Czech Technical University in Prague
Faculty of Nuclear Sciences nad Physical Engineering

Obor: Fyzikální inženýrství
Zaměření: FTTF

Support Vector Machine in Application to Massive Tokamak Data Analyses

RESEARCH WORK

Autor: Bc. Michal Odstrčil
Supervisor: RNDr. Jan Mlynář, Ph.D.
Year: 2011

Název práce:
Aplikace Support Vector Machine (SVM) v hromadné analýze dat z tokamaků

Autor: Bc. Michal Odstrčil

Obor: Fyzikální inženýrství
Druh práce: Výzkumná práce

Vedoucí práce: RNDr. Jan Mlynář, Ph.D.
ÚFP AV ČR v.v.i.
Konzultant: Dr. Andrea Murari
EFDA JET

Title:
Support Vector Machine in Application to Massive Tokamak Data Analyses

Author: Bc. Michal Odstrčil

Contents

Introduction **6**

1 Learning Algorithms Background **7**

1.1 Basic Properties of Learning Machines 7

1.2 Probabilistic Learning 8

1.3 Kernel Based Methods 10

2 Learning Machines **13**

2.1 Logistic Regression 13

2.1.1 Regularisation Methods for Logistic Regression 15

2.2 Relevance Vector Machine – RVM 16

2.3 Support Vector Machine – SVM 16

2.3.1 Different Precision of Data Points 18

2.3.2 Confidence Region 18

2.4 Testing 21

3 Model Selection **23**

3.1 Cross-Validation – CV 23

3.2 Bayesian Model Selection 24

3.3 Dimension Reduction 26

3.3.1 Wrappers 27

3.3.2 Embedded Methods 28

3.3.3 Model Validation 28

3.4 Scaling 29

3.5 Data Preprocessing 30

4 Results **31**

4.1 GOLEM database 31

4.2 Random probes 32

4.3 Univariate feature elimination 33

4.4 Recursive feature elimination with linear SVM 34

4.5 Recursive feature elimination with RBF SVM 35

Summary **39**

Introduction

Learning machines have wide field of use. They are used in physics, medicine, biology, economy, etc. Many different algorithms already exist for every possible problem such as optimisation, regression or classification. Algorithms that can deal with huge number of dimensions are required in biomedicine and learning algorithms for prediction of markets are often used in economy. This paper is focused on utilisation for plasma physics. In plasma physics, particularly in tokamak,learning algorithms are used for long time because physicists have to deal with huge amount of data to process. The learning algorithms can help in postprocessing to identify interesting events or classify data according some criterions. During the postprocessing phase it is possible to use ability of the algorithms to learn from new data and thus improve classification or regression.

Next great advantage of some learning machine algorithms is fast predictive phase. Thanks to this property, learning machines can be used for real-time systems, for example real-time tomography [1], real-time L-H mode transition detection [2, 3], real-time plasma disruptions detection [4]. Advantage of learning machines is faster processing compared the regular analysis like the tomography. On the other hand, learning machines are not able to reliably extrapolate the training data and thus if observed problem is significantly different from the training data, the prediction can be useless.

This paper is mainly focused on the postprocessing phase and massive data analyses. Two databases will be used as the data source: database of breakdowns on tokamak GOLEM and database of disruptions on tokamak JET. Both databases are approximately similar in number of samples and number of variables. The main difference is that GOLEM classification is binary and JET classification is multiclass. The aim was to add probabilities to the databases and predictions. This can be used to identify outliers and also probability prediction of individual phenomena in plasma could be used as input for some more advanced methods such as Bayesian networks.

Chapter 1

Learning Algorithms Background

1.1 Basic Properties of Learning Machines

Before introducing various algorithms, it is necessary to define some of the key differences among them. The number of different learning machines is huge and only some groups are used in this paper:

- *Regression, Classification*: If the data labels can take values only from a small set than the goal of machine learning is to predict which group the data belong to. On contrary, in case of continuous labels the aim is to perform regression. However, this two cases are basically very similar and some learning algoriths can do both.
- *Supervised, unsupervised, semisupervised*: If learning machines are trained on data labelled by expert, then it is called supervised learning. If the data are not labeled then it is called unsupervised learning. The unsupervised learning algorithms can for example detect separated groups or can be used for detection of outliers if there is only one group. However, the unsupervised learning is commonly much less reliable then the supervised. It is possible to use a combination so-called semisupervised learning, where some data are labelled by expert and some data are unlabeled. The unlabeled values are classificated according to the labelled data. Finally, newly labeled data are used to following learning.
- *Online or offline learning*: Online learning means that each newly predicted value is used to improve model. On the other hand, offline learning means that only one set of training points is used and once the algorithm is trained then the model is used only for predictions. Usually, the sparse models are faster for prediction however there are usually slow in learning while dense models, i.e Nearest neighbours, are fast in learning and thus can be used for online learning but are slow in prediction. Only offline learning is used for purposes of this work.
- *Generative or discriminative models*: Discriminative models are usually better for data classification, because they focus mainly on the features that distinguish the groups. On contrary, generative models are focused on the main characteristics of the datasets and are able to generate new data points with the same properties as

the original set. Generative models are usually easier and thus faster to fit and they can be trained for each class separately. The advantage of the discriminative models is easy possibility to use different preprocessing of the input vector $\boldsymbol{\phi}(\mathbf{x})$ [5] (sec. 1.3).

- *Fully observed or partially observed*: Partialy observed means that the training data have unknown variables in some dimensions. The training on partially observed data is much more problematic. Generally, it can be done only for generative models, where the missing values are replaced by the most probable ones. However, also the discriminative models ware trained on partially observed data [6].

1.2 Probabilistic Learning

This section introduces basic methods of probabilistic learning that will be used in this work. In this secion were used sources [7, 8, 9, 10, 11, 12]

Learning machines, used in this paper, belongs to the group of supervised learning. The input vectors are denoted $\mathbf{x_i}$ and the observed values are denoted t_i .

The regression function that assign t_i to $\mathbf{x_i}$ can be arbitrary but because of computation reasons a combination of non-linear functions is usually searched

$$y(\mathbf{x}, \mathbf{w}) = \sum_{m=1}^{M} w_m \boldsymbol{\phi}_m(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi} \tag{1.2.1}$$

The parameterised function consists of sum of non-linear basis functions $\boldsymbol{\phi}(x)$ and the parameters of model w_m are generally called *weights*. The assumptions of this form gives enough flexibility and allows to use many optimisation algorithms described in this paper.

Learning machines that are based on probability classification usually use a few common approaches to learn their parameters from data.

Prevalent approach is *maximum-likelihood* (ML). The goal of the ML is to identify the most probable parameters. This is also often called frequentist model. If the model is defined as *conditional probability distribution* (CPD) than it can be written as $P(\mathbf{y}|\boldsymbol{\Theta}) = \prod_{i=1}^{N} P(y_m|\boldsymbol{\Theta})$, where y_m are the observed values and $\boldsymbol{\Theta}$ are computed parameters. The goal is to maximize the probability of the model

$$\hat{\boldsymbol{\Theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\Theta}} \left(P(\mathbf{y}|\boldsymbol{\Theta})\right)$$

the log-likelihood for the observed data is

$$\hat{\boldsymbol{\Theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\Theta}} \sum_{m=1}^{N} \log P(y_m|\boldsymbol{\Theta}) \tag{1.2.2}$$

ML is widely used because of simple computation and also often an analytical solution exists. For example, the least squares method is the analytical solution for Gaussian

distribution. The minimised error E for least-squares (ML) solution of regression is

$$E = \frac{1}{2}\sum_{n=1}^{N}\left[t_n - \sum_{m=1}^{M} w_m \boldsymbol{\phi}_m(\mathbf{x_n})\right]^2 \tag{1.2.3}$$

If Φ is defined as $\Phi_{mn} = \boldsymbol{\phi}_m(\mathbf{x_n})$ then the maximum likelihood $\mathbf{w}_{\mathrm{ML}}$ is

$$\mathbf{w}_{\mathrm{ML}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{t} \tag{1.2.4}$$

The maximum likelihood has many interesting asymptotic properties for large number of samples. For example consistency: if the number of samples tends to infinity then the ML probability converges to the true value.

Disadvantages are problems for low number of samples or high number of free parameters. In this case, the ML can reach overfitting [9]. This can be mitigated by a minor change in the eq. (1.2.2) to include a prior belief $P(\boldsymbol{\Theta})$ on the parameters

$$\hat{\boldsymbol{\Theta}}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\Theta}} P(\mathbf{y}|\boldsymbol{\Theta})P(\boldsymbol{\Theta}) = \arg\max_{\boldsymbol{\Theta}}(\log P(\mathbf{y}|\boldsymbol{\Theta}) + \log P(\boldsymbol{\Theta})) \tag{1.2.5}$$

This is called *maximum a posteriori* MAP solution. This assumption can avoid the overfitting, on the other hand, it can introduce new problems through the apriori information.

Again, for Gaussian distribution and Gaussian prior, the ML has an analytical solution. If the expected noise is Gaussian with variance σ^2 and the prior distribution over parameters $\mathbf{w}$ is

$$p(\mathbf{w}|\lambda) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I}) = \prod_{m=1}^{M}\left(\frac{\lambda}{2\pi}\right)^{1/2}\exp\left(-\lambda w_m^2/2\right) \tag{1.2.6}$$

then the MAP solution of least squares method is

$$\mathbf{w}_{\mathrm{MAP}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sigma^2\lambda\mathbf{I})^{-1}\boldsymbol{\Phi}^T\mathbf{t} \tag{1.2.7}$$

where constant λ is called *ridge regression parameter*. This parameter must be selected according to some other knowledge, for example cross-validation, see Section 3.1.

