Fault Tolerant Regression for Sensor Data

Indrė Žliobaitė and Jaakko Hollmén

Aalto University, Dept. of Information and Computer Science,
Helsinki Institute for Information Technology (HIIT), Finland
{indre.zliobaite,jaakko.hollmen}@aalto.fi

Abstract. Many systems rely on predictive models using sensor data, with sensors being prone to occasional failures. From the operational point of view predictions need to be tolerant to sensor failures such that the loss in accuracy due to temporary missing sensor readings would be minimal. In this paper, we theoretically and empirically analyze robustness of linear predictive models to temporary missing data. We demonstrate that if the input sensors are correlated the mean imputation of missing values may lead to a very rapid deterioration of the prediction accuracy. Based on the theoretical results we introduce a quantitative measure that allows to assess how robust is a given linear regression model to sensor failures. We propose a practical strategy for building and operating robust linear models in situations when temporal sensor failures are expected. Experiments on six sensory datasets and a case study in environmental monitoring with streaming data validate the theoretical results and confirm the effectiveness of the proposed strategy.

Keywords: missing data, data streams, linear models, sensor failure.

1 Introduction

The amount of sensors installed in the urban and natural environments is rapidly increasing. It is predicted that sensor data collected from satellites, mobile devices, outdoor and indoor cameras will become the largest information trove for our society in the coming years [3]. Predictive models using sensor readings as inputs are widely applied in real-time systems, such as production quality control, air pollution monitoring, detecting traffic jams or severe road conditions, route recognition, road navigation, cargo tracking and many more [6].

Physical sensors are exposed to various risks due to, for instance, severe environmental conditions or exposure to physical damage. Moreover, typically sensors rely on batteries, are installed in remote or hardly accessible locations, or are unaccessible during operation runtimes. Sensors may break causing a sudden failure until replaced. Sensors may get covered in snow or water causing a seasonal temporary disruption. Some sensors may lose sensitivity due to wear and tear. Under such circumstances it is very common to have time intervals when readings from some sensors are missing. At the same time a predictive model needs to operate continuously and deliver predictions in real time.

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part I, LNAI 8188, pp. 449–464, 2013.
© Springer-Verlag Berlin Heidelberg 2013

We study how to make predictive models robust to temporary sensor failures during real-time operation. We focus on linear regression models. We assume that the observed process is stationary and the predictive model remains fixed during real-time operation. Our goal is to maximize prediction accuracy not only when all the input sensors are available, but also when readings from several sensors are missing for continuous periods of time. The problem is more challenging than it may seem due to the temporal nature of the missing data and limited computational resources under the stream setting.

Discarding the incoming instances that contain missing values is not an option, since there will be no predictions for continuous periods of time. The iterative multiple imputation (MI) [14] carries high computational costs and is not considered for real-time operation. Deploying additional predictive models for filling in missing values given the available sensors is computationally impractical, since an exponential number of models would be required to cover all combinations of failing sensors. One could consider an adaptive regression, e.g. [17]. However, the time for learning a stable model before recovery or the next failure is very limited, while the previous model is in principal correct. This applies to persistent temporal failures as well as once-off outlier failures. Hence, adaptation is not considered when to cover for frequent temporary failures.

A simple replacement of the missing values by the sensor mean value is a popular and easy to implement strategy in industrial applications. Unfortunately, in the stream setting where values of the same sensors are missing for a continuous period of time, this strategy may lead to a drastic deterioration of the prediction accuracy, particularly if the input sensors are highly correlated and a regression model exploits that correlation. Therefore, if sensor failures are expected in real-time operation it is not enough to replace missing values by the mean; we also need to ensure that the predictive model is robust to temporary missing data.

This paper presents a theoretical analysis of the predictive performance under sensor failures and formulates robustness criteria for real-time operation. We introduce the *deterioration index* that allows to assess robustness of a given linear model to partial loss of input data. We propose a practical strategy for building robust linear models that is based on a de-correlating transformation and a subsequent regularization of the model parameters in the transformed space. Experimental validation on six sensor datasets and a case study in environmental monitoring domain confirms the effectiveness of the proposed strategy.

Our study contributes a theoretically supported methodology for diagnosing robustness of linear regression models to loss of input data. This methodology makes it possible to *assess* the robustness of alternative models prior to deployment in real-time operation. The second contribution is a practical strategy for *optimizing* linear regression models such that they are robust to sensor failures.

The paper is organized as follows. Section 2 outlines the setting. In Section 3 we theoretically analyze how sensor failures affect the prediction accuracy and develop an index for diagnosing the performance. Section 4 gives practical recommendations. Experimental analysis is reported in Section 5 and the case study in Section 6. Section 7 discusses related work and Section 8 concludes the study.

2 Background and Problem Setting

We start by formalizing the problem and presenting a recap on linear models.

Setting. Suppose we have r sensors generating multidimensional streaming data vectors $\mathbf{x} \in \Re^r$ (e.g. weather observation sensors). Our task is to predict the target variable $y \in \Re^1$ (e.g. solar radiation) using these sensor readings as inputs. Data arrives in real time, the predictions need to be available in real time. The expected loss in accuracy due to sensor failures should be minimum. We assume that the observed process is stationary and the predictive model remains fixed during real-time operation. To keep the focus, we do not explicitly model potential spatial or temporal correlation between sensors.

Prerequisites. Without loss of generality we assume that the data (including the target variable) is standardized to zero mean and unit variance. To keep the focus we also assume that we know when a sensor fails (we do not need to detect it). We also assume that when a sensor fails, the missing values are automatically replaced with a constant value, say *zero* or the *median* value [2].

2.1 Linear Regression

In this study, we consider linear regression models for prediction [9], which assume that the relationship between r input sensors $\mathbf{x} = (x_1, \ldots, x_r)$ and the target variable y is linear. The model takes the form

$$y = b_1 x_1 + b_2 x_2 + \ldots + b_r x_r + \epsilon = \mathbf{x}\beta + \epsilon, \tag{1}$$

where ϵ is the error variable and the vector $\beta = (b_1, b_2, \ldots, b_r)^T$ contains the parameters of the linear model (regression coefficients). Since the data is assumed to have been standardized, the bias term in the regression model is omitted.

