Big Data Classification - Many Features, Many Observations

Claus Weihs, Daniel Horn, and Bernd Bischl

Department of Statistics, TU Dortmund University
{claus.weihs, daniel.horn, bernd.bischl}@tu-dortmund.de

Abstract. In this paper we discuss the performance of classical classification methods on Big Data. We distinguish the cases many features and many observations. For the many features case we look at projection methods, distance based methods, and feature selection. For the many observations case we mainly consider subsampling. The examples in this paper show that standard classification methods should not be blindly applied to Big Data.

1 Introduction

This paper is on Big Data Analytics (BDA). But what is Big Data? Unfortunately, the answer depends on whom you have asked when. In Machine Learning (ML) Benchmarks in the 1990s (e.g. in the UCI repository) maximum 100s to 1000s of data points were available. In modern benchmarks we often have more than $10^6$ data points. When you ask, e.g., Google, the answer might be 'Big Data means that data are much too big for your computer storage, only streaming is possible from a cloud, only distributed analytics, ...' Another possibility is to define a 'Big Data problem' by the impossibility to exactly solve the learning problem by computer time reasons.\footnote{Thanks to T. Glasmachers for providing this definition.} Therefore, information in the data is not optimally utilizable. This definition is used in the very last example of this paper, where the question is: Which information brings us as fast as possible as near as possible to the solution and what is a 'perfect' approximation algorithm?\footnote{This part of the paper was supported by the Mercator Research Center Ruhr, grant Pr-2013-0015, see http://www.largescalesvm.de/} Note that those who think that the data in our paper is not big enough for being Big Data might also call our topic of interest 'Large Scale Data Analysis'.

Another modern buzzword is 'Data Science'. This is sometimes defined as the study of generalizable extraction of knowledge from data. In the context of this paper, data science mainly builds on techniques and theories from machine
learning, statistics, and high performance computing. Data science is by no means restricted to Big Data, but methods scaling up with data size get more and more important. 'Classical' data science methods are classification and regression methods together with dimension reduction, feature selection, and hyperparameter tuning.

In this paper we will discuss typical classification methods in the context of Big Data Analytics. The message of this paper is that for BDA not all classical methods are adequate and that Big Data might even long for special methods. In order to demonstrate the extremes, we will particularly discuss the cases of many features (and small no. of observations) in Section 2, and the case of many observations (and small no. of features) in Section 3.

2 Many Features

With the advent of high throughput biotechnology data acquisition platforms such as micro arrays, SNP chips and mass spectrometers, data sets with many more variables than observations are now routinely being collected (see, e.g., Kiiveri (2008)). Most often, however, only a small part of these $p$ features or a small number of directions in $p$-space are important for classification. Therefore, one might be tempted to thoughtlessly apply standard methods which are known to be only adequate for $p<n$ (not too big), but problematic in high dimensions (curse of dimensionality) and for very large $n$. In this paper, we will discuss some of the many available classification methods in this context. Let us start with projection based methods.

2.1 Projection Based Methods

One of the best known and most used projection based classification methods in statistics is Fisher discrimination. The performance of this method in the case of more features than observations is discussed by Bickel and Levina (2004) showing the following property:

Consider 2 classes with Gauss distributions $\mathcal{N}(\mu_1, \Sigma), \mathcal{N}(\mu_2, \Sigma)$. Let the corresponding a priori probabilities be equal, i.e., $\pi_1 = \pi_2 = 0.5$. Then, for Fisher discrimination the classification function has the form $\delta_F(x) = (x - \mu)^T \Sigma^{-1}(\mu_1 - \mu_2)$ with $\mu = (\mu_1 + \mu_2)/2$. Let the corresponding samples be observed with equal sample sizes, i.e. $n_1 = n_2$. Then, the sample version of the classification rule is: Assign class 1 iff $\hat{\delta}_F(x) = (x - \bar{x})^T S^{-1}(\bar{x}_1 - \bar{x}_2) > \log(\pi_2/\pi_1) = 0$. If $p > n$, then the inverse of estimated pooled covariance matrix $S$ does not exist and the Moore-Penrose generalized inverse is used instead. For this situation, the following result is true under some regularity conditions: If $p \to \infty$, $n \to \infty$, and $p/n \to \infty$, then error $\to 0.5$, i.e. the class assignment is no better than random guessing.