However, MAP is still only a small step toward the full Bayesian probability. If the probability distribution over parameters and noise in data is Gaussiam as in the MAP solution, it is theoretically possible to integrate out (marginalise over) the unknown parameters $\mathbf{w}, \lambda, \sigma$ and obtain posterior distribution from Bayes' rule

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{normalisation}} \tag{1.2.8}$$

Distribution over parameters $\mathbf{w}$ is

$$p(\mathbf{w}|\mathbf{t}, \mathbf{x}, \lambda, \sigma^2) = \frac{p(\mathbf{t}|\mathbf{w}, \mathbf{x}, \lambda, \sigma^2)p(\mathbf{w}|\lambda)}{\int p(\mathbf{t}|\mathbf{w}, \mathbf{x}, \lambda, \sigma^2)p(\mathbf{w}|\lambda)\mathrm{d}\mathbf{w}} \tag{1.2.9}$$

$$p(\mathbf{w}|\lambda) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I}) \tag{1.2.10}$$

then the final analytical solution of the posterior for the Gaussian example is [10]

$$p(\mathbf{w}|\mathbf{t}, \lambda, \sigma^2) = \mathcal{N}(\mu, \mathbf{\Sigma}) \tag{1.2.11}$$

where the mean μ and covariance matrix $\mathbf{\Sigma}$ denote

$$\mu = (\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\sigma^2\mathbf{I})^{-1}\mathbf{\Phi}^T\mathbf{t} \tag{1.2.12}$$

$$\mathbf{\Sigma} = \sigma^2(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\sigma^2\mathbf{I})^{-1} \tag{1.2.13}$$

It is obvious that the mean value is identical to the MAP solution because the prior over weights have no bias. The advantage is the knowlenge of the whole posterior distribution over weights $\mathbf{w}$. Moreover, it is possible to apply the Bayesian rule (1.2.8) and the posterior prediction distribution can be obtained directly. This is important because usually the new predictions are the expected result

$$p(t_n^*|\mathbf{x}, \lambda, \sigma^2) = \int p(t_n^*|\mathbf{w}, \mathbf{x}, \sigma^2)p(\mathbf{w}|\lambda)\mathrm{d}\mathbf{w} \tag{1.2.14}$$

The prediction (1.2.14) is independ of the training data because the distribution $p(t_n^*|\mathbf{w}, \mathbf{x}, \sigma^2)$ already includes the information. Solution of the previous equation can be again found in closed-form. After marginalisation the distribution over the weights $\mathbf{w}$ the final prediction 1.2.14 is independent of the weight vector $\mathbf{w}$ and also the predicted value is known with its probability distribution.

One problem of the previous derivation is the assumption of the Gaussian distribution of the values t_n around the true function. It is not possible to find an analytical solution for other distributions. Thus it cannot be used for classification, see Section 2.2, it is only used for regression.

The previous equation can be also used to predict distribution over the model parameters

$$p(\mathbf{w}, \lambda, \sigma|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{w}, \sigma)p(\mathbf{w}|\lambda)p(\sigma^2)}{\int p(\mathbf{t}|\mathbf{w}, \sigma)p(\mathbf{w}|\lambda)p(\sigma^2)d\mathbf{w}d\lambda d\sigma^2}$$

but this equation usually cannot be solved analytically even for Gaussian distrubution. However, numerical integration can be used, i.e the Monte-Carlo method [10].

1.3 Kernel Based Methods

Before introducing some important machine learning algorithms, it is necessary to mention so called *kernel trick*. The basic principle of the data classification is separation of two groups by a hyperplane. When the hyperplane is known then the classification is done by a simple substitution inputs $\mathbf{x}$ into the formula of the trained hyperplane

$$y(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + b \tag{1.3.1}$$

where $y(\mathbf{x})$ is the distance from the hyperplane, $\mathbf{w}$ is the normal vector of the hyperplane, $\boldsymbol{\phi}(\mathbf{x})$ is a non-linear regular feature space mapping. The classification dependends on the sign of eq. (1.3.1).

This is the basic principle of all methods used in this paper, i.e Support vector machine, Relevant vector machine, Logistic sigmoid regression, Neural networks and many others. However, there exists a way how to improve this approach, if the training data points, or some subset of them, is used also in the prediction phase, then these data points are called *support vectors* and each of the support vectors are centers of *kernel functions*. This can be done only if the learning machine model (1.3.1) can be estimated in *dual representation* also called method of potentials:

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}') \tag{1.3.2}$$

This kernel is a scalar product and in the simplest case of mapping $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$ the result is kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}$ called *linear kernel*. The idea of the kernel methods is simple. If the learning method can be estimated using the scalar product kernel 1.3.2, then the scalar kernel can be replaced by some different type of kernel. Some basic types of kernels are in thr following table

$$k(\mathbf{x}, \mathbf{x}') = x^T x' \qquad \text{Linear kernel} \tag{1.3.3}$$

$$k(\mathbf{x}, \mathbf{x}') = (x^T x' + 1)^M \qquad \text{Polynomial kernel of M degree} \tag{1.3.4}$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(\|x - x'\|^2 / 2\sigma^2) \qquad \text{Gaussian kernel} \tag{1.3.5}$$

The purpose of the kernel is to estimate the similarity of vectors $\mathbf{x}, \mathbf{x}'$.

The linear kernel is a basic example of usage of the kernel method. Suppose that the normal vector $\mathbf{w}$ from eq. (1.3.1) can be expressed as a linear combination of a subset from training points $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x_i}^T$, then

$$y(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \mathbf{x_i}^T \mathbf{x} + b$$

with non-linear mapping $\boldsymbol{\phi}(\mathbf{x})$ it can be rewritten

$$y(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(\mathbf{x_i})^T \boldsymbol{\phi}(\mathbf{x}) + b$$

and finally after the kernel substitution

$$y(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x_i}) + b$$

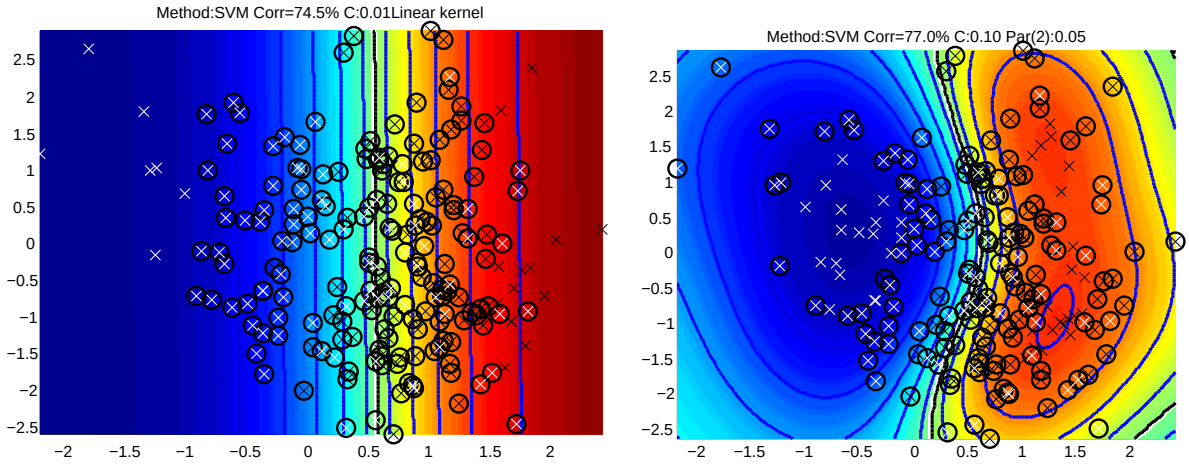


Figure 1.1: Difference between linear kernel (left) and RBF kernel (right). Prediction was done by Support vector machine algorithm. Points in circles are support vectors, both methods use the same dataset.

Advantage of the kernels is possibility to use of more general shapes of border (fig. 1.1) compared to nonkernel methods without significant increase of computational complexity [9]. Disadvantage is that there is no analytical equation of the border except the case of the linear kernel. The knowledge of the normal (weight) vector $\mathbf{w}$ of the linear hyperplane can be used to determine dependence of the model on different features, assuming that values in different dimensions were properly normalised (see Section3.4).

The next challenge is to choose the kernel suitable for the training data. The choice of the kernel and its parameters is often more important than the choice of the learning machine algorithm (see fig. 1.1). Typically, it is recommended [13] to use the linear kernel first. It usually gives the most stable results and has no kernel parameters to be tuned. The next advantage is more stable classification in high dimensional space when the number of samples is comparable with the number of dimensions. Gaussian and polynomial kernels are more flexible; on the other hand, the flexibility can lead to overfitting. If the number of dimensions is low and number of samples is much higher than dimension, then the Gaussian kernel is usually used. But it is necessary to determine the correct kernel parameter σ that corresponds to the size of the kernel. Usually, cross-validation introduced in the next section is used.

Chapter 2

Learning Machines

This work is mainly focused on the classification models that can also give the prediction probability conditioned on the observed data. The knowledge of the probability can help to determine incorrect values in the database (outliers) or can be used as feeding data for more advanced methods such as Bayesian networks that can connect many different prediction sources together and get some new information from the database.

The problem of probability predicting is ill-conditionality of this problem. Therefore, outliers and noise in the learning data can significantly influent the probability in case of small learning dataset.

2.1 Logistic Regression

Logistic regression is a simple classification method based on data regression by a logistic sigmoid. It is basically a transformed ordinary linear regression

$$y(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{m=1}^{M} w_m \boldsymbol{\phi}_m(\mathbf{x})\right) \tag{2.1.1}$$

The only difference is transformation of output by sigmoid function $\sigma(x) = 1/(1+\exp(x))$. The reason for sigmoid shape can be obtained from Bayesian theorem for class C_1

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)}$$

where C_1, C_2 are classes of data, after substitution

$$a = \ln \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)}$$

the probability can be rewritten

$$p(C_1|\mathbf{x}) = \frac{1}{1+\exp(a)} = \sigma(a)$$

for multiclass model, the probability can be written as

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln(p(\mathbf{x}|C_k)p(C_k))$. This function is called *softmax function.*

The logistic regression searches probability of class C_1 in shape

$$p(C_1|\mathbf{x}) = y(\boldsymbol{\phi}(\mathbf{x})) = \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}))$$

This principle is not used only for transformation of linear regression, it is used frequently used for transformation regression problem to classification. The same principle can be used, i.e. Neural networks or Relevance vector machines.

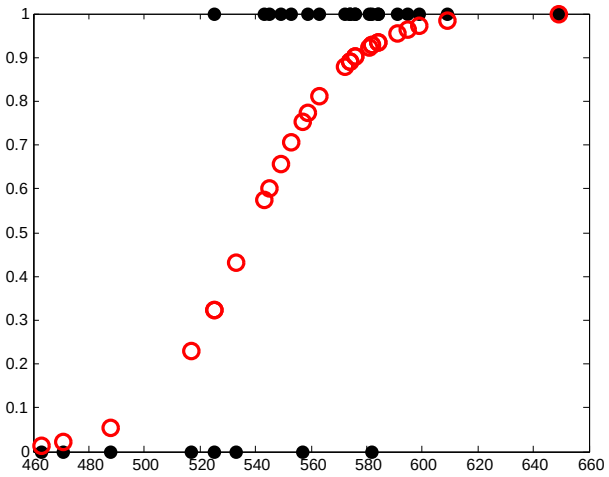


Figure 2.1: Logistic regression in 1D classification. Black points are training data, red points are predicted probability. Y-axe corresponds to probability.

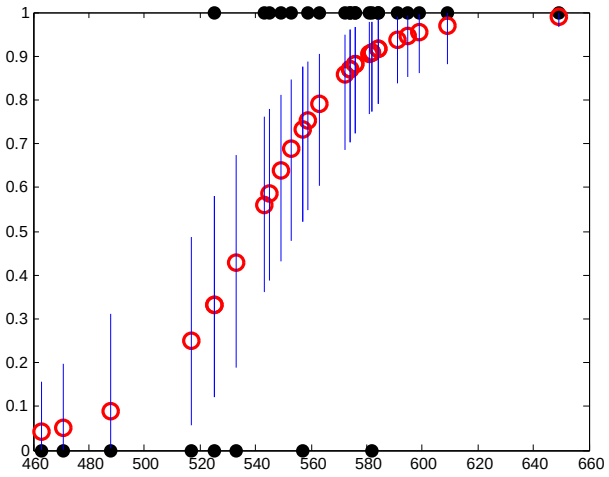


Figure 2.2: Logistic regression in 1D classification using full Bayesian approach. Black points are learning data, red points are predicted probability with blue errorbars. Y-axe corresponds to probability.