There are different ways to estimate the regression parameters [8,9]. Ordinary least squares (OLS) is a simple and probably the most common estimator. It minimizes the sum of squared residuals giving the following solution

$$\hat{\beta}_{OLS} = \arg\min_{\beta} \big((\mathbf{X}\beta - \mathbf{y})^T(\mathbf{X}\beta - \mathbf{y})\big) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \tag{2}$$

where $\mathbf{X}_{n\times r}$ is a sample data matrix containing n records from r sensors, and $\mathbf{y}_{n\times 1}$ is a vector of the corresponding n target values. Having estimated a regression model $\hat{\beta}$ the predictions for on new data $\mathbf{x}_{new}$ can be made as $\hat{y} = \mathbf{x}_{new}\hat{\beta}$.

If the input data is correlated, regularization is often used for estimating the regression parameters. The Ridge regression (RR) [9, 10] regularizes the regression coefficients by imposing a penalty on their magnitude. The RR solution is

$$\hat{\beta}_{RR} = \arg\min_{\beta} \big((\mathbf{X}\beta - \mathbf{y})^T(\mathbf{X}\beta - \mathbf{y}) + \lambda\beta^T\beta\big) = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}, \tag{3}$$

where $\lambda > 0$ controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage. $\mathbf{I}_{r\times r}$ is the identity matrix.

2.2 Prediction Error

The mean squared error (*MSE*) is a popular measure to quantify the discrepancy between the true target y and the prediction $\hat{y}$. For a test dataset it is computed as $MSE = \sum_{l=1}^{n}(\hat{y}^{(l)} - y^{(l)})^2/n = \hat{\mathbf{y}}^T\mathbf{y}/n$, where n is the number of samples. We use *MSE* since it punishes large deviations from the true values, that is relevant to industrial applications. Also, *MSE* has interesting analytical properties. We can decompose the expected mean squared error as

$$E[MSE] = E\big[\frac{1}{n}\sum_{l=1}^{n}(\hat{y}^{[l]} - y^{[l]})^2\big] = E[(\hat{y} - y)^2] = E[(\hat{y})^2 - 2\hat{y}y + y^2]$$
$$= E[\hat{y}^2] - 2E[\hat{y}y] + E[y^2] = Var[\hat{y}] - 2\,Cov[\hat{y}, y] + Var[y] \qquad (4)$$

The last equation follows from $Var[z] = E[z^2] - (E[z])^2$ and $Cov[x, z] = E[xz] - E[x]E[z]$. $E[y] = 0$ and $E[\hat{y}] = 0$, since the data has been standardized.

Let the prediction be $\hat{y} = \mathbf{x}\beta$. Then the variance of this prediction is

$$Var[\hat{y}] = Var\big[\sum_{i=i}^{r} b_i x_i\big] = \sum_{i=1}^{r}\sum_{j=i}^{r} b_i b_j\, Cov[x_i, x_j] = \beta^T\mathbf{\Sigma}\beta, \qquad (5)$$

where $\mathbf{\Sigma} = \mathbf{X}^T\mathbf{X}/(n-1)$ is the covariance matrix of the input data.

The covariance of the prediction is

$$Cov[\hat{y}, y] = E[\hat{y}y] = E\big[y\sum_{i=1}^{r} b_i x_i\big] = \mathbf{y}^T\mathbf{X}\beta/(n-1). \qquad (6)$$

In real-time predictive systems if a sensor fails, typically, a constant value is displayed. For convenience but without loss of generality we assume that the missing values are replaced by the mean (*zero*, since the data is standardized) as they arrive. Detecting sensor failures is beyond the scope of this work. We assume that the data collection system can signal sensor failures automatically. If this is not the case one can set up a simple rule based detector, such as: if the value is constant for a period of time declare sensor failure.

3 Theoretical Analysis of the Effect of Sensor Failures

Let us consider theoretically the prediction error of a linear model when the input sensors start to fail. Surprisingly, the jump in error can be nonlinear in the number of sensors failed and highly depends on the correlation of the inputs.

Denote by MSE_m the mean square error after m sensors have failed. Let MSE_0 be the error when all the sensors are working. Correspondingly, Var_m and Cov_m denote the variance and the covariance after m sensors have failed. Note that cross-validtion MSE_0 is often the only consideration when assessing the performance of a model or deploying in practice.

Proposition 1. *Suppose sensors fail independently with the uniform prior probability. With m failed sensors the expected error of a linear model is*

$$E[MSE_m] = \frac{r-m}{r} E[MSE_0] + \frac{m}{r} - \frac{(r-m)m}{r(r-1)} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta,$$

where r is the number of input sensors, β is a vector of the regression coefficients, $\mathbf{\Sigma}$ is the covariance matrix of the input data.

MSE can be decomposed into variance of the prediction, covariance of the prediction and the target and the variance of the target, as given in Eq.(4). For proving Proposition 1 we will analyze each component separately.

Proposition 2. *Suppose sensors fail independently with the uniform prior probability. With m failed sensors the expected variance of the prediction $\hat{y} = \mathbf{x}\beta$ is*

$$Var_m[\hat{y}] = \frac{r-m}{r} Var_0[\hat{y}] - \frac{(r-m)m}{r(r-1)} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta.$$

Proof (of Proposition 2). We decompose the variance from Eq. (5) into $Var_0[\hat{y}] = \beta^T \Sigma \beta = \beta^T \beta + \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta$. The first part $\beta^T \beta = \sum_{i=1}^{r} b_i Var[x_i]$ describes the variance when the inputs are linearly independent. The second part $\beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta = 2 \sum_{i=1}^{r-1} \sum_{j=i+1}^{r} b_i b_j Cov(x_i, x_j)$ is due to correlation of the inputs.