This result states a strong warning concerning the application of Fisher discrimination in the case of many more features than observations. As have
been motivated by Bickel and Levina (2004), the bad performance of Fisher discriminant analysis is due to the fact that the condition number of the estimated covariance matrix goes to infinity as dimensionality diverges even though the true covariance matrix is not ill-conditioned.

Noise accumulation can also be reduced by ignoring the covariance structure, i.e. by using a diagonal matrix as an estimate of the covariance matrix. In this context, Bickel and Levina (2004) derived the following asymptotic result for the so-called independence rule (ir), i.e. linear discriminant analysis with diagonal covariance matrix:

Let \( \Gamma \) be a ‘regular’ space of possible means and covariance matrices of the two classes, \( \Sigma \) the full covariance matrix in the two classes, \( \Sigma_0 \) the corresponding correlation matrix, \( \lambda(\Sigma_0) \) an eigenvalue of \( \Sigma_0 \), and \( \Phi \) the distribution function of the standard normal. Then, the following result is true:

If \( \log(p)/n \to 0 \), then \( \lim \sup n \to \infty \) \( (\text{maximal error in } \Gamma) = 1−\Phi(\sqrt{K_0}c) \), where \( K_0 = \max \{\lambda_{\max}(\Sigma_0)/\lambda_{\min}(\Sigma_0)\} \) and \( c = \min \{((\mu_2 − \mu_1)^T \Sigma^{-1}(\mu_2 − \mu_1))\} \).

Therefore, if \( p \) is going slower to infinity than \( e^n \), then for big data sets there is a bound for the maximal error in the space of possible data situations. In practice, this property may lead to a superiority of ir over the full lda. Finally note that for normal distributions the independence rule is equivalent to the Naive Bayes method. In practice, however, the Naive Bayes method (NB) is typically implemented in a non-parametric way and not by assuming a certain type of distribution like the normal distribution. This generally leads to implementations different from the independence rule. For normal distributions as in our examples, NB is thus expected to be inferior to ir. Additionally, the linear support vector machine (svm), also looking for linear separations, will be discussed as an alternative to lda which can be adapted to the actual data by tuning the cost parameter.

Generic Data Generation (GDG). Let us demonstrate the above theoretical results by means of data examples. Let us start with a GDG step. We will always consider the ideal situation for the linear discriminant analysis (lda), i.e. two classes where the influential features are multivariate normally distributed with different mean vectors and the same covariance matrix. In the case where \( pr \) features influence class separation we choose the class means \( m_1(i) = −md/2 \), \( m_2(i) = md/2 \), where \( md \) = difference between the two class means, \( i = 1, \ldots, pr \). The covariance matrices are built so that \( \Sigma = \Sigma_R + d \cdot I \), where \( \Sigma_R \) is built of independent uniform random numbers between 0.1 and 1 and the multiple \( d \) of the identity is added in order to generate positive definiteness. By choosing different distances \( md \) between the mean vectors or different \( d \) the Bayes error, interpreted as the difficulty of the classification problem, can be varied. Sometimes we add noise by means of features which do not have any influence on class separation by adding \( (p−pr) \) normally distributed features with mean 0 and variance 25. Overall, we assume that we have \( p \) features. Note that possibly \( p = pr \). We typically use \( n = 2 \cdot nel << p \) observations, \( nel \) observations for each class. Thus, \( p \) tends to be much bigger
Table 1. Comparison of mean error rates (%): a) all, b) only $p/6$ features influence