The joint probability of the observed data for labels $t \in \{0, 1\}$ and outputs are $y_n = \sigma(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_n}))$ can be estimated as

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1-t_n}$$

This "regression" problem (fig. 2.1) can be optimalised directly via maximum likelihood with iterative reweighted least squares [9, 12]. Disadvantages of this direct approach is possible overfitting if the data sets are linearly separable. In this case, the sigmoid gets infinitely steep in feature space. This can be prevented by adding some apriori assumption and finding the MAP solution. And the next problem is that it is originally nonkernel method.

The overfitting problem can be solved under assumption of a prior Gaussian distribution over weights $\mathbf{w}$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$$

The problem can be transformed to a standard optimisation MAP problem similar to the ridge regression in eq. (1.2.5).

In case that it is needed to use kernel, it is possible to define output y similarly to SVM machines (sec. 2.3)

$$y(\mathbf{x}) = \sum_{n=1}^{N} w_n k(\mathbf{x}, \mathbf{x_n}) + b$$

The output of the logistic regression is probability that the observed data $\mathbf{x}$ are in class C_k : $p(\mathbf{x}|C_k)$, but it is even possible to obtain precision of this prediction (fig. 2.2) if Bayesian rule and Laplace approximation of probability distribution is used. More details can be found in [9, 12].

2.1.1 Regularisation Methods for Logistic Regression

The loss function penalizing different values of model parameters $\mathbf{w}$ is used in two basic versions, so called L1-regularisation and L2-regularisation. These regularisers can be used generally but in this paper are used only for Logistic regression.

- L2-regularisation are ordinary least squares with assumption that distribution over parameters $\mathbf{w}$ is Gaussian $\mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$. The L2-regulariser was already used in the introduction Section 1.2

$$L(w) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \sigma(\mathbf{w}^T \mathbf{x_i}))^2 + \lambda \|\mathbf{w}\|_2^2$$

- L1-regularisation is basically similar, only there is L1-norm for model parameters $\mathbf{w}$.

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \sigma(\mathbf{w}^T\boldsymbol{\phi}(x_i)))^2 + \lambda\|\mathbf{w}\|_1$$

This model has an important property. If λ is sufficiently large, some of the parameters gets equal to zero and it leads to a sparse model very similar to the Relevance vector machine called *Sparse logistic regression* [12].

2.2 Relevance Vector Machine – RVM

Relevance vector machine is a probabilistic algorithm based on Bayesian rule. It was developed [7] to reach advantages of Support vector machine (Section 2.3) such as sparse model, kernels and to remove disadvantages like no native probability output and additional model parameters C or ν. Another advantage is that the model is usually much sparser than SVM model and thus the prediction can be faster (fig. 2.4)

RVM for data regression is basically derivated in Section 1.2, the main different to the logistic sigmoid are hyperparameters λ_i for each of the weight parameters w_i instead of single λ. The final marginal likelihood (evidence)

$$p(t_n^*|\mathbf{t}, \mathbf{x}, \boldsymbol{\lambda}, \sigma^2) = \int p(t_n^*|\mathbf{w}, \mathbf{x}, \sigma^2)p(\mathbf{w}|\boldsymbol{\lambda})\mathrm{d}\mathbf{w} \tag{2.2.1}$$

Now, only the hyperparameters λ_i and σ^2 must be determined. It can be obtained from the type II maximum likelihood method eq. (3.2). Complete derivation is in [10]. The most important property of RVM is that some parameters λ_i are almost infinitely peaked at zero and thus corresponding kernels can be removed. This is called ARD prior – *Automatic Relevance Determination* and it results to a very sparse model. The remaining vectors are called *Relevance vectors* .

In this paper, the implementation of RVM in MatLab was used – SparseBayes 2.0 [11] with a few speed improvements based on usage of sparse matrices.

2.3 Support Vector Machine – SVM

Support Vector Machine is a simple pattern recognition (classification) kernel based method introduced by Vapnik [14]. SVM is based on searching a hyperplane that separates two groups of data points. The SVM is kernel based methodsearched bordering hyperplane should separate the data and the data with the highest distance from the hyperplane (border). The SVM is kernel based method so the border between the groups can be very flexible with non-linear mapping of the hyperplane to the feature space but it is not probabilistic method. But it is possible to add a guess of the prediction probability [9, 15, 16]. This will be discussed later in the Section 2.3.1.

Maximisation of the distance from the border can be reformulated to maximisation of term $1/\|\mathbf{w}\|$ which is equivalent to minimizing $\|\mathbf{w}\|^2$. It results to following constrained

quadratic optimisation problem

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.3.1}$$

with constrains

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad i = 1, ..., N \tag{2.3.2}$$

It can be improved to allow data points to exceed the border with some penalty. This principe is called *soft margins*. New variables ξ_i must be introduced to measure this overlapping. And C is constants that controls the trade-off between data overlapping the border penalty and size of margin.

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \tag{2.3.3}$$

with constrains

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, ..., N, \quad \xi_i \geq 0 \tag{2.3.4}$$

These equations can be optimised directly but if the kernel representation of this equation used, the equation must be transformed to its dual representation. The final optimized equation is

$$L(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

where L is minimised Lagrangian, a_n are searched dual parameters, t_n are target values normalised to set $\{-1; 1\}$, input vectors $\mathbf{x}_n$ and $k(\mathbf{x}_n, \mathbf{x}_m)$ is value of kernel. Lagrangian L is minimised with respect to transformed constrains

$$0 \leq a_n \leq C_n$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

The prediction output of the SVM algorithm is proportional to the distance from the border

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x})$$

Only vectors $\mathbf{x_n}$ with nonzero parameters a_n are used for prediction and these vectors are called support vectors. Support vectors are always point at the border or exceeding the border. The rest of the points is ignored after the learning phase, therefore it leads to a sparse model.

SVM algorithm is not based on a probabilistic model, therefore the only known information about uncertainty of prediction is the distance from the hyperplane. But the distance is normalised to be one for the support vector laying on the border. Hence, the absolute value of distance from the border does not directly correspond to the uncertainty of the prediction.

2.3.1 Different Precision of Data Points

The original version of SVM algorithm expects that all points and also all classes have the same weight or precision. However, in practice it is not often satisfied.

Often it is needed to train algorithm with data where each group has significantly different number of members. In this case, the learning machine can completely ignore the smaller group. It is possible to choose different weights C_i eq. (2.3.3) for each class to prevent this issue. The weight can be selected as inverse number of data points in classes. The result is usually worse total classification score but better scores for the smaller group. Disadvantage can be also worse convergence of SVM algorithm (LinearSVM)

Another common problem is different precisions (errorbars) for each data point. The easiest solution is to use different weights C_i but now C_i must be different for each point. The weights can be selected as inverse points precision.

The main advantage of this solution is the fact that the weighting is fully supported by the LIBSVM library. On contrary, disadvantage is that the algorithm ignores space distribution of each data point with errorbars. It is not able to use the information about the space distribution from the errorbars about uncertainty of position of the given point.

The best solution of this problem is usage of the original data from which the errorbars were obtained. The SVM algorithm can process all the data and get more realistic results.

If the original data cannot be used, for example, they are not accessible or only precision of the diagnostics is known, another way can be used. Every data point with errobar can be replaced by a set of artificial points with random Gaussian distribution. Then of each the Gaussians corresponds to position original point and variance is set according to the known errorbars. For high number of the artificial data, the properties converge to the expected behaviour. As the result, the border will correctly reflect the data with errorbar. Disadvantage of this approach is increased number of training points for SVM algorithm and therefore decreasing solving speed.

2.3.2 Confidence Region

The main problem connected to probability estimation is determining confidence interval around border where the prediction of target value can be done with certain probability. In case of linear kernel, the exact equation of the bordering hyperplane can be found. In case of other kernels, border can be estimated by appropriate root searching algorithm.

The confidency intervals cannot be solved by SVM algorithm itself but in [15] was proposed method of fitting the outputs of trained SVM with the logistic sigmoid

$$p(x|C_1) = \sigma(Ay(\mathbf{x}) + B)$$

where $y(\mathbf{x})$ is distance from border (output of trained SVM algorithm) and values of parameters A, B are found by cross-entropy. Data used for fitting sigmoid should be independent of the data used for learning in order to avoid severe overfitting. Advantage is that this algorithm is already implemented in LIBSVM. LIBSVM uses 5-fold crossval-

idation [16] instead of new independent data to train the sigmoid, therefore probability prediction should not be used inside another cross-validation because it significantly delay the training phase. Moreover, SVM is not designed to estimate probability, therefore the results can be poor [9].

If is known probability estimations, it is easy to identify points, where the confidence of prediction is lower than 95%. The result for different size of errorbars is in figures 2.3, 2.4,2.5.

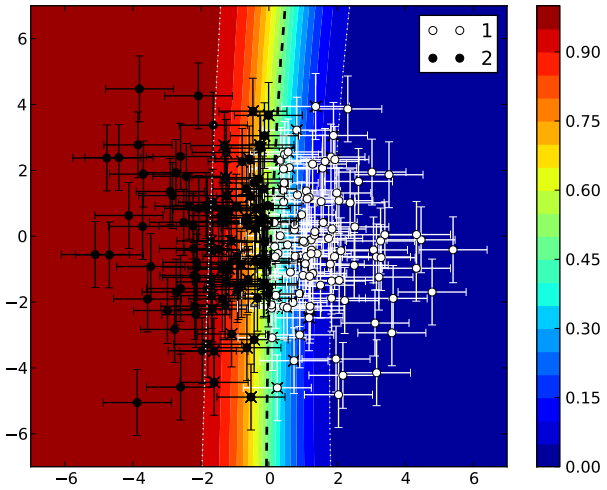


Figure 2.3: Border and probability distribution of data for binary classification. The original data were generated separable by border $x = 0$ and around each point was added artifical points with Gaussian distribution with variance $\sigma^2 = 1$.

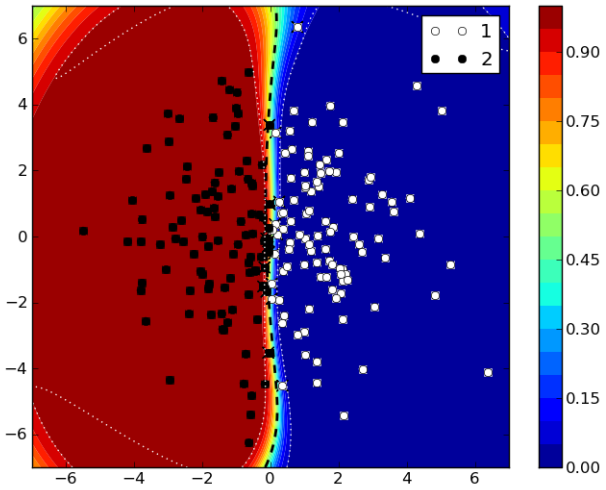


Figure 2.4: Border and probability distribution of data for binary classification. The original data were generated separable by border $x = 0$ and around each point was added artifical points with Gaussian distribution with variance $\sigma^2 = 0.01$.

