The handle http://hdl.handle.net/1887/29691 holds various files of this Leiden University dissertation.

**Author:** Kieft, Nataša  
**Title:** Evaluation of different design space description methods for analysing combustion engine operation limits  
**Issue Date:** 2014-11-11
A. Calculation hardware and routine specifications

A.1. Computer specifications

All calculations have been performed on a laptop computer, provided by AMG, with the following specifications:
Laptop type: Lenovo Thinkpad T510, 4484Y11
Processor: Intel(R) Core(TM) i5 CPU M 540 @ 2.53GHz (dual core)
Memory (RAM): 4 GB Operating system: Windows 7, 64 Bit
Matlab version: Matlab 2011b

A.2. Calculations overview

The calculations have been performed in Matlab. The general layout of the script is given below, where each subroutine is presented in the following sections. The classes referred to are inside (+1) or outside (-1). The boundary as such is defined by the limit on the output.

A.2.1. Routines to generate training points

Create a sobolset.
Scramble the sobolset according to the Matousek-Affine-Owen type.
Select the training points according to the required number.
Scale the data to the original variable minima and maxima.

A.2.2. Routines to generate test points

Define the number of steps per axis.
Generate a set of grid points within the variable minima and maxima.
Select the training points according to the required number.
A. Calculation hardware and routine specifications

A.2.3. Routine to train and test the CH

Scale the training points.
Select the inside class.
Start calculation measurement.
Train the convex hull with the inside class with ‘convhulln’.
Store the boundary defining data.
(‘A’ and ‘b’ of inequality constraints with ‘vert2con’)
Stop calculation measurement.

Scale the test points.
Start calculation measurement.
Determine for each test point (i):
  if A * x_test(i) <= b.
  Assign a class.
Stop calculation measurement.

A.2.4. Routine to train and test the PEV

Assign a value for the percentile.
Scale the training points.
Select the inside class.
Start calculation measurement.
Train the PEV with the inside class.
Store the boundary defining data.
(‘quantile’ of the variance based on the defined percentile)
Stop calculation measurement.

Scale the test points.
Start calculation measurement.
Calculate the variance of the test data.
Determine for each test point:
  variance <= percentile.
  Assign a class.
Stop calculation measurement.
Assign a value for sigma and gamma.
Scale the training points.
Start calculation measurement.
Train the LS-SVM with all the training points and classes.
Store the boundary defining data.
(‘alphas’ and ‘b’)
Stop calculation measurement.

Scale the test points.
Start calculation measurement.
Determine for each test point (i):
  if alpha * kernel(x,x_test(i)) + b <= 0.
    Assign a class.
Stop calculation measurement.
A.2.6. Routine to train and test the LS-SVM-LOO

Scale the training points.
Start calculation measurement.
Minimise the tuning parameter sigma and gamma by ’fmincon’.
   Invert the LS-matrix.
   Train the LS-SVM with all the training points and classes.
   Store of the boundary defining data.
      (‘alphas’ and ‘b’)
   Calculate the residual.
Store the optimal values for sigma and gamma.
Train the LS-SVM with all the training points, classes, sigma and gamma.
Store the boundary defining data.
(‘alphas’ and ‘b’)
Store of the boundary defining data.
Stop calculation measurement.

Scale the test points.
Start calculation measurement.
Determine for each test point (i):
   if alpha * kernel(x,x_test(i)) + b <= 0.
      Assign a class.
Stop calculation measurement.

A.2.7. Routine to compare output

Compare the test point assigned class to the true class:
   test class +1 == true class +1 --> label ‘true positive’.
   test class -1 == true class -1 --> label ‘true negative’.
   test class -1 == true class +1 --> label ‘false positive’.
   test class +1 == true class -1 --> label ‘false negative’.
A.2.8. Main routine

Define the problem dimension (number of variables).
Define the minima and maxima for each variable.
Load the problem model.

Define the number of training points.

Repeat the following 25 times:
  Generate a space-filling set of training points.
  Generate output data for the training points by means of the model.
  Define the output limiting value.
  Based on the limit value assign the training points to their class.

Define the number of test points.
Generate a random set of test points.
Generate output data for the test points by means of the model.
Based on the limit value assign the training points to their class.

Define the solving method.

Start calculation measurement.
Train the solver by means of the training data and class.
Store of the boundary defining data.
Stop calculation measurement.

Start calculation measurement.
Based on the boundary assign the test points to their class.
Stop calculation measurement.

Compare the test point assigned class to the true class.
Assign assessment values according to the confusion matrix.

Calculate the average assessment values and calculations times for the 25 repetitions.
A.3. Data for the 2D reference shapes

In the table A.1 the polynomial data for the 2D reference shapes can be found. The grid used for generation points was spanned from $-2$ to $2$ for both variables. The data for the 3D, 4D and 7D are not explicitly presented in this report, but can be made available at request.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>2D Convex shape</th>
<th>2D Non-convex shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial order</td>
<td>4</td>
<td>False negative</td>
</tr>
<tr>
<td>Overlap</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Number of coefficients</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>1st coefficient</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2nd coefficient</td>
<td>0.5186</td>
<td>0.2653</td>
</tr>
<tr>
<td>3rd coefficient</td>
<td>0.9730</td>
<td>0.8244</td>
</tr>
<tr>
<td>4th coefficient</td>
<td>0.6490</td>
<td>0.9827</td>
</tr>
<tr>
<td>5th coefficient</td>
<td>0.8003</td>
<td>0.7302</td>
</tr>
<tr>
<td>6th coefficient</td>
<td>0.4538</td>
<td>0.3439</td>
</tr>
<tr>
<td>7th coefficient</td>
<td>0.4324</td>
<td>0.5841</td>
</tr>
<tr>
<td>8th coefficient</td>
<td>0.8253</td>
<td>0.1078</td>
</tr>
<tr>
<td>9th coefficient</td>
<td>0.0835</td>
<td>0.9063</td>
</tr>
<tr>
<td>10th coefficient</td>
<td>0.1332</td>
<td>0.8797</td>
</tr>
<tr>
<td>11th coefficient</td>
<td>-</td>
<td>0.8178</td>
</tr>
<tr>
<td>12th coefficient</td>
<td>-</td>
<td>0.2607</td>
</tr>
<tr>
<td>13th coefficient</td>
<td>-</td>
<td>0.5944</td>
</tr>
<tr>
<td>14th coefficient</td>
<td>-</td>
<td>0.0225</td>
</tr>
<tr>
<td>Threshold</td>
<td>3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Table A.1.: Values of the polynomial parameters for the 2D hypothetical data set
B. Supporting mathematics for PEV

This appendix summarises the used mathematical and statistical relations. More detailed information and derivation can be found in [1, 11, 13, 26, 27, 28, 35].

B.1. Probability

The probability of an event can be defined as a real number between 0 and 1 that expresses how likely an event is to occur for a large number of trials. The number 1 means that the event will certainly occur. If it is absolutely not possible for the event to occur, its probability is 0.

A discrete random variable has a 'probability mass function' [26]:

\[ p(x) = P(X = x) \quad (B.1) \]

A continuous random variable has a the 'probability density function'. For a continuous random variable \( X \in B \) the 'probability density function' \( P(x) \) is a function satisfying:

\[ P(X \in B) = \int_B f(x) \, dx. \quad (B.2) \]

Where \( f \) satisfies the following two conditions [1]:

1. \( f(x) \geq 0 \) on \([x_{\text{min}}, x_{\text{max}}]\),
2. \( P(-\infty < x < \infty) = \int_{-\infty}^{\infty} f(x) \, dx = 1 \)

For both types of variables the cumulative distribution function \( F(x) = P(X \leq x) \) can be given. For discrete variables the function is defined by Equation (B.3) and for continuous functions by Equation (B.4) [26]:

\[ F(x) = \sum_{i=1}^{n} p(x_i) \quad \text{for } x \text{ in ascending order} \quad (B.3) \]

\[ F(x) = \int_{-\infty}^{x} f(x) \, dx \quad (B.4) \]
B.1.1. Normal distribution

A common way to depict a normal distribution is by means of the distribution density function as in Figure B.1 and the cumulative density plot in Figure B.2 for different values of $\mu$ and $\sigma^2$. The standard normal distribution has $\mu = 0$ and $\sigma^2 = 1$.