Consider the first component $\beta^T \beta$. If a sensor fails, the individual variance becomes zero and the term vanishes. The total variance decreases by $\frac{1}{r} b_1^2 + \frac{1}{r} b_2^2 + \cdots + \frac{1}{r} b_p^2 = \frac{1}{r} \beta^T \beta$. Likewise, if m sensors fail, the variance decreases by $\frac{m}{r} \beta^T \beta$.

Now consider the second component $\beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta$. Since $(\mathbf{\Sigma} - \mathbf{I})$ has zeros on the diagonal, the component is a weighted sum of $r(r-1)$ covariances from the covariance matrix. If one sensor (say, sensor i) fails, all the covariances of other sensors with x_i will become zero and all the terms containing covariance with x_i will vanish. The sum will lose $2(r-1)$ elements (such is the amount of elements with $Cov[x_i, \ldots]$), the total loss will be $\frac{2(r-1)}{r(r-1)} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta$.

However, if two sensors fail then only $2(r-1) + 2(r-2)$ elements will be lost from the sum. If sensors i and j fail, there will be $2(r-1)$ lost containing covariance with x_i, but only $2(r-2)$ more terms lost containing covariance with x_j, as the term $Cov(x_i, x_j)$ has already been lost earlier. Hence, if m sensors fail then $2(r-1) + 2(r-2) + \cdots + 2(r-m) = (2r-1-m)m$ elements will be lost and the collinearity component will decrease by $\frac{(2r-1-m)m}{r(r-1)} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta$.

Plugging the terms into $Var[\hat{y}]$ expression gives
$Var_m[\hat{y}] = \beta^T \beta - \frac{m}{r} \beta^T \beta + \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta - \frac{(2r-1-m)m}{r(r-1)} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta = \frac{r-m}{r} \beta^T \beta +$
$+ \frac{r-m}{r} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta - \frac{(r-m)m}{r(r-1)} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta = \frac{r-m}{r} Var_0[\hat{y}] - \frac{(r-m)m}{r(r-1)} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta.$ □

Proposition 3. *Suppose sensors fail independently with the uniform prior probability. With m failed sensors the expected covariance of the prediction $\hat{y} = X\beta$ is*

$$Cov_m[\hat{y}, y] = \frac{r-m}{r} Cov_0[\hat{y}, y].$$

Proof (of Proposition 3). The covariance is $Cov_0[\hat{y}, y] = E\big[y \sum_{i=1}^{r} b_i x_i\big]$. If a sensor fails, the expected value of that sensor becomes zero, the term becomes independent from y and vanishes. If one sensor fails, the expectation decreases by $\frac{1}{r} y b_1 x_1 + \frac{1}{r} y b_2 x_2 + \cdots + \frac{1}{r} y b_r x_r = \frac{1}{r} y \sum_{i=1}^{r} b_i x_i$. If m sensors fail the expectation decreases by $\frac{m}{r} y \sum_{i=1}^{r} b_i x_i$. Plugging that into the covariance expression gives $Cov_m[\hat{y}, y] = E\big[y \sum_{i=1}^{r} b_i x_i - \frac{m}{r} y \sum_{i=1}^{r} b_i x_i\big] = E\big[\frac{r-m}{r} y \sum_{i=1}^{r} b_i x_i\big] = \frac{r-m}{r} Cov_0[\hat{y}, y]$. Note that the effect of sensor failure on $Cov[\hat{y}, y]$ is the same no matter whether the input data is correlated. □

Given Proposition 2 and Proposition 3 we can now prove Proposition 1.

Proof (of Proposition 1). Following Eq. (4) we decompose the error with m failed sensors into $E[MSE_m] = Var_m[\hat{y}] - 2\,Cov_m[\hat{y}, y] + Var_m[y]$. Failing input sensors do not affect the variance of the true target, thus $Var_m[y] = Var_0[y] = 1$. From Propositions 2, 3 and Eq. (4) we get $E[MSE_m] = \frac{r-m}{r} Var_0[\hat{y}] - 2\frac{r-m}{r} cov_0[\hat{y}, y] + Var_0[y] - \frac{(r-m)m}{r(r-1)} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta = \frac{r-m}{r} E[MSE_0] + \frac{m}{r} - \frac{m(r-m)}{r(r-1)} \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta$. □

For constructing fault tolerant models we will need the next proposition.

Proposition 4. *Given a $r \times r$ covariance matrix $\mathbf{\Sigma}$ and a vector $\beta \in \Re^r$ with at least one non-zero element, the term $\beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta$ is bounded by*

$$-\beta^T \beta \leq \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta \leq (r-1) \beta^T \beta.$$

Proof (of Proposition 4). The Rayleigh quotient of the covariance matrix is defined as $\frac{\beta^T \Sigma \beta}{\beta^T \beta}$, for non-zero $\beta \in \Re^r$ and is bounded by the maximum and the minimum eigenvalues of Σ: $\ell_{min} \leq \beta^T \Sigma \beta / \beta^T \beta \leq \ell_{max}$, where ℓ are eigenvalues, and takes the extreme values when β is equal to the corresponding eigenvectors.

Since Σ is a covariance matrix, all eigenvalues are non-negative and their sum is equal to the sum of the trace. As the data is standardized the sum of eigenvalues is r, hence the maximum eigenvalue does not exceed r: $0 \leq \beta^T \Sigma \beta / \beta^T \beta \leq r$. Algebraic manipulations give the bound $-\beta^T \beta \leq \beta^T (\mathbf{\Sigma} - \mathbf{I}) \beta \leq (r-1) \beta^T \beta$. □

Our analysis relies on theoretical variance and covariance of the prediction. Potentially it could be extended to higher order regression models (e.g. quadratic), that would require much more involved theoretical analysis due to interaction terms. Alternatively, one could obtain non linear prediction models by using the same linear regression with non-linear input features.

Proposition 1 has an important implication. If input data is uncorrelated then *MSE* is increasing linearly with the number of sensors failed. If some sensor fails, the predictive information is lost, there is no source for replacement.