<table>
<thead>
<tr>
<th>$p$</th>
<th>12</th>
<th>20</th>
<th>24</th>
<th>30</th>
<th>36</th>
<th>40</th>
<th>60</th>
<th>1080</th>
<th>2040</th>
<th>sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) all</td>
<td>md = 1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lda</td>
<td>45</td>
<td>43</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>43</td>
<td>43</td>
<td>40</td>
<td>35</td>
<td>32</td>
</tr>
<tr>
<td>ir</td>
<td>42</td>
<td>28</td>
<td>22</td>
<td>17</td>
<td>15</td>
<td>13</td>
<td>7</td>
<td>4</td>
<td>32</td>
<td>9</td>
</tr>
<tr>
<td>NB</td>
<td>45</td>
<td>38</td>
<td>34</td>
<td>31</td>
<td>29</td>
<td>27</td>
<td>23</td>
<td>17</td>
<td>37</td>
<td>22</td>
</tr>
<tr>
<td>1NN</td>
<td>45</td>
<td>34</td>
<td>29</td>
<td>25</td>
<td>22</td>
<td>19</td>
<td>13</td>
<td>10</td>
<td>37</td>
<td>15</td>
</tr>
<tr>
<td>svm</td>
<td>44</td>
<td>29</td>
<td>21</td>
<td>15</td>
<td>14</td>
<td>11</td>
<td>7</td>
<td>3</td>
<td>34</td>
<td>8</td>
</tr>
<tr>
<td>b) $p/6$</td>
<td>md = 2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lda</td>
<td>46</td>
<td>42</td>
<td>41</td>
<td>41</td>
<td>42</td>
<td>42</td>
<td>41</td>
<td>36</td>
<td>26</td>
<td>24</td>
</tr>
<tr>
<td>lda+fs/6</td>
<td>36</td>
<td>30</td>
<td>27</td>
<td>27</td>
<td>26</td>
<td>27</td>
<td>26</td>
<td>27</td>
<td>20</td>
<td>11</td>
</tr>
<tr>
<td>ir</td>
<td>41</td>
<td>26</td>
<td>18</td>
<td>13</td>
<td>10</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>NB</td>
<td>45</td>
<td>35</td>
<td>32</td>
<td>28</td>
<td>26</td>
<td>25</td>
<td>18</td>
<td>13</td>
<td>32</td>
<td>11</td>
</tr>
<tr>
<td>1NN</td>
<td>47</td>
<td>41</td>
<td>38</td>
<td>35</td>
<td>33</td>
<td>31</td>
<td>25</td>
<td>18</td>
<td>38</td>
<td>19</td>
</tr>
<tr>
<td>svm</td>
<td>43</td>
<td>26</td>
<td>18</td>
<td>14</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>30</td>
<td>4</td>
</tr>
</tbody>
</table>

Note: $p =$ no. of dimensions, sec = mean training time over both md in seconds for $p = 2040$, md = mean difference of classes in each dimension.
lda = linear discriminant analysis (lda, package MASS, software R (R Core Team (2014)),
fs/6 = feature selection (best $p/6$ features, mutual information (symmetrical.uncertainty) criterion), package FSelector in R).
ir = independence rule = lda with diagonal covariance matrix
(sda, package sda in R, no shrinkage, diagonal = TRUE)
NB = naive Bayes method (naiveBayes, package e1071 in R),
1NN = 1 nearest neighbor rule (knn, package class in R),
svm = linear support vector machine
(svm in R, package e1071, cost parameter tuned on grid $2^{-4}, \ldots, 2^4$ by leave-one-out).

than $n$, the case we discuss in this section. The generation of $n$ data points from the above normal distributions in $p$ dimensions is repeated $rp = 200$ times using different random covariance matrices $\Sigma$. For the estimation of error rates, corresponding test samples with $nelt = 1000$ observations per class are generated from training distributions.