![Probability density function for a normal distribution with different $\mu$ and $\sigma$](image)

Figure B.1.: Normal distribution density plot

B.1.2. Expected or mean value

The mean $\mu$ of a random variable $X$ with probability mass function $P(X = a_i)$ is defined as:

$$\mu = \sum_{i=1}^{n} x_i p_i \quad \text{for } x \text{ in ascending order} \quad (B.5)$$

For a continues random variable with the probability density function $f(x)$ the relation can be written as:

$$\mu = \int_{-\infty}^{\infty} xf(x) \, dx \quad (B.6)$$

The mean is also called the expected value $E(X)$ of the random variable $X$:

$$\mu = E(X) \quad (B.7)$$
B.1. Probability

Cumulative distribution function for a normal distribution with different $\mu$ and $\sigma$

Figure B.2.: Cumulative density distribution

B.1.3. Variance

The variance $\sigma^2$ is the square of the expected value of the distance between $X$ and its mean value $\mu$:

$$\sigma^2 = \sum_{i=1}^{n} (x_i - \mu)^2 p_i$$  \hspace{1cm} (B.8)

Or for continuous variables:

$$\sigma^2 = \int_{-\infty}^{\infty} (x - \mu)^2 f(x) \, dx$$  \hspace{1cm} (B.9)

In terms of expected values the equation can be rewritten as:

$$\sigma^2 = Var(X)$$

$$= E((X - \mu)^2)$$

$$= E(X^2) - \mu^2$$

$$= E(X^2) - (E(X))^2$$  \hspace{1cm} (B.10)
B. Regression analysis

By means of regression analysis one tries to find the distribution of a response (or random or dependent) variable or some characteristics of the distribution (e.g. its mean) as a function of one or more deterministic (or independent or controlled) variables. Or with other words one tries to establish the relation between variables and how to describe the relation. For more information on regression analysis, linear regression and least-squares see also [13, 26]. In the following section the complete derivation is given to determine the expression for the ‘Prediction Error Variance’ used for the PEV method. Part of the derivations are repeated from Chapter 3 to give the complete derivation.

B.2.1. Simple regression model

A linear model of the first order that describes the (measured) relation of the independent \( x \) and dependent variable \( y \) can be written as:

\[
y = \beta_0 + \beta_1 x + \epsilon
\]  

(B.11)

Where \( \epsilon \) represents the unknown measurement error. An extension to a second order linear model can be written as:

\[
y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon
\]  

(B.12)

It is also possible that one has several measurements or observation of the same system, that is one has several, let it be \( n \) observations, of the independent variables \( (x_1, x_2, \ldots, x_n) \) and accordingly \( n \) responses \( (y_1, y_2, \ldots, y_n) \). So a second order model can be written as the following set of equations:

\[
\begin{align*}
y_1 &= \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \epsilon_1 \\
y_2 &= \beta_0 + \beta_1 x_2 + \beta_2 x_2^2 + \epsilon_2 \\
&\vdots \\
y_n &= \beta_0 + \beta_1 x_n + \beta_2 x_n^2 + \epsilon_n
\end{align*}
\]

If one has more independent variables (e.g. \( x_1 \) and \( x_2 \)) a second order linear model can be written as:

\[
y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \beta_4 x_2^2 + \beta_5 x_1 x_2 + \epsilon
\]  

(B.13)

It is called multiple regression, where the results lead to a linear model, since the systems is linear in its unknown coefficients \( \beta_0, \ldots, \beta_k \). In vector notation a second (or higher) order linear model that describes the (measured) relation between the independent \( (X_i) \) and dependent variables \( (y) \) can be written as:

\[
y = X \beta + \epsilon
\]  

(B.14)
B.2. Regression analysis

Where

Response vector

\[ y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad (n \times 1) \]

Design matrix

\[ X = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{bmatrix}, \quad (n \times (k + 1)) \]

Coefficient vector

\[ \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}, \quad ((k + 1) \times 1) \]

Residual vector

\[ \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}, \quad (n \times 1) \]

where \( X \) is also called the model or regression matrix. A linear model that can be used to predict an outcome is given as:

\[ \hat{y} = X b \]  

(B.15)

In general the goal of regression analysis is to find the fitted coefficient vector \( b \) that minimises the residuals. An approach to accomplish this is given in the next section.

B.2.2. Minimising the residuals by means of least-squares

The Gauss-Markov theorem defines that for a linear regression model, with uncorrelated errors, a zero expectation of the errors and a constant variance of the errors, the best linear unbiased estimator (BLUE) is found by the least-squares approach [13]. The approach tries to minimise the residuals and is explained in this section.

The residuals can be written as:

\[ e = y - \hat{y} \]  

(B.16)

where \( e = [e_1, e_2, \ldots, e_n] \) represents the vector of residuals.

The method of least-squares minimises the sum of the squared residuals, which can be written as:

\[ e^2 = e^T e \]

\[ = (y - \hat{y})^T (y - \hat{y}) \]

\[ = (y - X b)^T (y - X b) \]

\[ = (y^T - b^T X^T)(y - X b) \]

\[ = y^T y - y^T X b - b^T X^T y + b^T X^T X b \]

\[ = y^T y - 2y^T X b + b^T X^T X b \]  

(B.17)

The last simplification is possible since the result is a scalar and in that case \( y^T X b = b^T X^T y \) holds.
Minimising the result with respect to the unknown \( b \) can be obtained by taking the first derivative to \( b \) and to setting it equal to 0. Hence the best estimate for to \( b \) will be:

\[
\frac{d}{db} e^T e = -2X^T y + 2X^T X b \\
\frac{d}{db} e^T e = 0 \\
-2X^T y + 2X^T X b = 0 \\
X^T X b = X^T y \\
b = (X^T X)^{-1} X^T y
\] (B.18)

The result is unbiased and has a minimum variance. The equation \( \hat{y} = X b \) accordingly can be written as:

\[
\hat{y} = X (X^T X)^{-1} X^T y \\
= H y
\] (B.19)

Where \( H \) is called the hat matrix and has the special properties of being symmetric, \( H^T = H \), and idempotent, \( H^2 = H \). Some applications and use of the hat matrix are given in Section B.4.

