pseudocode of AdaBoost.M1[16]. AdaBoost.M1, which is a multi-class classifier, is the most straightforward extension of AdaBoost. The main difference is in the replacement of the error \( |h_t(x_i) - y_i| \) for the binary case by \([h_t(x_i) \neq y_i]\) where, for any predicate \([\pi]\) to be 1 if \(\pi\) holds and 0 otherwise. Also, the final hypothesis of the weak hypotheses predicts that label.
**AdaBoost**

**Input**: sequence of $N$ labeled examples $((x_1, y_1), \ldots, (x_N, y_N))$

- Distribution $D$ over the $N$ examples
- Weak learning algorithm $\text{WeakLearn}$
- Integer $T$ specifying number of iterations

**Initialize** the weight vector: $w_i^{t} = D(i)$ for $I=1, \ldots, N$.

**Do for** $t=1,2, \ldots, T$

1. set $p^t = \frac{w_i^t}{\sum_{i=1}^{N} w_i^t}$

2. Call $\text{WeakLearn}$, Providing it with the distribution $p^t$; get back a hypothesis $h_i : X \rightarrow [0,1]$

3. Calculate the error of $h_i$:
   $$\epsilon_i = \frac{1}{N} \sum_{i=1}^{N} p_i^t | h_i(x_i) - y_i |$$

4. Set $\beta_i = \epsilon_i / (1 - \epsilon_i)$

5. Set the new weights vector to be
   $$w_i^{t+1} = w_i^t \beta_i^{1-|h_i(x_i)-y_i|}$$

Output the hypothesis

$$h_f(x) = \begin{cases} 
1 & \text{if } \sum_{i=1}^{T} \left( \frac{1}{\beta_i} \log \frac{1}{\beta_i} \right) h_i(x) \geq \frac{1}{2} \sum_{i=1}^{T} \log \frac{1}{\beta_i} \\
0 & \text{otherwise}
\end{cases}$$

**Figure 4** Pseudocode of AdaBoost algorithm
A-boosting

1. Start with weights $w_i = 1/n$, $I=1, \ldots, n$.

2. Repeat for $m=1, \ldots, M$;
   
   (a) Fit a hypothesis $f_m$ on $F$ minimizing the weighted misclassification error rate with weights $\{w_i\}$ on the training examples.

   (b) Calculate the misclassification error $\varepsilon_m$ of $f_m$ on the original training examples by
   
   $$
   \varepsilon_m = \frac{\sum_{i=1}^{n} I(y_i \neq f_m(x_i))}{n}
   $$

   (c) Let $\beta_m = \log((1 - \varepsilon_m) / \varepsilon_m) / 2$

   (d) Update $w_i = \exp\left(-y_i \sum_{i=1}^{m} \beta_i f_i(x_i) / m\right)$. $I=1, \ldots, n$ and renormalize so that $\sum_i w_i = 1$

3. Output the classifier $\text{sign} \left( \sum_{m=1}^{M} \beta_m f_m(x) \right)$

Figure 5 Pseudocode of A-boosting algorithm
Algorithm AdaBoost.M1

Input: sequence of N examples \(((x_1, y_1), \ldots, (x_N, y_N))\) with labels \(y_i \in \{1, \ldots, k\}\)

Distribution D over the examples
Weak learning algorithm \textbf{WeakLearn}
Integer T specifying number of iterations

Initialize the weight vector: \(w_i^1 = D(i)\) for \(i=1,\ldots,N\).

1. set \(P^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}\)

2. Call \textbf{WeakLearn}, providing it with the distribution \(P^t\), get back a hypothesis \(h_t: X \rightarrow Y\)

3. Calculate the error of \(h_t: \varepsilon_t = \sum_{i=1}^{N} P_i^t [h_t(x_i) \neq y_i]\) if \(\varepsilon_t > 1/2\), the set \(T=t-1\) and abort loop.