On the other hand, if input data is correlated, *MSE* changes quadratically in the number of sensors lost. From Proposition 4 we see that this quadratic term can be positive or negative depending on the regression model (β). The good news is that a well chosen β may reduce the loss in accuracy to sub-linear. The next section considers strategies for building regression models such that the expected *MSE*, when sensors are failing, is minimized.

Comparison of three regression models

	$\mathbb{d}$	MSE_m
$\beta_1 = (1,0,0,0)^T$	0	$0.25m$
$\beta_2 = (\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})^T$	-0.75	$(0.25m)^2$
$\beta_3 = (2,-\frac{3}{2},1,-\frac{1}{2})^T$	6.5	$1.7m - 0.4m^2$

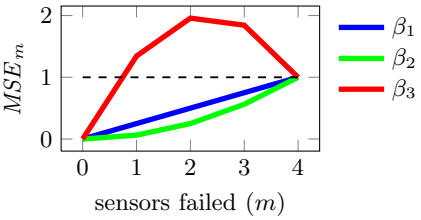

Fig. 1. Three regression models and the deterioration of the prediction accuracy as a function of number of failed sensors (m)

4 How to Build Fault Tolerant Regression Models

Based on the theoretical results next we propose a quantitative measure for assessing robustness of regression models and present practical guidelines on how to build fault tolerant regression models.

4.1 Deterioration Index

In Proposition 1 the term $-\frac{(r-m)m}{r(r-1)}\beta^T(\mathbf{\Sigma}-\mathbf{I})\beta$ decides whether MSE increases linearly, quadratically or sub-linearly due to sensor failures. Hence, for diagnosing model robustness to sensor failures we define a *deterioration index* as

$$\mathbb{d} = -\beta^T(\mathbf{\Sigma}-\mathbf{I})\beta.$$

When input data is uncorrelated, i.e. $\mathbf{\Sigma} = \mathbf{I}$, then $\mathbb{d} = 0$. When input data is correlated $\mathbb{d}$ may be positive or negative (see Proposition 4). If $\mathbb{d} > 0$ then MSE deteriorates quadratically, if $\mathbb{d} < 0$ then MSE deteriorates sub-linearly. Thus, the lower the *deterioration index* the more robust the predictive model is.

4.2 An Illustrative Example

For illustrative purposes, let us consider a small regression problem where four input sensors are perfectly correlated with each other and the target variable: $x_1 \sim \mathcal{N}(0,1)$, $x_1 = x_2 = x_3 = x_4 = y$. Note that $\mathit{\Sigma} = \mathbf{1}_{4\times4}$. Figure 1 gives three regression models that would give perfect predictions if all sensors are working, i.e. $MSE_0 = 0$ and their respective MSE_m after m sensors fave failed (from Proposition 1). Figure 1 plots the expected errors when sensors start to fail.

Model β_1 utilizes only one input sensor and the deterioration of MSE_m is linear to the number of sensors failed (m). Can we do better? In fact we can do better with model β_2, which makes use of the redundancy in sensors. As a result, the loss in accuracy is lower. Model β_3 represents a really bad case of overfitting with the regression, although this model can predict perfectly well, the weights grow unnecessary high. In such a case, if a sensors start failing, the variance of the prediction grows really high and so does the MSE_m. We can observe that even if a single sensor has failed $MSE_1 > 1$ making the predictions worse than a naive baseline that always predicts constant value ($MSE_{constant} = 1$).

4.3 Comparing Robustness of Several Regression Models

Since our goal is to deploy an accurate model that would also be robust to sensor failures, the models that have *small* initial MSE_0 and *small* $\mathbb{d}$ are preferred.

Suppose we have a choice between two regression models A and B. Two situations may occur. First, one model, say A, has a better *deterioration index* and at least equally good initial error: $MSE_0^{(A)} \leq MSE_0^{(B)}$ and $\mathbb{d}^{(A)} < \mathbb{d}^{(B)}$. In such a case for $m = 0 \dots r$ we have $MSE_m^{(A)} \leq MSE_m^{(B)}$ (results from Proposition 1); hence, model A is preferable. Example in Section 4.2 showed such a situation.

The other situation is more tricky. One model, say A, may have a better *deterioration index* , but worse initial error: $MSE_0^{(A)} > MSE_0^{(B)}$ and $\mathbb{d}^{(A)} < \mathbb{d}^{(B)}$. In this case model A is preferable if we expect less than $m^\star$ sensors to fail, and otherwise model B is preferable. We can find $m^\star$ from Proposition 1 by assigning $E[MSE_m^{(A)}] = E[MSE_m^{(B)}]$ and solving for m. The number of sensors to fail is

$$m^\star = (r-1)(MSE_0^{(A)} - MSE_0^{(B)})/(\mathbb{d}^{(B)} - \mathbb{d}^{(A)}). \tag{7}$$

4.4 Building Fault Tolerant Regression Models

A regression model β obtained using the ordinary least squares procedure OLS minimizes MSE_0. The index $\mathbb{d}$ takes its minimum when β is equal to the eigenvector with the maximum eigenvalue (Proposition 4). Unfortunately, such β does not guarantee correct predictions, since eigenvectors are obtained not taking into account the target variable. Hence, for making fault tolerant models we need an optimization criteria that would minimize $\mathbb{d}$ and MSE_0 at the same time.

For a predictive model $\hat{y} = \mathbf{x}\beta$ the *deterioration index* can be decomposed into $\mathbb{d} = -\beta^T(\mathbf{\Sigma} - \mathbf{I})\beta = -Var[\hat{y}] + \beta^T\beta$. We can rewrite Eq. (4) as $Var[\hat{y}] = MSE_0 + 2Cov[\hat{y}, y] - Var[y] \to 1$. Accurate prediction requires $Cov[\hat{y}, y] \to 1$ and $Var[y]$ is fixed. Thus, we cannot vary $Var[\hat{y}]$ without affecting the error.