### B.2.3. Ridge regression

Although optimized, the coefficients \( b \) can result in bad estimates if \( X^T X \) is not close to the unit matrix \( I \) [16, 17]. Regularisation of ill-posed problems can be achieved by the biased estimation approach of ridge regression [13]. Ridge regression is the term used in statistic application, for other applications also the term 'Tikhonov regularization' is used [7]. The basic idea behind ridge regression is to have a large reduction in the coefficient variance at the cost of a small bias on the coefficient estimates [13]. It can be expressed as:

\[
b = (X^T X - \gamma I)^{-1} X^T y
\] (B.20)

Here \( I \) presents the unit matrix and \( \gamma \) is the ridge regression parameter. The choice of the ridge regression parameter \( \gamma \) generally requires knowledge about the estimator. Often optimisation algorithms have to be implemented. For the effect of the ridge parameter it can be said that the larger the value of the ridge regression parameter, the bias on the least-squares estimator will be larger but the variance of the least-squares estimator will be smaller.
B.3. Expectation and variance for the regression results

B.3.1. Expectation and variance for the measurement error - Assumptions

The measurement error is defined as \( \epsilon \). It is assumed that the distribution of the measurement errors each can be given by a normal distribution, with a mean value of zero and a standard deviation of that of the response variable or \( \epsilon \sim \mathcal{N}(0, \sigma^2 \epsilon I_n) \) [13]. Furthermore the expectation of the covariance matrix for the measurement errors \( \epsilon_{ij} \) will be zero where \( i \neq j \) and thus will only have values on its diagonal \( E(\epsilon_i, \epsilon_j) = 0 \) for all \( i \neq j \) and \( E(\epsilon_i, \epsilon_j) = \sigma^2 \epsilon \) for all \( i = j \). Therefore:

\[
E(\epsilon) = 0 \quad \text{(B.21)}
\]
\[
\text{Var}(\epsilon) = I \sigma^2 \epsilon \quad \text{(B.22)}
\]

B.3.2. Expectation and variance for the response variable

Since \( y = X\beta + \epsilon \) the expectation of \( y \) can be written as:

\[
E(y) = E(X\beta + \epsilon) = E(X\beta) + E(\epsilon) = X\beta + E(\epsilon) = X\beta \quad \text{(B.23)}
\]

Substituting the result of Equation (3.15) and rewriting \( y = X\beta + \epsilon \), the variance can be given as:

\[
\text{Var}(y) = E((y - \mu)^2) = E((y - E(y))^2) = E((y - E(y))^T(y - E(y))) = E((y - X\beta)^T(y - X\beta)) = E(\epsilon^T\epsilon) = I \sigma^2 \epsilon \quad \text{(B.24)}
\]
B. Supporting mathematics for PEV

B.3.3. Expectation and variance for the residual

Since the following relation holds (see Equation (3.12)):

\[ e = y - \hat{y} = y - H y = (I - H) y \]

(B.25)

The expectation of \( e \), while using the result of Equation (3.15), can be written as:

\[ E(e) = E((I - H)y) = (I - H)E(y) = (I - H)X\beta \]

(B.26)

The variance of \( e \) can be derived as:

\[
\text{Var}(e) = E((e - E(e))^2) = E((e - E(e))^T(e - E(e))) = E(((I - H)y - (I - H)X\beta)^T((I - H)y - (I - H)X\beta)) = E((y - X\beta)^T((I - H)(y - X\beta))) = E(\epsilon^T(I - H)^T(I - H)\epsilon) = (I - H)^T(I - H)E(\epsilon^T\epsilon) = (I - H)\sigma^2_{\epsilon} \]

(B.27)

The last step can be explained since \((I - H)\) is idempotent and symmetric as well, which can easily be proved analogue to the proof of the properties for the hat matrix \( H \) in B.4.

B.3.4. Expectation and variance for the predicted response variable

The predicted output variable is defined by Equation (3.7) as \( \hat{y} = Xb \). But using the result of Equation (3.12) \( \hat{y} = Hy \) and or Equations (3.15) one can also write the expectation of \( \hat{y} \) as:

\[ E(\hat{y}) = E(Hy) = HE(y) = HX\beta \]

(B.28)
B.3. Expectation and variance for the regression results

In a similar way the variance of \( \hat{y} \) can be derived as:

\[
\text{Var}(\hat{y}) = E((\hat{y} - E(\hat{y}))^2) \\
= E((H y - E(H y))^T (H y - E(H y))) \\
= E((H(y - E(y)))^T (H(y - E(y)))) \\
= E((y - E(y))^T H^T H(y - E(y))) \\
= H^T H E((y - E(y))(y - E(y))^T) \\
= H \sigma^2_\epsilon \tag{B.29}
\]

B.3.5. Expectation and variance of the coefficients

The model coefficients can be estimated with \( b = (X^T X)^{-1}X^Ty \). Therefore the expectation of \( b \) can be written as:

\[
E(b) = E((X^T X)^{-1}X^Ty) \\
= (X^T X)^{-1}X^T E(y) \\
= \left(\begin{array}{c} 
X^T X^{-1}X^T \beta \\
\end{array}\right) \\
= \beta \tag{B.30}
\]

With that result, the estimation of \( b \) and the definition of \( y \) the variance \( \text{Var}(b) \) can be derived:

\[
\text{Var}(b) = E((b - E(b))^2) \\
= E((b - \beta)^2) \\
= E(((X^T X)^{-1}X^Ty - \beta)^2) \\
= E(((X^T X)^{-1}X^T(X\beta + \epsilon) - \beta)^2) \\
= E(((X^T X)^{-1}X^T \beta + (X^T X)^{-1}X^T \epsilon - \beta)^2) \\
= E((\beta + (X^T X)^{-1}X^T \epsilon - \beta)^2) \\
= E(((X^T X)^{-1}X^T \epsilon)^2) \\
= E(((X^T X)^{-1}X^T \epsilon)^T((X^T X)^{-1}X^T \epsilon)) \\
= E(\epsilon^T X((X^T X)^{-1})^T(X^T X)^{-1}X^T \epsilon) \\
= \left(\begin{array}{c} 
X \otimes (X^T X)^{-1} X^{-1} (X^T X)^{-1} X^T \epsilon \\
\end{array}\right) \\
= (X^T X)^{-1} \sigma^2_\epsilon \tag{B.31}
\]
B. Supporting mathematics for PEV

B.3.6. Expectation and variance for a new measurement

The expectation and variance of the outcome of a new measurement is analogous to the derivation
for the predicted response variable making use of the previous result. Let \( \hat{y}_{\text{new}} = X_{\text{new}} b \) and
insert the best possible estimation for \( b \), that is \( b = (X^T X)^{-1} X^T y \), see (3.11). It gives:

\[
\hat{y}_{\text{new}} = X_{\text{new}} (X^T X)^{-1} X^T y
\]

(B.32)

The expectation of \( \hat{y}_{\text{new}} \) can assumed by the expectation of the true response variable:

\[
E(\hat{y}_{\text{new}}) = X_{\text{new}} \beta
\]

(B.33)

Since \( \hat{y}_{\text{new}} = X_{\text{new}} b \) the variance can be written as:

\[
\text{Var}(\hat{y}) = E((\hat{y}_{\text{new}} - E(\hat{y}_{\text{new}}))^2)
= E((X_{\text{new}} b - X_{\text{new}} \beta)^2)
= E((X_{\text{new}} (b - \beta))^2)
= E((X_{\text{new}} (b - \beta))^T (X_{\text{new}} (b - \beta)))
= (X_{\text{new}}^T (b - \beta)^T (b - \beta) X_{\text{new}})
= X_{\text{new}}^T \underbrace{E((b - \beta)^T (b - \beta))}_{\text{Var}(b)} X_{\text{new}}
= X_{\text{new}}^T (X^T X)^{-1} \sigma^2_{\epsilon} X_{\text{new}}
= X_{\text{new}}^T (X^T X)^{-1} X_{\text{new}} \sigma^2_{\epsilon}
\]

(B.34)

Let there exist a function \( X_{\text{new}} = f(x_i) \) then for any predicted value \( \hat{y}_i \) the variance of \( \hat{y}_i \) at some
point \( x_i \) is called the prediction error variance and is given by [35]:

\[
\text{Var}(\hat{y}_i) = \left(\frac{f(x_i)}{P E V}\right)^T (X^T X)^{-1} \frac{f(x_i)}{P E V} \sigma^2_{\epsilon}
\]

(B.35)
B.4. The hat matrix and leverage

B.4.1. Matrix idempotence

A matrix is called idempotent if $H^2 = H$. The hat matrix $H$ is idempotent since:

\[
H = X(X^T X)^{-1}X^T \\
H^2 = (X(X^T X)^{-1}X^T)(X(X^T X)^{-1}X^T) \\
= X(X^T X)^{-1}(X^T X)(X^T X)^{-1}X^T \\
= X(X^T X)^{-1}X^T \\
= H
\]