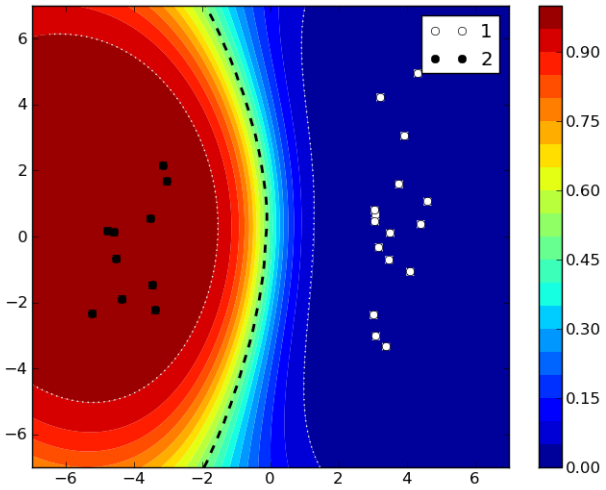


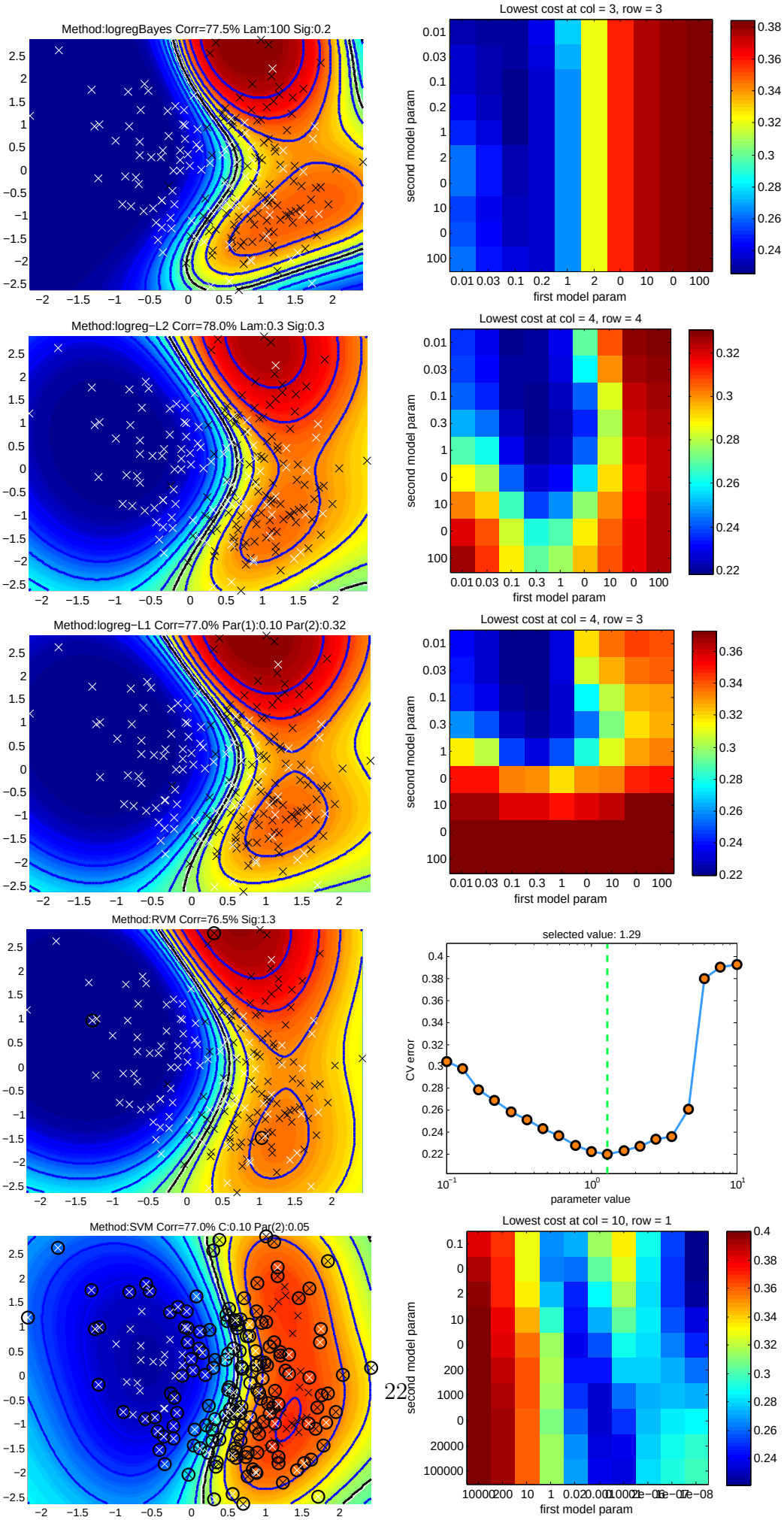
Figure 2.5: Border and probability distribution of data for binary classification. The original data were generated separable by border $x = 0$ and around each point was added artifical points with Gaussian distribution with variance $\sigma^2 = 1$. Distance between groups is 6.

2.4 Testing

Some important algorithms described in previous text are compared on the following figures. It is important to note a few things. The first, all the predictions are very similar. It is not so expectable, because the principe of all algorithms is quite different, first two models are dense (logreg-Bayes, logreg-l2), the rest are sparse, the first (logreg-Bayes) and the fourth (RVM) use full Bayesian probability, and finally the SVM maximize distance border instead of some probability fitting. Also, if you compare size of the kernel for different algorithms, it oscillates from 0.05 to 1.3. And even quite small change in the cross-validation (see Section 3.1) parameters significantly changes resulting probability although classification rate stays almost the same.

Testing data were loaded from `http://research.microsoft.com/~cmbishop/PRML/webdatasets/datasets.htm`. Data were generated from a mixture of 2 Gaussians. Points in cicles are support vectors.

Example of predictions with different learning machines and output of CV.



Chapter 3

Model Selection

The direct approach to model selection would be to choose the model that gives the best classification rate on training data according to some loss function. Unfortunately, this approach does not work, since the most complex model is always choosen, for example polynomial of the highest degree in regression. This solution has low predictive value and it is called overfitting. There exist two ways to avoid this problem: the Bayesian way and the non-Bayesian way.

3.1 Cross-Validation – CV

Crossvalidation (CV) is a non-Bayesian (frequentist) statistical method to avoid overfitting caused by choice of model parameters [17]. The aim of crossvalidation is to generalise estimated results for an independent data set. Bayesian methods should be able to choose the model parameters automatically [7]. However, it is valid only for regression (see Section 1.2), not for classification. So even for Bayesian methods the kernel parameters are determined by CV. Also the choice of the correct kernel is either based on some apriori knowledge or the kernel with best results in CV is used.

A common way to perform CV is to use a grid search in N_p-dimensional space for N_p parameters that gives the best prediction. Instead of grid search minimum searching algorithms can be used but they have problems with local minimums and also the grid can be easily parallelised to increase speed.

The training data are randomly separated to N fold, usually to 5-fold [12], $N-1$ folds are used for training and the last one is used for testing, but many different types of CV exists. The training process is repeated N-times for different combinations of learning/testing folds. The result of the CV is the mean classification score for given parameters. It is possible to use even the knowledge of the variance of the CV mean. So called "S1-rule" [12] can be used in case of flat bottom (fig 3.1) of the loss function. According to the S1-rule the best parameter is chosen from the set of parameters, whose score is better than best score minus one sigma with some apriori knowledge. For example the largest kernel can be preferred. The parameters from the grid that have the best mean score (or the best S1-rule) are finally selected.

The problem of CV is that it is a non-probabilistic (non Bayesian) method and all learning machines compute probability of predicted data under the assumption that the used kernel parameter is correct. Thus the probability prediction can be significantly different for different kernels and their parameters.

In this paper the used apriori knowledge for S1-rule is that larger Gaussian kernel size σ usually means less complex model. This knowledge usually leads to more reliable predictions.

The second problem of CV is connected to the way how the CV score is computed. So called *loss functions* are used to rate success of the models. A common way is to use a zero-one loss function that returns number of misfited points

$$L = \frac{1}{N} \sum_{y_i \neq \tilde{y}_i} 1$$

where $\tilde{y}_i$ is the predicted value and y_i is the expected value. If the sizes of classified groups are not similar, it is better to normalise the loss function in order to prefer the smaller group

$$L = \sum_{i \in Y} \sum_{j=1}^{N} \frac{y_{ij} \neq \tilde{y}_{ij}}{N_i}$$

The problem of this loss function is that it gives no importance to the probability prediction so the results can prefer all prediction near to 50% (binary classification – fig. 3.3). A better result gives function that penalises predictions according to their probability:

$$L = \sum_{y_i \neq \tilde{y}_i} P(x_i|C_1) + \sum_{y_i = \tilde{y}_i} (1 - P(x_i|C_1)) \tag{3.1.1}$$

where $P(x_i|C_1)$ is conditional probablility that data point x_i belongs to the first group in case of binary classsification. This loss function also gives less flat shape of the CV curve compared to the standard zero-one loss function (fig. 3.1)

3.2 Bayesian Model Selection

The cross-validation has several major drawbacks. The first and the most obvious is the necessarity to repeat the model training many times to find the best parameters, moreover the number of points in the grid grows exponentially with number of parameters. The usage of the grid also implies that the searched parameters are discredited to a finite number of values. Another problem of CV is estimation of the uncertainty in the selected parameters. It means that probability given by learning machines is based on assumption that parameters learned by CV are the only correct values.

