Time-Series Prediction of a Waste Water Treatment Plant

A case study in data analysis using support vector machines

NIKLAS COLLIN

Master’s Degree Project
Stockholm, Sweden 2004

TRITA-NA-E04125
Time-Series Prediction of a Waste Water Treatment Plant

A case study in data analysis using support vector machines

NIKLAS COLLIN

TRITA-NA-E04125

Master’s Thesis in Computer Science (20 credits) at the School of Computer Science and Engineering, Royal Institute of Technology year 2004
Supervisor at Nada was Stefan Arnborg
Examiner was Stefan Arnborg
ABSTRACT

This thesis concerns a time-series prediction problem on a waste water treatment plant in Barcelona, Spain. Using a fairly new method called Support Vector Machines (SVM), the problem was to find a model to reasonably predict the output of solids at sample day t+1, where samples are obtained each day. The problem was defined as a two class problem where one class signifies normal operating conditions and the other an excess in the output. The goal was to find the single best SVM model during the time available.

This problem needed considerably more time than was available during the project. The most serious factors in affecting performance negatively were unknown important input variables, access to only a year's worth of data and missing values.

The final model identified 73% of the excess cases in an independent test data set, previously unseen, which contained only 6% of these cases. However, due to false alarms, only 50% of the predictions actually came true. The results were thus encouraging but inconclusive.

This thesis will serve the reader best as a data analysis case study. The methods employed herein could prove to be a useful addition to the reader’s own toolbox. These include:

- Data pre-processing by
  - Missing value interpolation
  - Normalisation
  - Adaptation of time series data by time windowing
  - Data partitioning into training and test data sets for reliable performance analysis

- Feature selection by
  - Principal Component Analysis (PCA)
  - RELIEF algorithm

- SVM model selection through
  - Parameter optimisation
  - Kernel selection
  - Performance measurement by leave-one-out cross validation
TIDSSERIEPROGNOS AV ETT VATTENRENINGSVerk

SAMMANFATTNING

Detta examensarbete behandlar ett tidsserieprognosproblem av ett vattenreningssverk i Barcelona, Spanien. Projektet gick ut på att hitta en modell för att förutspå utsläppsmängd vid avläsningsdag t+1, då avläsningar gjordes dagligen, genom användandet av en relativt ny metod kallad ”Support Vector Machines” (SVM). Problemet definierades som ett tvåklassproblem där ena klassen representerade normalt tillstånd och den andra ett gränsvärdesöverskridande utsläpp. Målet var att finna bästa möjliga SVM-modell under tillgänglig tid.

Detta var ett svårt problem som skulle kräva långt mer tid än vad som var tillgängligt under projektet. De störst bidragande faktorerna till svårigheter var avsaknad av viktiga indatavariabler, tillgång till endast ett års data och saknade mätvärden i datat.

Den slutgiltiga modellen lyckades identifiera 73% av gränsvärdesöverskridande fall i en oberoende, ej tidigare använd testdatamängd i vilken endast 6% av sådana fall existerade. Trots detta, på grund av falsklarm, överensstämde bara 50% av förutsägelserna med verkligheten. Resultaten var således uppmuntrande men otillräckliga i praktiken.

Denna rapport bistår läsaren på bästa sätt i form av en exempelstudie. Metoderna som används här skulle kunna bli ett värdefullt tillskott till läsarens verktøyglåda. Dessa inkluderar bland annat:

- Dataförbehandling genom
  - Interpolation av saknade mätvärden
  - Normalisering
  - Anpassning av tidsseriedata genom tidsfönsterpartitionering
  - Datapartitionering i tränings- och testdatamängd för pålitlig prestandaanalys
- Variabelanalys via
  - Principalkomponenten
  - RELIEF-algoritmen
- Val av SVM-modell genom
  - Parameteroptimisering
  - Utvärdering av kernel
  - Prestandautvärdering via ”leave-one-out cross validation”
## CONTENTS

**Background** ............................................................................................................................... 1
- Problem definition ................................................................................................................ 2
- Difficulties .......................................................................................................................... 2

**Support Vector Machines theory** .......................................................................................... 4
- The classification problem ............................................................................................. 4
- Linear learning machines ........................................................................................... 5
- Feature space ...................................................................................................................... 5
- Kernels ............................................................................................................................... 6
- Generalisation theory ...................................................................................................... 7
- Sparse representation ..................................................................................................... 7
- Soft margin SVM ............................................................................................................. 7
- Incremental SVM .......................................................................................................... 9

**Input variable specification** ................................................................................................... 10

**Data pre-processing** ................................................................................................................. 11
- Outline ............................................................................................................................. 11
- Missing value treatment ............................................................................................... 11
- Normalisation ................................................................................................................... 14
- Time windowing ............................................................................................................... 15
- Variable importance analysis ....................................................................................... 16
- Principal Component Analysis (PCA) ........................................................................... 17
- RELIEF algorithm ......................................................................................................... 20
- Comparison between Relief and PCA results ............................................................ 23

**SVM model selection** .............................................................................................................. 24
- Definition ........................................................................................................................ 24
- Outline ............................................................................................................................. 24
- Performance measurement by cross validation ............................................................. 24
- Data partitioning into training and test data sets ......................................................... 25

**Parameter correlation with SVM performance** .................................................................. 26
- Goal ................................................................................................................................. 26
- Outline ............................................................................................................................. 26
- Results ............................................................................................................................. 27
- Analysis ........................................................................................................................... 29
- Conclusions ..................................................................................................................... 30
FIGURES

Figure 1 – The optimal separating hyper plane maximises the margin. ........................................ 4
Figure 2 – Non-linearly separable case that becomes linearly separable in feature space........... 5
Figure 3 – The support vectors alone define the maximal margin hyper plane......................... 7
Figure 4 – Slack variables (ξ) allow the margin constraints to be violated. ............................. 8
Figure 5 – Maximal margin SVM fitting of data where all points are correctly fitted. ............. 9
Figure 6 – Soft margin SVM fitting of data where two points are incorrectly fitted............... 9
Figure 7 – Simplified WWTP schematic ................................................................................. 10
Figure 8 – Example of linear interpolation .............................................................................. 12
Figure 9 – Interpolation plot of variable V30-2 with P = 1..................................................... 13
Figure 10 – Interpolation plot of variable V30-2 with P = 10.................................................. 14
Figure 11 – PCA analysis example on two variables X and Y ................................................. 17
Figure 12 – PCA scores for all input variables except MES-S ................................................. 19
Figure 13 – Pseudo code for RELIEF algorithm.................................................................... 20
Figure 14 – Second degree polynomial kernel on all input variables .................................... 27
Figure 15 – Third degree polynomial kernel on all input variables ......................................... 28
Figure 16 – Fourth degree polynomial kernel on all input variables ....................................... 28
Figure 17 – Gaussian kernel on all input variables ................................................................. 29
Figure 18 – Results visualised per input variable exclusion for all kernels......................... 32
Figure 19 – Results visualised per kernel for all input variable exclusions.......................... 33
Figure 20 – Example of an irregular performance curve with MES-S excluded..................... 34
Figure 21 – Data from previous experiments in the area of the minima............................... 38
Figure 22 – Finer granularity by four times as many sampling points. ............................... 39
TABLES

Table 1 – The set of input variables suggested by the plant engineers................................. 10
Table 2 – Number of missing values in data per input variable............................................ 11
Table 3 – Normalisation example of two variables X and Y ............................................... 15
Table 4 – Example of a four samples wide time window of one variable X ....................... 15
Table 5 – The new input set of 28 inputs created by time windowing.................................. 16
Table 6 – Factors produced by PCA-analysis of input variables........................................ 19
Table 7 – Results of the RELIEF-algorithm ...................................................................... 22
Table 8 – Most important variables according to RELIEF ................................................. 22
Table 9 – Least important variables according to RELIEF .................................................. 22
Table 10 – Variables of indeterminable importance according to RELIEF ......................... 22
Table 11 – Important variables according to both PCA and RELIEF ................................. 23
Table 12 – Unimportant variables according to both PCA and RELIEF ............................. 23
Table 13 – Important variables according to PCA but not RELIEF .................................... 23
Table 14 – Cross-validation errors for all kernels and input variable exclusions............... 32
Table 15 – Performance ranking of input variable exclusions............................................. 35
Table 16 – Performance improvement of Gaussian over other kernel types ....................... 36
Table 17 – Definition of confusion matrix.......................................................................... 41
Table 18 – Performance of final model on test data set....................................................... 42
BACKGROUND

The control and prediction of Wastewater Treatment Plants (WWTP) is important in order to keep the system stable under a wide range of circumstances and avoid disturbing the environmental balance. Investigating the availability of models characterising WWTP behaviour as a dynamic system is thus a necessary first step. However, due to the high complexity of the involved processes and the heterogeneity, incompleteness and imprecision of WWTP data, finding suitable models entails substantial problems.