The derivation above makes use of the following general matrix rules

\[
(AB)^T = B^TA^T \\
(AB)^{-1} = B^{-1}A^{-1} \\
(A^{-1})^T = (A^T)^{-1}
\]
C. Supporting mathematics for SVM

C.1. Optimisation

C.1.1. Basic constrained optimisation problem

The basic constrained optimisation problem can be given by [7, 42]:

\[
\begin{align*}
\text{Minimise} & \quad f(x) \quad x \in \mathbb{R}^n \\
\text{Subject to} & \quad g_i(x) = 0 \quad \text{for } i = 1, 2, \ldots, m \\
& \quad h_j(x) \leq 0 \quad \text{for } j = 1, 2, \ldots, p
\end{align*}
\]

Here \( x \) is the optimisation variables vector, \( f(x) \) is the objective or cost function, \( g_i(x) \) are the equality constraints and \( h_i(x) \) are the inequality constraints.

The definition of an optimisation problem is to minimise a function. It requires to find the value of the variable(s) of the function that achieve the minimisation. If the optimisation problem is to maximise a function, the procedure remains the same, the optimisation criterion has to be negated [50].

It means that \( \max[f(x)] = \min[-f(x)] \) but also that for a given \( h_i(x) \geq 0 \) one can rewrite the constraint as \( -h_i(x) \leq 0 \).

Furthermore if \( g_i(x) = c \), where \( c \) is some positive constant, one can rewrite it as \( \tilde{g}_i(x) = g_i(x) - c \).

C.1.2. Linear program

A linear program or linear optimisation is finding the extreme of a linear objective function subjected to a set of linear (in)equalities:

\[
\begin{align*}
\text{Minimise } x & \quad f(x) = c^T x \\
\text{Subject to} & \quad A_{ineq} x \leq b_{ineq} \\
& \quad A_{eq} x = b_{eq}
\end{align*}
\]

Where \( A \) is some matrix and \( c \) and \( b \) are vectors.
C.1.3. Quadratic program

A convex optimisation problem is a quadratic program (QP) if the objective function is quadratic convex, but where the constraint functions are still affine [7]. A quadratic programming problem is an optimisation problem, where the objective function is quadratic and convex and the constraints are linear. The constraints can be given by both equality and inequality constraints. In general the problem can be written as:

Minimise $x$ for $f(x) = \frac{1}{2}x^T H x + h^T x$  \hspace{1cm} (C.2)
Subject to $A_{ineq} x \leq b_{ineq}$
and/or $A_{eq} x = b_{eq}$

Where $H$ is a symmetric matrix, $A$ some matrix and $h$ and $b$ are vectors.
If only equality constraints exist, the problem can be easily be solved by making use of Lagrange multipliers and finding the extreme values for the converted problem. If in equality constraints are present a relaxation of the problem is achieved by rewriting the inequality as equality constraints.

C.1.4. Convex set and functions

A set $S$ of points in $\mathbb{R}^n$ is convex if for all $x_1, x_2 \in S$ and $\theta \in [0, 1]$ it holds that:

$$(1 - \theta)x_1 + \theta x_2 \in S$$  \hspace{1cm} (C.3)

And in general for all $x_1, x_2, ..., x_n \in S$ and $\sum_{i=1}^{n} \theta_i = 1$:

$$\sum_{i=1}^{n} \theta_i x_i \in S$$  \hspace{1cm} (C.4)

An intersection of two convex sets is also a convex set. The convex hull of a set of points is the smallest convex set that contains all points of the original set [7, 42].

A function $f$ is convex on a set $S$ if:

$$f((1 - \theta)x_1 + \theta x_2) \leq (1 - \theta)f(x_1) + \theta f(x_2)$$  \hspace{1cm} for all $x_1, x_2 \in S$ and $\theta \in [0, 1]$  \hspace{1cm} (C.5)

The epigraph is the set of points that lie on or above the graph of $f(x)$ and is also convex [7, 42].

A graphical understanding of convex problems in two dimensions is given by Figures C.1-C.5. Assume one has a quadratic function of the form $f(x) = x^T H x$, where $H$ represents the coefficient matrix.
The matrix $H$ can be split into five different categories according to the properties of the eigenvalues.

- $H$ is positive definite for $x^T H x > 0$, which means all eigenvalues are positive (see Figure C.1). In this case the problem is convex.

- $H$ is semi-positive definite for $x^T H x \geq 0$, which means the eigenvalues are positive or equal to zero (see Figure C.2). Also in this case the problem is convex.

- $H$ is negative definite for $x^T H x < 0$, which means all eigenvalues are negative (see Figure C.3) and the problem is as such not convex.

- $H$ is semi-negative definite for $x^T H x \leq 0$, which means all eigenvalues are negative or equal to zero (see Figure C.4). Also by definition this problem is not convex.

- $H$ is indefinite if $x^T H x$ can have both positive and negative values. This also means that the eigenvalues have both positive and negative values (see Figure C.5). In this case it is clear from the graph that the problem is not convex.

![Figure C.1: Positive definite matrix](image1.png)

![Figure C.2: Positive definite matrix](image2.png)

![Figure C.3: Positive definite matrix](image3.png)
C. Supporting mathematics for SVM

Figure C.2.: Semi-positive definite matrix

Figure C.3.: Negative definite matrix
C.1. Optimisation

Figure C.4.: Semi-negative definite matrix

Figure C.5.: Indefinite matrix
C. Supporting mathematics for SVM

C.1.5. The method of Lagrange multipliers

Let there be a function $f(x, y)$ that is continuous and has continuous (partial) derivatives. Now apply a constraint to the problem: $g(x, y) = c$ that is also continuous and continuously differentiable, where $c$ is some positive constant. A possible way to find the extreme values of $f(x, y)$ for the constrained optimisation problem is by means of applying Lagrange multipliers, $\alpha$. The Lagrangian is then defined as [1]:

$$L(x, y, \alpha) = f(x, y) + \alpha(g(x, y) - c) \quad (C.6)$$

Solving the following set of equations results in solutions for $x_0$, $y_0$ and $\alpha_0$:

$$\frac{\partial L}{\partial x} \bigg|_{(x_0, y_0, \alpha_0)} = 0 \quad (C.7)$$
$$\frac{\partial L}{\partial y} \bigg|_{(x_0, y_0, \alpha_0)} = 0 \quad (C.8)$$
$$\frac{\partial L}{\partial \alpha} \bigg|_{(x_0, y_0, \alpha_0)} = g(x, y) - c = 0 \quad (C.9)$$

Here the last equation is zero, because it is exactly the constraint. The found point(s) is(are) called the critical point(s) of the Lagrangian, but it(they) might not be the extreme point(s) of the function $f(x, y)$ itself. More compactly and for any number of variables, where $x = (x_1, x_2, ..., x_n)$, one can also write:

$$\nabla f(x) = \alpha \nabla g(x) \quad (C.10)$$

The Lagrange multipliers might also be introduced for multiple constraints, as introduced in the section of the basic constrained optimisation formulation C.1.1. Let $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$ be the Lagrange multipliers for the first set of equality constraints and $\nu = (\nu_1, \nu_2, ..., \nu_p)$ the Lagrange multipliers for the second set of equality constraints. One can write the Lagrangian with two different constraints as:

$$L_P(x, \alpha, \lambda) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{j=1}^{p} \nu_j h_j(x) \quad (C.11)$$

C.1.6. Karush–Kuhn–Tucker conditions

The Method of Lagrange multipliers can be given in a more general form and can at the same time be extended to inequality constraints by means of the Karush–Kuhn–Tucker (KKT) conditions [42].

