# SVM in Oracle Database 10g: Removing the Barriers to Widespread Adoption of Support Vector Machines

Boriana L. Milenova

Data Mining Technologies
Oracle

boriana.milenova@oracle.com

Joseph S. Yarmus

Data Mining Technologies
Oracle

joseph.yarmus@oracle.com

Marcos M. Campos

Data Mining Technologies
Oracle

marcos.m.campos@oracle.com

## Abstract

Contemporary commercial databases are placing an increased emphasis on analytic capabilities. Data mining technology has become crucial in enabling the analysis of large volumes of data. Modern data mining techniques have been shown to have high accuracy and good generalization to novel data. However, achieving results of good quality often requires high levels of user expertise. Support Vector Machines (SVM) is a powerful state-of-the-art data mining algorithm that can address problems not amenable to traditional statistical analysis. Nevertheless, its adoption remains limited due to methodological complexities, scalability challenges, and scarcity of production quality SVM implementations. This paper describes Oracle's implementation of SVM where the primary focus lies on ease of use and scalability while maintaining high performance accuracy. SVM is fully integrated within the Oracle database framework and thus can be easily leveraged in a variety of deployment scenarios.

## 1. Introduction

Data mining is an analytic technology of growing importance as large amounts of data are collected in government and industry databases. While databases traditionally excel at data retrieval, data mining poses new challenges. Successful applications of data mining technology usually require complex methodologies and

'hands-on' involvement of data mining analysts. In addition, data mining is typically a computationally intensive activity that requires significant dedicated system resources. Support Vector Machines (SVM) [1] is a typical example of a data mining algorithm that can produce results of very good quality when used by an expert and given sufficient system resources. In fact, SVM has emerged as the algorithm of choice for modeling challenging high-dimensional data where other techniques under-perform. Example domains include text mining [2], image mining [3], bioinformatics [4], and information fusion [5]. In comparison studies, SVM performance has been shown to be superior to the performance of algorithms like decision trees, neural networks, and Bayesian approaches [2, 4, 5, 6].

The success of SVM is largely attributed to its strong theoretical foundations based on the Vapnik-Chervonenkis (VC) theory [1]. The algorithm's regularization properties ensure good generalization to novel data. There are, however, certain limitations inherent in the standard SVM framework that decrease the algorithm's practical usability:

- out-of-the-box performance is often unsatisfactory – SVM parameter tuning and data preparation are usually required;
- scalability with number of records is poor (quadratic); and
- non-linear models can grow very large in size, making scoring impractically slow.

This paper describes how these challenges have been addressed in Oracle's SVM implementation. Most design solutions are focused on improved usability and making SVM accessible to database users with limited data mining expertise. The paper demonstrates that these goals can be achieved without compromising the integrity of the SVM framework. Here the focus is on new techniques augmenting the current best practices in order to achieve the scalability and usability expected in a production quality system. This paper is *not* a primer on how to implement a standard SVM. Excellent discussions on

standard SVM implementation approaches and data structure representations can be found in a number of sources [7, 8, 9, 10, 11].

To our knowledge, Oracle's SVM is the only SVM production quality implementation available in a database product. As an alternative, one can consider stand-alone academic SVM implementations (e.g., LIBSVM [8], SVMLight [9], SVMTorch [10], HeroSVM [12]). However, such tools have limited appeal in the context of real-world applications. Oracle's full integration of SVM into the database infrastructure offers a number of advantages, including:

- data security and integrity during the entire mining process;
- distributed processing and high system availability;
- centralized view of the data and data transformation capabilities; and
- flexible model deployment including scheduling of model builds and model deployments.

The SVM feature is part of the Oracle Data Mining (ODM) product [13]. ODM is an option to the Oracle Database Enterprise Edition. ODM supports all major data mining activities: association, attribute importance, classification, clustering, feature extraction, and regression. ODM's capabilities can be accessed through Java or PL/SQL APIs or a graphical user interface (Oracle Data Miner).

The ease of use and good scalability of the ODM SVM feature has been corroborated by independent evaluation studies conducted at the University of Rhode Island [14] and the University of Genoa [15]. ODM SVM has been also leveraged as part of production systems used by several government agencies to tackle challenging data mining problems. Independent software vendors such as SPSS and InforSense have integrated the ODM SVM feature into their data mining products (SPSS Clementine, InforSense KDE Oracle Edition).

This paper is organized as follows. Section 2 introduces key SVM concepts. Section 3 discusses our improvements to the quality of out-of-the-box SVM models, in particular parameter selection, and provides comparisons with previously published results. Section 4 describes our solution to SVM's scalability problems and relates it to alternative approaches in the field. Section 5 discusses SVM's functional integration with other Oracle database components. Section 6 presents the conclusions and directions for future work.

## 2. Support Vector Machines

The SVM algorithm is capable of constructing models that are complex enough to solve difficult real world problems. At the same time, SVM models have a simple functional form and are amenable to theoretical analysis. SVM includes, as special cases, a large class of neural networks, radial basis functions, and polynomial classifiers. The output of an SVM binary classification model is given by:

$$f_i = b + \sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) \qquad (1),$$

where $f_i$ (also called margin) is the distance of each point (data record) to the decision hyperplane defined by setting $f_i = 0$; $b$ is the intercept; $\alpha_j$ is the Lagrangian multiplier for the $j^{\text{th}}$ training data record $\mathbf{x}_j$; and $y_j$ is the corresponding target value ($\pm 1$). Eq. 1 is a linear equation in the space of attributes $\theta_j = K(\mathbf{x}_j, \mathbf{x}_i)$. The kernel function, $K$, can be linear or non-linear. If $K$ is a linear kernel, Eq. 1 reduces to a linear equation in the space of the original attributes $\mathbf{x}$. If $K$ is a non-linear kernel, Eq. 1 defines a linear equation on a new set of attributes. The number of new attributes in the kernel-induced space can be as many as the number of rows in the training data. This is one of the key aspects of SVM and accounts for both its simplicity and power. For the non-linear case, while there is a linear relationship in the new set of attributes $\theta$, the relationship in the original input space is non-linear.