Besides routine analysis to monitor the process there is an increasing tendency in sensors and probes to measure different variables in real-time. These include the flows, pH, temperature, suspension solids, nutrients and organic matter. There are also other vital parameters that cannot be measured online such as the amount of foam present either in the bioreactor surface or in the settlers, the proliferation of lamentous microorganisms or the sludge volume index (SVI).

Besides microbiological observation of the sludge, the analytic parameter SVI reasonably predicts bulking episodes, which is an important phenomenon. It consists of an activated sludge instability characterised by a deficient separation of liquids and solids in the secondary settlement phase of the WWTP process. The phenomenon appears occasionally or even permanently in most biological WWTP:s. Consequently, it has been extensively investigated. Correlation studies between the presence of specific lamentous microorganisms and the control parameters of various WWTP:s have led to the conclusion that the phenomenon is due to the proliferation of the former.

Research contributions in this field have been formulated from many different points of view. However, a direct cause-effect relationship for WWTP performance has been established only in a few cases and, even in those, experimental results could lead to contradictory conclusions (see [7]), precluding the formulation of deterministic cause-effect relationships that could be used as prediction models. The identification of a model that could reasonably predict bulking episodes in real-time and with reasonable accuracy is thus of great practical importance in view of a potential improvement of treatment plant efficiency and cost savings (see [8]).

 Artificial neural networks have been used to develop prediction models for bulking, with some degree of success (see [12]). The support vector machine (SVM) is a recent approach and has attracted a great deal of attention within the machine learning community due to its good theoretical properties and its growing and successful practical use. An incremental SVM has been developed within the LSI department at UPC, Barcelona in Spain (Universitat Politecnica de Catalunya) where this project was performed.
The aim of this project was to create an SVM based model of the WWTP capable of predicting the output of solids based on a subset of the available parameters that were measured. There is a legislative limit (EU directive 91/271/CEE) to the amount of solids that the plant is allowed to output. The point of interest was whether this amount might be exceeded in a near future. Such anomalies in the process are mainly caused by malfunctions of a microbiological nature, such as the bulking phenomenon, which is difficult to reasonably predict with classical fault detection systems.

See [6] for more information on modelling, control and diagnosis of WWTP:s.

**Problem definition**

The task at hand was to find an SVM based classification model that could reasonably predict the output of solids of the WWTP at sample day t+1 (samples were obtained each day). The set of models would be narrowed down to the single best SVM model possible during the time available, i.e. the one exhibiting the best performance. In the context of this thesis, SVM-performance is equivalent to prediction quality.

The main interest was to reasonably predict whether the output of solids would exceed 35 mg/l/day (an EU legislative limit) and cause the plant to be fined. Prediction would allow the abnormality to be rectified sooner, perhaps even before it occurred. The problem was therefore formulated as a classification problem with two classes:

- **Class A** - “Not exceeded” ($\leq 35$ mg/l/day).
- **Class B** - “Exceeded” ($> 35$ mg/l/day)

**Difficulties**

Of course there are several ways of doing this kind of prediction where SVMs are but one promising way of dealing with the problem. The problem is hard and a graduate project is time-limited. It was therefore decided that investigating the possibility of using SVMs to attack this problem would be a reasonably well-defined and implementable project given the time and computational resources available. It is important to note that the problems described in this thesis can be solved in many other ways.

There are several factors causing difficulties in this project. The data contains missing values for which there is no default way of interpolation for assessment. Before using the SVM one must know which variables of the data are most important and which are not important since inclusion of unimportant variables will affect performance negatively.

The responsibility of selecting the set of relevant variables lies primarily but not entirely on the chemical engineers at the WWTP. The computer scientist must also use various algorithms to try and select the best
possible subset of variables from the set of variables originally chosen. However, a close cooperation with the engineers at the plant is necessary to get the most relevant selection which was not a possibility during this project.

The SVM's performance is controlled by several parameters for which there is no obvious method of optimisation. Performance also greatly depends on a function called the "kernel"-function which defines the non-linearity and expressive power of the machine of which there are several types. However, the difficulties particular to the SVM will be described in more detail in upcoming chapters.
Support vector machines are based on an assortment of different theories. This chapter first explains the way they collectively make up the simplest SVM called the maximal margin classifier. Then follows the type of SVM used for the work in this thesis called the soft margin classifier which is based on the maximal margin classifier. There also SVMs for regression analysis. This thesis, however, only involves classification SVMs.

The explanations contained in this report are deliberately brief since there are other better sources for detailed information on the subject. The aim here is to portray the general idea of an SVM and its unique advantages. For more information on SVMs and their foundations see [1].

The classification problem

The classification problem can be restricted to consideration of the two-class problem without loss of generality. In this problem the goal is to separate the two classes by a function which is induced from available examples. The goal is to produce a classifier that will work well on unseen examples, i.e. it generalises well.

Consider the example in figure 1. Here there are many possible linear classifiers that can separate the data, but there is only one that maximises the margin (maximises the distance between it and the nearest data point of each class, marked by dotted lines in the figure). This linear classifier is termed the optimal separating hyperplane. Intuitively, one would expect this boundary to generalise better than other possible boundaries at least in most cases.

Figure 1 – The optimal separating hyperplane maximises the margin. (source: [1]).
Linear learning machines

Perceptrons are linear machines designed to find the optimal separating hyperplane. Consider a training set with \( l \) points. The perceptron will find a decision function that can be used to classify new points in the input space whose class is unknown.

Formula 1 presents the decision function which represents the optimal separating hyperplane. It accepts a new point in the input space and returns the class to which the point belongs based on which of side of the plane the point is located.

\[
h(x) = \text{sgn} \left( \sum_{j=1}^{l} \alpha_j y_j \langle x_j, x \rangle + b \right)
\]  

(1)

The decision function performs a sum over the training points of which \( y_j \) and \( x_j \) are the class and training point respectively. \( x \) is the new point in the input space to be classified. The weights \( \alpha_1, \ldots, \alpha_l \) and displacement \( b \) are pre-calculated during training of the perceptron on the training set.

Note in this decision function how the training points and the new point do not appear independently but only inside a dot product. This is of significant importance in the next section.

Feature space

The previous section presented a simple linear case for which the perceptron is adequate. In our next example the points are not linearly separable but instead only non-linearly separable as in figure 2. In this case, they could be made linearly separable by a non-linear mapping \( \phi \) to a new space called feature space as shown in formula 2 below.

\[
\phi(x) = \phi(y_j \langle x_j, x \rangle + b)
\]

Figure 2 – Non-linearly separable case that becomes linearly separable in feature space. (source: [1]).
\[ \mathbf{x} = (x_1, \ldots, x_n) \rightarrow \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_n(\mathbf{x})) \] (2)

Feature space has a higher dimensionality than the input space. The idea is that even if the input is not linearly separable in the input space, it will be in the feature space where added dimensionality gives a higher degree of freedom to fit the hyper plane. The mapping could also spread out the data points over the additional dimensions in feature space making it even simpler to fit the hyper plane.

This means that when the decision function is used, the new data points in need of classification will first be mapped into feature space before looking at which side of the hyper plane they lie to find out their class. To facilitate this, the decision function needs to be adapted by substituting the training points and the unclassified point for their mapped positions in feature space. This makes the perceptron implicitly fit the plane in feature space. Formula 3 shows the modified decision function:

\[ h(\mathbf{x}) = \text{sgn} \left( \sum_{j=1}^{l} \alpha_j y_j \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle + b \right) \] (3)

**Kernels**

The feature space is a good idea and it can be taken further. It would be useful if one could directly compute the inner product and mapping into feature space in a single step. Such a direct computation method is called a "kernel" function. A kernel is thus a function \( K \) as defined in formula 4:

\[ K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \] (4)

\( \mathbf{x}, \mathbf{z} \in X \) (where \( X \) is the input space)

Once such a function is defined, the decision rule can be evaluated as portrayed in formula 5:

\[ h(\mathbf{x}) = \text{sgn} \left( \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}) + b \right) \] (5)

The kernel function is a dot product in the feature space but when it is defined it is not necessary to think about it as such. Any function \( K(\mathbf{x}, \mathbf{z}) \) will do as long as it complies with a special set of conditions. These conditions will make sure that \( K(\mathbf{x}, \mathbf{z}) \) actually defines a dot product in some feature space. See [1] for more information on kernel conditions.

