Classification Using Optimization: Application to Credit Ratings of Bonds

Vladimir Bugera, Stan Uryasev, and Grigory Zrazhevsky

University of Florida, ISE, Risk Management and Financial Engineering Lab,
uryasev@ufl.edu

Summary. The classification approach, previously considered in credit card scoring, is extended to multi class classification in application to credit rating of bonds. The classification problem is formulated as minimization of a penalty constructed with quadratic separating functions. The optimization is reduced to a linear programming problem for finding optimal coefficients of the separating functions. Various model constraints are considered to adjust model flexibility and to avoid data overfitting. The classification procedure includes two phases. In phase one, the classification rules are developed based on "in-sample" dataset. In phase two, the classification rules are validated with "out-of-sample" dataset. The considered methodology has several advantages including simplicity in implementation and classification robustness. The algorithm can be applied to small and large datasets. Although the approach was validated with a finance application, it is quite general and can be applied in other engineering areas.

Key words: Bond ratings, credit risk, classification.

1 Introduction

We consider a general approach for classifying objects into several classes and applies it to a bond-rating problem. Classification problems become increasingly important in the decision science. In finance, they are used for grouping financial instruments according to their risk, or profitability characteristics. In the bonds rating problem, for example, the debt instruments are arranged according to the likelihood of debt issuer to default on the defined obligation.

Mangasarian et al. (1995) used a *utility function* for the failure discriminant analysis (applications to breast cancer diagnosis). The utility function was considered to be linear in control parameters and indicator variables and it was found by minimizing the error of misclassification. Zopounidis and Doumpos (1997), Zopounidis et al. (1998) and Pardalos et al. (1997) used linear utility functions for trichotomous classifications of credit card applications. Konno and Kobayashi (2000) and Konno et al. (2000) considered utility

functions, quadratic in indicator parameters and linear in decision parameters. The approach was tested with the classification of enterprises and breast cancer diagnosis. Konno and Kobayashi (2000) and Konno et al. (2000) imposed convexity constraints on utility functions in order to avoid discontinuity of discriminant regions. Similar to Konno and Kobayashi (2000) and Konno et al. (2000), Bugera et al. (2003) applied a quadratic utility function to trichotomous classification, but instead of convexity constraints, monotonicity constraints reflecting experts' opinions were used. The approach by Bugera et al. (2003) is closely related to ideas by Zopounidis and Doumpos (1997), Zopounidis et al. (1998) and Pardalos et al. (1997); it considers a multi-class classification with several levelsets of the utility function, where every levelset corresponds to a separate class.

We extend the classification approach considered by Bugera et al. (2003). Several innovations improving the efficiency of the algorithm are suggested:

- A *set* of utility functions, called *separating functions*, is used for classification. The separating functions are quadratic in indicator variables and linear in decision variables. A set of optimal separating functions is found by minimizing the misclassification error. The problem is formulated as a linear programming problem w.r.t. decision variables.
- To control flexibility of the model and avoid overfitting we impose various *new constraints* on the separating functions.

Controlling flexibility of the model with constraints is crucially important for the suggested approach. Quadratic separating functions (depending upon the problem dimension) may have a very large number of free parameters. Therefore, a tremendously large dataset may be needed to "saturate" the model with data. Constraints reduce the number of degrees of freedom of the model and adjust "flexibility" of the model to the size of the dataset.

This paper is focused on a numerical validation of the proposed algorithm. We rated a set of international bonds using the proposed algorithm. The dataset for the case study was provided by the research group of the RiskSolutions branch of Standard and Poor's, Inc. We investigated the impact of model flexibility on classification characteristics of the algorithm and compared performance of several models with different types of constraints. Experiments showed the importance of constraints adjusting the flexibility of the model. We studied "in-sample" and "out-of-sample" characteristics of the suggested algorithm. At the first stage of the algorithm, we minimized the empirical risk, that is, the error of misclassification on a training set (in-sample error). However, the real objective of the algorithm is to classify objects outside the training set with a minimal error (out-of-sample error). The in-sample error is always not greater than the out-of-sample error. Similar issues were studied in the Statistical Learning Theory (Vapnik, 1998). For validation, we used the leave-one-out cross-validation scheme. This technique provides the highest confidence level while effectively improving the predicting power of the model.

The advantage of the considered methodology is its simplicity and consistency if compared to the proprietary models which are based on the combination of various techniques, such as expert models, neural networks, classification trees etc.

The paper is organized as follows. Section 2 provides a general description of the approach. Section 3 considers constraints applied to the studied optimization problem. Section 4 discusses techniques for choosing model flexibility. Section 5 explains how the errors estimation was done for the models. Section 6 discusses the bond-rating problem used for testing the methodology. Section 7 describes the datasets used in the study. Section 8 presents the results of computational experiments and analyses obtained results. We finalize the paper with concluding remarks in Section 9.