Example 1. Let us first assume that all involved features in fact influence the class choice, i.e. $p = pr = 12, \ldots, 2040$, and let $d = 25, nel = 6, md \in \{1.5, 2.5\}$. By means of this variation of $p$ with constant $n = 2 \cdot 6 = 12$ we vary the ratio $p/n$ from 1 to 170. For $md = 1.5$ the classification problem is obviously much harder than for $md = 2.5$. On the accordingly generated data (see GDG) different classification methods are compared. Let us start the discussion of the mean error rates in Table 1a) with (Fisher’s) linear discriminant analysis ($lda$), the independence rule ($ir$), the naive Bayes method ($NB$), and the linear support vector machine ($svm$).3 Obviously, $lda$ does not benefit from higher dimensions. However, it is also not suffering from increasing $p$ as possibly suspected from the above theory, but mainly stays as bad as with small $p$. Obviously, even $p = 2040$, although being $170 \cdot n$, is not high enough to see the effect. Also note that $p = 2040$ is much lower than

3 This simulation was carried out using the R-packages BatchJobs (Bischl et al. (2014)) and mdr on the SLURM cluster of the Statistics Department of TU Dortmund University.
\( e^n = e^{12} = 162754.8 \) (cp. the asymptotic result for the independence rule).

So, we see that the behavior of lda is not as bad in practice as suspected from theory. In contrast to lda, svm and ir benefit very much, NB somewhat less. Since svm needs by far the most training time, ir is definitely to be preferred (cp. column sec).

Let us now compare this behavior with the case where only \( pr = p/6 \) features influence the classes. Moreover, the situation further differs since \( d = 10, md \in \{2.5, 5\} \). Looking at the results in Table 1b), again, lda does not benefit from more features. Only in the easier case \( md = 5 \), there is a benefit from \( p = 12 \) to \( p = 120 \). Again, method NB only slowly benefits from more features, at least in the harder case \( md = 2.5 \). In contrast, ir and svm benefit much faster. Moreover, in the easy case \( (md = 5) \) the benefit is much greater than in the hard case \( (md = 2.5) \).

### 2.2 Feature Selection

Note that in Example 1 ignoring the full covariance structure leads to an improvement in high dimensions. Another idea could be to ignore the fact that all \( p \) dimensions influence the class membership, and just take the most important features for classification, e.g. in lda.

Simple filters are the fastest feature selection methods. In filter methods, numerical scores \( s_i \) are constructed for the characterization of the influence of feature \( i \) on the dependent class variable. Filters are generally independent of classification models. Easy example filters are the \( \chi^2 \)-statistic for the evaluation of independence between (discretized) feature \( i \) and the class variable, the p-value of a t-test indicating whether the mean of feature \( i \) is different for the two classes, the correlation between feature \( i \) and the class variable, and the mutual information in feature \( i \) and the class variable.

Filters can be easily combined with a classification method. First calculate filter values (scores). Then sort features according to scores and choose the best \( k \) features. Finally, train the classification method on these \( k \) features.

**Example 1 (cont.)** When only \( p/6 \) features influence the classes, the correct number of features is selected by feature selection \((\text{lda+fs}/6)\). The corresponding error rates are then much lower than without feature selection (see Table 1b)). In the easier case \( (md = 5) \) the result was even nearly optimal but at the price of higher computation times (see column "sec") caused by the usage of a mutual information criterion for feature selection.