Let $\bar{x}$ be the solution (the minimum) of the above convex constrained optimisation problem. Naturally $\bar{x}$ also minimises $L(x, \alpha)$. There exist KKT multipliers $(\alpha, \nu)$ such that the gradient
of \( L(x, \alpha) \) vanishes:
\[
\nabla f(\bar{x}) + \sum_{i=1}^{m} \alpha_i \nabla g_i(\bar{x}) + \sum_{j=1}^{p} \nu_j \nabla h_j(\bar{x}) = 0 \tag{C.12}
\]

With the last equation the Karush–Kuhn–Tucker conditions can be summarised:
\[
\begin{align*}
    g_i(\bar{x}) &\leq 0 \quad i = 1, \ldots, m \\
h_j(\bar{x}) &\leq 0 \quad j = 1, \ldots, p \\
\alpha_i &\geq 0 \quad i = 1, \ldots, m \\
\alpha_i g_i(\bar{x}) & = 0 \quad i = 1, \ldots, m \\

\nu_j &\geq 0 \quad j = 1, \ldots, p \\
\nu_j h_j(\bar{x}) & = 0 \quad j = 1, \ldots, p \\
\nabla L(\bar{x}, \alpha, \nu) & = 0
\end{align*}
\]

### C.1.7. Primal and dual forms

Assume a convex optimisation problem of the same form as given in Section C.1.1:
\[
\begin{align*}
    \text{Minimise} \quad & f(x) \\
    \text{Subject to} \quad & g_i(x) = 0 \quad \text{for} \quad i = 1, 2, \ldots, m \\
                    & h_j(x) \leq 0 \quad \text{for} \quad j = 1, 2, \ldots, p
\end{align*}
\]

As explained in Section C.1.5 the primal form of the Lagrangian for the optimisation can be given as:
\[
L_P(x, \alpha, \nu) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{j=1}^{p} \nu_j h_j(x) \tag{C.13}
\]

The dual function of the problem can be given as the minimum value of \( x \) for the above Lagrangian:
\[
l(\alpha, \nu) = \inf_{x} L_P(x, \alpha, \nu) = \inf_{x} (f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{j=1}^{p} \nu_j h_j(x)) \tag{C.14}
\]

The solution of the dual problem therefore defines the lower bound to the solution of the primal problem that is as a standard form defined as a minimisation problem. Furthermore the dual formulation is always convex [7].

If the solution can be solved for \( x \) than it can be proved that:
\[
f(x) \geq l(\alpha, \nu) \tag{C.15}
\]
Since $\nu \geq 0$ and $h_j(\mathbf{x}) \leq 0$ it can be written that:

\[
\begin{align*}
f(\mathbf{x}) & \geq f(\mathbf{x}) + \sum_{i=1}^{m} \alpha_i g_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x}) \\
& \geq \inf_{\mathbf{x}} (f(\mathbf{x}) + \sum_{i=1}^{m} \alpha_i g_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x})) \\
& \geq l(\alpha, \nu)
\end{align*}
\]

As an example how to implement the result consider the following linear problem.

\[
\begin{align*}
\text{Minimise} & \quad \mathbf{c}^T \mathbf{x} \\
\text{Subject to} & \quad \mathbf{A} \mathbf{x} - \mathbf{b} \leq 0
\end{align*}
\]

The Lagrangian with $\alpha \geq 0$ can be written as:

\[
L_P(\mathbf{x}, \alpha) = \mathbf{c}^T \mathbf{x} + \alpha^T (\mathbf{A} \mathbf{x} - \mathbf{b}) = -\alpha^T \mathbf{b} + (\alpha^T \mathbf{A} + \mathbf{c}^T) \mathbf{x}
\]

Thus the minimum of the Lagrangian is given by:

\[
L(\alpha) = \begin{cases} 
-\alpha^T \mathbf{b} & \text{if } \alpha^T \mathbf{A} + \mathbf{c}^T = 0 \\
-\infty & \text{otherwise}
\end{cases}
\]

It can be seen that the function is convex.

In dual form the constrained problem can be written as

\[
\begin{align*}
\text{Maximise} & \quad -\alpha^T \mathbf{b} \\
\text{Subject to} & \quad \alpha^T \mathbf{A} + \mathbf{c}^T = 0 \\
& \text{and } \quad \alpha \geq 0
\end{align*}
\]

An example on implementation of the primal and dual form for a quadratic problem consider the following.

\[
\begin{align*}
\text{Minimise} & \quad \mathbf{x}^T \mathbf{H} \mathbf{x} \\
\text{Subject to} & \quad \mathbf{A} \mathbf{x} \leq \mathbf{b}
\end{align*}
\]

Where $\mathbf{H}$ is positive definite.
The Lagrangian with $\alpha \geq 0$ can be written as:

$$
L_P(x, \alpha) = x^T H x + \alpha^T (A x - b) = x^T H x + \alpha^T A x - \alpha^T b = -\alpha^T b + (\alpha^T A + x^T H) x
$$

Thus the minimum of the Lagrangian is given by:

$$
\frac{\partial L_D}{\partial x} = 0 \quad \rightarrow \quad \alpha^T A + \frac{1}{2} x^T H = 0
$$

$$
\quad \rightarrow \quad x^T = -\frac{1}{2} \alpha^T A H^{-1}
$$

$$
\quad \rightarrow \quad x = -\frac{1}{2} (\alpha^T A H^{-1})^T
$$

$$
\quad \rightarrow \quad x = -\frac{1}{2} (H^{-1})^T A^T \alpha
$$

Step 1: Rewrite the condition as:

$$
L(\alpha) = \begin{cases} 
-\alpha^T b & \text{if } -\frac{1}{2} (H^{-1})^T A^T \alpha = 0 \\
-\infty & \text{otherwise}
\end{cases}
$$

It can be seen that the function is convex.

In dual form the constrained problem can be written as

Maximise $-\alpha^T b$

Subject to $-\frac{1}{2} (H^{-1})^T A^T \alpha = 0$

and $\alpha \geq 0$

Step 2: Substitute the result of $x$ in the primal form of the Lagrangian:

Maximise $-\frac{1}{2} ((H^{-1})^T A^T \alpha)^T H x - \frac{1}{2} (H^{-1})^T A^T \alpha \alpha^T (A x - b)$

Maximise $\frac{1}{4} \alpha^T A (H^{-1})^T A^T \alpha + \alpha^T A - \frac{1}{2} (H^{-1})^T A^T \alpha - \alpha^T b$

Maximise $\frac{1}{4} \alpha^T A (H^{-1})^T A^T \alpha - \frac{1}{2} \alpha^T A (H^{-1})^T A^T \alpha - \alpha^T b$

Maximise $-\frac{1}{4} \alpha^T A (H^{-1})^T A^T \alpha - \alpha^T b$

Subject to $\alpha \geq 0$
C.1.8. Mercer’s condition

A real-valued function \( k(\mathbf{x}, \mathbf{x}') \) satisfies Mercer’s condition for all square integrable functions \( g(\mathbf{x}) \) if [42]:

\[
\int k(\mathbf{x}, \mathbf{x}') g(\mathbf{x}) g(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0
\]  

(C.16)

Only kernels that satisfy Mercer’s condition can be used as a mapping for solving the learning problem. That is, in addition to the above condition, there exists a mapping \( \varphi(\mathbf{x}) \) and \( k(\mathbf{x}, \mathbf{x}') = \sum_i \varphi(\mathbf{x}_i) \varphi(\mathbf{x}_i)' \). Furthermore \( \int g(\mathbf{x})^2 d\mathbf{x} \) should be finite.