The kernel functions, $K$, can be interpreted as pattern detectors or basis functions. Each kernel used in Eq. 1 is associated with a specific input record in the training data. In general, in an SVM model only a subset of $\alpha_j$ are non-zero. The input records with non-zero $\alpha_j$ are called support vectors. Learning in SVM is the process of estimating the values of $\alpha_j$. This is accomplished by solving a quadratic optimization problem. The objective function is constructed in such a way that the trade-off between model complexity and accuracy in the training data is controlled in a principled fashion. Such balancing of model capacity versus accuracy, also known as regularization, is a key aspect of the SVM algorithm. This feature is responsible for SVM's good generalization on new data.

The SVM formulation discussed so far addresses a binary target classification problem. In classification problems with more than two target classes, there are two common approaches: 1) one-versus-all where a separate binary SVM model is built for each class; and 2) one-versus-one where a binary SVM model is built for each pair of classes. The second approach is considered more accurate, however, it involves building more SVM models. If there is a limited amount of data, one-versus-all is the preferred strategy.

Standard binary supervised classification algorithms require the presence of both positive and negative examples of a target class. The negative examples are often referred to as counterexamples. In contrast, one-class SVM is a classification algorithm that requires only the presence of examples of a single target class. The model learns to discriminate between the known members of the positive class and the unknown non-members.

One-class SVM models [16] can be easily cast as outlier detectors – the typical examples in a distribution are separated from the atypical (outlier) examples. The

distance from the separating plane (decision boundary) indicates how typical a given record is with respect to the distribution of the training data. Outliers can be ranked based on their margins. In addition to outlier detection, one-class SVM models can be used to discriminate the known class from the universe of unknown counterexamples. This is appropriate in domains where it is a challenge to provide a useful and representative set of counterexamples. The problem exists mostly in cases where the target of interest is easily identifiable but the counterexamples are either hard to specify, expensive to collect, or extremely heterogeneous.

In addition to classification targets, SVM can also handle targets with a continuous range. The SVM model learns a linear regression function. In the linear kernel case, the linear function is defined in the original input space. In the non-linear kernel case, the linear function is defined in the transformed, kernel-induced space. The latter is equivalent to learning a non-linear function in the original input space.

SVM regression uses an $\varepsilon$-insensitive loss function, that is, the loss function ignores values that are within a certain distance from the true value. The epsilon parameter ($\varepsilon$) in SVM regression represents the width of the insensitivity tube around the actual target value. Predictions falling within $\varepsilon$ distance of the true target are not penalized as errors.

The 10g release of ODM includes a suite of SVM algorithms: SVM-classification (binary and multi-class), SVM-regression, and one-class SVM. Both Gaussian and Linear kernels are supported across all three classes of models.

## 3. SVM Ease of Use

SVM is often daunting to people with limited data mining experience. Even though SVM models are generally easier to train than neural networks, there are many potential pitfalls that can lead to unsatisfactory results. The 'tricks of the trade' for building SVM models with good accuracy fall within two broad categories:

- Data preparation; and
- Model selection/parameter tuning.

### 3.1 Data Preparation Issues and Solutions

The SVM algorithm operates on numeric attributes and requires a one-to-one relationship between the object of study and its attributes. For example, a customer has a one-to-one relationship to its demographic data. In many cases, data are stored in multiple tables, one-to-many relationships are common, and data types other than real numbers are ubiquitous. To achieve the data representation required by SVM, the data have to undergo multiple non-trivial transformations. For example, a categorical column needs to be "exploded" into a set of new binary columns (one per category value). These early

stage transformations are application specific and are usually accomplished in an ad-hoc manner.

Using ODM significantly streamlines this processing stage. ODM SVM accepts a single data source. The source can be either a table or a view. It can include combinations of scalar columns (numeric or categorical) and nested table columns (collections of numeric or categorical values). The transformation of categorical data into binary attributes takes place internally without user intervention. The nested table columns provide several advantages: 1) they can be used to capture simple one-to-many relationships; 2) they provide efficient representation for sparse data; and 3) they allow data of arbitrary dimensionality to be mined (circumventing the upper limit on number of columns in a database table).

Even when the data are in the format required by the SVM algorithm, they may still not be suitable for training SVM models. Individual data attributes need to be normalized (placed on similar scale). Normalization prevents attributes with a large original scale from biasing the solution. Additionally, it makes computational overflows and underflows less likely. Furthermore, normalization brings the numerical attributes to the same scale [0, 1] as the exploded binary categorical columns. Outliers and long tailed distributions can adversely affect normalization by producing very compressed scales with low resolution. To prevent this, outlier removal and power transformations are often employed. ODM provides normalization and outlier removal capabilities in the `dbms_data_mining_transform` package.

While SVM is uniquely equipped to successfully handle high-dimensional data, it is sometimes desirable (e.g., for model transparency, performance, or noise reduction) to reduce the number of features. ODM provides two feature selection mechanisms – 1) an MDL-based (minimum description length) univariate attribute importance metric, and 2) a Non-Negative Matrix Factorization multivariate feature extraction method. Either of these techniques can be used for dimensionality reduction in data preparation for an SVM model build.