The most important problem with feature selection is the adequate choice of \( k \). Another idea is to apply dimension reduction, e.g. by principal components analysis \((\text{pca})\), before application of classification methods (see, e.g., Bair et al. (2006)) since there is hope that projection dimensions put much more weight on features having large classification power. Unfortunately, the
Table 2. Mean error rates (%) of \textit{lda} on the first 2 principal components based on the best \( m \) features

\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
\( md \) & \( m \) & 2 & 10 & 50 & 90 & 110 & 200 & 500 & 800 & 900 & 1000 \\
\hline
1 & 42 & 34 & 48 & 47 & 46 & 46 & 45 & 44 & 45 & 45 & 45 \\
1.2 & 41 & 29 & 38 & 30 & 27 & 23 & 24 & 28 & 31 & & \\
1.4 & 39 & 26 & 11 & 8 & 8 & 7 & 8 & 10 & 15 & 20 & \\
\hline
\end{tabular}

above result for \textit{lda} can be generalized to the application of \textit{lda} to any general projection on linear combinations. This is because such projection directions are constructed with probability 1 using essentially all features so that the misclassification error tends to be big because of noise accumulation when not all features are relevant for class separation (Fan et al. (2011)). This affects, e.g., \textit{lda} applied to principal components, but also combinations with other projection methods like partial least squares (as proposed, e.g., by Boulesteix (2004)).

Let us discuss whether a, at least nearly, correct finding of the real number of influential dimensions is helpful for \textit{lda} and look at an example combining the two ideas, feature selection and \textit{pca}.

\textbf{Example 2.} 4 Consider two classes in \( p = 1000 \) dimensions, where only \( pr = 100 \) dimensions really influence class membership. The idea is, first, to identify those \( m \) features with the highest effect on class separation, \( m \in \{2, \ldots, p\} \), by means of feature selection on training data. Here, we use the linear correlation criterion, which is much faster than mutual information but only approximate for binary outputs (again from FSelector in R). In the above GDG we use \( nel = 50, md \in \{1, 1.2, 1.4\}, d = 10, rp = 200, netl = 1000 \). Second, class separation is tried by means of \textit{lda} on the first two principal components (\textit{pcs}). Note that \textit{pcs} are only determined up to sign. The sign might even differ for training and test sets resulting in an interchange of class labels in the test data. Therefore, \( \min(mcr, 1 - mcr) \), \( mcr \) = estimated misclassification error, is used as the error rate. Table 2 shows mean \textit{lda} error rates on the first two principal components of the same \( m \) dimensions of the test data identified on the training data. Obviously, choosing \( m \) near the correct \( pr = 100 \) is only optimal for the easiest problem (\( md = 1.4 \)). For the somewhat harder problem with \( md = 1.2 \), higher \( m \) gave more acceptable results. This may be caused by a nearly inevitable imperfect feature selection. Except for the hardest problem \( md = 1 \), where astonishingly enough \( m = 10 \) gave the distinctly best result, higher \( m \) appear to be on the save side.

\section{2.3 General Distance Based Methods}

Let us now switch to general distance based classifiers and discuss whether there are such methods, other than \textit{lda}, better suited in high dimensions.

\footnote{This example is inspired by Fan et al. (2011).}
Table 3. Mean error rates (%) of 1NN (left) and mean class distances (right)

<table>
<thead>
<tr>
<th>ip</th>
<th>p</th>
<th>12</th>
<th>120</th>
<th>240</th>
<th>360</th>
<th>480</th>
<th>600</th>
<th>1080</th>
<th>2040</th>
<th>12</th>
<th>120</th>
<th>240</th>
<th>360</th>
<th>480</th>
<th>600</th>
<th>1080</th>
<th>2040</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>15</td>
<td>30</td>
<td>60</td>
<td>75</td>
<td>135</td>
<td>255</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>2.7</td>
<td>3.2</td>
<td>3.5</td>
<td>3.8</td>
<td>4.0</td>
<td>4.6</td>
<td>5.4</td>
</tr>
<tr>
<td>4</td>
<td>45</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>2.7</td>
<td>3.2</td>
<td>3.5</td>
<td>3.8</td>
<td>4.0</td>
<td>4.6</td>
<td>5.4</td>
</tr>
<tr>
<td>8</td>
<td>45</td>
<td>25</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>2.0</td>
<td>2.2</td>
<td>2.3</td>
<td>2.4</td>
<td>2.5</td>
<td>2.6</td>
<td>2.9</td>
</tr>
<tr>
<td>16</td>
<td>45</td>
<td>27</td>
<td>14</td>
<td>10</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
<td>1.9</td>
<td>1.9</td>
<td>2.0</td>
<td>2.1</td>
</tr>
<tr>
<td>32000</td>
<td>45</td>
<td>33</td>
<td>28</td>
<td>23</td>
<td>20</td>
<td>17</td>
<td>11</td>
<td>4</td>
<td>1</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
</tbody>
</table>