A better approach is to use the Bayesian rule (1.2.8) and integrate out model parameters $\boldsymbol{\Theta}$ as is shown in Section 1.2. The total probability $p(y|y_0, C_k)$ that vector $\mathbf{x}$ is in class C_k

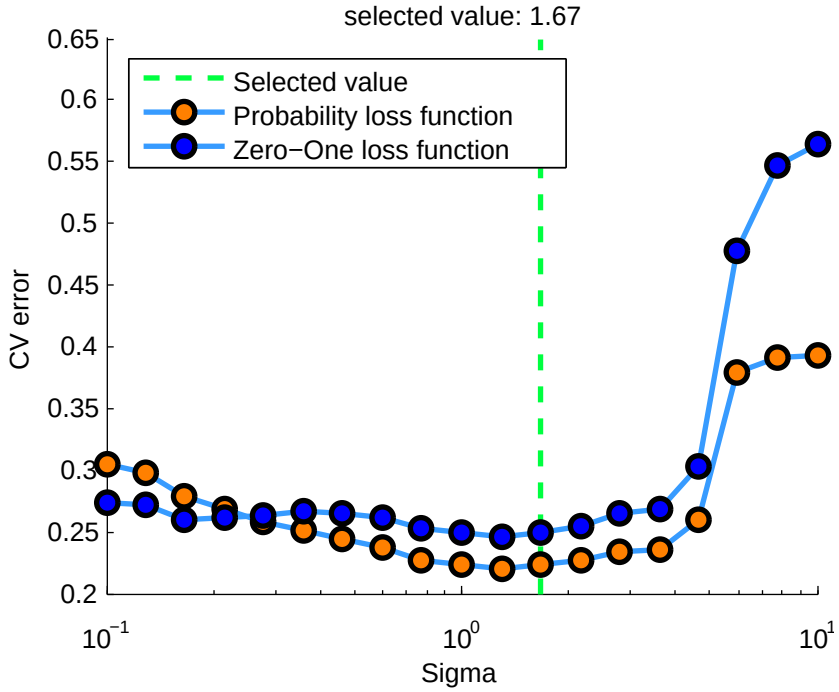


Figure 3.1: CV tuning of kernel size for the Relevance vector machine. The "S1-rule" was used. The blue points correspond to the ordinary zero-one loss function, the red points are probability loss function (3.1.1).

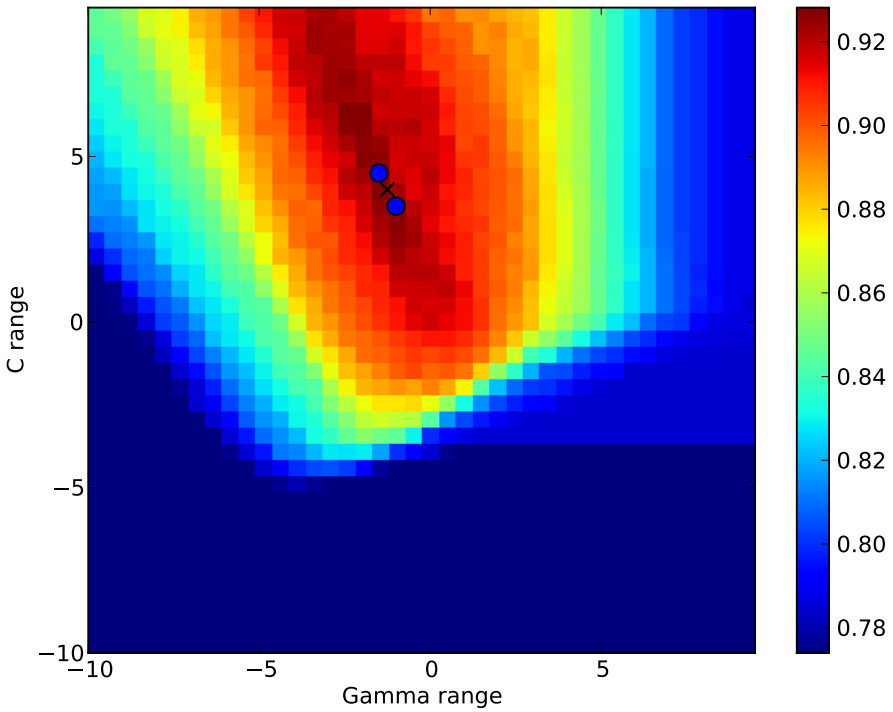


Figure 3.2: CV tuning of kernel parameters for Support vector machine. Loss function is $L = \frac{1}{N}\sum_{y_i \neq \tilde{y}_i} 1$. Best values are signed by blue circle.

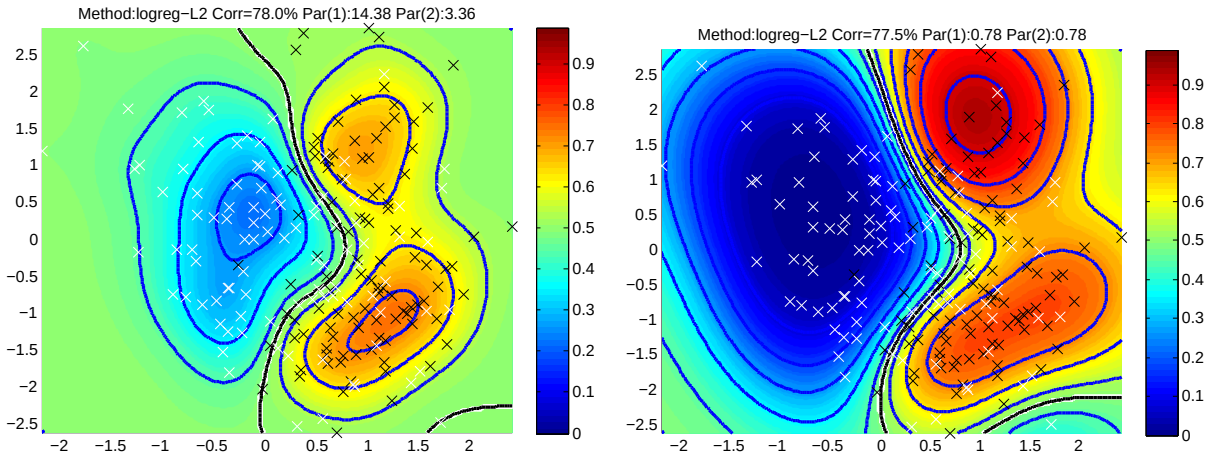


Figure 3.3: Difference of prediction for optimal parameters selected according zero-one loss function (left) and loss function based on probability (right). The prediction was done by L2 Logistic regression.

if y_0 are labelled training data, is generally

$$p(y|y_0, C_k) = \int p(y|\mathbf{\Theta})p(\mathbf{\Theta}|y_0, C_k)d\mathbf{\Theta}$$

The term $p(y|y_0, C_k)$ is marginal likelihood or sometimes also called *evidence* for class C_k.

The probability over searched parameters can be estimated directly from Bayes rule

$$p(\mathbf{\Theta}|y_0, C_k) = \frac{p(y|\mathbf{\Theta})p(\mathbf{\Theta}|y_0, C_k)}{\int p(y|\mathbf{\Theta})p(\mathbf{\Theta}|y_0, C_k)d\mathbf{\Theta}}$$

or it is possible to numerically optimise the likelihood $p(\mathbf{\Theta}|y_0, C_k)$.

In case that distribution $p(\mathbf{\Theta}|y_0, C_k)$ can be solved analytically over some set of parameters $\mathbf{\Theta}_i$ while for other parameters, for example kernel size σ, it cannot be directly solved, then it is possible to approximate σ by its ML value and use only distribution over the rest of parameters. This approach is called type-II maximum likelihood approximation and is used for RVM regression [10].

The great advantage of the Bayesian method is the possibility to integrate out the model parameters w and thus automatically avoid overfitting. It follows the principle of Occam's razor.

3.3 Dimension Reduction

It is important to note that even a lot of data do not guarantee that the overfitting will be avoided. The amount of the data must be measured relatively to the number of free parameters in the model. Usually the number of free parameters grows with the dimensionality of the problem and also with the choice of the kernel. The Gaussian kernel (RBF) is very flexible therefore it provides a huge number of free parameters. The linear

kernel is less flexible but still, in case of high dimensionality or insufficiency of data, it can reach overfitting. Another advantage of dimension reduction is possible better understanding of the physical context if only the most important variables are selected.

One way, how to avoid overfitting, is to use linear kernel and decrease dimensionality of the problem only to the most significant dimensions (properties of the problem). The only issue is to find the most relevant properties.

It is possible to perform "pruning" from an apriori knowledge or via learning machines. Many different algorithms were developed to provide some feature ranking and elimination [18].

A basic way is to use a single variable classifier and thus to classify individual predictive power of each dimension separately and choose dimensions with the best score. This method is called *univariate feature selection*. It statistically rank differences in distribution using ANOVA filter[1] and identifies the most relevant dimensions; the results are similar to the linear separation [19, 20]. Advantage of this attitude is good theoretical background, on the other hand, this can work only for special cases when the data are easily separable.

A bit more advanced method is to separate data by a linear hyperplane. In case that the data are correctly normalised then dimensions with lower absolute weight $\mathbf{w}$ are less relevant for the linear separation. This can be used to remove the least successful dimensions. Disadvantage is the expectation of linear separability. Advantage is quite fast selection compared to i.e. wrapppers described in following text.

Some learning methods can be improved to automaticly remove some dimensions, if l_0-norm or l_1-norm of weights $\mathbf{w}$ is used in the MAP solution. The l_0-norm prefers low number of non-zero weights. The number of variables ($N_p = \|\mathbf{w}\|_0$) can be added as next regularisation term to the optimised error size. But it needs changes in the learning machine algorithm. In paper [21] a simple method was proposed to solve l_0-norm approximately for the SVM algorithm. The process is the following:

1. Train linear SVM
2. Rescale the input data with their weights $\mathbf{w}$
3. Repeat until convergence

This procedure prunes variables approximately similary to the l_0-norm. Although, sometimes is enough to use only l_1-norm for example in the L1 logistic regression (Section 2.1).

3.3.1 Wrappers

Wrappers denotes class of powerful method for features elimination regardless the used learning algorithm. Wrappers are algorithms searching the optimal set of variables in feature space and propose different combinations of variables according to some loss function. This loss functions is usually some learning machine algorithm that is trained on the

[1] Basic statistical analysis of variance

data proposed by the wrapper and returns a number corresponding to the mean loss of the testing phase. The learning algorithm is used only as a black box and thus it is easy to implement such wrapper for many different learning machine algorithms. Therefore, wrappers use the predictive performance of the learning machines or other ranking algorithms such as ANOVA filter and only need to know how to interpret the result and how to search in space of all possible subsets of variables. The problem is that checking all subsets combinations is NP-hard[2]. For larger datasets it cannot be solved by a brute-force method and thus many strategies were developed [19]. For example genetic algorithms can be used [22].

However, usually forward selection and backward elimination algorithms are used. The basic principle of the forward selection is selection of feature according to their predictive power. Only the most important features are added to the training subset. On contrary, the backward elimination algorithm starts with all features and removes dimensions with the smallest additional information. Advantage of the backward elimination is better resistance to the variable correlation, on the other hand this algorithm is more computationally demanding.

These algorithms can search in possible subsets of variables, but it is also possible to rank each dimension (variable) separately according their influence to the total classification score and, again, remove only the least significant.

Great advantage of the wrappers is their universality and also, compared to the previous methods, possibility to deal with non-linear feature dependencies if they are supported by the used learning machine and also easy possibility to handle multi-class problem. Disadvantage is generally higher demand on the computing power.

3.3.2 Embedded Methods

Some methods have built-in selection of the most relevant features. One example is decision trees algorithm – CART (Classification And Regression Tree). Tree methods need to decide which feature split the data best and sort layers in the tree according to this. Disadvantage of the CART method is that it searches only classification performance separately for each dimension.