However, we could vary $\beta^T\beta$ to a certain extent with little impact to MSE_0, for instance, as in the toy example in Section 4.2. Hence, $\mathbb{d}$ will be minimized when $\beta^T\beta$ shrinks. To achieve that we recommend using regularization for building regression models, such as the Ridge regression (Section 2.1).

In addition, we recommend reducing the dimensionality rotating the input data towards the first k principal components. Let $\mathbf{X} = \mathbf{UDV}^T$ be the singular value decomposition of the training data. Let the rotation matrix $\mathbf{R}_{p\times k}$ be composed of the eigenvectors in $\mathbf{V}$ that correspond to the largest eigenvalues recorded in the diagonal of $\mathbf{D}$. Then new k-dimensional input data is $\mathbf{X}^\star = \mathbf{XR}$. Let $\beta^\star$ be a vector of regression coefficients in the transformed k-dimensional space. The model $\beta = \mathbf{R}\beta^\star$ would give the same predictions in the original.

In order to minimize $\mathbb{d}$ we need to minimize $\beta^T\beta$ in the original space, but at the same time we need to find the optimal $\beta^\star$ in the transformed space. Hence, our optimization criteria is $\hat{\beta}^\star = \arg\min_{\beta^\star} \left((\mathbf{X}^\star\beta^\star - y)^T(\mathbf{X}^\star\beta^\star - y) + \lambda\beta^T\beta\right)$. Since $\mathbf{R}$ is orthogonal, thus $\beta^T\beta = \beta^{\star T}\mathbf{R}^T\mathbf{R}\beta^\star = \beta^{\star T}\beta^\star$. Therefore, optimizing the above criteria is equivalent to the Ridge regression in the $\mathbf{X}^\star$ space.

Given these considerations, our recommendation for building fault tolerant models is to apply PCA and then train the Ridge regression in the new space.

5 Experimental Analysis

Next we experimentally analyze the robustness of selected regression techniques against sensor failure on a synthetic benchmark and real sensor datasets.

5.1 Datasets

Chemi dataset from the INFER project[1] describes a chemical production process via 70 real valued sensor variables sampled once per hour over a two years period (17562 instances). The goal is to predict concentration of a product.

ChemiR extends the Chemi data, an additional variable indicates the concentration of the product an hour ago (71 sensors, 17562 instances).

Catalyst dataset[2] is a chemical modeling dataset. Given 13 input variables the goal is to predict catalyst activity. The dataset contains 8687 instances.

Wine dataset from the UCI repository[3] presents 10 chemical measurements as inputs and 4897 instances. The goal is to predict a wine quality score.

CPU dataset from the DELVE repository[4] collects computer systems activity measures described by 19 real valued attributes. The goal is to predict the portion of time that cpus run in user mode. The dataset contains 8192 instances.

Gaussian is a synthetic dataset in 30-dimensional space. Input data is sampled from $\mathcal{N}(\mathbf{0}, \Sigma)$, a random covariance matrix is generated as $\Sigma = s^T s$, where $s \sim \mathcal{U}(-1, 1)$. The target variable is set to $y = \sum_{i=1}^{30} x_i + u$, where $u \sim \mathcal{N}(0, 6)$.

5.2 Regression Models

We test the following regression models.

ALL uses all r sensors as inputs. A regression model is built using the ordinary least squares (OLS) optimization approach.

rALL uses all r sensors and the Ridge regression (RR) optimization.

SEL builds OLS regression on k sensors that have the largest absolute correlation with the target variable (measured on the training data).

PCA rotates the input data using principal component analysis (PCA) and builds the OLS regression on k new attributes with the largest eigenvalues.

rPCA extracts attributes using PCA, but then RR is used instead of OLS.

sPCA rotates the data using PCA, selects k new attributes that are the most correlated with the target. A regression model is built using OLS.

PLS regression is very popular in chemometrics [18]. It is similar to PCA, but instead of maximizing the variance with the rotation, a covariance between the inputs and the target is maximized. We keep k new attributes.

[1] Source: `http://infer.eu/`
[2] Source: `http://www.nisis.risk-technologies.com/`
[3] Source: `http://archive.ics.uci.edu/ml/`
[4] Source: `http://www.cs.toronto.edu/~delve/`

Table 1. Testing MSE_0 and *deterioration index* on the six sensory datasets

		rPCA	PCA	rALL	SEL	sPCA	PLS	ALL
Chemi	MSE_0	0.47	0.47	0.38	0.52	0.42	0.41	0.41
	$\mathbb{d}$	-0.25	-0.25	1.03	0.67	6.42	6.54	7.44
ChemiR	MSE_0	0.42	0.43	0.36	0.35	0.37	0.35	0.34
	$\mathbb{d}$	-0.34	-0.33	0.09	0.53	4.90	3.83	5.57
Catalyst	MSE_0	0.51	0.51	0.45	0.82	0.47	0.44	0.43
	$\mathbb{d}$	-0.11	-0.08	0.21	0.69	0.63	0.82	1.71
Wine	MSE_0	0.83	0.83	0.73	0.76	0.74	0.73	0.73
	$\mathbb{d}$	-0.07	-0.07	0.04	0.12	0.04	0.08	0.18
CPU	MSE_0	0.31	0.31	0.28	0.30	0.29	0.28	0.28
	$\mathbb{d}$	-0.31	-0.32	-0.25	-0.26	-0.16	-0.14	-0.11
Gaussian	MSE_0	0.23	0.23	0.12	0.25	0.14	0.12	0.12
	$\mathbb{d}$	-0.32	-0.33	-0.13	-0.16	-0.10	0	7.58

5.3 Experimental Protocol and Parameters

Each dataset is split into training and testing at random (equal sizes). Some data may have temporal dependencies, hence some predictive information (such as autocorrelation) cannot be utilized, that applies to all the tested models in the same way, while random splits allow multiple tests. We repeat every experiment 100 times and report averaged results. The input data and the target variable is standardized, the mean and the variance for standardization is calculated on the training data. The regression models are trained on the training part and the reported errors and sample covariances are estimated on the testing part. The regression coefficients are always reported in the original (not transformed) feature space. For feature selection SEL, PCA and PLS models we set the number of components to be a half of original number of features: $k = r/2$. The regularization parameter in the Ridge regression experiments is fixed to 200.