For a plausible distance-based classifier \(g\) we only assume the following two properties: (a) \(g\) assigns \(X\) to class 1 if it is closer to each of the \(X_i\)s in class 1 than it is to any of the \(X_j\)s in class 2. (b) If \(g\) assigns \(X\) to class 1, then \(X\) is closer to at least one of the \(X_i\)s in class 1 than to the most distant \(X_j\) in class 2. For such a method the following property is true (Fan et al. (2011)):

Consider the model \(X_{ij} = \mu_{kj} + \epsilon_{ij}, i \in G_k, k = 1, 2\), where \(X_{ij}\) is the \(j\)th component of \(X_i\), \(\mu_{kj}\) the \(j\)-th component of mean vector \(\mu_k\), and the \(\epsilon_{ij}\) are independent identically distributed with mean 0 and finite 4th moment. Then, the probability that a distance based classifier of the above kind classifies a new observation correctly converges to 1 iff \(p = o(\|\mu_2 - \mu_1\|^4)\) for \(p \to \infty\).

This property shows that with distance based classifiers perfect class prediction is possible, but only if the distance of class means grows with the number of influential features so that \(p^{1/4}/\|\mu_2 - \mu_1\| \to 0\), i.e. that \(md = \|\mu_2 - \mu_1\|\) grows faster than \(p^{1/4}\). Note that this result is independent of sample size \(n\).

Let us look at an example to illustrate this property.

**Example 3.** Consider the \(k\)NN method with \(k = 1\) based on the Euclidean distance. Let the mean distance \(md\) between the two classes increase with dimension \(p\) so that \(md = p^{1/4p} \cdot 1.5/12^{1/4p}\) guaranteeing a start distance of 1.5, \(ip = 1, 2, 4, 8, 16, 3200\). In GDG we additionally choose \(nel = 6, nelt = 1000, rp = 200\) and \(\Sigma_R = 0 \cdot I, d = 25\), meaning that we sample from independent normal distributions with mean distance \(md\) and standard deviation 5 in each dimension (cp. the above theoretical property). Table 3( left) shows that the start distance of 1.5 leads to a high mean test error rate of 45\%. However, the error rate benefits from more features. This can be seen by comparing mean test error rates for different \(p\) with (nearly) the same mean class difference. E.g., compare the error rates for \(ip = 4, p = 120\) and \(ip = 8, p = 1080\) or for the different \(p\) in the case \(ip = 32000\) (see Table 3(left and right)). Indeed, in the model 'error = \(\alpha_0 + \alpha_1 md + \alpha_2 p + \alpha_3 md \cdot p\)', \(md \cdot p\) representing the interaction between \(md\) and \(p\), all coefficients appear to be significant. Particularly note that the increase of mean distances is only moderate for \(ip = 4\), the inverse power in the above theoretical property. Thus, this example shows that an increase of \(md = p^{1/4}\) leads to better and better class separation and does not have to mean a big restriction.
Example 1 (cont.) Let us finish this section by discussing the performance of the one-nearest-neighbor rule 1NN on the data sets of Example 1. There, it performs similarly as naive Bayes (NB) (see Tables 1a,b)). Note, however, that runtime is near zero for 1NN because the training data set only consists of \( n = 12 \) observations.