The perceptron has no knowledge of a feature space since it is implicitly defined in the kernel function. As far as it is concerned, it is only doing a linear fitting of the hyper plane. This permits many non-linear problems to be easily coped with using different kinds of kernels.
**Generalisation theory**

The introduction of kernels greatly increases the expressive power of the learning machines while retaining the underlying linearity that will ensure that learning remains tractable. The increased flexibility, however, increases the risk of over fitting as the choice of separating hyper plane becomes increasingly ill-posed due to the number of degrees of freedom.

SVMs comply with a set of statistical constraints that theoretically bounds the generalisation error and in effect makes it hard for SVMs to over or under fit the data and enables them to easily handle high-dimensional inputs. See [1] for more information on generalisation theory and SVMs.

**Sparse representation**

To define the optimal hyper plane all training points are not necessary. Only the training points on the margin (see figure 3 below) define the position of the hyper plane. These are called support vectors and give Support Vector Machines their name. This leads to another of the SVM’s big advantages that the data representation of an SVM is extremely small and efficient.

![Figure 3 – The support vectors alone define the maximal margin hyper plane. (source: [1]).](source)

The decision function only needs to use the support vectors to classify a new data point. The support vectors typically consist of but a fraction of the training points. This makes SVMs both fast and memory efficient.

**Soft margin SVM**

The maximal margin classifier is an important concept, as a starting point for the analysis and construction of more sophisticated SVMs, but it cannot be used in many real-world problems. If the data is noisy, there will in general be no linear separation in the feature space (unless we are ready to use very powerful kernels, and hence over fit the data). The main
problem is that it always produces a perfectly consistent hypothesis with no training errors making it sensitive to the idiosyncrasies of a few points that could be outliers or noise.

To deal with this one introduces a soft margin. This means that one allows for the margin and hyper plane constraints to be violated, i.e. some training points might lie inside the margin or on the wrong side of the plane. It is implemented by so called margin slack variables. There is one slack variable $\xi_i$ for each training point $i$ making it possible to define how much it is allowed to violate the soft margin $\gamma$ as presented in figure 4:

![Figure 4 - Slack variables ($\xi$) allow the margin constraints to be violated. (source: [1]).](image)

The overall tolerance of the soft margin is controlled by the C-parameter which will be referred to later when the SVM is used for the particular problem presented in this thesis.

$C$ is bounded by $0 < C < \infty$. At the two extremes $C \to \infty$ and $C \to 0$ the machine tends to over fit (training errors decrease at the expense of fitting noise, i.e. generalisation error increases) and under fit (generalisation error increases because the model is not capturing an essential part of the underlying function), respectively. $C \to \infty$ equals the maximal margin case and $C \to 0$ equals having no margin constraints at all.
Figure 5 and figure 6 give an example of the difference between a soft and maximal margin classifier. The maximal margin classifier makes sure all training points are classified correctly whereas the soft margin SVM allows for two outliers to violate the margin constraints.

**Incremental SVM**

The actual SVM used in this project was an incremental SVM which has further advantages in that it is fully online. This means that it can be updated by removing or adding training points in real-time while the machine is running, whereas the previous SVM has to do a complete recalculation when any change occurs in the training data.

For more information, see [9] and [10].
The engineers at the plant proposed to use the data of the specific subset of input variables presented in table 1 and visualised in figure 7. The variables were measured for a period of 587 consecutive days. MES-S is the variable this project aimed to reasonably predict at day t+1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>Input flow of soiled water into plant.</td>
<td>m³</td>
</tr>
<tr>
<td>MLSS-2</td>
<td>Solids at the bioreactor.</td>
<td>mg/l</td>
</tr>
<tr>
<td>V30-2</td>
<td>Analytical test where a bucket with a litre of water from the reactor is left to settle for 30 minutes whereupon the volume of settled sludge is measured. This gives a rough idea of sludge settling.</td>
<td>ml</td>
</tr>
<tr>
<td>RASS-2</td>
<td>Recirculation flow which is performed to recover part of the biomass that gets lost along with the outgoing water.</td>
<td>m³</td>
</tr>
<tr>
<td>WAS-2</td>
<td>Secondary purge flow. The purge is performed to get rid of an excess of biomass.</td>
<td>m³</td>
</tr>
<tr>
<td>WAS-3</td>
<td>Purge flow at thickener.</td>
<td>m³</td>
</tr>
<tr>
<td>MES-S</td>
<td>Plant output of suspended solids, i.e. a measurement of how clean the water recirculated back out into nature is.</td>
<td>mg/l</td>
</tr>
</tbody>
</table>
DATA PRE-PROCESSING

A crucially important part of a data analysis project is to understand the data and pre-process it correctly. No method would succeed if this step is not done. The steps taken in this chapter were performed in the order written.

Outline

The following outline was used for data pre-processing:

- Missing value treatment – Interpolation on each input variable separately using other values of the same variable that lie close to the missing value. This prepares data for use with the SVM which cannot handle missing values.
- Normalisation – Normalisation prevents the SVM from adding more importance to certain input variables due to a difference in scale of their values compared to those of other variables.
- Time windowing – Adaptation of time series data for use with non-time series compatible learning machines such as the SVM.
- Variable importance analysis – Done by comparing the results of a set of distinctly different general algorithms designed for assessment of variable importance. This has implications for how that variable might affect performance when used with the SVM.

Missing value treatment

In the original data a few values were missing for some of the input variables as presented in table 2:

<table>
<thead>
<tr>
<th>Input variable</th>
<th>Number of missing values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLSS-2</td>
<td>8</td>
</tr>
<tr>
<td>Q</td>
<td>1</td>
</tr>
<tr>
<td>V30-2</td>
<td>14</td>
</tr>
<tr>
<td>RASS-2</td>
<td>0</td>
</tr>
<tr>
<td>WAS-2</td>
<td>2</td>
</tr>
<tr>
<td>WAS-3</td>
<td>0</td>
</tr>
<tr>
<td>MES-S</td>
<td>3</td>
</tr>
</tbody>
</table>

The SVM is not adapted to handle missing values which, among other reasons, is why this problem must be handled during data pre-processing. This is a hard problem, with many possible solutions and algorithms, where intuition plays a big role. There is no right answer other than that the chosen method should optimise the performance of the SVM, i.e.
prediction quality. SVM performance, however, depends on a number of factors. This is why it is not practical to use the SVM as a performance measurement for a missing value treatment method. To avoid spending too much time on a highly speculative process, things were kept simple and two algorithms were tested:

- Nearest neighbour weighted interpolation
- Linear interpolation

**Nearest neighbour weighted interpolation**
Consider a time series of \( n \) values \((x_i, y_i), i = 1 \ldots n\). \( x_i \) holds the time point in days when the measurement was taken and \( y_i \) the measurement value. To interpolate the missing value \( y_m \) on day \( x_m \) (where \( m \) stands for missing value) one can use the equation in formula 6:

\[
y_m = \sum_{i=1}^{n} \frac{y_i}{\left|x_m - x_i\right|^p} \frac{1}{\sum_{i=1}^{n} \left|x_m - x_i\right|^p}
\]  

(6)

Note how the denominator in the top sum in formula 6 gives a measure of how close the point being estimated is to the sample points. Naturally if a sample is close then it has a greater influence on the estimate than if the sample is distant. \( p \) is a parameter to control the relative importance of distant samples. As \( p \) increases closer samples will have more impact on \( y_m \). See [2] for more information on nearest neighbour interpolation.

**Linear interpolation**
Consider the time series described in the section above. To interpolate a missing value at day \( x_m \) simply find the closest point to the left of \( x_m \), i.e. \( \min(x_m - x_i) : x_i \leq x_m, i = 1 \ldots n - 1 \) and calculate the value of \( y_m \) as shown in figure 8:

\[
(y_{m+1}, y_{m+1})
\]

\[
(x_m, y_m + (x_m - x_i) \frac{y_{i+1} - y_i}{x_{i+1} - x_i})
\]

\[
(x_i, y_i)
\]

*Figure 8 – Example of linear interpolation*
Results
Figure 9 and figure 10 illustrate the effects of both algorithms by two plots of V30-2. This is the variable with the most missing values and is a good example. Shown is a part of the curve containing a large set of missing values.

In the plots, a square represents a measured value and the dotted line connecting the squares is the linear interpolation curve. A circle represents an interpolated missing value using nearest neighbour and the solid curve that connects them is the nearest neighbour interpolation curve. Two plots are shown, one with P = 1 and the other with P = 10.