4. Set \(\beta_t = \varepsilon_t / (1-\varepsilon_t)\)

5. Set the new weights vector to be \(w_i^{t+1} = w_i^t \beta^{t-[h_t(x_i)=y_i]}\)

output the hypothesis

\[ h_j(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left( \log \frac{1}{\beta_t} \right) [h_t(x) = y] \]

Figure 6 Pseudocode of AdaBoost.M1 algorithm
4.4 Weak Learner

4.4.1 Prototype Algorithm

Input: sequence of N examples \( ((x_1, y_1), \ldots, (x_N, y_N)) \) with labels 
\( y_i \in \{1, \ldots, k\} \), number of classes K, and weight \( w_i (i=1\ldots N) \)

1. let \( w_i = w_i \cdot N \)

2. According to the labels \( y_i \), separate the N examples to K subsets, \( S_k \).

3. Calculate the weighted center \( C_k \) of each subsets \( S_k \),

\[
C_k = \frac{\sum_{i \in S_k} x_i \cdot w_i}{\sum_{i \in S_k} w_i}
\]

4. For each \( x_i \) (I=1,\ldots N), calculate the hypothesis \( h: X \rightarrow Y \)

4.1 Calculate the distance \( d_{i,k} \) between \( x_i \) and \( C_k \) (k=1,\ldots K)

4.2 If \( d_{i,m} = \min_{k=1\ldots K} d_{i,k} \), \( x_i \) is labeled by m.

4.4.2 Knn Algorithm

Input: sequence of N examples \( ((x_1, y_1), \ldots, (x_N, y_N)) \) with labels 
\( y_i \in \{1, \ldots, k\} \), number of classes K, weight \( w_i (i=1\ldots N) \), and number of nearest neighbor (NN) used to decide the labels.

let \( w_i = w_i \cdot N \)

For each \( x_i, i=1\ldots N \)

1. Calculate the distances \( d_{i,j} \) between \( x_i \) and \( x_j, j=1\ldots i-1, i-1\ldots N \)

2. Sort \( d_{i,j} \) in increasing order,
3. According the sorted distances vector, put the first m records into the neighbor set $NS$ so that $\sum_{j \in NS} w_j < NN$

4. Separate the neighbor set $NS$ into k subsets $S_k$ (k=1…K) according to the labels $y_i$, and calculate the weights $W_{S_k}$ of each subset $S_k$ (k=1…K)

$$W_{S_k} = \sum_{i \in S_k} w_i$$

5. Output the hypothesis $h : X \rightarrow Y$

Label $x_i$ as $m$ if $W_{S_m} = \max_{k=1…K} W_{S_k}$

### 4.5 AdaBoost-M and A-boosting-M

The training procedure as follows:

1. Label training examples in set B1AaO and B1AaE as ‘0’, Label training examples in set B2O and B2E as ‘1’,

2. Call a binary boosting algorithm(Adaboost or A-Boosting). Output the classifier $C_0$

3. Label training examples in set B1AaO as ‘0’, label training examples in set B1AaE as ‘1’,

4. Call a binary boosting algorithm(Adaboost or A-Boosting). Out put the classifier $C_1$

5. Label training examples in set B2O as ‘0’, label training examples in set B2E as ‘1’,

6. Call a binary boosting algorithm(Adaboost or A-Boosting). Out put the
The testing procedure as the follows:

Input: a set of unlabeled testing data T.