3 Many Observations

Let us also briefly look at cases with many more observations than features. In such cases, the standard idea is to split the data into smaller blocks, analyze these blocks, and recombine the results to an overall result. Let us concentrate here on the case where we actively split a too big data set and analyze the corresponding blocks. This idea is obviously adapted from cross validation and bagging. Then, we try to find a recombination method that gives a reasonable, as optimal as possible, approximation to that result which we would have seen if we would have looked at all observations at the same time. This leads us back to the definition of Big Data based on approximations in the introduction. Please note that we skip the streaming case here because of space restrictions. In such a case, the data is arriving already in blocks with the possibility of structural breaks in new blocks.

One example for splitting the data actively into subsamples and try to estimate the overall error rate from estimated error rates in the subsamples is the so-called cascade-svm (Graf et al. (2005)), a version of which was realized in Meyer et al. (2013) in the following way:

1. Partition the data into \( k \) subsets of possibly the same size.
2. Parallelly train \( \text{svm} \) independently on each subset.
3. Generate new data sets by combination of the support vectors (\( \text{svs} \)) of pairs (or triplets,...) of such analyses.
4. Repeat steps 2 and 3 for some time.
5. Train an \( \text{svm} \) on all \( \text{svs} \) in step 4 leading to an \( \text{svm} \) model.

Here, the main idea for the estimation of the overall error rate is that for \( \text{svm} \) already the \( \text{svs} \) contain all information necessary for model building. For other classification methods, analogues might be constructed by identifying the important observations by means of the distance to the decision border.

Meyer et al. (2013) tested the method on examples with 67 000 to 581 000 observations comparing the full linear \( \text{svm} \), pure bagging with majority voting, and different versions of the cascade-\( \text{svm} \). The latter sometimes resulted in much better results than pure bagging and was much faster with only a little worse results than full \( \text{svm} \).

In another example we took a closer look on the approximation of the result of the full kernel \( \text{svm} \) by means of subsampling in the following way:
1. Optimize the cost parameter, the kernel-width, and the duality gap of the kernel SVM method as well as the subsampling rate \( k \) with respect to two targets, namely the misclassification error and required training time, by means of sequential model-based multi-criteria optimization.

2. Randomly split the subsample once into 50% training, 25% test, and 25% validation samples. For this, the order of the observations was permuted.

3. During the optimization, generate a training subsample by using the first \( k \) percent of the training data. This way, the training sample for \( k = 0.1 \) is a subset of the sample for \( k = 0.2 \).

4. Train the kernel SVM on the training sample, calculate the training time, and estimate the misclassification rate on the validation sample.

5. Analyze the trade-off between the 2 targets by means of the Pareto-front.

With this method, we received very promising results for several large data sets\(^6\) in that the subsample results built very promising alternatives to the full data result with only slightly higher errors produced in much less training time. In Figure 1 on the left we see an expected Pareto-front. Using 93% of the data, a small speed up is possible without a significant error loss, whereas using only 21% we observe a speed up by factor 10, but twice the validation error. On the right we see a very promising result where we were able to reduce the training time by factor 100 nearly without any loss in accuracy.

\(^5\) We used the R-library libSVM, see http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

4 Summary and Conclusion

In this paper we discussed the performance of standard classification methods on Big Data. We distinguished the cases many features and many observations. For the many features case we looked at projection methods, distance based methods, and feature selection. lda appeared to be problematic for Big Data, whereas svm, the independence rule ir, naive Bayes NB, and 1NN were able to benefit from high dimensions. Note, however, that the behavior of lda was not as bad in practice as suspected from theory. Also, feature selection might help for finding better models in high dimensions. ir and svm performed best in high dimensions, ir in much less time than svm. For the many observations case, subsampling generated promising alternatives to the full data result by producing only slightly higher errors in much less training time. Overall, the paper shows that classification on Big Data is a very important research field which is by no means fully understood until now.

References

Bickel, P. J. and Levina, E. (2004). Some theory for Fisher’s linear discriminant function, “naive Bayes”, and some alternatives when there are many more variables than observations. Bernoulli 10, 989-1010