3.3.3 Model Validation

The last step of variable elimination is to check if the number of selected dimensions is optimal or more dimensions should be removed. If pruning leads to better predictive results (classification score) then the algorithm can continue. But even if pruning decrease predictive accuracy it still can improve generalisation of the predictions and remove overfitting.

One way is to repeatedly randomly permutate labels of the data and perform classification. The statistical p-value of the original classification is equal to percentage of random permutations that have higher classification score then the score of the original data. The

[2]the complexity is higher than polynomial

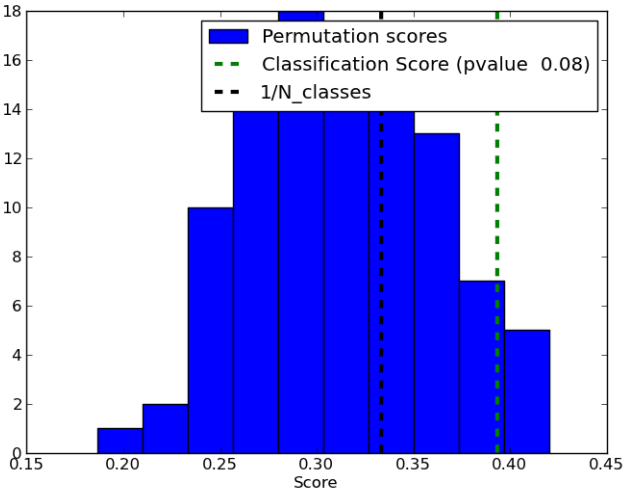


Figure 3.4: Histogram of relevance determination using random permutations of labels

final p-value should be almost zero (fig. 3.4) This approach can give only the upper limit for the number of dimensions and also it can be used only for smaller data sets because of high demand on computing performance.

The classification success also must be compared relatively to random selection. It means that classification score 60% for binary classification is worse than 60% for ten classes classification.

The next problem that must be considered is correlation between dimensions. It is possible that two dimensions can have good predictive power but the information they add to the system is redundant. Some feature selection algorithms can choose both these dimensions although it do not improve total prediction power.

One way to solve this is to use correlation analysis to detect these redundant dimensions. The second way is to use another elimination algorithm for example the recursive feature elimination that should completely ignore this problem.

Another interesting method of model validation uses artificial "fake" dimensions with random Gaussian noise. These dimensions in the training data set are used as a probe. If weight of some "real" variable is lower or similar to weights of the probes then this dimension can be removed. This method can be futher improved if random permutation of a real variable is used as fake variable instead of a Gaussian noise.

3.4 Scaling

Kernel learning algorithms are not generally independent of scaling and translation. Some kernels are invariant to translation, i.e. $k(x, x') = c(x - x')$, some kernels depend only on the magnitude (e.g. *radial basis functions* – RBF). Linear kernel is not invariant to the translation or scaling, but the prediction stays the same, if all dimensions are scaled or shifted with the same number. RBF kernel is invariant to a translation and also scaling

in case that σ is scaled too as denoted in this equation:

$$k(a\mathbf{x}+b, a\mathbf{x}'+b) = \exp(-a^2\|\mathbf{x}-\mathbf{x}'\|^2/\sigma^2)$$

The size of kernel is the only element that depends directly on the scale of inputs x, the rest of SVM algorithm is independent. This is true only if the scaling constant a is the same for all dimensions.

It is commonly recommended [9] to rescale data to be more similar in all dimensions. However, it is important to normalise the training and the testing data with the same number. Either normalisation in form $(x_i - \max(x))/(\max(x) - \min(x))$ or normalisation to reach zero mean and standard deviation equal one is usually used used. Latter normalisation is more robust to outliers. In order to completely avoid to outliers problem it is possible to use robust statistics and median instead if mean and median absolute deviation instead of standard deviation. The final normalisation would be

$$\tilde{X}_k = \frac{X_k - \text{median}_i(X_i)}{\text{median}_i(\|X_i - \text{median}_j(X_j)\|)} \tag{3.4.1}$$

If the data have errorbars, it is necessary to scale them too. The purpose of the scaling for linear kernel is mainly to reach the same importance for all dimensions. On contrary, for RBF kernel the aim is to achieve similar distance between data points in all dimensions.

The normalisation of the data to unit vectors reduces dimensionality by one since the data are projected on the unit sphere. This can cause problem in analyses of low dimensional data.

3.5 Data Preprocessing

Creating good data representation is the art of the learning machines. All learning machines algorithms described in this paper expects properties as amplitude, therefore all features must be converted to the intensity. It is also possible to include an apriori knowlendge to the feature extraction, i.e. exctract frequency via wavelets transformation, size of derivation, variation of data or many others features. Also removing outliers from data and performing data smoothing in case of time depence data can improve the prediction.

It is also possible to use some generic feature extraction methods such as clustering or singular value decomposition to increase orthogonality of the data.

Chapter 4

Results

All learning machines described in previous sections were applied on data set based on database of shots from tokamak Golem but some algorithms for feature elimination were used only for SVM.

4.1 GOLEM database

The Golem database is used to determine probability of breakdown from parameters that were set up before the shot. These parameters are

Shortcut	Description	Range
H2	Enabled H2 filling	0,1
U_b	charge of capacitors of magnetic field	0-800 V
U_{cd}	charge of capacitors of current drive	0-1200 V
U_{bd}	charge of capacitors of breakdown field	0-500 V
U_{st}	charge of capacitors of vertical stabilisation field	0-500 V
P	pressure of H2	0-250 mPa
T_{cd}	delay of current drive trigger to main trigger	0-10 ms
T_{bd}	delay of breakdown trigger to main trigger	0-10 ms
T_{st}	delay of stabilisation field trigger to main trigger	0-10 ms
PreIonisation	Preionisation electrode	0,1

The reason, why was used this model, is that it shows many problems described in previous chapters while the problem can be quite easily analysed because it is less complex.

The first problem was the high number of dimensions and their correlations. It is important to remove less important dimensions to avoid overfitting. Here, it could be done from apriori knowledge. The first property H2 filling is necessary condition to make breakdown thus it can be removed and also all shots when there was breakdown without filling gas, because it is only a minor unimportant group – outliers.

The next step was to join some dimensions to ensure the independence of variables. In this case U_b plus T_{cd} or U_b plus T_{bd} gives magnetic field in time of maximal current drive

or breakdown field. Also instead of absolute values of T_{cd} and T_{bd} is important only their time shift, thus only the difference should be used.

Also stabilisation should not have any significant effect on the probability of breakdown although from tests in this section proved that the affect was quite strong. In this case, the explanation is quite easy: the stabilisation was used when the aim was to make some nice plasma. The result is a non-physical correlation, so the stabilisation settings should be also removed. Totally, only six dimensions from original 10 should rest.

The results of SVM algorithm with RBF kernel before this elimination of features by apriori knowledge are

Group	Precision	Recall	F1-score	CV score	N_{SV}	N_{VEC}
0	0.834	0.627	0.648	0.338	265	367
1	0.885	0.990	0.467	0.786	230	1064
Total:	0.858					
Total SV:	34.591%					
Best parameters:	C=1000	G=0.005				

and after the these changes were the results

Group	Precision	Recall	F1-score	N_{SV}
0	0.93	0.67	0.78	302
1	0.91	0.99	0.95	1034
avg / total	0.92	0.91	0.91	1336

It is important to notice two things. The first and the most important is that the dimensions cut-off can lead to worse results in the total precision. This was expected and it even quite successful result when the dimensionality of the problem was decreased by 40% and the prediction probability rest almost the same.

The second issue is common property of described learning algorithms; they usually give worse results to the smaller class. In this case, it was group 0 (*no breakdown*), because breakdowns were usually required. Numbers of shots with breakdown were 1034 and shots without breakdown were only 302. This disproportion should be compensated. The ways, how to compensate it for SVM, were already described in the section 2.3.1.

Results for the same dataset with the linear kernel and SVM method are much worse. **The total precision decreased to 80%** [check] but it is important to notice that value 50% is only random guess. This problem is caused by linear non-separability of the data, as is shown in Fig. 4.1.

4.2 Random probes

Methods of random probes introduced in Section 3.3.3 was used. Three probes were created, the first probe *test_1* was random permutation of numbers from 1 to number of

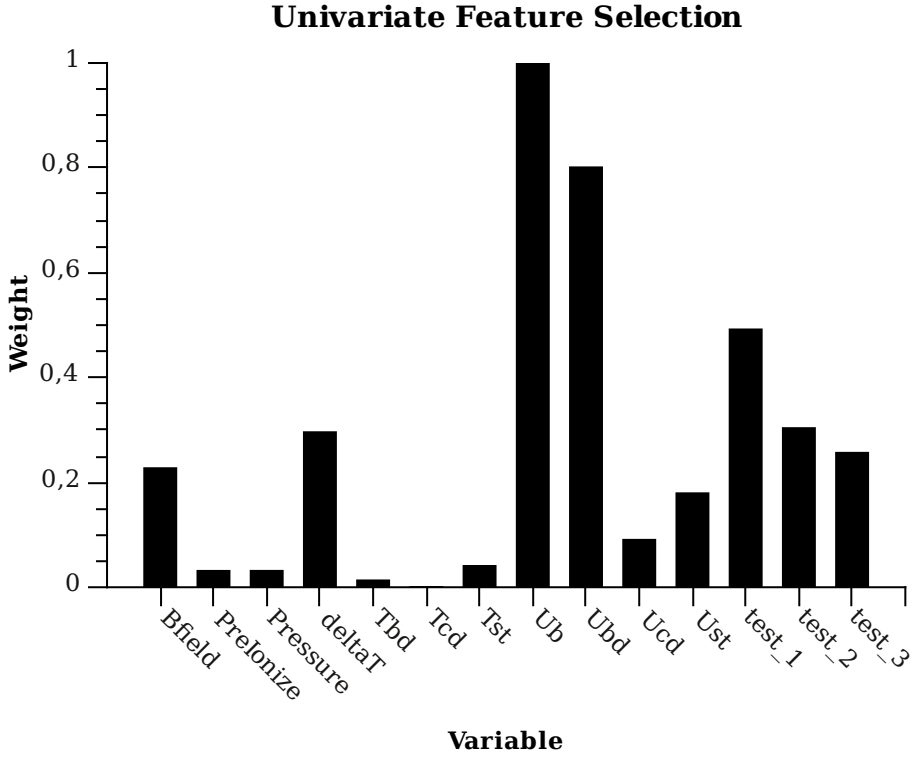


Figure 4.1: Results of univariate feature elimination with ANOVA.

samples, the second *test_2* was random permutation of U_b and the last *test_3* was random permutation of U_cd. These probes should have the weight much lower than the rest of valid variables.