1. Using classifier $C_0$ to classify T into two groups. We put the test examples being labeled as ‘0’ in the set TB1Aa and the examples being labeled as ‘1’ in the set TB2

2. Using classifier $C_1$ to classify TB1Aa into two groups. We put the test examples being labeled as ‘0’ in the set TB1AaO, and the examples being labeled as ‘1’ in the set TB1AaE

3. Using classifier $C_2$ to classify TB2 into two groups. We put the test examples being labeled as ‘0’ in the set TB2O, and the examples being labeled as ‘1’ in the set TB2E
Chapter 5 Results

5.1 Results for the Data from the First Set of Experiments

In order to compare the Adaboost-M and A-boosting-M with Adaboost.m1 and the corresponding weak learners, we use the same training data set and the same testing data set for all the methods. There are 33 records with known label (1 corresponding to odd B1Aa, 2 corresponding to even B1Aa, 3 corresponding to odd B2, and 4 corresponding to even B2) available. We ran 5 group of experiments totally. In the first group of experiments, one record was randomly selected from the 33 as the testing data and the rest 32 records as the training data. Repeat the process 10 times. In the other 4 group of experiments, we did the same thing except selecting 2, 3, 4 and 5 records as testing data set rather than 1. The testing accuracies is summarized in table2.

<table>
<thead>
<tr>
<th>Group1 (10 records here)</th>
<th>Adaboost-M(KNN)</th>
<th>A-boosting-M(KNN)</th>
<th>Adaboost.m1(KNN)</th>
<th>KNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Group2 (10 records here)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Group3 (10 records here)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.667</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.667</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Accuracy of the tests
A paired t-test was used to determine whether there is a significant difference between the average values of the same measurements made under two different conditions. For our study here, we use the same training data set and testing data set to exam the accuracy of 4 methods. With the accuracy percentages obtained, we carried out the following comparisons for force x, y, and Z separately.

(1) Adaboost-M(KNN), A-boosting-M(KNN), Adaboost.M1(KNN) against the weak leaner KNN

(2) Adaboost-M(KNN), A-boosting-M(KNN) against Adaboost.M1(KNN)

(3) Adaboost-M(Proto), A-boosting-M(Proto), Adaboost.M1(Proto) against the weak leaner Proto

(4) Adaboost-M(Proto), A-boosting-M(Proto) against Adaboost.M1(Proto)

5.1.1 The Comparisons of Boosted Methods against the Corresponded Weak Learners

Table 3 gives the comparisons of Adaboost-M(KNN), A-boosting-M(KNN), Adaboost.M1(KNN) against the weak leaner KNN. It shows that Adaboost-M(KNN) and A-boosting-M(KNN) are significantly better than the weak learner KNN for force x. However Adaboost.M1(KNN) is not significant.

Table 4 gives the comparisons of Adaboost-M(Proto), A-boosting-M(Proto), Adaboost.M1(Proto) against the weak leaner Proto. It shows that Adaboost-M(Proto), A-boosting-M(Proto), and Adaboost.M1(Proto) are significantly better than the weak learner Prototype.
Table 3 The boosted methods against the weak leaner KNN for force X

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.0265</td>
<td>0.0265</td>
<td></td>
</tr>
<tr>
<td>Test result ($\alpha = 0.05$)</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4 The boosted methods against the weak leaner prototype for force X

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.00000026545</td>
<td>0.00000026545</td>
<td>0.9975</td>
</tr>
<tr>
<td>Test result ($\alpha = 0.05$)</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

From table 5, we can observe that only the comparison of A-boosting-M(KNN) against KNN is significant for force Y.

From table 6, only the comparison of Adaboost.M1(Proto) against Prototype is significant for force Y.

Actually, we don’t expect that our methods can get high accuracy when using force Y because force Y doesn’t contain much useful information. It’s basically noise.
Table 5 The boosted methods against the weak leaner KNN for force Y

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.386</td>
<td>0.00017085</td>
<td>0.0701</td>
</tr>
<tr>
<td>Test result((\alpha = 0.05))</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6 The boosted methods against the weak leaner prototype for force Y

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.2085</td>
<td>0.1815</td>
<td>0.002</td>
</tr>
<tr>
<td>Test result((\alpha = 0.05))</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7 gives the comparisons of boosted methods against the corresponded weak learner KNN. It shows that, for force Z, Adaboost-M(KNN), A-boosting-M(KNN) and Adaboost.M1(KNN) are significantly better than the weak learner KNN.