### 3.2 Model Selection

Model selection is the single most challenging and error prone step in the process of building SVM models. Also described as parameter tuning, model selection deals with choosing appropriate values for the regularization and kernel parameters. SVM models can be very sensitive to parameter choices – inappropriate values may lead to severe underfitting (e.g., for classification, the model always predicts the dominant class), severe overfitting (i.e., the model memorizes the training data), or slow convergence.

In practice, parameter values are usually chosen using a grid search approach. A grid search defines a grid over the parameter space. Each grid point represents a possible combination of parameter values. The grid resolution

determines the search space size. This can be computationally expensive as the search space may span several orders of magnitude for individual parameter values. Since SVM parameters interact, the complexity and computational cost of the search increases with the number of parameters. Often, the tuning process starts with a coarse grid search. Once a promising region is identified, a finer grid is applied. A number of speedups of the classic grid search approach have been proposed. In [17], an iterative refinement of the grid resolution and search boundaries was used. In [18], the search space was divided into a bad (underfitting/overfitting) region and a good region and a simplified line search was performed in the good region. In [19], the authors advocated a pattern search approach that iteratively searched a neighbourhood in a fixed pattern to discover an optimal operating point.

In order to evaluate model quality, parameter-tuning methods require a measure that captures their generalization performance. The most reliable, albeit most computationally expensive, method is to use $N$-fold cross-validation. The training data is divided into $N$ partitions. A model is built on $N$-1 partitions and the remaining partition is held aside for evaluation. Cross-validation involves building $N$ models for each parameter value combination. Once a good set of parameter values is found, a model is trained on the entire data. To speedup this process, the authors of [20] proposed that the coarse grid search be performed on a sample of the data; the entire data is considered only for the second stage of the search when a finer grid is used.

For datasets of reasonable size, it is feasible to hold aside a portion of the data for evaluation purposes and thus build one model per parameter value set and avoid the $N$-fold cross-validation.

While grid searches are usually effective, building and testing multiple SVM models can be an onerous and time-consuming activity. There has been some theoretical work on estimating generalization performance without the use of held-aside data. In [21], several leave-one-out bounds on generalization performance for SVM classifiers were defined. Given a set of parameter values, such bounds can be used for model evaluation. That is, if used within a grid search, the model test on held-aside data can be avoided. More importantly, such bounds can be optimized directly to derive the optimal set of parameter values. The leave-one-out bound is optimized using a gradient descent approach. A major bottleneck in the process is that each gradient step involves solving a separate SVM model. The leave-one-out bounds approach was also extended to SVM regression [22].

### 3.3 Oracle's Solution to SVM Model Selection

ODM tackles the model selection problem from a different perspective – emphasis is placed on usability and good out-of-the-box performance. Instead of investing time and system resources in finding an optimal set of parameters, ODM performs computationally inexpensive, data-driven, 'on-the-fly' estimates that place the algorithm parameters in reasonable operating ranges. The estimated values prevent severe model underfitting/overfitting, produce models with competitive accuracy, and support rapid convergence. These estimates also represent an excellent initial point for the parameter tuning methodologies outlined above. The following sections outline the parameter estimation techniques in the ODM SVM implementation.

*Complexity Factor Estimation*

SVM's complexity factor $C$, also referred to as capacity or penalty, is a regularization parameter that trades off training set error for margin maximization. In other words, the complexity of the model is counterbalanced by model robustness in order to achieve good generalization on new data.

In classification, our estimation approach for the complexity factor $C$ is based on the argument that an SVM model should be given sufficient capacity to separate typical representatives of the target classes. We randomly select a small number of examples per class (e.g., 30 positive and 30 negative records). We compute the prediction (using Eq. 1) for each record under the assumption that all Lagrangian multipliers are bounded (equal to $C$):

$$f_i = \sum_j Cy_j K(\mathbf{x}_j, \mathbf{x}_i) ,$$

where $y_j$ is the target label ($\pm 1$) and $K(\mathbf{x}_j, \mathbf{x}_i)$ is the kernel function. The summation is over all examples in the random sample. For this calculation, the focus is on the scale of the margins. Therefore, the bias is set to zero.

A small number of records may be predicted incorrectly under this assumption (where $f_i$ has the opposite sign from the actual target label). Within the random sample, such records are more similar to the examples from the opposite class and therefore can be interpreted as noise. We exclude their $f_i$ values from further consideration. For each of the other records, we calculate the $C$ value ($C^{(i)}$) that would make the $i$th record a non-bounded support vector (i.e., setting $f_i = \pm 1$):

$$C^{(i)} = sign(f_i) \Big/ \sum_j y_j K(\mathbf{x}_j, \mathbf{x}_i) .$$

The summation in the denominator is over all records in the sample (including the noisy ones).

The resulting set of $C$ values is sorted in descending order and the 90th percentile value is selected as the operating point. Thus the model is given sufficient capacity to achieve good, but not perfect, discrimination for the specific data, which is the intent of regularization.

In regression, we follow a heuristic proposed by [23]: $C = 3\sigma$ where $\sigma$ is the standard deviation of the target. This approach ensures that the approximating function has sufficient capacity to map target values within a reasonable range around the mean. Selecting a large $C$

value is reasonable for regression, since regularization is more effectively controlled via a different parameter: epsilon.

### Epsilon Estimation

The value of epsilon is supposed to reflect the intrinsic level of noise present in the regression target. In real-world problems, however, the noise level is generally not known. Therefore, a good epsilon estimator includes a method to estimate the noise level [23].

In ODM SVM, we use the following estimation procedure:

1. The initial value of epsilon is set to a small fraction of the mean of the target ($\mu_y$): $\varepsilon = 0.01 * \mu_y$.