Figure 9 – Interpolation plot of variable V30-2 with P = 1
Conclusions

None of the nearest neighbour curves seemed intuitive. In figure 9 (P=1) the algorithm seems to always pull the interpolated value towards the mean and in figure 10 (P=10) seems to set the interpolated value equal to the nearest measurement. Using other values of P in the range 1 < P < 10 produced somewhat better curves.

There is a missing value at day one in the plot (the circle to the far left). This value is very hard to guess at since it is not enclosed by measurements to use for interpolation.

The linear algorithm was selected because it seemed to be the least risky alternative. It is in any case highly speculative which interpolation method will give the best results. The missing value on the far left was simply set to the measured one next to it.

Normalisation

Un-normalised data might contain a significant difference in variance and mean when comparing values of data vectors belonging to different input variables. If this difference is large some variables might influence the fitting of the hyper-plane more than others. There is no reason to assume that these variables are more important than others because of this difference. Therefore the data is normalised to move the mean of all variables to zero and rescale their variance.
The input variables were normalised one by one individually using the equation in formula 7 on all values $x \in \overline{X}$ where $\overline{X}$ is the training data vector of values for one variable. $x_N$ is the normalised output value. Please refer to table 3 for an explanatory example.

$$x_N = \frac{x - \text{mean}(\overline{X})}{\text{std}(\overline{X})}$$  \hspace{1cm} (7)

### Table 3 – Normalisation example of two variables X and Y.

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Mean(X)</th>
<th>Std(X)</th>
<th>Norm(X)</th>
<th>Norm(Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,00</td>
<td>100</td>
<td></td>
<td></td>
<td>−1,34</td>
<td>−1,34</td>
</tr>
<tr>
<td>2,00</td>
<td>200</td>
<td>3,5</td>
<td>1,87</td>
<td>−0,80</td>
<td>−0,80</td>
</tr>
<tr>
<td>3,00</td>
<td>300</td>
<td></td>
<td></td>
<td>−0,27</td>
<td>−0,27</td>
</tr>
<tr>
<td>4,00</td>
<td>400</td>
<td></td>
<td></td>
<td>0,27</td>
<td>0,27</td>
</tr>
<tr>
<td>5,00</td>
<td>500</td>
<td>350</td>
<td>187</td>
<td>0,80</td>
<td>0,80</td>
</tr>
<tr>
<td>6,00</td>
<td>600</td>
<td></td>
<td></td>
<td>1,34</td>
<td>1,34</td>
</tr>
</tbody>
</table>

---

**Time windowing**

The type of data that was dealt with in this project was time series data. There are many algorithms to deal with this kind of data but classification SVMs are not among them. These SVMs are only designed to learn a non-linear function from an N-dimensional input space to a class label, in this case one of two labels.

Since the SVM is not adapted for time series data a conversion is necessary. The conversion process is called Time Windowing. This essentially means that a lag is given to all the variables so that the input to the machine consists of the variables at the present, one sample ago and so on.

### Table 4 – Example of a four samples wide time window of one variable X

<table>
<thead>
<tr>
<th>X(t–3)</th>
<th>X(t–2)</th>
<th>X(t–1)</th>
<th>X(t–0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,00</td>
<td>1,00</td>
<td>2,00</td>
<td>3,00</td>
</tr>
<tr>
<td>2,00</td>
<td>2,00</td>
<td>3,00</td>
<td>4,00</td>
</tr>
<tr>
<td>3,00</td>
<td>3,00</td>
<td>4,00</td>
<td>5,00</td>
</tr>
<tr>
<td>4,00</td>
<td>4,00</td>
<td>5,00</td>
<td>6,00</td>
</tr>
<tr>
<td>5,00</td>
<td>5,00</td>
<td>6,00</td>
<td></td>
</tr>
<tr>
<td>6,00</td>
<td>6,00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Consider the example in table 4 where a time window of four samples is used on an input variable. This variable then expands into four input
variables. In the resulting data matrix only three rows of data are left out of the original six. This is because each row needs four values. The first row thus needs to aggregate four rows of the original data vector into one, using one value for each lag. The rest of the rows are created by simply shifting all values one cell to the left, discarding the left most value and importing a new value for $X(t-0)$, the right most value, from the original data vector of $X$.

The specific lag to use was decided by the plant engineers, which, hopefully, would catch the inherent feedback loops of the system. This report will not go into further details regarding their decision since it is a civil engineering issue and not within the scope of computer science.

The engineers chose a time window that was four samples wide. This means that each of the seven input variables would be input at four different lags from zero to three samples. Please refer to table 5 for the new set of 28 inputs (7x4):

<table>
<thead>
<tr>
<th>Variable</th>
<th>Q(t)</th>
<th>Q(t-1)</th>
<th>Q(t-2)</th>
<th>Q(t-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLSS-2(t)</td>
<td>MLSS-2(t-1)</td>
<td>MLSS-2(t-2)</td>
<td>MLSS-2(t-3)</td>
<td></td>
</tr>
<tr>
<td>V30-2(t)</td>
<td>V30-2(t-1)</td>
<td>V30-2(t-2)</td>
<td>V30-2(t-3)</td>
<td></td>
</tr>
<tr>
<td>RASS-2(t)</td>
<td>RASS-2(t-1)</td>
<td>RASS-2(t-2)</td>
<td>RASS-2(t-3)</td>
<td></td>
</tr>
<tr>
<td>WAS-2(t)</td>
<td>WAS-2(t-1)</td>
<td>WAS-2(t-2)</td>
<td>WAS-2(t-3)</td>
<td></td>
</tr>
<tr>
<td>WAS-3(t)</td>
<td>WAS-3(t-1)</td>
<td>WAS-3(t-2)</td>
<td>WAS-3(t-3)</td>
<td></td>
</tr>
<tr>
<td>MES-S(t)</td>
<td>MES-S(t-1)</td>
<td>MES-S(t-2)</td>
<td>MES-S(t-3)</td>
<td></td>
</tr>
</tbody>
</table>

**Variable importance analysis**

To further understand the data it is necessary to explore it before actually using the SVM. Perhaps all of the variables at all of the different lags are not important. Sometimes certain lags for certain input variables are highly important and others have no bearing at all. It is also possible that entire input variables are unimportant to successful prediction.

There are a number of general algorithms to test and understand the importance of input variables at different lags. The word general in this context implies that they do not use specific information about the problem at hand or share the same performance measurement as the SVM. Therefore it is possible that the assumptions they make regarding the information content and relevancy of the data might be wrong or not entirely true.

For this reason, two different algorithms were used which employ distinctly different assumptions and measurements of information content and relevancy. This was to see if a pattern could be discerned over multiple algorithms as opposed to risk basing a decision on the possible idiosyncrasies of a single algorithm.
Principal Component Analysis (PCA)

PCA is a well known algorithm that assumes that high variance equals high information content and tries to find a new orthogonal base for the data. This new base is created by defining the first component in the direction where the data exhibits the most variance, the next one in the same manner with the constraint that it has to be orthogonal to the first and so on for the following components. When the dimensionality of the output equals that of the input the process is complete. Figure 11 illustrates the result of this process on a two dimensional input space (for more information on PCA, see [3]):

![Figure 11 – PCA analysis example on two variables X and Y](image)

Results

PCA results consist of a set of factors for each input variable. The factors are used to linearly combine the input variables for creation of each component. The farther a factor is from zero, the larger is its significance which can be interpreted as the variable contributing with relevant information for a specific component. If a variable contributes with relevant information for many components the assumption is that this variable in general contributes with relevant information describing the data.

PCA also calculates how many percent of the variance that is captured by each component. This value is referred to as the information content percentage below.
The factors and the percentage combined determine each input variables importance. A unified expression named PCA score was defined based on this fact as shown in formula 8:

\[ S_i = \text{the PCA score for input variable } i \]
\[ n_c = \text{number of components} \]
\[ P_j = \text{information content percentage for component } j \]
\[ f_{ij} = \text{factor for variable } i \text{ and component } j \]

\[ S_i = \sum_{j=1}^{n_c} |P_j f_{ij}| \]  

(8)

\( S_i \) is the sum of all factors belonging to variable \( i \) weighted by each component’s information content percentage. The absolute value of the factors is used in this sum, since the distance from zero determines factor importance as opposed to the sign.

Table 6 presents the factors produced by PCA-analysis multiplied by a constant to present smaller numbers and conserve space. Only the relationship between the factors is interesting so this does not affect the results. Only a subset of the most important components is shown, also to conserve space.