Table 8 gives the comparisons of boosted methods against the corresponded weak learner prototype. It shows that Adaboost-M(Proto), A-boosting-M(Proto) and Adaboost.M1(Proto) are significantly better than the weak leaner Proto for force Z.
Table 7 The boosted methods against the weak leaner KNN for force Z

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.000001666</td>
<td>0.0242</td>
<td>0.00000000005551</td>
</tr>
<tr>
<td>Test result</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha = 0.05$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8 The boosted methods against the weak leaner prototype for force Z

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.0000059574</td>
<td>0.00016441</td>
<td>0.0023</td>
</tr>
<tr>
<td>Test result</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha = 0.05$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.1.2 The Comparisons of Adaboost-M and A-boosting-M against Adaboost.M1

From Table 9 and Table 10, we can conclude that, for the x-forces, Adaboost-M(KNN) and A-boosting-M(KNN) are significantly better than Adaboost.M1(KNN). Adaboost-M(Proto) and A-boosting-M(Proto) are significantly better than Adaboost.M1(Proto).
Table 9 Adaboost-M(KNN) and A-boosting-M(KNN) against Adaboost.M1(KNN) for Force X

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.0265</td>
<td>0.0265</td>
</tr>
<tr>
<td>Test result ( $\alpha = 0.05$)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 10 Adaboost-M(Proto) and A-boosting-M(Proto) against Adaboost.M1(Proto) for force X

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.00000003482</td>
<td>0.00000003482</td>
</tr>
<tr>
<td>Test result ( $\alpha = 0.05$)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

From Table 11, Adaboost-M(KNN) and A-boosting-M(KNN) are not significantly better than Adaboost.M1(KNN) for force Y.

From Table 12, Adaboost-M(Proto) and A-boosting-M(Proto) are not significantly better than Adaboost.M1(Proto) for force Y.

Table 11 Adaboost-M(KNN) and A-boosting-M(KNN) against Adaboost.M1(KNN) for Force Y

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value(0.05)</td>
<td>0.1461</td>
<td>0.877</td>
</tr>
<tr>
<td>Test result</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 12 Adaboost-M(Proto) and A-boosting-M(Proto) against Adaboost.M1(Proto) for force Y

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.9926</td>
<td>0.9926</td>
</tr>
<tr>
<td>Test result ($\alpha = 0.05$)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

From Table 13, Adaboost-M(KNN) and A-boosting-M(KNN) are not significantly better than Adaboost.M1(KNN) for force Z.

From Table 14, Adaboost-M(Proto) and A-boosting-M(Proto) are significantly better than Adaboost.M1(Proto) for force Z.

Table 13 Adaboost-M(KNN) and A-boosting-M(KNN) against Adaboost.M1(KNN) for Force Z

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>0.997</td>
<td>0.1647</td>
</tr>
<tr>
<td>Test result ($\alpha = 0.05$)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 14 Adaboost-M(Proto) and A-boosting-M(Proto) against Adaboost.M1(Proto) for force Z

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value(0.05)</td>
<td>0.0016</td>
<td>0.0956</td>
</tr>
<tr>
<td>Test result</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
5.2 Results for the Data from the Second Set of Experiments

Three data sets for the second set of experiments were put together to test the performance of the developed methodology for grinding wheel condition monitoring. The first data set comprises of 160 AE signal segments extracted from those generated in grinding some alumina specimens using different grinding conditions when the wheel was sharp as well as when the wheel was dull. All 160 signal segments in the second data set come from grinding a different material, i.e., silicon nitride. The third data set is the combination of both first and second data sets. The 10-fold cross validation method is consistently used in testing each data set.

The $k$-nearest neighbor classifier requires the specification of the $k$ value. To determine the effect of number of nearest-neighbors $k$, a range of $k$ values were tried for each data set. Figures 7-9 plot the classification accuracy as a function of $k$ for each boosted classifier when applied to the first, second, and third data set, respectively. The following observations can be made based on these results:

1) The KNN weaker learner is far superior to the prototype leaner.