2. We create a small training and a small held-aside validation set via random sampling of the available data.

3. An SVM model is built on the training sample and residuals are measured on the held-aside dataset.

4. Epsilon is updated on the basis of the mean absolute residual value ($\mu_r$):

$$\varepsilon^{new} = (\varepsilon^{old} + \mu_r)/2$$

Steps 3 and 4 are repeated several times (8 in the current implementation). The epsilon value that produced the lowest residuals on the held-aside data is used for building an SVM model on the full training data. Working with more data (e.g., the entire dataset) during the epsilon search phase is likely to produce a more accurate epsilon estimate. However, in our experience, samples produce results of good accuracy and achieve fast out-of-the-box performance.

### Standard Deviation Estimation

Standard deviation is a kernel parameter for controlling the spread of the Gaussian function and thus the smoothness of the SVM solution. Since distances between records grow with increasing dimensionality, it is a common practice to scale the standard deviation accordingly. For example, a simple, but not very effective, approximation of the standard deviation is $\sqrt{d}$, where $d$ is the dimensionality of the input space. The ODM SVM standard deviation estimation uses a data-driven approach to find a standard deviation value that is on the same scale as distances between typical data records.

In SVM classification, our approach is to estimate the typical distance between the two target classes. We randomly draw pairs of positive and negative examples and measure the Euclidean distance between the elements of each pair. We rank the distances in descending order and pick the 90th percentile distance as the standard deviation. We exclude the last decile to allow for some overlap of the two target classes so as to produce a smooth solution.

In SVM regression, the standard deviation estimate is also based on typical distances between randomly drawn data records. We draw pairs of random records and measure the Euclidean distances between them. We sort the values, exclude the smallest distances in the tail, and choose the 90th percentile distance as the standard deviation. The selected standard deviation value has sufficient resolution and an adequate degree of smoothness.

### 3.4 Model Selection Results

To illustrate the performance of our methodology, we compare our approach to a standard grid-search method. The results of an exhaustive grid search with fine granularity can be considered optimal - a 'gold standard'. We chose LIBSVM for all comparison tests in this paper, since it is very well known, and has achieved wide adoption as a standalone SVM tool.

We compared our classification parameter estimation approach to published results using the LIBSVM grid search tool [8]. Table 1 illustrates the performances of ODM SVM and LIBSVM on three benchmark classification datasets. Details about these datasets can be found in [20]. We report the following accuracy measurements: out-of-the-box ODM SVM, LIBSVM after grid-search and cross validation, and LIBSVM with default parameters. All of these tests were run on appropriately scaled (normalized) data. To emphasize the importance of data preparation in SVM, we also report LIBSVM's accuracy on the raw datasets.

**Table 1. SVM classification accuracy (% correct)**

| | ODM SVM | LIBSVM grid+xval | LIBSVM default | Raw data |
|---|---|---|---|---|
| **Astroparticle Physics** | 97 | 97 | 96 | 67 |
| **Bio-informatics** | 84 | 85 | 79 | 57 |
| **Vehicle** | 71 | 88 | 12 | 2 |

As expected, the combination of grid search and cross-validation produced the best accuracy results overall. Our approach to parameter estimation closely matched the grid search results in two of the three cases. Even in the third case, the benefit of using a computationally inexpensive data-driven estimation method is significant – it represents a great improvement over using LIBSVM's default parameters. It can also provide a very good starting point for model selection, if further system tuning is required. The main advantage of our on-the-fly estimation approach is that only a single model build takes place. The grid search and cross-validation methodology requires multiple SVM builds and, as a result, is significantly slower.

For SVM regression, we report in Table 2 the results on three popular benchmark datasets (details can be found in [24]). As in classification, our method shows a small loss of accuracy with respect to a grid search method. However, the ODM SVM methodology is still advantageous in terms of computational efficiency.

**Table 2. SVM regression accuracy (RMSE)**

|  | ODM SVM | LIBSVM grid |
|---|---|---|
| **Boston housing** | 6.57 | 6.26 |
| **Computer activity** | 0.35 | 0.33 |
| **Pumadyn** | 0.02 | 0.02 |

### 3.5. Other Usability Features

Other noteworthy usability features in ODM SVM include an automated kernel selection mechanism, one-class SVM enhancements, and unbalanced data treatment.

*Kernel Selection*

The SVM kernel is automatically selected according to the following rule: if the data source has high dimensionality (e.g., 100 dimensions), a linear kernel is used, otherwise a Gaussian kernel is used. This rule takes into account not only the number of columns in the original data source but also factors in categorical attribute cardinalities and information from any nested table columns.

*One-Class SVM*

One-class SVM models are a relatively new extension to the class of SVM models. To facilitate their adoption by a wider audience we incorporated several usability enhancements into ODM SVM. For models with linear kernels, our implementation automatically performs unit-length normalization. That is, all data records are mapped to the unit sphere and have a norm equal to 1. While such a transformation is essential for the correct operation of the one-class linear model, it may not be obvious to users having little experience with SVM models.

The other usability enhancement involves tuning the regularization parameter in one-class models. Typically, the outlier rate is controlled via a regularization parameter, $\eta$, which represents a lower bound on the number of support vectors and an upper bound on the number of outliers. In our experience, specifying the desired number of outliers via $\eta$ can result in outlier rates that significantly diverge from the desired value. To rectify this, we introduced a user setting for outlier rate. The specified outlier rate is achieved as closely as possible by an internal search on the complexity factor parameter. The search is conducted on a small data sample.

*Unbalanced data*

Real world data often have unbalanced target distributions. That is, certain target values/ranges can be significantly under-represented. Often these rare targets are of particular interest to the user. SVM models trained on such data can be very successful in identifying the dominant targets, but may be unable to accurately predict the rare target values. ODM SVM addresses this potential problem for both classification and regression by automatically employing a stratified sampling mechanism with respect to the target values. Our stratified sampling strategy is discussed in more detail in Section 4.3. Its overall effect is that the unbalanced target problem is significantly mitigated. In the case of classification, the user also has the option of supplying a weight vector (often referred to in the SVM literature as costs) that biases the model towards predicting the rare target values. By giving higher weights to the rare targets, the model can better characterize the targets of interest.