5.4 Robustness versus Accuracy

Table 1 reports the testing errors and the *deterioration index* ($\mathbb{d}$) on the six sensory datasets. The models are grouped according to the potential deterioration of their accuracies. We can distinguish three groups of models.

The first group contains PCA and regularized rPCA. These models consistently achieve a very good *deterioration index* (below zero), that guarantees preservation of prediction accuracy. The initial MSE_0 of PCA and rPCA is typically larger than the peer approaches, that is the price to pay for robustness. The superior performance of rPCA and PCA is consistent across the six datasets.

The second group contains regularized rALL and SEL, which have varying *deterioration index* , but typically not too high. rALL maintains a reasonable accuracy (typically better than the first group); however, the accuracy of SEL varies a lot, due to varying predictive power of the individual sensors (depends on the prediction task at hand). The third group contains sPCA, PLS and ALL,

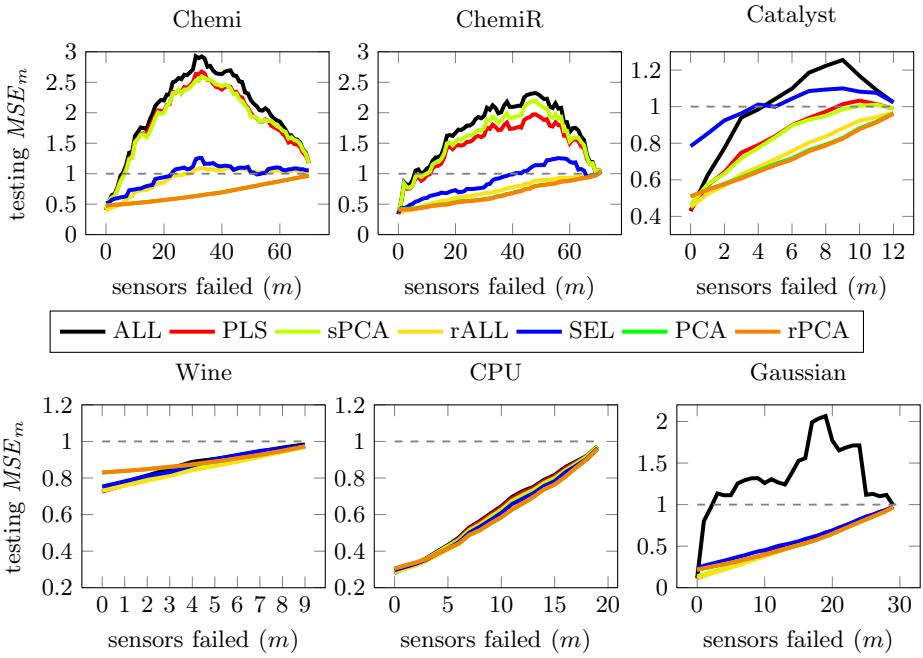

Fig. 2. Empirical MSE_m versus the number of sensors failed

these models mostly show a very high *deterioration index* , especially in Chemi, ChemiR and Catalyst datasets, where the inputs are highly correlated.

Thus, we recommend using rPCA (or PCA) if sensors are expected to fail often and predictions are needed continuously. If failures are rare we recommend rALL that is less robust but more accurate with all the sensors working.

5.5 Empirical Analysis of Deterioration of Accuracy

Next we investigate how the error depends on the number of sensors that have failed. Figure 2 shows testing MSE_m as a function of the failed sensors. We chose sensors to fail uniformly at random, we report the results over 100 runs.

Advantages of PCA and rPCA are prominent in Chemi and ChemiR, where the dimensionality is large and the input data is strongly correlated. All the models perform similarly (nearly linear loss) in Wine and CPUact, where the input data is not much correlated. In Catalyst PCA and rPCA have an advantage as expected based on *deterioration index* . On this dataset SEL has notably worse performance. As the overall number of features is low (12), quite a lot of initial accuracy is lost by dropping half of the features. Gaussian data strictly follows the normal distribution, and the contribution of each sensor to the target variable is uniformly distributed. ALL performs notably badly, but we see that any regularization attempt (all the other methods) leads to a good performance.

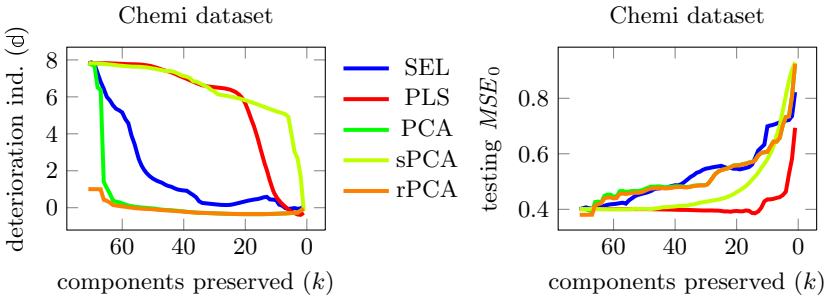

Fig. 3. Deterioration index and prediction error as a function of components preserved

5.6 Sensitivity to the Number of Extracted Components

In our analysis number of components extracted in PCA, rPCA, sPCA and PLS, as well as the number of features selected in SEL was fixed. Next we analyze how the *deterioration index* and the prediction error depends on the number of components on the Chemi dataset that has high dimensionality ($r = 70$). We analyze all five models that take the number of components as a parameter: PCA, rPCA, sPCA, SEL and PLS. Our goal is to assess the stability with respect to sensor failures at arbitrary selected number of components. Note, that if all the components are preserved ($k = r$) then PCA, sPCA, PLS and SEL are equivalent to ALL, and the regularized rPCA is equivalent to rALL.