2) Both boosted algorithms produce the same or better performance than the prototype weak learner for all three data sets tested.

3) Both boosted algorithms yield slightly worse result than the KNN for some $k$ values. This means that a boosting algorithm does not guarantee improved performance.

4) A-Boosting does not always outperform AdaBoost and vice versa.
5) The best classifiers for the first data set are KNN and both boosted-KNN algorithms that yield 95.63% average classification accuracy when the optimal $k$ value of one is used.

6) The best classifiers for the second data set are again KNN and both boosted-KNN algorithms that produces 100% average classification accuracy when the optimal $k$ value of one or three is used.

7) For the third data set, three classifiers including A-Boosting-KNN, AdaBoost-KNN, and KNN, produce the best accuracy of 97.5% with all having the optimal $k$ value of one. Note that this accuracy rate falls between those of the first and second data sets, which seems to make sense.

![1st Data Set (Al2O3)](image)

**Figure 7** Mean classification accuracies of classifiers applied to data set #1.
2nd Data Set (Si3N4)

Figure 8 Mean classification accuracies of classifiers applied to data set #2.

3rd Data Set (Al2O3+Si3N4)

Figure 9 Mean classification accuracies of classifiers applied to data set #3.
Figure 10  Average false positive rate of the 1\textsuperscript{st} data set.

Figure 11 Average false positive rate of the 2\textsuperscript{nd} data set.
3rd Data Set (Al2O3+Si3N4)

Figure 12 Average false positive rate of the 3rd data set

1st Data Set (Al2O3)

Figure 13 Average false negative rate of the 1st data set
### 2nd Data Set (Si3N4)

<table>
<thead>
<tr>
<th>K (number of neighbors)</th>
<th>False Negative Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0000</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
</tr>
<tr>
<td>10</td>
<td>0.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

- **A-Boosting_KNN & AdaBoost-KNN**
- **KNN**
- **A-Boosting-Proto**
- **AdaBoost-Proto**
- **Prototype**

**Figure 14** Average false negative rate of the 2\textsuperscript{nd} data set

### 3rd Data Set (Al2O3+Si3N4)

<table>
<thead>
<tr>
<th>K (number of neighbors)</th>
<th>False Negative Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0000</td>
</tr>
<tr>
<td>5</td>
<td>0.0000</td>
</tr>
<tr>
<td>10</td>
<td>0.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

- **A-Boosting_KNN & AdaBoost-KNN**
- **KNN**
- **A-Boosting-Proto**
- **AdaBoost-Proto**
- **Prototype**

**Figure 15** Average false negative rate of the 3\textsuperscript{rd} data set
Chapter 6 Conclusion

The most important conclusion we can reach from this study is that the ‘Regression model with auto correlated errors’ is a good model to extract features from both the force signals and AE signals collected during the grinding process.

Furthermore, from the results based on the force data set, we can see that Adaboost-M, A-boosting-M and Adaboost.M1 are significantly better than the correspond weak leaner (KNN or Proto). Adaboost-M and A-boosting-M are better than Adaboost.M1. From the analysis based on the AE data set, we can conclude that the choosing of weak learners is very important with the boost methods. For our application, the KNN weaker learner is far superior to the prototype learner.
References


18. Dae Kyun Baek, Tae Jo Ko, Hee Sool Kim, Real Time Monitoring of Tool Breakage in a Milling Operation Using a Digital Signal Processor, *Journal of Materials Processing Technology* 100(200) 266-272


23. SAS online document


Vita

Ms. Fengming Tang is from Sichuan province in China. She enrolled in the doctorate program in Engineering Science in Louisiana State University in the spring of 2003. She will earn the degree of Master in Industrial Engineering in December, 2006. She is expected to finish her doctorate study in May of 2008.