## 4. SVM Scalability

SVM has a reputation of being a data mining algorithm that places high requirements on CPU and memory resources. Scalability can be problematic both for the model training and model scoring operations. The next two sections outline techniques developed within the SVM community to somewhat alleviate the scalability issues. We used these techniques as a starting point for most of our methodological approaches and design choices in the ODM SVM implementation (Section 4.3).

### 4.1 SVM Build Scalability

The training time for standard SVM models scales quadratically with the number of records. For applications where model build speed is important, this limits the algorithm applicability to small and medium size data (less than 100K training records). Since the introduction of SVM, scalability improvements have been, and remain, the focus of a large body of research. Some noteworthy results include:

- Chunking and decomposition – the SVM model is built incrementally on subsets of the data (working sets), thus breaking up the optimization problem into small manageable subproblems and solving them iteratively [25];
- Working set selection – the working set is chosen such that the selected data records provide greatest improvement to the current state of the model, thus speeding up convergence;
- Kernel caching – some parts of the kernel matrix are maintained in memory (subject to memory constraints), thus speeding up computation;
- Shrinking – records with Lagrangian multipliers that appear to have converged can be temporarily excluded from optimization, thus shrinking the

size of the training data and speeding up convergence;

- Sparse data encoding – the algorithm can handle training data in a compressed format (zero values are not stated explicitly), thus reducing the number of computations;
- Specialized linear model representation – SVM models with linear kernels can be represented in terms of attribute coefficients in the input space. This form is more efficient than the traditional representation (Eq. 1), since it shortcuts the kernel computation step, thus significantly reducing computational and memory requirements.

These optimizations (or subsets of them) have been successfully leveraged in many SVM tools (implementation details can be found in LIBSVM [8], SVMLight [9], SVMTorch [10], HeroSVM [12]). These techniques have gained wide acceptance and are commonly leveraged in contemporary SVM implementations. ODM SVM also adopts variations of these optimizations (Section 4.3).

While these techniques alleviate some of the build scalability problems, their effectiveness is often unsatisfactory when applied to real world data. The situation is especially problematic for SVM models with non-linear kernels where computational and memory demands are higher. Large amounts of training data usually result in large SVM models (the number of support vectors increases linearly with the size of the data). While linear models can sidestep this issue by employing a coefficient representation, non-linear models require storage of the support vectors along with their Lagrangian multipliers. As a result, large non-linear SVM models have high disk and memory usage requirements. They are also very slow to score. This makes these models impractical for many types of applications.

## 4.2  SVM Scoring Scalability

In the context of real world applications, scoring poses even more stringent performance requirements than the build operation. Models are usually built offline and can therefore be scheduled, taking into account system loads. In contrast, scoring requires the processing of large quantities of data both offline and in real time. Real time operation necessitates fast computation and a small memory footprint. This effectively excludes large non-linear SVM models as likely candidates for applications where fast response is required.

Some recently developed methods have focused on decreasing SVM's model size. While these methods often improve build scalability, from an application point of view their main benefit lies in producing small models capable of fast and efficient scoring. Such reduced size models are also expected to have comparable accuracy to their full-scale counterparts. That is, gains in scoring

performance should not be offset by significant loss of accuracy.

A simple solution to the model size problem is to use only a fraction of the training data. Since random sampling often results in suboptimal models, to improve model quality, alternative sampling methods have been considered [26, 27, 28, 29]. Of particular interest are the active sampling methodologies [28, 29], where the goal is to identify data records that have the potential to maximally improve model quality. Many active sampling approaches follow a similar learning paradigm: 1) construct an initial model on a small random sample; 2) test the entire dataset against this model; 3) select additional data records by analyzing the predictions; 4) retrain the model on the new augmented training sample. The procedure iterates until a stopping criterion (usually related to limiting the model size) is met. These studies proposed selecting the records(s) nearest to the decision hyperplane because such records are expected to have the most significant effect in changing the decision hyperplane. A major criticism of the active sampling approaches is that finding the next 'best' candidate requires a full scan of the dataset. Ultimately, repeated scans of the data can dominate the execution time [29]. ODM SVM uses a variation of active sampling that avoids full scans.

## 4.3  Oracle's Solution to SVM Scalability

Oracle's ODM SVM implementation strives to meet performance requirements that are relevant within a typical database environment. Emphasis is placed on the efficient processing of large quantities of data, low memory requirements, and fast response time. Since the individual SVM features (classification, one-class, and regression) share a common code base, the performance optimizations described here apply to all three types of SVM models unless explicitly stated otherwise.

*Stratified Sampling*

When dealing with large collections of data, practitioners usually resort to sampling and building SVM models on subsets of the original dataset. Since random sampling is rarely adequate, specialized sampling strategies need to be employed. To remove this burden from the user, we include data selection mechanisms in our SVM implementation. These mechanisms are feature specific and were designed to achieve good scalability without compromising accuracy.

An important data selection requirement for classification tasks is that all target values be adequately represented. We use a stratified sampling approach where sampling rates are biased towards rare target values. Even though the relevant data statistics (record count, target cardinality, and target value distribution) are unknown at the time of the data load, the algorithm performs a single pass through the data and loads into memory a subset of

the data that is as balanced as possible across target values. The subset of records kept in memory consists of at most 500,000 records with a maximum of 50,000 per target value. If some target values cannot fill their allocated quotas, the quotas of other targets can be increased.