Figure 3 shows the *deterioration index* and the prediction error as a function of extracted components (over 100 runs). The regularized rPCA performs much better đ than PCA when nearly all of the components are retained. PCA and rPCA demonstrates superior đ across all k in line with the previous experiments. SEL demonstrates a mediocre đ and sPCA together with PLS keeps a dangerously high đ until the majority of the components are discarded (k is below 10). We see from the right plot in Figure 3 that, unfortunately, at such a low k a lot of prediction accuracy is lost, *MSE* is nearly twice as large with all the sensors.

Overall, we see a tendency to achieve a better *deterioration index* at an expense of a lower initial prediction accuracy. The regularized rPCA demonstrates the most stable performance and superior results throughout all the range of k.

5.7 Worse Than Blind Guessing

Blind guessing is a naive prediction, that does not use any input data and always predicts the average of the target variable. Next we analyse how many sensors can fail before predictions become worse than blind guessing. Table 2 reports empirical results on the six datasets averaged over 100 runs.

We see that in Gaussian the problem of sensor failure is very serious, it is enough for two sensors out of 30 to fail and the predictive model is useless. In case of Chemi and ChemiR it is enough for 6-7 sensors to fail out of 70-71 to make ALL or even PLS useless. PLS is a very popular state of the art technique

Table 2. No. of sensors to fail before the prediction becomes worse than blind guessing

	rPCA	PCA	rALL	SEL	sPCA	PLS	ALL	sensors
Chemi	-	-	29	23	7	7	6	out of **70**
ChemiR	-	-	70	41	10	9	6	out of **71**
Catalyst	-	-	-	4	10	9	5	out of **12**
Wine	-	-	-	-	-	-	-	out of **9**
CPU	-	-	-	-	-	-	-	out of **19**
Gaussian	-	-	-	-	-	-	3	out of **30**

often used in chemometrics applications [13, 18], such as Chemi. From Figure 2 we can see that if just *one* sensor is lost, the error of PLS or ALL in Chemi already jumps up by nearly 40%. We see that the mean imputation of missing sensor values is a serious problem for these regression models.

The experimental results suggest that careful regularization measures are needed for ensuring that predictive models stay functional during real-time operation. The experimental results confirm our theoretical findings and the indications of the *deterioration index* that the proposed rPCA and PCA can effectively prevent rapid boosts in errors due to sensor failures. If a user does not have the capacity to determine the optimal k, as a rule of thumb from our practical experience we recommend using $k = r/2$, where r is the number of input sensors.

6 Case Study in Environmental Monitoring

To validate our findings we perform a case study in environmental monitoring where sensor failures are happening frequently. The task is to predict the *level of solar radiation* from meteorological sensor data (such as temperature, precipitation, wind speed). We use a data stream recorded at SMEAR II station in Finland [12]. This station can measure solar radiation; hence, the ground truth is available for us. In general, measuring solar radiation is delicate and expensive. Not many stations can afford to measure solar radiation and would be interested in predicting it from other data that can be collected much cheaper and easier.

We use data over a five years period (2007-2012), recorded every 30 min. from 39 meteorological sensors at one station. The data coming from the station has about 7% of missing values. There is no single sensor that would provide non interrupted readings over those five years; for any sensor from 1% up to 30% values are missing. The solar radiation (target variable) is available 99% of the times, we eliminate from the experiment the instances having no target value.

Our goal is to verify if the proposed *deterioration index* can effectively diagnose the performance of regression models and test the performance of our regression models with naturally occurring missing data. We use the first two years of data as a training set and the remaining three years as a testing set. From the training set we eliminate all the instances that contain missing values (-25% of train data). We standardize the training set (zero mean, unit variance). Then we standardize the testing set using the mean and the variance values obtained from the training set. After standardization we replace all the missing values in the testing set by zeros and test the regression models.

Table 3. Accuracy and stability on the environmental monitoring data

	rPCA	PCA	rALL	SEL	sPCA	PLS	ALL
MSE on all test data	0.37	0.37	0.51	15.67	8.55	0.93	7.39
MSE on non-missing test data	0.33	0.33	0.29	0.35	0.29	0.28	0.28
MSE on missing test data	0.39	0.39	0.63	23.88	12.99	1.27	11.21
MSE on train data (cross-validation)	0.40	0.39	0.37	0.34	0.31	0.32	0.29
₫ on train data	-0.22	-0.20	1.61	574.58	253.86	8.66	302.84

Table 3 reports the testing results of the regression models ALL, rALL, SEL, PCA, rPCA, sPCA and PLS ($k = 20$, which is half of the input sensors following the rule of thumb suggested in Section 5.6). The recommended rPCA and PCA demonstrate an outstanding performance ($MSE = 0.37$), followed by rALL (0.51). The performance of PLS (0.91) is more than twice worse as of PCA and rPCA. ALL, sPCA and SEL perform much worse by a large margin.

Next we split of the test data into non-missing (35%) and missing data (65%) parts and inspect the errors separately. We see that the performance data of all the models is very similar when there is no missing data. However, the non-regularized models (ALL, SEL, sPCA and PLS) fail badly when there is missing data, except for PCA, which is consistent with our theoretical findings. Moreover, we can see from the last part of the table that if we selected a model for deployment based on cross-validation *MSE*, we would probably deploy ALL. It would perform on non-missing data well, but the performance would deteriorate very drastically when sensors started to fail. Finally, we can see that the proposed *deterioration index* computed on the training data indicates very well the future robustness of the model. Hence, after seeing a comparable cross-validation performance of all models we would deploy rPCA that gives the minimum ₫.

The results support our recommendation to use PCA and rPCA when temporal sensor failures are expected. The case study also confirms the effectiveness of the *deterioration index* in diagnosing robustness of predictive models.

7 Related Work

Our study is closely connected with handling missing data research, see e.g. [1,2, 4,14]. The main techniques are: imputation procedures where missing values are filled in and the resulting compete data is analyzed, reweighing procedures where instances with missing data are discarded or assigned low weights, and model-based procedures that define models for partially missing data. We investigate what happens *after* missing values are imputed during real-time operation using a very popular and practical mean value imputation. In our setting discarding streaming data is not suitable, since there would be continuous periods when we have no input data and thus no predictions. Model-based procedures could handle one-two missing sensors; however, when many sensors may fail, such a procedure is computationally impractical and likely infeasible, as we would need to keep an exponential number of models to account for all possible situations.