Figure 12 shows the PCA scores for all variables except MES-S, which was removed from the figure because it was obviously important, judged by its factors in table 6, and would adversely affect the scale of the chart.
Table 6 – Factors produced by PCA-analysis of input variables

<table>
<thead>
<tr>
<th>Component</th>
<th>INFO(%)</th>
<th>24</th>
<th>17</th>
<th>13</th>
<th>10</th>
<th>8</th>
<th>5</th>
<th>5</th>
<th>3</th>
<th>2</th>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLSS-2(t)</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>–3</td>
</tr>
<tr>
<td>MLSS-2(t-1)</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>–2</td>
</tr>
<tr>
<td>MLSS-2(t-2)</td>
<td>0</td>
<td>2</td>
<td>1</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>2</td>
</tr>
<tr>
<td>MLSS-2(t-3)</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>–2</td>
</tr>
<tr>
<td>Q(t)</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Q(t-1)</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>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Q(t-2)</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>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Q(t-3)</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>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>V30-2(t)</td>
<td>0</td>
<td>4</td>
<td>–7</td>
<td>–1</td>
<td>1</td>
<td>10</td>
<td>–2</td>
<td>–2</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>–16</td>
</tr>
<tr>
<td>V30-2(t-1)</td>
<td>0</td>
<td>5</td>
<td>–7</td>
<td>–1</td>
<td>2</td>
<td>9</td>
<td>–3</td>
<td>–2</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>V30-2(t-2)</td>
<td>0</td>
<td>5</td>
<td>–7</td>
<td>–1</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>–1</td>
<td>–1</td>
<td>–5</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>V30-2(t-3)</td>
<td>1</td>
<td>5</td>
<td>–7</td>
<td>–1</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>–11</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>RASS-2(t)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RASS-2(t-1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RASS-2(t-2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RASS-2(t-3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WAS-2(t)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>–2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>–12</td>
</tr>
<tr>
<td>WAS-2(t-1)</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>–3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>–3</td>
</tr>
<tr>
<td>WAS-2(t-2)</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>–3</td>
<td>3</td>
<td>0</td>
<td>–1</td>
<td>0</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>WAS-2(t-3)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>–4</td>
<td>3</td>
<td>0</td>
<td>–3</td>
<td>–2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>WAS-3(t)</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>–1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>WAS-3(t-1)</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>–1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>WAS-3(t-2)</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WAS-3(t-3)</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>–6</td>
<td></td>
</tr>
<tr>
<td>MES-S(t)</td>
<td>17</td>
<td>13</td>
<td>26</td>
<td>–99</td>
<td>40</td>
<td>18</td>
<td>27</td>
<td>196</td>
<td>–284</td>
<td>16</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>MES-S(t-1)</td>
<td>17</td>
<td>13</td>
<td>29</td>
<td>–115</td>
<td>56</td>
<td>9</td>
<td>60</td>
<td>114</td>
<td>–95</td>
<td>55</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>MES-S(t-2)</td>
<td>17</td>
<td>11</td>
<td>27</td>
<td>–115</td>
<td>58</td>
<td>–6</td>
<td>93</td>
<td>–51</td>
<td>133</td>
<td>43</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>MES-S(t-3)</td>
<td>16</td>
<td>8</td>
<td>21</td>
<td>–100</td>
<td>45</td>
<td>–16</td>
<td>99</td>
<td>–171</td>
<td>253</td>
<td>–123</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Figure 12 – PCA scores for all input variables except MES-S
Analysis
It is easy to see in table 6 that the MES-S variable has significantly higher factors than the rest of the variables for all lags. This is in accordance with intuition, since this variable is the one we are trying to reasonably predict at day t+1. It is also interesting to see that V30-2 seems to exhibit the highest PCA score besides MES-S for all lags.

Generally, all the variables end up showing off at the end, in greater or lesser degree. However, it is easy to see that variables Q and RASS-2 boast a lot of zeroes for the most important components making their PCA score low in comparison with the other input variables.

It is interesting to see how the PCA scores for each input variable are almost identical for all of the variable’s lags. This is not surprising, since almost the same values exist in each of the lagged data vectors for the variable and therefore their variances will be almost identical. However, this fact is probably inconsistent with reality, since the lag of an input variable might be highly significant to SVM performance due to causes in the physical process of the WWTP.

Conclusions
Two variables stand out and it was concluded that Q and RASS-2 could perhaps be discarded to the benefit of SVM performance.

RELIEF algorithm
RELIEF calculates a weight W[A] for each attribute A and a good attribute will result in a high value for W[A]. The rationale is that the value of a good attribute should differ between points of different classes and be approximately the same for points of the same class.

RELIEF randomly selects m points from the data set. For each point R, it searches for its two nearest neighbours, i.e. one from the same class, called nearest hit, and the other from a different class called nearest miss. Figure 13 presents the pseudo code for how these are used in the calculation of W[A]:

Set all weights W[A] := 0
For i := 1 to m do
Begin
Randomly select an point R
Find nearest hit H and nearest miss M
For A := 1 to all attributes do
+ diff(A,R,M)/m
End

Figure 13 – Pseudo code for RELIEF algorithm
Diff(A,R,H) calculates the difference between the values of attribute A for two points R and H. Diff is also used for calculating the distance between points to find the nearest neighbours. The total distance is simply the
Euclidian distance using all attributes. Since parameter m is the number of points selected from the data, the larger m implies more reliable approximation.

**Converting weights into a subset of important variables**
Since RELIEF gives as output an ordered list of variables $x_i$ according to their weight $w_i$, a filtering criterion is necessary to transform this solution to a subset of variables deemed important. The procedure used here is simple: since the interest is in determining a good cut point, first the weights are sorted; then those $w_i$ further than two variances from the mean are discarded, i.e. very high or very low weights. Then the definitions in formula 9 are used:

\[
s_i = w_i + w_{i-1}, \quad \sigma_j = \sum_{i=2}^{l} s_i \quad \text{and} \quad \delta_j = \frac{\sigma_j - n - j}{\sigma_n} n \quad (9)
\]

The objective is to search for the feature $x_j$ that maximises $\delta_j$. The cut point is then set between $x_j$ and $x_{j+1}$. The reason is simple: on one hand, it is beneficial to accumulate as much relevancy as possible relative to the total, i.e. maximise $\sigma_j / \sigma_n$; on the other hand, it is preferable to select a low number of variables (only the most important ones), i.e. maximise $n - j / n$.

Since RELIEF has a random element, the algorithm is run multiple times to make sure that the output is somewhat stable. For more information on RELIEF see [4,5]. For more information on the filtering criterion see [11].

**Results**
Table 7 shows the output from running the RELIEF algorithm ten times. Each line represents the result of one execution of the algorithm. The lags for each variable are presented in the order t–0 → t–3 from left to right. Each lag and variable are marked with a “1” or a “0”. “1” means important and “0” unimportant according to the filtering criterion:
Table 7 – Results of the RELIEF-algorithm

<table>
<thead>
<tr>
<th>Variables</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLSS-2</td>
<td>All lines support this</td>
</tr>
<tr>
<td>V30-2</td>
<td>All lines support this</td>
</tr>
<tr>
<td>RASS-2</td>
<td>All lines support this</td>
</tr>
<tr>
<td>WAS-2</td>
<td>All lines support this</td>
</tr>
<tr>
<td>WAS-3</td>
<td>Majority support this</td>
</tr>
<tr>
<td>MES-S</td>
<td>Majority support this</td>
</tr>
</tbody>
</table>

Table 8 – Most important variables according to RELIEF

<table>
<thead>
<tr>
<th>Variables</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLSS-2</td>
<td>All lines support this</td>
</tr>
<tr>
<td>MES-S</td>
<td>All lines support this</td>
</tr>
<tr>
<td>V30-2</td>
<td>All lines support this</td>
</tr>
<tr>
<td>V30-2</td>
<td>Majority support this</td>
</tr>
<tr>
<td>WAS-2</td>
<td>Majority support this</td>
</tr>
<tr>
<td>WAS-2</td>
<td>All lines support this</td>
</tr>
</tbody>
</table>

Table 9 – Least important variables according to RELIEF

<table>
<thead>
<tr>
<th>Variables</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>All lines support this</td>
</tr>
<tr>
<td>RASS-2</td>
<td>All lines support this</td>
</tr>
<tr>
<td>WAS-3</td>
<td>All lines support this</td>
</tr>
<tr>
<td>MES-S</td>
<td>All lines support this</td>
</tr>
</tbody>
</table>