C.1.9. Kernels

The extension of solving SVM’s for non-linear separable data is made possible by applying the kernel trick. Since the data in the (training) problem only appears as a dot product one can consider transforming (or mapping) the data into a different (high dimensional) feature space \( H \) [8]. A possibility to do the transformation is to simply replace \( \mathbf{x} \) with \( \varphi(\mathbf{x}) \) and apply the kernel trick. It should be noted that \( \varphi(\mathbf{x}) \) might actually be infinite dimensional. It restricts the possible solving methodologies. Where for linear separable data the finding of the optimal hyperplane could be done by solving both the primal or dual problem, it is no longer possible if one applies the kernel trick. After using the transformation \( \varphi(\mathbf{x}) \), as \( \mathbf{w} \) is transformed into a different, possibly infinite, space. Therefore also the solution is then found in that different space. To obtain the solution is the original space once has to apply an inverse transformation to the solution, which is usually complicated [42].

There are three advantages of mapping the data into a feature space [38]:

1. Similarities can be defined from the dot product in \( H \):

\[
k(\mathbf{x}_i - \mathbf{x}_j) := \mathbf{x}_i \cdot \mathbf{x}_j = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)
\]

2. The data and learning algorithms can be analysed geometrically in the feature space, because it has a richer mathematical structure. The use of linear algebra and analytic geometry is therefore possible.

3. Since the choice of the non-linear mapping \( \varphi(\mathbf{x}) \) is open, one can choose a representation that is most suitable for each problem. Also the similarity measures as well as the learning algorithms can be widely varied.

Some suitable choices of kernels, depending on the problem at hand, may be [8, 38, 42]:

- for a linear support vector machine:

\[
k(\mathbf{x}_i - \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j
\]

- the polynomial kernel:

\[
k(\mathbf{x}_i - \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d
\]

- the Gaussian kernel:

\[
k(\mathbf{x}_i - \mathbf{x}_j) = e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}}
\]

- the sigmoid kernel:

\[
k(\mathbf{x}_i - \mathbf{x}_j) = \tanh(\kappa_1 (\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa_2)
\]

* Note that Mercer’s condition only holds for certain values of \( \kappa_1 \) and \( \kappa_2 \).
C.2. Linear algebra

C.2.1. Distance of a point to a plane

Let there be a plane $\Pi: ax + by + cz = d$ and a point $P_0 = (x_0, y_0, z_0)$. The distance between the plane and the point can be determined as follows.

The normal of the plane $\Pi$ can be written as $\mathbf{n} = ai + bj + ck$. The positioning vector from the origin to point $P_0$ is $\mathbf{r}_0$.

There is a point $P_1 = (x_1, y_1, z_1)$ on the plane $\Pi$ that is closest to point $P_0$ and the vector $\overrightarrow{P_1P_0}$ is perpendicular to the plane $\Pi$ and therefore parallel to its normal vector.

The distance $\overrightarrow{P_1P_0}$ can be defined as $s_1 = \left\| \overrightarrow{P_1P_0} \right\|$. For some other point $P = (x, y, z)$ on the plane $\Pi$, the distance vector is defined as $\mathbf{r}$ and the distance between the arbitrary point $P$ to $P_0$ is $s = \left\| \overrightarrow{PP_0} \right\| = \| \mathbf{r}_0 - \mathbf{r} \|$. Along the direction of the normal vector to the plane the distance becomes:

$$s = \left\| \frac{\mathbf{PP}_0 \cdot \mathbf{n}}{\| \mathbf{n} \|} \right\|$$

$$s = \left\| \frac{\| (\mathbf{r}_0 - \mathbf{r}) \cdot \mathbf{n} \|}{\| \mathbf{n} \|} \right\|$$

$$s = \left\| \frac{\| (\mathbf{r}_0 \cdot \mathbf{n}) - (\mathbf{r} \cdot \mathbf{n}) \|}{\| \mathbf{n} \|} \right\|$$

(C.17)

Since $P = (x, y, z)$ lies on plane $\Pi$: $\mathbf{r} \cdot \mathbf{n} = ax + by + cz = d$.

Substituting the result and $P_0 = (x_0, y_0, z_0)$ in Equation (C.17) one gets [1]:

$$s = \frac{\| ax_0 + by_0 + cz_0 - d \|}{\sqrt{a^2 + b^2 + c^2}}$$

(C.18)

C.2.2. Vector space

A function $Q: H \times H \to \mathbb{R}$ like $(x, x') \to Q(x, x')$ is a bilinear form on a vector space $H$. It has the following properties for all $x, x', x'' \in H$ and for all $\lambda, \lambda' \in \mathbb{R}$ [38]:

$$Q((\lambda x + \lambda' x'), x'') = \lambda Q(x, x'') + \lambda' Q(x', x'')$$

(C.19)

$$Q(x'', (\lambda x + \lambda' x')) = \lambda Q(x'', x) + \lambda' Q(x'', x')$$

(C.20)

A symmetric bilinear form is defined by the additional requirement that for all $x, x' \in H$ the following holds [38]:

$$Q(x, x') = Q(x', x)$$

(C.21)

For an n-dimensional vector space $H$ over a field of real numbers, a symmetric bilinear functional $Q$ might exist that can be an inner product or dot product on the vector space $H$. It is only possible if $Q$ is symmetric, as defined in Equation (C.21), and if $Q$ is strictly positive for all $x \in H$. 

---

125
C. Supporting mathematics for SVM

[38, 42]:

\[ x \cdot x \geq 0 \]  \hspace{1cm} (C.22)

Where \( x \cdot x = 0 \) is only valid for \( x = 0 \).

A Euclidean space can be defined by a real finite dimensional vector space \( H \), for which the inner product is defined [42].

A real (or complex) space \( H \) for which the inner product is defined is called a real (or complex) inner product space [42].

A Hilbert space in turn is defined as a complete inner product space [42].

C.2.3. Block matrix inversion

A matrix \( M \) can be decomposed following \( M = LDU \) in its lower \((L)\) and upper \((U)\) unit triangular matrix for some diagonal matrix \( D \).

\[
M = LDU = \begin{bmatrix}
1 & 0 \\
\ell_{21} & 1
\end{bmatrix}
\begin{bmatrix}
d_{11} & 0 & 1 \\
0 & d_{22} & u_{12}
\end{bmatrix}
\]  \hspace{1cm} (C.23)

For the following 2-by-2 square matrix it can be derived that:

\[
M = \begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
= \begin{bmatrix}
d_{11} & d_{11}u_{12} \\
\ell_{21}d_{11} & \ell_{21}d_{11}u_{12} + d_{22}
\end{bmatrix}
\]  \hspace{1cm} (C.24)

Where:

\[
d_{11} = A
\]

\[
u_{12} = A^{-1}B
\]

\[
\ell_{21} = CA^{-1}
\]

\[
d_{22} = D - CA^{-1}B
\]

In general one can write for a block matrix LDU decomposition:

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
= \begin{bmatrix}
I & 0 \\
CA^{-1} & I
\end{bmatrix}
\begin{bmatrix}
A & 0 \\
0 & D - CA^{-1}B
\end{bmatrix}
\begin{bmatrix}
I & A^{-1}B \\
0 & I
\end{bmatrix}
\]  \hspace{1cm} (C.25)

where \( A \) and \( D \) need to be square matrices, and \( A \) and \( D - CA^{-1}B \) need to be nonsingular and \( D - CA^{-1}B \) is called the Schur complement of \( A \) [26]. Inverting a block upper triangular matrix (or equivalently a block lower triangular matrix) gives the following convenient result:

\[
\begin{bmatrix}
I & X \\
0 & I
\end{bmatrix}^{-1}
= \begin{bmatrix}
I & -X \\
0 & I
\end{bmatrix}
\]  \hspace{1cm} (C.26)
Inverting a diagonal matrix gives:

\[
\begin{bmatrix}
D_1 & 0 \\
0 & D_2
\end{bmatrix}^{-1} = 
\begin{bmatrix}
D_1^{-1} & 0 \\
0 & D_2^{-1}
\end{bmatrix}
\]  \hspace{1cm} (C.27)