Handling missing values in regression is reviewed in [14]. The majority of research focuses on training regression models from data with partially missing values. In our setting discarding some *training* data with missing values is not a problem, since the volumes of data are typically large. The problem arises during real-time operation. We not only need to input missing values, but also make the regression models fault tolerant. Hence, our work solves a different problem and is not directly comparable with missing value imputation techniques.

Topic-wise our work relates to fault tolerant control that is widely researched and applied in industrial and aerospace systems [16, 19]. The main focus is on detecting the actual fault, not operating with a fault present. In our setting there is no fault in the system, just sensors fails, our model needs to remain accurate.

Redundancy in engineering duplicates critical components of a system to increase reliability, see e.g. [5]. A common computational approach is to to use an average the redundant sensors to reduce the impact of possible sensor failure. In fact, this is the effect we are aiming to achieve by minimizing the *deterioration index* . The main difference from our setting is in availability of backup sensors, it is even possible to install duplicate sensors on demand. In our setting; however, the data is given as is and we aim at exploiting it in the best way.

Robust statistics aims at producing models that are robust to *outliers* or other small departures from model assumptions, see e.g. [11]. The main idea is to modify loss functions so that they do not increase so rapidly, to reduce the impact of outliers. In our setting there are no large deviations in the input data due to sensor failure, in fact the opposite, the variance of a failed sensor goes to zero. Hence, robust statistics approaches target a different problem.

Our theoretical analysis of the mean squared error resembles bias-variance analysis (see e.g. [7]) in the way we decompose *MSE* into components. Regarding the connection of the bias-variance decomposition to the Ridge regression solution, we well know that enforcing strong regularisation is likely to decrease variance and to increase bias. Further investigation is left for future work.

Finally, the setting relates to concept drift [20] and transfer learning [15] settings in a sense that the training and the testing data distributions are different. However, in our setting there is no model adaptation during real-time operation.

8 Conclusion

Systems relying on predictive models should be robust with regard to missing input values, due to transient failures in the sensors, for instance. We focused on linear models for predictions, and theoretically analyzed the criteria for linear regression to be robust to sensor failures. Based on this analysis we introduced the *deterioration index* measure that allows to quantify how robust is a given linear regression model to sensor failure. We also proposed a practical strategy for building robust linear models. Our experiments with real data confirmed the theoretical results and demonstrated the effectiveness of the proposed strategy.

The current work assumes that input sensors fail with the uniform prior probability, but does not quantify any distribution on how many are likely to fail,

or how the failures form correlated patterns among the sensors. These questions would make an interesting follow up investigation. Mapping the findings of the current study to predictive models in the evolving data stream setting offers another important avenue for future research.

Acknowledgments. We thank the INFER project for Chemi dataset. This work has been supported by the Academy of Finland grant 118653 (ALGODAN).

References

1. Alippi, C., Boracchi, G., Roveri, M.: On-line reconstruction of missing data in sensor/actuator networks by exploiting temporal and spatial redundancy. In: Proc. of the 2012 Int. Joint Conf. on Neural Networks, IJCNN, pp. 1–8 (2012)
2. Allison, P.: Missing data. Sage, Thousand Oaks (2001)
3. Brobst, S.: Sensor data is data analytics' future goldmine (2010), `http://www.zdnet.com`
4. Ciampi, A., Appice, A., Guccione, P., Malerba, D.: Integrating trend clusters for spatio-temporal interpolation of missing sensor data. In: Di Martino, S., Peron, A., Tezuka, T. (eds.) W2GIS 2012. LNCS, vol. 7236, pp. 203–220. Springer, Heidelberg (2012)
5. Frank, P.: Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: a survey and some new results. Automatica 26(3), 459–474 (1990)
6. Gama, J., Gaber, M. (eds.): Learning from Data Streams: Processing Techniques in Sensor Networks. Springer (2007)
7. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Comput. 4(1), 1–58 (1992)
8. Golub, G., Van Loan, C.: Matrix Computations. Johns Hopkins Un. Press (1996)
9. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference, and prediction. Springer (2001)
10. Hoerl, A., Kennard, R.: Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 42(1), 55–67 (1970)
11. Huber, P.: Robust statistics. Wiley (1981)
12. Junninen, H., Lauri, A., Keronen, P., Aalto, P., Hiltunen, V., Hari, P., Kulmala, M.: Smart-SMEAR: on-line data exploration and visualization tool for smear stations. Boreal Env. Res., 447–457 (2009)
13. Kadlec, P., Gabrys, B., Strandt, S.: Data-driven soft sensors in the process industry. Computers & Chemical Engineering 33(4), 795–814 (2009)
14. Little, R.: Regression with missing X's: A review. Journal of the American Statistical Association 87(420), 1227–1237 (1992)
15. Pan, S., Yang, Q.: A survey on transfer learning. IEEE Trans. on Knowl. and Data Eng. 22(10), 1345–1359 (2010)
16. Patton, R.: Fault-tolerant control: the 1997 situation. In: Proc. of the 3rd IFAC Symp. on Fault Detection, Superv. and Safety for Tech. Proc., pp. 1033–1055 (1997)
17. Qin, J.: Recursive PLS algorithms for adaptive data modeling. Computers & Chemical Engineering 22(4-5), 503–514 (1998)
18. Wold, S., Sjostroma, M., Eriksson, L.: PLS-regression: a basic tool of chemometrics. Chemometrics and Intelligent Laboratory Systems 58(2), 109–130 (2001)
19. Zhang, Y., Jiang, J.: Bibliographical review on reconfigurable fault-tolerant control systems. Annual Reviews in Control 32(2), 229–252 (2008)
20. Zliobaite, I.: Learning under concept drift: an overview. CoRR, 1010.4784 (2010)