Table 10 – Variables of indeterminable importance according to RELIEF

<table>
<thead>
<tr>
<th>Variables</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAS-2</td>
<td>Considered important in almost half of the executions</td>
</tr>
</tbody>
</table>

Conclusions

Observations made from the RELIEF results are summarised in table 8, table 9 and table 10:
Comparison between Relief and PCA results

By comparing the results of both PCA and RELIEF it is possible to derive the following combined results as summarised in table 11, table 12 and table 13:

Table 11 – Important variables according to both PCA and RELIEF

<table>
<thead>
<tr>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLSS-2 [t–0,t–1,t–2,t–3]</td>
</tr>
<tr>
<td>MES-S [t–0,t–1]</td>
</tr>
<tr>
<td>WAS-2 [t–0,t–1,t–2, t–3]</td>
</tr>
<tr>
<td>V30-2 [t–0,t–1,t–2,t–3]</td>
</tr>
</tbody>
</table>

Table 12 – Unimportant variables according to both PCA and RELIEF

<table>
<thead>
<tr>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q [t–0,t–1,t–2,t–3]</td>
</tr>
<tr>
<td>RASS-2 [t–0,t–1,t–2,t–3]</td>
</tr>
</tbody>
</table>

Table 13 – Important variables according to PCA but not RELIEF

<table>
<thead>
<tr>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAS-3 [t–0,t–1,t–2,t–3]</td>
</tr>
<tr>
<td>MES-S [t–2,t–3]</td>
</tr>
</tbody>
</table>

Conclusions

Variables Q and RASS-2 were the only variables that were deemed to possibly be unimportant by both algorithms which loosely suggests that it might be beneficial to remove these as inputs to the SVM during training.

It must be remembered that the algorithms used are general and do not consider domain specific information. To avoid loss of important information, the decision was taken to be conservative and experiment using the SVM with several input variable selections including the exclusion of Q and RASS-2.
SVM MODEL SELECTION

Definition
In this thesis, the reference to a model is always meant to be an SVM based classification model. Model selection thus refers to the selection of the final parameterised and trained SVM-model, i.e.
1) determining
   - the optimal parameters for the SVM
   - the best set of input variable to train the SVM with
2) training the SVM on the training data set
Model selection will thus result in a performance optimised SVM model that is completely ready to classify new data. In the context of this thesis, SVM-performance is equivalent to prediction quality.

Outline
There is no simple way of selecting a model even though this project was limited to the sole use of classification SVMs. These machines are very general and there are some important parameters that influence their performance. These parameters are:
- C
- Kernel type
- K-scale
C defines how much the margin and hyper plane constraints are allowed to be violated, as described in the "Soft margin SVM"-section above.
In the package used for our work developed by the Computer Science department at the university where this work was conducted (UPC, Barcelona), the following kernel types were available:
- Second degree polynomial
- Third degree polynomial
- Fourth degree polynomial
- Gaussian
K-scale is a smoothing parameter specific to the SVM software package used in the project. It specifies the variance of the kernel function.

Performance measurement by cross validation
An inherent necessity in model selection is a well defined measurement of performance which, in the context of this thesis, is equivalent to prediction quality. The one used in this project is leave-one-out cross validation
where, for a training data set of N points, the SVM is trained and run N times on N–1 points. Each time, before training, a point is removed from the training set. The same point is then classified by the trained SVM and the classification is compared to the correct class of the point. During every execution, a different point is selected so that the SVM is tested on all N points.

The number of correctly classified points divided by N is used to compare different models to find the best one with the highest rate of correct classifications.

**Data partitioning into training and test data sets**

The input data was partitioned into two sets:

- The training data set which was to be used for model selection.
- The test data set which was to be used for performance analysis of the final model, i.e. the single best model found during model selection.

Out of the 587 consecutive measurements in the data, the first 312 measurements chronologically were chosen to constitute the training data set. This was done before time windowing. The remaining 275 measurements were chosen to constitute the test data set. The reason for this particular partitioning was simply that the last 275 measurements did not become available until the near-end of the project.

It is worth noting that neither the test nor the training data contains one full year’s worth of data which has implications for dealing with seasonal properties of the same.
PARAMETER CORRELATION WITH SVM PERFORMANCE

Goal

The goal of this stage was to see how the C and K-scale parameters affected performance.

Outline

Prerequisites

Several executions were performed on the SVM over a two-dimensional input grid of K-scale and C-values. This was done in combination with the four available kernels on a single input variable selection consisting of all input variables.

There were no theoretical results to suggest the shape of the three-dimensional height field that the performance curve would consist of or which range of parameter values to use. It was not known whether there were several local error minima or one global minimum. This first stage would hopefully give us some kind of parameter boundaries for a future more detailed search over a smaller area and a hint of the curve’s shape.

Parameter start value and variation heuristic

The K-scale value is dependent on the kernel chosen. For the polynomial kernels the influence of K-scale is very strong. It somewhat normalises the scalar product used in the decision function of the SVM’s linear machine. A value proportional to 1/N where N is the number of variables was deemed to be a good starting value. This was based on the empirical experience of my supervisor Lluis Belanche. In our case, using all inputs N=28 makes 1/N = 0.0357. The starting point for the Gaussian K-scale value was chosen ad hoc.

Departing from the start value for K-scale, the variation heuristic chosen would travel in both directions by halving and doubling it. This is also a heuristic that comes from practical experience. It has been found that small changes do not infer a large effect on performance, so this binary search is a quicker way to find a minimum.

For the C-value, a step size of orders of magnitude was used to make an even larger difference. There was no practical heuristic to help in choosing the C-value so a large range was decided upon. Values were sampled at equal logarithmical distance within this range. The number of sampled values, the granularity, was doubled with each consecutive iteration to see if this would improve performance.

The maximum granularity that was used is the one displayed in the figures in the “Results”-section below. It is theoretically possible to use a higher granularity but the calculations were time consuming and needed to be
completed within reasonable time. Each point in the figures represents hundreds of SVM training runs to calculate the number of cross-validation errors. Due to the same reason the number of different K-scale values used was limited even more than for the C-parameter. The C-parameter is more important since it directly affects the fitting of the hyper plane in the SVM.

**Actual parameter values**

C-values were sampled at equal logarithmical distance in the range \((10^{-3}, 10^4)\). Three consecutive iterations over the same range were made with 7, 14 and 29 sampled C-values.

These K-scale values were used with the polynomial kernels:

\((1/28)/4, \ (1/28)/2, \ 1/28 \text{ and } (1/28)*2\)

These K-scale values were used with the Gaussian kernel:

\((1/4)/4, \ (1/4)/2, \ 1/4 \text{ and } (1/4)*2\)

**Results**

The following figures summarise the results by showing the number of cross-validation errors for every combination of C, K-scale and kernel type:

![Graph showing cross-validation errors for different C-values and K-scales.](image)

*Figure 14 – Second degree polynomial kernel on all input variables*
Figure 15 – Third degree polynomial kernel on all input variables

Figure 16 – Fourth degree polynomial kernel on all input variables
Analysis

In the left and right parts of the plot the curves seem to level out. This is true for all of the four kernel types and K-scale values. The leftmost levelled part of the square and triangle curves, in figure 15 and figure 16, is not within the visible region. However, it can be clearly seen that these curves as a whole are shifted sideways to the left. The trend seems to indicate that these curves would level out as well.

There also seems to be some kind of global minimum for each kernel type that seems to shift sideways depending on the K-scale value. Without a more precise granularity, which was not available due to unreasonable computation time, it is not possible to be sure but it was deemed to be a reasonable assumption.

The minima in the plot occur at different C-values depending on K-scale but the number of cross-validation errors does not change significantly. The C-value seems to be of much greater importance to performance than K-scale as previously suspected. This is true for all of the four kernels.
The granularity was considered to be sufficient at this stage. This decision was based on two factors:

- The computation time would become too great at higher granularities.
- Only slight changes in performance occurred for any of the kernel types or different K-scale values during the last iteration while doubling the granularity of the C-value. The increase in granularity also reinforced the notion that the curves seemed to gently slope down towards the minimum as can be seen in the figures above, i.e. no large spikes or dips appeared.

**Conclusions**

The most important observations made at this stage can be summarised as follows:

- There seems to exist a global error minimum within the C-value range of \((10^{-3}, 10^4)\).
- The global minimum seems to be mostly determined by the C-parameter.
- The granularity of 29 C-values for the specified range seems to allow a reasonably accurate approximation of the global minimum.
- The K-scale parameter seems to shift the global minimum sideways but only slightly impact performance even during large exponential variations.
INPUT VARIABLE AND KERNEL TYPE CORRELATION WITH SVM PERFORMANCE

Goal

The goal of this stage was to determine the best input variable and kernel type selection.