4.3 Univariate feature elimination

The first tested method was univariate feature elimination based on statistical ANOVA test. ANOVA filter returns weight corresponding to statistical difference between groups for given variable. Great advantage was almost immediate speed of filtering. Disadvantage is quite poor performance compared to other methods. The resulting weights are in fig. 4.1. According to the ANOVA are the random probes one of the most important variables and on the other hand pressure is negligible. This is significantly different from the expected results. It is caused because the classes in database are not linearly separable variable. For example, the breakdown fails for too high pressure and also for too low pressure.

The weights of the random probes could be caused by unsatisfied assumptions of the ANOVA model, i.e. independence of cases, normality of the residuals, homogeneity – variance of the groups should be the same.

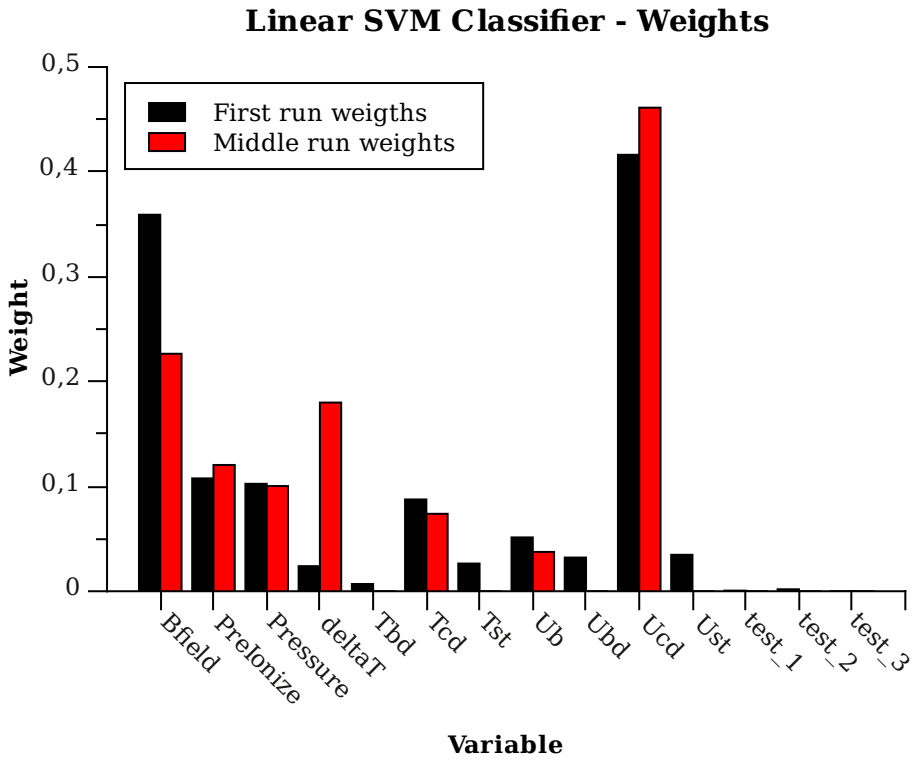


Figure 4.2: Recursive feature elimination with linear SVM. The black bars are weights for all variables, the red bars are weights for seven the most important variables.

4.4 Recursive feature elimination with linear SVM

The next tested method is based on recursive feature elimination (RFE) with linear support vector machine, but the results are similar for all described methods with linear kernel. The solving speed was slower than ANOVA filter but still it took only a few minutes. The weights are normalised absolute values of the normal vector of the SVM linear hyperplane. It is possible to make prediction for all variables and use only n of the most important or remove only the least important and perform the training and prediction again. The first way is faster but the second way gives result that should be less affected by random variable that were removed in previous steps.

The weights for linear classifier are in fig. 4.2. The results are significantly different from the ANOVA weights. Linear weighting successfully ignored random probes. Also it is important to note that the weight stayed almost similar during elimination, the only exception is ΔT variable.

In the following table are basic statistical characteristics of the fitted model. Recall for the group 0 is very low; it means that linear model almost ignored the smaller group. But it was expected since the model is not linearly separable.

Group	Precision	Recall	F1-score	CVscore
0	0.77	0.29	0.21	0.39
1	0.80	0.97	0.44	0.78
Total:	0.80			

The next step is to use the weights from linear hyperplane estimate the influence of each dimension.

keys:	Bfield	PreIon	P	ΔT	T_{bd}	T_{cd}	U_{bd}	U_{cd}
weights:	0.34	0.34	0.33	0.12	0.08	-0.13	-0.20	0.53

Almost all variables increased breakdown probability with increasing variable size, the only interesting exception is U_{bd}. However, the breakdown field should always improve the chance to reach breakdown. This was some non-physical effect.

4.5 Recursive feature elimination with RBF SVM

This method was also described in Section 3.3.1. The weights correspond to increase of loss function when the belonging variable was removed. Full crossvalidation was performed to find the best parameters for each variable combination. It was also possible to use the results from the crossvalidation to estimate errorbars for the weights. Disadvantage was that the training (elimination) was very time demanding. The full elimination took almost 1000 h of CPU usage.

The advantage is that it is fully nonlinear method and thus the weights are not biased by assumption of linear separability.

The results are in fig. 4.3. The results fit very well to the expected behaviour. Weights of random probes are not zero, but within the errorbars they are correct. Also, on the contrary to the other method, pressure was determined as very important variable with magnetic field and current drive. The fact that parameters of stabilisation are not zero was described in previous text, but still it is quite low. It is also quite interesting that ΔT, delay between breakdown and current drive, have no importance although linear SVM and ANOVA filter predicted quite high importance to this variable.

In Figure 4.4 is order of eliminated variables. The variables eliminated at beginning have very similar weights, so the order is not important. Interesting is that probe variable *test_3* alive quite long.

Finally, the evolution of predictive probability for each step of RFE algorithm is on fig. 4.5. During the first four steps the prediction is almost constant then the classification rate decreased although the *test_3* variable was still in training set, so it was still overfitting.

The optimal number of variables is five or six, it depends on allowing of T_{st}, U_{st} although they have no real information about probability.