Stratified target sampling is also advantageous in regression tasks. In many problems the target distribution is non-uniform. A random sampling approach could result in areas of low support being poorly represented; however, often, such areas are of particular interest. Our SVM regression implementation takes an initial sample of up to 500,000 data records, estimates the distribution density over 20 equi-width intervals, and then creates a sub-sample that is biased towards areas with low support. The objective of the biased sampling is to select an equal number of data records from each target interval. That is, each interval is initially given the same quota. However, if there are not enough data records to meet the quota for a given interval, the quotas of the other intervals are increased.

### Decomposition and Working Set Selection

The core optimization module in our implementation uses a decomposition approach – the SVM model is built incrementally by optimizing small working sets towards a global solution. The model is trained until convergence on the current working set; then this working set is updated with new records and the model adapts to the new data. The process continues iteratively until the convergence conditions on the entire training data are met. Models with linear kernels use a fixed size working set. For models with non-linear kernels, the working set size is chosen such that the kernels associated with each working set record can be cached in the dedicated kernel cache memory.

The iterative nature of the optimization process makes the method prone to oscillations – the model can overfit individual working sets thus slowing down convergence towards the global solution. To avoid oscillations, we smoothly transition between working sets. A significant portion (up to 50%) of the working set is retained during working set updates. Retention preference is given to non-bounded support vectors since their Lagrangian multipliers are most likely to need further adjustment. To achieve rapid model improvement, we select the new members of the working set from amongst data records that perform worse than average under the current model (i.e., have worse than average violation of the model convergence constraints). Traditionally, working set selection methods evaluate all training data records, rank them and select the top candidates as working set members. In our implementation, we choose not to incur the cost of sorting the candidate data records. Instead, during the evaluation stage we compute an average violation. We perform a scan of the candidate data

records. If the violation is larger than average, the data record is included in the working set. If this scan does not produce a sufficient number of records to fill up the working set, a second scan is initiated where every record that does not meet the convergence conditions (i.e., it has non-zero violation) is included in the working set. Both scans start at a random point in the list of candidate records and can wrap around if needed. The scans terminate when the working set is filled up. This approach has been found to be advantageous over sorting, not only in terms of computational efficiency, but also in smoothing the solution from one working set to the next. Selecting the maximum violators tends to produce large model changes and oscillations in the solution. On the other hand, the random element in our working set selection contributes to smoother transitions and faster convergence.

### Active Learning

In classification, additional convergence speedup and model size reduction can be achieved via active learning. Under active learning, the algorithm creates an initial model on a small sample, scores the remaining training data, adds to its working set the data record that is closest to the current decision boundary, and then locally refines the decision hyperplane. The incremental model updates continue until the model converges on the training data or the maximum allowed number of support vectors is reached. To avoid the cost of scanning the entire dataset, the new data records are selected from a smaller pool of records (up to 10,000). This pool is derived from the target-stratified data sample obtained during the data load (see above). The candidate support vectors in the pool are balanced with respect to their target values. In a binary classification task, each target class provides half of the pool. If not enough examples are available for either of the target classes, the remainder of the pool is filled up with records from the other class. In multi-class classification tasks, we employ a one-versus-all strategy (i.e., a binary model is built for each target class and records associated with other target values are treated as counterexamples). The active learning pool is equally divided between examples of the current positive target and counterexamples – the latter being selected via stratified sampling across the other target values. Even though active learning takes place by default, it can be turned off to help assess its benefits over the standard approach.

### Linear Model Optimizations

The ODM SVM implementation has a separate code path for linear models. A number of linear kernel specific optimizations have been instrumented. Models with linear kernels can be considered a special case of SVM models, whose decision hyperplanes can be defined in terms of input attribute coefficients alone. This has an important

computational advantage that we exploit. It circumvents the explicit usage of a kernel matrix during the training optimization phase. Thus, we allocate kernel cache memory only for models with non-linear kernels. The compact representation for linear models also allows efficient model storage and fast scoring with low memory requirements. An added benefit is the easy interpretability of the resulting SVM model.

Shrinking is another enhancement that has a significant performance impact for ODM SVM linear models. If a training record meets the convergence conditions and its Lagrangian multiplier is at the lower limit (not a support vector) or upper limit (bounded support vector) for several successive iterations, this training record can be temporarily excluded from the training data, thus reducing the number of records that need to be evaluated at every iteration. Once the model converges on the remainder of the training data, the 'shrunk' records are re-activated and learning continues until convergence conditions are met.

Having a separate code path for linear models in ODM SVM introduces complexity in the implementation. However, the performance benefits more than outweigh the associated development costs. Linear SVM models have become the data mining algorithm of choice for a number of high-dimensional domains (e.g., text mining, bioinformatics, hyperspectral imagery) where computational efficiency is essential.

### Other Optimizations

Other optimizations include scalable memory management and sparse data representation. By default, the ODM SVM build operation has a small memory footprint. Active learning requires that only a small number of training examples (10,000) be maintained in memory. The dedicated kernel cache memory for non-linear models is set by default to 40MB and can be adjusted by the user depending on system resources. Two of the most expensive operations during model build – sorting of the training data and persisting the model into index-organized tables for faster access – have constraints on the physical memory available to the process. They spill to disk when physical memory constraints are exceeded. These operations are supported by database primitives and the amount of memory available to the process is controlled by the *pga_aggregate_target* database parameter. This allows graceful performance degradation for large datasets and reliable execution without system failures even with limited memory resources.

Sparse data encoding is also supported (Section 4.1). This allows SVM to be used in domains such as text mining and market basket analysis. The Oracle Text database functionality currently leverages the SVM implementation described here for document classification.