Outline

This chapter outlines the significant computational task of running the SVM using:

- the same ranges and granularities for the C- and K-scale parameters as in the previous chapter
- all four kernel types
- multiple input variable combinations (the previous chapter amounted to only a single combination)

It was not possible to try all different combinations of input variables due to limited computation time. Therefore the selection was limited to all combinations involving the exclusion of any single input variable. This was to allow seeing how the exclusion of each input variable affected performance in comparison with the case where all input variables were used. Based on the analysis performed in the “Variable importance analysis”-section, an exclusion combination of both Q and RASS-2 was also scheduled at the very beginning. They were the only two variables deemed unimportant by both of the general algorithms.

The kernel represents the best possible separation of the data and the input combination represents the relevance of the data presented to the SVM. It is important to avoid choosing a variable exclusion case based on performance improvement caused by fitting to noise by an overly adaptive kernel function. A candidate exclusion case was therefore expected to improve performance for all kernels to be considered suitable.

This approach of using the same algorithm for input variable selection as for performance is called a wrapper strategy. It is more computationally demanding but usually yields better results than using general algorithms.
Results

The results are summarised in table 14, figure 18 and figure 19. The performance for each combination of kernel type and input variable exclusion is represented by the minimum error encountered for all SVM executions on that combination.

Table 14 – Cross-validation errors for all kernels and input variable exclusions

<table>
<thead>
<tr>
<th></th>
<th>2nd poly</th>
<th>3rd poly</th>
<th>4th poly</th>
<th>Gaussian</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>All input variables</td>
<td>47 50 49 45</td>
<td>47,8 45 50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>51 50 50 48</td>
<td>49,8 48 51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASS-2</td>
<td>47 49 51 46</td>
<td>48,3 46 51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MES-S</td>
<td>71 70 66 60</td>
<td>66,8 60 71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAS-3</td>
<td>48 49 49 45</td>
<td>47,8 45 49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAS-2</td>
<td>45 49 49 44</td>
<td>46,8 44 49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLSS-2</td>
<td>45 49 51 43</td>
<td>47,0 43 51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V30-2</td>
<td>45 46 47 44</td>
<td>45,5 44 47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q and RASS-2</td>
<td>49 50 50 47</td>
<td>49,0 47 50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V30-2 and WAS-2</td>
<td>42 42 46 43</td>
<td>43,3 42 46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean (excl. MES-S)</td>
<td>46,6 48,2 49,1 45,0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min (excl. MES-S)</td>
<td>42 42 46 43</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max (excl. MES-S)</td>
<td>51 50 51 48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 18 – Results visualised per input variable exclusion for all kernels.
General analysis

Table 14 presents a few interesting facts:

- The minimum error count is 42 or 14% of 308 rows of data.
- The maximum error count is 71 or 23%.
- The difference between minimum and maximum is 29 errors or 9%.
  The span between worst and best solution is thus relatively large.

The exclusion of MES-S seems to degrade performance considerably. This is evident in both table 14 and figure 18. The MES-S exclusion adds another:

- 18–29 errors compared to the minimum error count of 42.
- 15–24 errors compared to when all input variables are used.

This is consistent with the analysis performed in the “Variable importance analysis”-section. Analysis by both PCA and the RELIEF-algorithm concluded MES-S to be important.

Plots of performance correlation with C and K-scale parameters for each of the 40 combinations of kernel type and input variable exclusion revealed a U- or V-shape as was found previously (figure 14 – figure 17). There was one exception, however, which corresponds to the exclusion of MES-S where the curve took on a different form. This is exemplified in figure 20. There a second degree polynomial kernel was used but this phenomenon occurred for all kernel types. Due to time constraints and because performance was degraded for this case, it was not examined in greater detail.
Figure 20 – Example of an irregular performance curve with MES-S excluded.

The maximum difference between the best solution of 42 errors and all other solutions is only 9 errors or 3% of the data when the MES-S exclusion case is disregarded. In the next sections where performance correlation with input variable exclusion and kernel type is analysed, this fact should be remembered. All results derived there will be based on this small variation in performance.

Conclusions

Only MES-S has had a huge impact on performance. In all other cases, performance varies only slightly but noticeably which has implications for the importance and relevance of the conclusions drawn in further performance analysis.
Input variable correlation with performance

All inputs case versus variable exclusion cases

The following variable ranking in table 15 is sorted from best to worst variable exclusion based on the error mean over kernels:

Table 15 – Performance ranking of input variable exclusions

<table>
<thead>
<tr>
<th>Variable exclusion</th>
<th>Error mean over kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>V30-2 and WAS-2</td>
<td>–4,5</td>
</tr>
<tr>
<td>V30-2</td>
<td>–2,3</td>
</tr>
<tr>
<td>WAS-2</td>
<td>–1,0</td>
</tr>
<tr>
<td>MLSS-2</td>
<td>–0,8</td>
</tr>
<tr>
<td>WAS-3</td>
<td>0,0</td>
</tr>
<tr>
<td>RASS-2</td>
<td>0,5</td>
</tr>
<tr>
<td>Q and RASS-2</td>
<td>1,3</td>
</tr>
<tr>
<td>Q</td>
<td>2,0</td>
</tr>
<tr>
<td>MES-S</td>
<td>19,0</td>
</tr>
</tbody>
</table>

The table calculates the difference between
- the exclusion cases (row 3–11, column 7 of table 14) and
- the all input variables case (row 2, column 7 of table 14)

Focusing primarily on single variable exclusions, only V30-2 and WAS-2 had unchanged or improved performance for all kernels as can be seen in table 14 and figure 18. All other exclusion cases degraded performance for one or more kernels. They also positioned themselves at the top of the mean based ranking in table 15. It is worth noting that V30-2 had fourteen missing values, which is 4,5% of the data.

Therefore the case of excluding both V30-2 and WAS-2 was tried with the encouraging result of being at the top of the ranking. Their combined exclusion improved performance marginally but noticeably. It is also possible to see in figure 19 that this exclusion represents the best solution for every kernel type.

Discrepancies with PCA and RELIEF

There are some discrepancies between the results of the analysis performed by PCA and RELIEF in the “Variable importance analysis”-section and those produced here. Two cases completely contradict the conclusions that were drawn based on both PCA and RELIEF:

- Both WAS-2 and V30-2 were previously considered bad candidates for exclusion.
- Q and RASS-2 were previously considered good candidates for exclusion.
Generally, this is probably due to the fact that PCA and RELIEF are general purpose algorithms and measure performance differently and without specific knowledge of the prediction problem.

PCA measures performance based on high variance and not on accurate prediction based on cross validation. A variable could be irrelevant meaning that it is an indecisive factor for predicting MES-S in the physical process of the WWTP. An irrelevant variable with high variance would only provide a lot of unnecessary information and could even degrade performance by confusing the SVM.

This is one interpretation of the fact that excluding WAS-2 and V30-2 improves performance instead of degrading it as expected since their variance is high. Q and RASS-2 might therefore be more relevant to the problem since their inclusion improves performance even though their variance is low according to the same principle.

**Conclusions**

By a small margin, the best exclusion seems to be V30-2 and WAS-2 by comparison with the all input variables case considering:

- means over kernels
- the best solution for each kernel
- that only the exclusion of V30-2 and WAS-2 together or separately improve performance for all four kernels

**Kernel type correlation with performance**

Figure 19 gives an overview of how performance is affected by the kernel type by displaying all results divided into groups by kernel type.

Table 14, row 13, column 2–5, show the means over the input variable selections without the irregular and performance degrading MES-S exclusion case. These values show that the mean for the Gaussian kernel gives better performance than the rest. This is illustrated in the following ranking in table 16 where the mean of the Gaussian kernel type has been subtracted from the other kernel types:

<table>
<thead>
<tr>
<th>Kernel type</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fourth degree polynomial</td>
<td>−4,3</td>
</tr>
<tr>
<td>Third degree polynomial</td>
<td>−3,3</td>
</tr>
<tr>
<td>Second degree polynomial</td>
<td>−1,7</td>
</tr>
</tbody>
</table>

Figure 18 shows that the Gaussian kernel type also has the best solution for all input variable selections except for V30-2 and WAS-2, but the difference to the next best solution is only 1 error. The general trend seems to be that the Gaussian improves performance.
It is worth noting that the Gaussian kernel improves performance considerably compared to the other kernels for MES-S as illustrated in figure 18. One interpretation of this is that the Gaussian is more adaptable and could be “improving” the solution by adapting to noise. This could possibly lead to poor performance on the test data set.