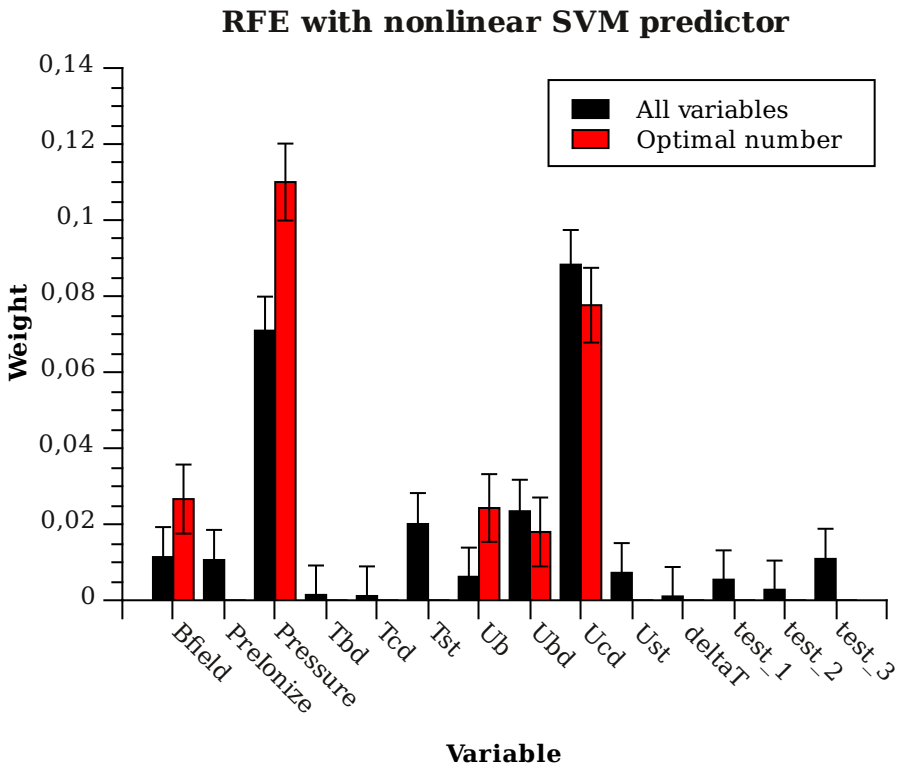


Figure 4.3: RFE with nonlinear SVM predictor

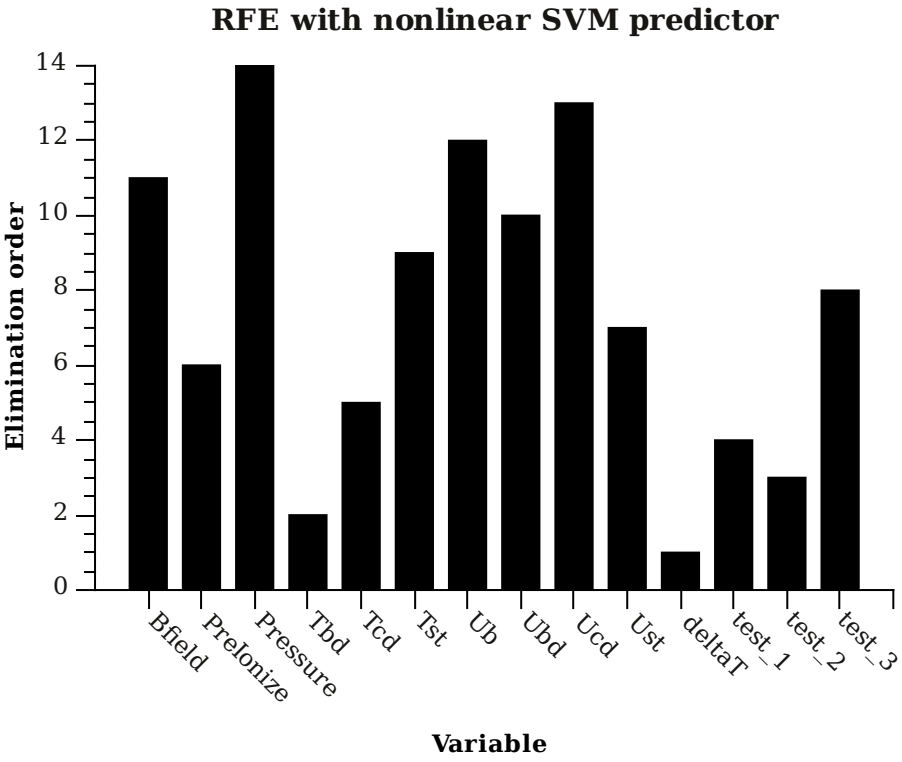


Figure 4.4: Elimination order of all variables by RFE with nonlinear SVM

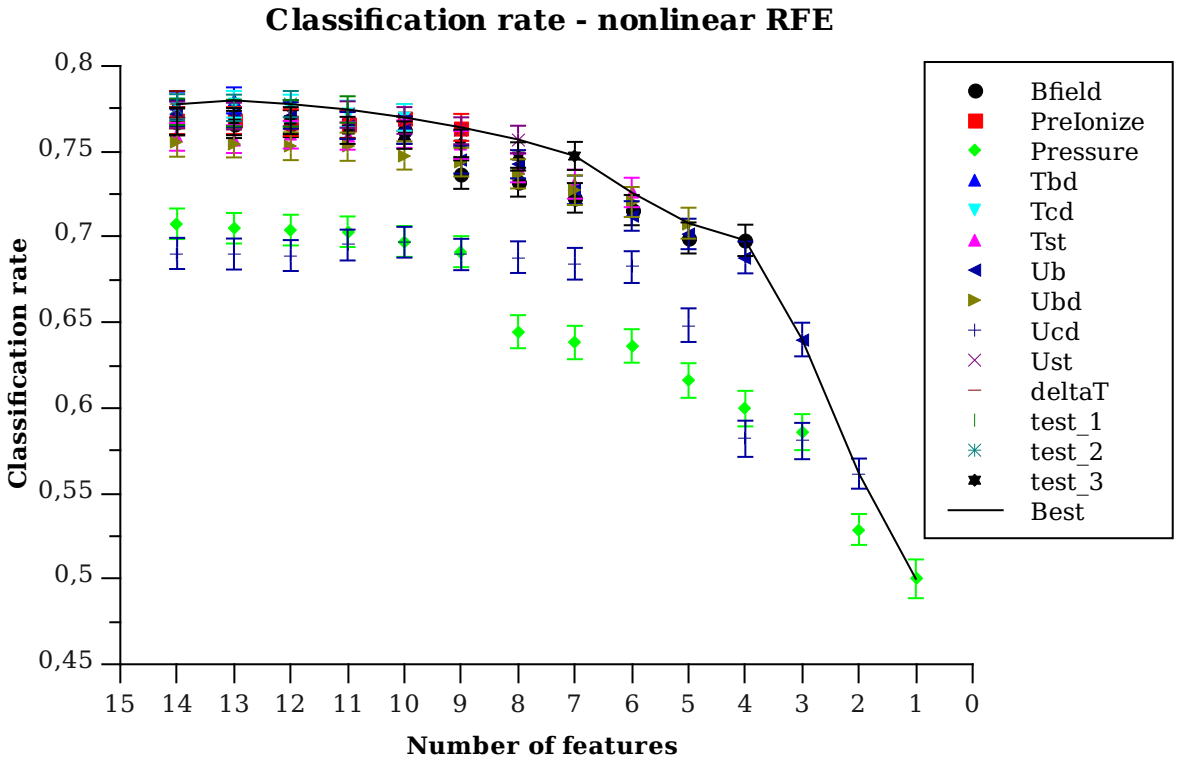


Figure 4.5: Classification rate for different number of variables with nonlinear SVM

In following table are results for different learning machines when only four dimensions were used. The differences between the misclassification are not significant for different machines because random changes between each run can produce different result even for on type of learning machine. Important property is sparsity of the model. The sparsest and thus usually the less complex model is the RVM. Interesting is that linear SVM have less sparse model but it is caused by linear nonseparability of the model and thus by high misclassification rate.

Machine	Total Missfit	Main vectors
RBF kernel - 4 dimensions		
SVM	0.941	34%
LogReg L1	0.926	100%
LogReg L2	0.959	100%
RVM	0.912	5%
linear kernel - all dimensions		
SVM	0.67	46%

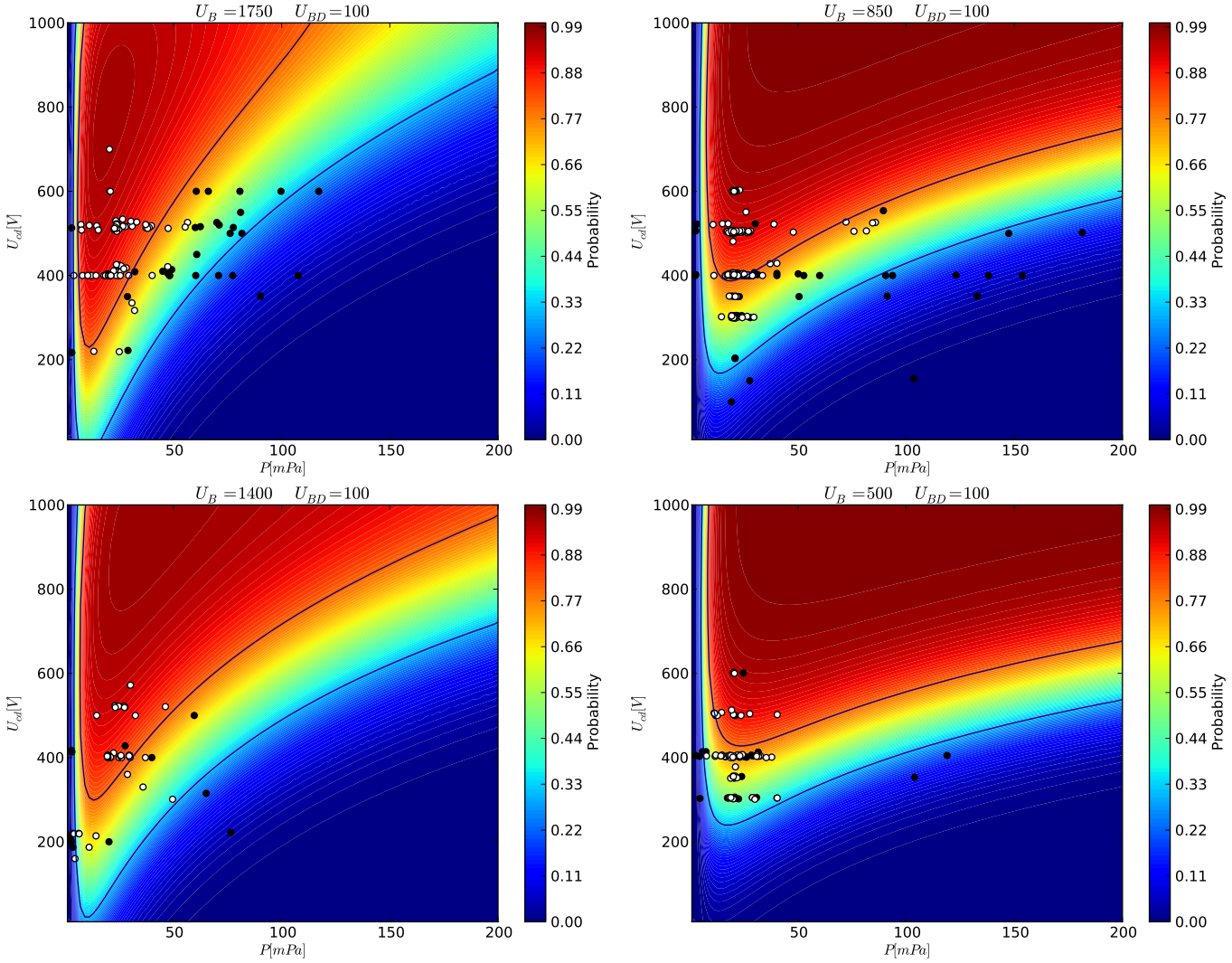


Table 4.1: Cuts through the predicted probability for breakdown on tokamak GOLEM, the prediction is output of nonlinear SVM algorithm. Black points are shots without breakdown and white points are with breakdown. Contour lines sign 30% and 80% decision border. It should be noted that the curve shape is very similar to Paschen's curve.

Summary

In this work, learning algorithms based on kernels were successfully tested: Support vector machine, Relevance vector machine and Logistic regression with L1 and L2 norm. These algorithms were applied on the database of breakdowns from GOLEM tokamak. Many methods to improve the classification reliability were introduced and applied to the dataset. Finally, the best parameters were selected and best pre-processing applied to obtain the probability of the breakdown.

It is possible to continue in this work by application of the tested principles to some bigger database. Another possibility to continue is implementation knowledge of data uncertainty to the learning machines to improve reliability.

Bibliography

[1] E. Ronchi, S. Conroy, E. Andersson Sundén, G. Ericsson, M. Gatu Johnson, C. Hellesen, H. Sjostrand, M. Weiszflog, and C. JET-EFDA. Neural networks based neutron emissivity tomography at jet with real-time capabilities. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 613(2):295–303, 2010.

[2] J. Vega, A. Murari, G. Vagliasindi, and GA Rattá. Automated estimation of l/h transition times at jet by combining bayesian statistics and support vector machines. *Nuclear Fusion*, 49:085023, 2009.

[3] S. González, J. Vega, A. Murari, A. Pereira, JM Ramírez, S. Dormido-Canto, and J.E.T.E. Contributors. Support vector machine-based feature extractor for l/h transitions in jet. *Review of Scientific Instruments*, 81:10E123, 2010.

[4] B. Cannas, A. Fanni, E. Marongiu, and P. Sonato. Disruption forecasting at jet using neural networks. *Nuclear fusion*, 44:68, 2004.

[5] K.P. Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, Citeseer, 2002.

[6] K. Pelckmans, J. De Brabanter, JAK Suykens, and B. De Moor. Handling missing values in support vector machine classifiers. *Neural Networks*, 18(5-6):684–692, 2005.

[7] M.E. Tipping. Sparse Bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244, 2001.

[8] C.M. Bishop and M.E. Tipping. Variational relevance vector machines. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, page 46–53. Citeseer, 2000.

[9] C.M. Bishop and SpringerLink (Online service). *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[10] M.E. Tipping. Bayesian inference: An introduction to principles and practice in machine learning. *Advanced lectures on machine Learning*, page 41–62, 2004.

[11] M.E. Tipping. An Efficient Matlab Implementation of the Sparse Bayesian Modelling Algorithm (Version 2.0). 2009.

[12] Murphy. *Machine Learning: a Probabilistic Perspective*. not yet published.

[13] C.W. Hsu, C.C. Chang, C.J. Lin, et al. A practical guide to support vector classification, 2003.

[14] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000.

[15] J. Platt. Probabilistic outputs for support vector machines. *Bartlett P. Schoelkopf B. Schurmans D. Smola, AJ, editor, Advances in Large Margin Classifiers*, page 61–74.

[16] H.T. Lin, C.J. Lin, and R.C. Weng. A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276, 2007.

[17] W. Martinez. Computational statistics handbook with MATLAB. 2001.

[18] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[19] Y.W. Chang and C.J. Lin. Feature ranking using linear svm. In *JMLR Workshop and Conference Proceedings: Causation and Prediction Challenge*, volume 3, page 53–64. Citeseer, 2008.

[20] S. González, J. Vega, A. Murari, A. Pereira, JM Ramírez, S. Dormido-Canto, et al. Svm-based feature extractor for l/h transitions in jet.

[21] J. Weston, A. Elisseeff, B. Scholkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *The Journal of Machine Learning Research*, 3:1439–1461, 2003.

[22] GA Rattá, J. Vega, A. Murari, et al. Improved feature selection based on genetic algorithms for real time disruption prediction at jet. *Plasma Phys. Control. Fusion, submitted for publication*.