### 4.3 Scalability Experiments

To illustrate the behavior of Oracle's ODM SVM implementation on large datasets, we use a publicly available dataset for a network intrusion detection task [30]. In the experiments, a binary SVM model is used for classifying an activity pattern as normal or as an attack. The number of entries in the training dataset is 494,020. There are 41 attributes. After exploding the categorical values into binary attributes, the number of dimensions increases to 118. Figure 1 shows the scalability of the algorithm with increasing number of records for models with linear and Gaussian kernels. The smaller datasets were created via random sampling of the original data. For comparison, results from the LIBSVM tool are also included. LIBSVM was used with default parameters and no grid search was performed. Tests were run on a machine with the following hardware and software specifications: single 3GHz i86 processor, 2GB RAM memory, and Red Hat enterprise Linux OS 3.0.
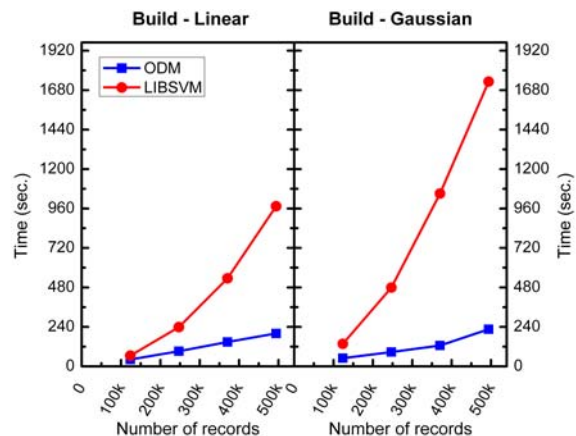


**Figure 1. SVM build scalability**

Oracle's ODM SVM implementation has better scalability than LIBSVM. While LIBSVM employs a number of optimizations, Oracle's ODM SVM has the advantages of active learning, sampling, and specialized linear model optimizations. On this particular dataset, the models built using either tool have equivalent discriminatory power. Since this is a binary model, performance is measured using the area under the Receiver Operating Characteristic (AUC). We compare the models built on the full training data (rightmost points in both subplots). Generalization performance is measured on an independent test set. Both linear models have AUC = 0.97. For models with Gausssian kernels, LIBSVM has AUC = 0.97 and ODM SVM has AUC = 0.98.

It should be also noted that in Figure 1 ODM's Gaussian kernel SVM models have comparable or even better execution times than the corresponding linear kernel models. This result is due to active learning where a small non-linear model size is enforced. Linear models do not have this constraint and the build effectively converges on a larger body of data.

A more important concern in the context of real world applications is that SVM models can be slow to score. Figure 2 illustrates the scoring performance when comparing SVM models built in ODM and LIBSVM on the full training data. All tested models have linear scalability. In the linear model case, the significant performance advantage of ODM SVM is due largely to the specialized linear model representation. For Gaussian models, the large difference is due mostly to differences in model sizes (132 support vectors in ODM SVM versus 9,168 in LIBSVM). The small number of support vectors in ODM SVM is achieved through active learning.
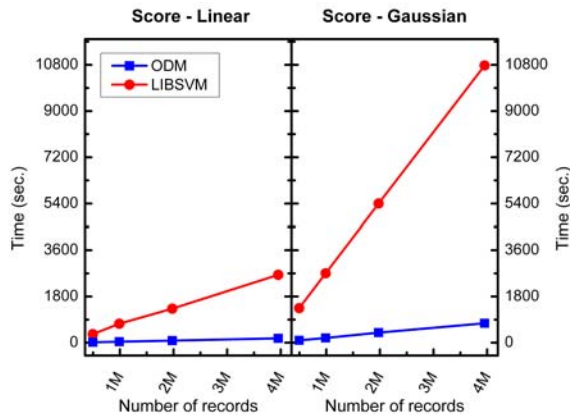


**Figure 2. SVM scoring scalability**

The scoring times in Figure 2 include persisting the results to disk. Oracle's PREDICTION SQL operator allows the seamless integration of the scoring operation in larger queries. Predictions can be piped out directly into the processing stream instead of being stored in a table. Table 3 breaks down the linear model timings into model scoring and result persistence.

**Table 3. ODM SVM scoring time breakdown**

|  | 500K | 1M | 2M | 4M |
|---|---|---|---|---|
| **SVM scoring (sec)** | 18 | 37 | 71 | 150 |
| **Result persistence (sec)** | 2 | 4 | 11 | 22 |

## 5.  SVM Integration

The tight integration of Oracle's data mining functionality within the RDBMS framework allows database users to take full advantage of the available database infrastructure. For example, a database offers great flexibility in terms of data manipulation. Inputs from different sources can be easily combined through joins. Without replicating data, database views can capture different slices of the data. Such views can be used directly for model generation or scoring. Database security is enhanced by eliminating the necessity of data export outside the database.

Using an SVM model representation that is native to the database greatly simplifies model distribution. Models are not only stored in the database but are executed in it as well. Oracle's scheduling infrastructure can be used to instrument automated model builds and deployments to multiple database instances.

Having scoring exposed through an SQL operator interface allows embedding of the SVM prediction functionality within DML statements, sub-queries, and functional indexes. The PREDICTION operator also enables hierarchical and cooperative scoring – multiple SVM models can be applied either serially or in parallel. The following SQL code snippet illustrates sequential (or 'pipelined') scoring:

```
SELECT id, PREDICTION(
              svm_model_1
              USING *)
FROM user_data
WHERE PREDICTION_PROBABILITY(
          svm_model_2,
          'target_val'
          USING *) > 0.5
```

In this example svm_model_1 is used to further analyze all records for which svm_model_2 has assigned a probability greater of equal 0.5 of belonging to the 'target_val' category. If this were an intrusion detection problem, svm_model_2 could be used for detecting anomalous behavior in the network and svm_model_1 would then assign the anomalous case to a category for further investigation.