Conclusions
The general trend is that the Gaussian kernel gives the best performance by:

- comparison between means over the input variable selections
- being the best kernel solution for almost all variable selections

Final conclusions
The most important observations gathered in this chapter are:

- Inclusion of MES-S seems critical to solving the problem.
- Exclusion of V30-2 and WAS-2 seems to improve performance slightly.
- The Gaussian kernel seems to give the best performance in general.

It must be kept in mind that although it is probable that the granularity of the C and K-scale ranges are adequate, it is still very possible that the exact minimum was not found in the experiments but rather the best approximation possible within reasonable computation time.

This could also account for the relatively small variance in performance when disregarding the MES-S exclusion case. Therefore it is still hard to reach the conclusion that a slight increase in performance actually means that the combination of kernel type and input selection in question was the true cause of the performance improvement.
**SVM PARAMETER FINE TUNING**

**Goal**

The goal of this stage was to determine the best $C$ and K-scale parameter values for the Gaussian kernel while excluding V30-2 and WAS-2.

**Outline**

The kernel type and input variable selection constraints were defined to allow for concentration of computing power on a smaller problem. To find the best parameter values, a fine grain search was performed around the minima found previously, i.e. $C = (10^0, 10^1)$. Granularity was doubled over several iterations until computation time became unreasonable.

**Results**

Figure 21 gives us a picture of the starting point showing the experiments previously performed for the Gaussian kernel while excluding V30-2 and WAS-2. Figure 22 shows us the results of an increase in granularity.

![Figure 21 – Data from previous experiments in the area of the minima.](image-url)
Analysis

The error minimum is as in the previous chapter for this kernel and variable combination 43 errors. It is interesting that the star curve presents three new minimum points at 43 in comparison to its previous minimum of 45. Perhaps this would indicate a performance peak for this combination.

The forms of the curves alter only slightly even though there are 4 times as many sampling points in the region of the minima. Although it is possible to see a few new dips in each of the curves, none of the dips are more than 3 errors in depth. The curves continue to look relatively smooth. The general interpretation of these facts was that the previous granularity level describes the real curve close enough to find a reasonable approximation of the minima.

Conclusions

One of the best minima was chosen as the final SVM model. It was not important which one since there was no other way to rank them except by performance and they all had the same number of cross-validation errors, i.e. 43. The minimum chosen corresponds to the middle of the minima located on the triangle curve. The complete model chosen was thus:

$K$-scale: 0.025. $C$-value: 2,831. Kernel type: Gaussian.
Input variable exclusion: WAS-2 and V30-2.
PERFORMANCE ANALYSIS OF FINAL SVM MODEL

Goal
The goal of this stage was to determine the performance of the chosen model on the independent test data set using a more detailed performance measurement.

Outline
It was known for how many days the SVM performed a correct prediction but not for which days. It could have been predicting correctly only those days where the limit of MES-S was not exceeded, which is the vast majority of the data. It could also have been too sensitive, i.e. predicting that the limit would be exceeded when it would not be in too many cases.

Using the model chosen in the previous chapter, the input data in the test data set was classified and compared to the correct output. The idea was to measure performance more realistically, by using data the SVM had not been trained on and which had not been used for performance measurement in model selection. The test data consisted of an additional 271 days of measurements following directly after the training data on the time line.

A new performance measurement method
It was necessary to define a new more detailed performance measurement to know exactly what the SVM was predicting and how relevant those predictions were. It would take into account the different types of correctly and incorrectly classified data which was done by creating a so called “confusion matrix”.

Definition:
- Class A signifies that the limit for MES-S was not exceeded.
- Class B signifies that the limit for MES-S was exceeded.
The confusion matrix is thus defined as in table 17:

*Table 17 – Definition of confusion matrix*

<table>
<thead>
<tr>
<th>Reality</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A'</td>
</tr>
<tr>
<td>A</td>
<td>AA'</td>
</tr>
<tr>
<td>B</td>
<td>BA'</td>
</tr>
</tbody>
</table>

- AA' – Number of days where prediction and reality agrees upon class A, i.e. a correct classification.
- AB' – Number of days where SVM predicts B when in reality class A is true, i.e. an incorrect classification.
- BA' – Number of days where SVM predicts A when in reality class B is true, i.e. an incorrect classification.
- BB' – Number of days where prediction and reality agrees upon class B, i.e. a correct classification.

From the confusion matrix, it is possible to calculate the two following quantities for each class:

- Sensitivity (SE): Correctly predicted proportion of data belonging to that class.
- Specificity (SP): Correctly predicted proportion of data predicted to be of that class.

Mathematically they are defined as in formula 10:

\[
SE_A = P(A' | A) = \frac{AA'}{AB' + AA'} \\
SP_A = P(A | A') = \frac{AA'}{AA' + BA'}
\]

(10)
Results

Table 18 shows the result of executing the final model on the test data set by confusion matrix, sensitivity and specificity:

Table 18 – Performance of final model on test data set

<table>
<thead>
<tr>
<th>Prediction</th>
<th>A'</th>
<th>B'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reality</td>
<td>A</td>
<td>245</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>11</td>
</tr>
</tbody>
</table>

Class A | Class B
---|---
SE | 95,7% | SE | 73,3%
SP | 98,4% | SP | 50,0%

Analysis and conclusions

Class B is the most important class since it corresponds to an abnormal operating condition for the WWTP. The sensitivity specifies how big percentage of the occurrences of class B that we predicted correctly. At 73% this is an encouraging result.

The specificity, however, is only 50%. There are as many wrong classifications of B as there are correct ones, i.e. $AB' = BB' = 11$. This seems a bit high, but one has to consider that only 15 days of class B exist in the test data set, i.e. $BA' + BB'$. This means that the SVM is at least not just solving the trivial part of the problem. It has managed to get at least 50% credibility to its predictions even though only close to 6% of the data is of class B. This is very encouraging.

The combined interpretation of sensitivity and specificity for class B suggests that the SVM is not being conservative enough in its predictions. It manages to predict a large percentage of a small number of days correctly but the number of false alarms limits the usability of those predictions. If the hyper plane separation could be fine tuned to minimise the number of false alarms, i.e. $AB'$, specificity could perhaps be improved.

Regarding class A, 96% of all occurrences were correctly predicted. A prediction of class A gives a 98% probability of that class becoming a reality. One reason for this is that 249 days out of 271 or 92% of the data belongs to class A. If the SVM always predicted class A, then 92% of those predictions would be correct and it would correctly predict 100% of those occurrences, i.e. $SE = 100\%, SP = 92\%$. Therefore, those results are not nearly as interesting as those regarding class B. They represent the trivial part of the problem.
FINAL CONCLUSIONS AND RECOMMENDATIONS

The most serious factors in affecting performance negatively were:

- lacking important input variables that the engineers at the plant did not specify
- access to only a year’s worth of data
- missing values

These factors combined meant that we lacked a large amount of relevant information necessary to successfully solve the problem. At least two or three years of accumulated information would be necessary to have a chance at successful prediction. More careful selection of variables and delays would probably also improve performance. This needs close cooperation with the experts on the plant process which was not possible during this project.

The WWTP involves a time-varying process that is seasonal and intuitively one cannot expect a more-than-fair prediction when splitting the data in two time-ordered sets. The presence of an independent test set is mandatory in order to have a more reliable measure of real performance. It is even more important than the number of leave-one-out cross-validation errors and in fact complements it. However, a better way to divide the data than was done in this project could be to present the machine with data from the whole year, not a smaller consecutive segment of it, though with artificially induced missing values. These missing values would constitute the test set and the rest of the data the training set.

The pre-processing that was performed to interpolate missing values and normalise the data really affects performance and the effect should be more closely studied. It is also not possible to fully ascertain whether the SVM is doing really well or not in the sense that other methods may perform better, although there is evidence to suspect that no other method would have beaten the SVM by a significant margin.

The results are inconclusive and much remains to be done. With the time at hand the analysis performed is as complete as can be expected and future work will certainly improve upon the results presented in this thesis. The problem remains unsolved but the work provides an interesting case study and much useful knowledge to the benefit of other people willing to tackle a similar problem in a similar way. This is perhaps the most important result of this project.
ACKNOWLEDGEMENTS

I would like to thank my supervisor Lluis Belanche at the computer science department at UPC (Universitat Politecnica de Catalunya) for his guidance during this project.
REFERENCES