Making use of \((ABC)^{-1} = (C)^{-1}(B)^{-1}(A)^{-1}\) a block matrix inversion for a matrix given in Equation (C.25) can be implemented as follows:

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}^{-1} = 
\begin{bmatrix}
I & -A^{-1}B \\
0 & I
\end{bmatrix} 
\begin{bmatrix}
A^{-1} & 0 \\
0 & (D - CA^{-1}B)^{-1}
\end{bmatrix} 
\begin{bmatrix}
I & 0 \\
CA^{-1} & I
\end{bmatrix}
\]

\[
= 
\begin{bmatrix}
A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\
-(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1}
\end{bmatrix}
\]  \hspace{1cm} (C.28)
C.3. Summary formulas SVM

The decision function always depends on the solution of $w_0$ and $b_0$ and can be given as:

$$y_{new} = \text{sign} \left[ w_0 \cdot x_{new} + b_0 \right] \quad \text{for linear data}$$

or

$$y_{new} = \text{sign} \left[ w_0 \cdot \varphi(x_{new}) + b_0 \right] \quad \text{for non-linear data}$$

the latter form is the form used for data for which the assumption that the data is linearly separable is not known.

C.3.1. Linearly separable data

Primal form

$$\min_{w, b} J_P(w) = \frac{1}{2} w \cdot w$$

under the constraint of $y_i (x \cdot w + b) \geq 1$ for $i = 1, 2, ..., n$

Dual form

$$\max_{\alpha} J_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i y_i \alpha_j y_j x_i \cdot x_j$$

where $\sum_{i=1}^{n} \alpha_i y_i = 0$

and $0 \leq \alpha_i$ for $i = 1, 2, ..., n$

The decision function is given as

$$w_0 = \sum_{i=1}^{nSV} \alpha_i y_i x_i$$

$$b_0 = \frac{1}{nSV} \sum_{i=1}^{nSV} \frac{1}{y_i} - w_0 \cdot x_i$$

$$y_{new} = \text{sign} \left[ \sum_{i=1}^{nSV} \alpha_i y_i x_i \cdot x_{new} + b_0 \right]$$
C.3.2. Quasi-linearly separable data

Primal form

\[
\min_{w, b, \xi} J_P(w, \xi) = \frac{1}{2} w \cdot w + c \sum_{i=1}^{n} \xi_i
\]

under the constraint of \(y_i(x \cdot w + b) \geq 1 - \xi_i\) for \(i = 1, 2, ..., n\)

and \(\xi_i \geq 0\) for \(i = 1, 2, ..., n\)

Dual form

\[
\max_{\alpha} J_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i y_i \alpha_j y_j x_i \cdot x_j
\]

where \(\sum_{i=1}^{n} \alpha_i y_i = 0\)

and \(0 \leq \alpha_i \leq c\) for \(i = 1, 2, ..., n\)

The decision function is given as

\[
w_0 = \sum_{i=1}^{n_{SV}} \alpha_i y_i x_i
\]

\[
b_0 = \frac{1}{n_{SV}} \sum_{i=1}^{n_{SV}} \frac{1}{y_i} - w_0 \cdot x_i
\]

\[
y_{new} = \text{sign} \left( \sum_{i=1}^{n_{train}} \alpha_i y_i x_i \cdot x_{new} + b_0 \right)
\]

C.3.3. Non-linearly separable data

Primal form (including slack variables)

\[
\min_{w, b, \xi} J_P(w, \xi) = \frac{1}{2} w \cdot w + c \sum_{i=1}^{n} \xi_i
\]

where \(y_i(x \cdot \phi(x_i) + b) \geq 1 - \xi_i\) for \(i = 1, 2, ..., n\)

and \(\xi_i \geq 0\) for \(i = 1, 2, ..., n\)
C. Supporting mathematics for SVM

Dual form

$$
\max_{\alpha} J_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j k(x_i, x_j) \alpha_i \alpha_j
$$

where $$\sum_{i=1}^{n} \alpha_i y_i = 0$$

and $$0 \leq \alpha_i \leq c$$ for $$i = 1, 2, ..., n$$

The decision function is given as

$$
\begin{align*}
\mathbf{w}_0 &= \sum_{i=1}^{n_{SV}} \alpha_i y_i \varphi(x_i) \\
\mathbf{b}_0 &= \frac{1}{n_{SV}} \sum_{j=1}^{n_{SV}} \left( \frac{1}{y_j} - \sum_{i=1}^{n_{SV}} \alpha_i y_i k(x_i, x_j) \right) \\
y_{new} &= \text{sign} \left[ \sum_{i=1}^{n_{train}} \alpha_i y_i k(x_i, x_{new}) + \mathbf{b}_0 \right]
\end{align*}
$$

C.3.4. LS-SVM

Primal form (including slack variables, assuming non-linear data)

$$
\begin{align*}
\min_{\mathbf{w}, \mathbf{b}, \mathbf{e}} J_P(\mathbf{w}, \mathbf{e}) &= \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + \frac{1}{2} \gamma \sum_{i=1}^{n} e_i^2 \\
\text{subject to } y_i (\mathbf{x} \cdot \varphi(x_i) + \mathbf{b}) &= 1 - e_i \text{ for } i = 1, 2, ..., n
\end{align*}
$$

Dual form

$$
\begin{bmatrix}
\Omega + \frac{1}{\gamma} \mathbf{y} & \mathbf{y}^T \\
\mathbf{y}^T & 0
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\mathbf{b}
\end{bmatrix}
= 
\begin{bmatrix}
\mathbf{1} \\
\mathbf{0}
\end{bmatrix}
$$

with

$$
\Omega_{i,j} = y_i y_j k(x_i, x_j)
$$

Since $$\mathbf{b}_0$$ is a direct result from the least squares solution the decision function is given as:

$$
\begin{align*}
\mathbf{w}_0 &= \sum_{i=1}^{n_{SV}} \alpha_i y_i \varphi(x_i) \\
y_{new} &= \text{sign} \left[ \sum_{i=1}^{n_{train}} \alpha_i y_i k(x_i, x_{new}) + \mathbf{b}_0 \right]
\end{align*}
$$
C.3.5. LS-SVM LOO

Primal form (including slack variables, assuming non-linear data)

\[
\min_{w, b, e} J_P(w, e) = \frac{1}{2} w \cdot w + \frac{1}{2} \gamma \sum_{i=1}^{n} e_i^2
\]

subject to \( w \cdot \varphi(x_i) + b + e_i - y_i = 0 \)

Dual form

\[
\begin{bmatrix}
K + \frac{I}{\gamma} & 1 \\
1^T & 0
\end{bmatrix}
\begin{bmatrix}
\alpha \\
b
\end{bmatrix}
= \begin{bmatrix}
0 \\
y
\end{bmatrix}
\]

where

\( K = k(x_i, x_j) \)

With \( b_0 \) as a direct result from the least squares solution the decision function is given as;

\[
w_0 = \sum_{i=1}^{n_{SV}} \alpha_i \varphi(x_i)
\]

\[
y_{new} = \text{sign} [\alpha_i k(x_i, x_{new}) + b_0]
\]
C.4. Tuning parameter variation

In the graph C.6 the effect of the variation of $\sigma$ and $\gamma$ on the squared and normalised predicted error can be seen. The error is presented in the legend on the right-hand side of the figure.