## 6.  Conclusion

The addition of SVM to Oracle's analytic capabilities creates opportunities for tacking challenging problem domains such as text mining and bioinformatics. While the SVM algorithm is uniquely equipped to handle these types of data, creating an SVM tool with an acceptable level of usability and adequate performance is a non-trivial task. This paper discusses our design decisions, provides relevant implementation details, and reports accuracy and scalability results obtained using ODM SVM. We also briefly outline the key advantages of a tight integration between SVM and the database platform. We plan to further enhance our methodologies to achieve higher ease of use, support more data types, and improve our automatic data preparation.

## References

[1] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, NY, 1995.

[2] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. European Conference on Machine Learning*, 1998, pp. 137-142.

[3] K.-S. Goh, E. Chang, and K.-T. Cheng. SVM Binary Classifier Ensembles for Image Classification. In *Proc. Intl. Conf. Information Knowledge Management*, 2001, pp. 395-402.

[4] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. Mesirov, T. Poggio, W. Gerald, M. Loda, E. Lander, and T. Golub. Multi-Class Cancer Diagnosis Using Tumor Gene Expression Signatures. *Proc. National Academy of Sciences U.S.A.*, 98(26), 2001, pp. 15149-15154.

[5] M. Pal and P. M. Mather. Assessment of the Effectiveness of Support Vector Machines for Hyperspectral Data. *Future Generation Computer Systems*, 20(7), 2004, pp. 1215-1225.

[6] S. Mukkamala, G. Janowski, and A. Sung. Intrusion Detection Using Neural Networks and Support Vector Machines. In *Proc. IEEE Intl. Joint Conf. Neural Networks*, 2002, pp. 1702-1707.

[7] B. Scholkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[8] C.-C. Chang and C.-J. Lin. *LIBSVM: A Library for Support Vector Machines*. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[9] T. Joachims. Making Large-Scale SVM Learning Practical. In B. Scholkopf, C. J. C. Burges, and A. J. Smola (eds.), *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999, pp. 169-184.

[10] R. Collobert and S. Bengio. SVMTorch: Support Vector Machines for Large-Scale Regression Problems. *Journal of Machine Learning Research*, 1, 2001, pp. 143-160.

[11] J. C. Platt. Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In B. Scholkopf, C. J. C. Burges, and A. J. Smola (eds.), *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999, pp. 185-208.

[12] J. X. Dong, A. Krzyzak, and C. Y. Suen. A Fast SVM Training Algorithm. In *Proc. Intl. Workshop Pattern Recognition with Support Vector Machines*, 2002, pp. 53-67.

[13] Oracle Corporation. *Oracle Data Mining Concepts, 10g Release 2*. http://otn.oracle.com/, 2005.

[14] L. Hamel, A. Uvarov, and S. Stephens. *Evaluating the SVM Component in Oracle 10g Beta*. Technical Report TR04-299, Dept. of Computer Science and Statistics, University of Rhode Island, http://homepage.cs.uri.edu/faculty/hamel/pubs/oracle-tr299.pdf, 2004.

[15] D. Anguita, S. Ridella, F. Rivieccio, A.M. Scapolla, and D. Sterpi. *A Comparison Between Oracle SVM Implementation and cSVM*. Technical Report, Dept. of Biophysical and Electronic Engineering, University of Genoa, http://www.smartlab.dibe.unige.it/smartlab/publication/pdf/TR001.pdf, 2004.

[16] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, and A. J. Smola. Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7), 2001, pp. 1443-1471.

[17] C. Staelin. *Parameter Selection for Support Vector Machines*. Technical Report HPL-2002-354, HP Laboratories, Israel, 2002.

[18] S. S. Keerthi and C.-J. Lin. Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. *Neural Computation*, 15(7), 2003, pp. 1667-1689.

[19] M. Momma and K. P. Bennett. A Pattern Search Method for Model Selection of Support Vector Regression. *Proceedings of SIAM Conference on Data Mining*, 2002.

[20] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. *A Practical Guide to Support Vector Classification*. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[21] O. Chapelle and V. Vapnik. Model Selection for Support Vector Machines. *Advances in Neural Information Processing Systems*, 1999, pp. 230-236.

[22] M.-W. Chang and C.-J. Lin. Leave-One-Out Bounds for Support Vector Regression Model Selection. To appear in *Neural Computation*, 2005.

[23] V. Cherkassky and Y. Ma. Practical Selection of SVM Parameters and Noise Estimation for SVM Regression. *Neural Networks*, 17(1), 2004, pp. 113-126.

[24] Delve: Data for Evaluating Learning in Valid Experiments. http://www.cs.toronto.edu/~delve/data/datasets.html

[25] E. Osuna, R. Freund, and F. Girosi. An Improved Training Algorithm for Support Vector Machines. In *Proc. IEEE Workshop on Neural Networks and Signal Processing*, 1997, pp. 276-285.

[26] Y.-J. Lee and O. L. Mangasarian. RSVM: Reduced Support Vector Machines. In *Proc. SIAM Intl. Conf. Data Mining*, 2001.

[27] J. L. Balcazar, Y. Dai, and O. Watanabe. Provably Fast Training Algorithms for Support Vector Machines. In *Proc. IEEE Intl. Conf. on Data Mining*, 2001.

[28] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. *Journal of Machine Learning Research*, 2, pp. 45-66.

[29] H. Yu, J. Yang, and J. Han. Classifying Large Data Sets Using SVM with Hierarchical Clusters. In *Proc. Intl. Conf. Knowledge Discovery Data Mining*, 2003, pp. 306-315.

[30] KDD'99 competition dataset. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.