Figure C.6.: The predicted failure while varying $\sigma$ and $\gamma$
D. Additional results for hypothetical data

In this appendix the results are shown for a randomly generated training data set, a non-separable data set (e.g., representing measurement noise) and the results of the calculations with the original ‘convhulln’ algorithm. The results follow the structure of the report and are therefore split into three sections: the assessment evaluation, the training time and allocation time.

D.1. Additional results test point assessment evaluation

D.1.1. Separable randomly generated training data

To be able to analyse the effect of the training data distribution both a random and space-filling distribution are tested. In this appendix the random distribution is shown for a two-dimensional data set. For a comparison to the space-filling generated data set see Figure 4.1 in Chapter 4. It can be seen that there is no relevant effect on the assessment distribution.

D.1.2. Non-separable training data

In Figure D.2 the results for a non-separable two-dimensional data set are shown. That type of training data has a clear effect on the results. Considering the convex boundary shape it can be seen that PEV now shows a small number of false-negative assessments. The CH method shows an increasing number of false-negative assessments, like for a separable data set but with a non-convex boundary. Both SVM-based methods show an increase in false-positive assessments.

D.1.3. Separable training data with the convhulln algorithm

The original convex hull algorithm has been considered for both a two- and a four-dimensional case in order to have a better base of comparison. The results are found in Figures D.3 and D.4. In this case only the results of the CH method need to be compared, since the other methods are obviously not influenced by it. There is no clear effect on the assessment results, the slight variation comes from the individual test cases. It means that both convex hull approaches produce the same quality of boundary description.
D. Additional results for hypothetical data

Figure D.1.: Assessment test points for a 2D randomly generated separable training data set

Figure D.2.: Assessment test points for a 2D non-separable training data set
D.1. Additional results test point assessment evaluation

Figure D.3.: Assessment test points for a 2D separable training data set with the original convhull

Figure D.4.: Assessment test points for a 4D separable training data set with the original convhull
D. Additional results for hypothetical data

D.2. Additional results training time

D.2.1. Separable randomly generated training data

Though the assessment might not have significantly changed by the random distribution, there might still be an effect found in training or allocation time. Comparing Figure D.5 with Figure 5.1 in Chapter 5 it can be seen that for both assessments, there is no effect on the training time.

D.2.2. Non-separable training data

As to be expected the non-separable training data does not influence the time to train the boundary in the presented two-dimensional case in Figure D.6.

D.2.3. Separable training data with the convhulln algorithm

The two-dimensional case trained with the original convex hull algorithm, Figure D.7, is a little faster than the modified algorithm in Figure 5.1. The original is a lot faster in four dimensions, even faster than the PEV method.
Figure D.5.: Training time with a 2D randomly generated separable training data set.
Figure D.6.: Training time with a 2D non-separable training data set
D.2. Additional results training time

Figure D.7.: Training time with a 2D separable training data set with the original convex hull
D. Additional results for hypothetical data

Figure D.8.: Training time with a 4D separable training data set with the original convhull
D.3. Additional results time to assess new points

D.3.1. Separable randomly generated training data

The time to allocate new test points is, as expected, not influenced by the training data distribution. The space-filling results can be found in Figure 6.1 and the randomly distributed data in Figure D.9.

D.3.2. Non-separable training data

Like for the randomly distributed data, it is not expected that a non-separable training data set will influence the allocation of any new points. It is confirmed in Figure D.10.

D.3.3. Separable training data with the convhulln algorithm

It is expected that the original convex hull algorithm will affect the time to allocate new points, due to the recalculation of the hull for every new point. And indeed Figure D.11 shows a large increase in allocation time for the CH method in comparison to Figure 6.1. The effect is even more present for the four-dimensional case, considering the different scaling of the y-axis.

D.4. Summary

For future purposes the space-filling distribution will be used, since it is a more comparable type of distribution that can be assumed for engine calibration purposes when DoE test plans are applied.

Furthermore it is assumed that the measurement data is of such good quality that the noise is not significant and will not influence the outcome of the calculated position of the boundary. Though the original convex hull method shows the same quality of results as the adjusted method in both two- and four-dimensional cases and is faster in training the boundary, it takes so much longer in assessing new points, that the latter method will be used for the future calculations.
Figure D.9.: Time to test new points with a 2D randomly generated separable training data set

Figure D.10.: Time to test new points with a 2D non-separable training data set
Figure D.11.: Time to test new points with a 2D separable training data set with the original convex hull.

Figure D.12.: Time to test new points with a 4D separable training data set with the original convex hull.
References


List of publications

2012
Conference: MathWorks Automotive Conference 2012 in Stuttgart, Germany
Paper title: AMG Automation Tool for engine calibration
Authors: Hasan Uzun, Nataša Kieft, Christian Manz, Steffen Waldmann

Conference: SDPS 2012 - The 17th International Conference on Transformative Science, Engineering, and Business Innovation in Berlin, Germany
Paper title: Application of Design Space Exploration Tools for Engine Calibration
Authors: Nataša Kieft, Geritt Kampmann, Oliver Nelles

Conference: World Congress on Engineering and Computer Science 2012 in San Francisco, USA
Paper title: Support Vector Machines for Design Space Exploration
Authors: Geritt Kampmann, Nataša Kieft, Oliver Nelles

2014
Conference: 6. Tagung Simulation und Test für die Automobilelektronik in Berlin, Germany
Paper title: Evaluation of Support Vector Machines as a Design Space Description Method in automotive applications
Authors: Nataša Kieft, René Linssen, Thomas Bäck

Conference: 3rd Conference on Design of Experiments (DoE) in Engine Development in Plymouth(MI), USA
Paper title: Evaluation of support vector machines as a design space description method for low-dimensional data sets
Authors: Nataša Kieft, René Linssen, Thomas Bäck
Curriculum Vitae

Nataša Kieft was born on November 27, 1981 in Amsterdam, the Netherlands. She finished her pre-university education (VWO) in 2000 while attending the secondary school ‘Helen Parkhurst’ in Almere. Motivated by her interest in technology, cars and planes, she decided to study Aerospace Engineering at Delft University of Technology. She undertook her internship with KLM Royal Dutch Airlines in the purchasing department in Amstelveen. The assignment, defined directly by Peter Hartman who at that time held the position of COO, included a research on how future developments in propulsion technology could benefit KLM in terms of fuel consumption and emissions and how this could support KLM’s decision on finding the best successors for the regional fleet.

After the internship Nataša decided she wanted to do her Master thesis in the automotive industry, to be able to compare the difference between the industries. She went to Porsche in Weissach, Germany, where she looked at the problem of reducing the brake temperatures from an aerodynamics point of view, for the brake system of a race car for long-distance races. With the Master thesis she completed her studies in April 2007.

In order to be able to complete the UNITECH Management Programme, an extra curricular programme for talented engineering students focusing on soft skills and international business, she completed a voluntary internship after finishing her studies. She got a position with Shell in Chester, the United Kingdom. Here she did research on the effect of viscosity on engine performance for V-Power Diesel fuels for passenger cars.

In April 2008 Nataša joined the Daimler trainee programme ‘CAReer’ in Stuttgart, Germany. She was responsible for the mechanical testing of several components in the development department for medium duty truck engines. During the trainee programme one of her projects was with Mercedes-AMG, where she contributed to the optimisation of undesired noise reduction of the new SLS powertrain project. In the beginning of 2009 she went to São Paulo, Brazil, to work on duration testing of the medium duty truck engines, but now for the Brazilian market.

Nataša worked another one-and-a-half year for Daimler Trucks in Stuttgart, until she decided it was time for something new. Motivated by her former manager at Mercedes-AMG, she decided to take up the challenge of starting a PhD research in the field of engine calibration with the focus on extending the possibilities and application of test methodologies.