2 Description of Methodology

The *object space* is a set of elements (objects) to be classified. Each element of the space has n quantitative characteristics describing properties of the considered objects.

We represent objects by n-dimensional vectors and the object space by an n-dimensional set $\Psi \subset \mathrm{R}^n$. The classification problem assigns elements of the object space Ψ to several classes so that each class consists of elements with similar properties. The classification is based on available prior information. In our approach, the prior information is provided by set of objects $S = \{\mathbf{x}_1, .., \mathbf{x}_m\}$ with known classification (*in-sample dataset*). The purpose of the methodology is to develop a classification algorithm that assigns a class to a new object based on the in-sample information.

Let us consider a classification problem with objects having n characteristics and J classes. Since each object is represented by a vector in a multidimensional space R^n, the classification can be defined by an integer-valued function $f_0(\mathbf{x}), \mathbf{x} \in \mathrm{R}^n$. The value of the function defines the class of an object. We call $f_0(\mathbf{x})$ a *classification function*. This function splits the object space into J non-intersecting areas:

$$\mathrm{R}^n = \bigcup_{i=1}^{J} F_i,\ F_i \cap F_j = \emptyset,\ F_i \neq \emptyset,\ F_j \neq \emptyset,\ i \neq j \quad , \tag{1}$$

where each area F_i consists of elements belonging to the corresponding class i:

$$F_i = \left\{ \mathbf{x} \in \mathrm{R}^n \middle|\ f_0(\mathbf{x}) = i \right\} \quad . \tag{2}$$

We can approximate the classification function $f_0(\mathbf{x})$ using optimization methods. Let $F(\mathbf{x})$ be a cumulative distribution function of objects in the object space R^n, and Λ be a parameterized set of discrete-value approximating functions. Then, the function $f_0(\mathbf{x})$ can be approximated by solving the following minimization problem:

$$\min_{f \in \Lambda} \int_{R^n} Q\left(f\left(\mathbf{x}\right) - f_0\left(\mathbf{x}\right)\right) dF\left(\mathbf{x}\right), \tag{3}$$

where $Q\left(f\left(\mathbf{x}\right) - f_0\left(\mathbf{x}\right)\right)$ is a penalty function defining the value of misclassification for a single object $\mathbf{x}$. The optimal solution $f\left(\mathbf{x}\right)$ of optimization problem (3) gives an approximation of the function $f_0\left(\mathbf{x}\right)$. The main difficulty in solving problem (3) is the discontinuity of functions $f\left(\mathbf{x}\right)$ and $f_0\left(\mathbf{x}\right)$ leading to non-convex optimization. To circumvent this difficulty, we reformulate problem (3) in a convex optimization setting.

Let us consider a classification function $\bar{f}(\mathbf{x})$ defining a classification on the object space R^n. Suppose we have a set of continuous functions $U_0(\mathbf{x}), ..., U_J\left(\mathbf{x}\right)$. We call them *separating functions for the classification function* $\bar{f}(\mathbf{x})$ if for every object $\mathbf{x}^*$ from class $i = \bar{f}(\mathbf{x}^*)$ values of functions with numbers lower than i are positive:

$$U_0\left(\mathbf{x}^*\right), ..., U_{i-1}\left(\mathbf{x}^*\right) > 0; \tag{4}$$

values of functions with numbers higher or equal to i are negative or zeros:

$$U_i\left(\mathbf{x}^*\right), ..., U_J\left(\mathbf{x}^*\right) \leq 0. \tag{5}$$

If we know the functions $U_0(\mathbf{x}), ..., U_J\left(\mathbf{x}\right)$ we can classify objects according to the following rule:

$$\left\{ \begin{array}{ll} \forall p = 0, .., i-1 : & U_p(\mathbf{x}^*) > 0 \\ \forall q = i, .., J : & U_q\left(\mathbf{x}^*\right) \leq 0 \end{array} \right\} \Leftrightarrow \left\{ \bar{f}\left(\mathbf{x}^*\right) = i \right\}. \tag{6}$$

The class number of an object $\mathbf{x}^*$ can be determined by the number of positive values of the functions: an object in i class has exactly i positive separating functions. Figure 1 illustrates this property.

Suppose we can represent any classification function $f\left(\mathbf{x}\right)$ from a parameterized set of functions Λ by a set of separating functions. By constructing a penalty for the classification $f_0\left(\mathbf{x}\right)$ using separating functions, we can formulate optimization problem (3) with respect to the parameters of the separating functions.

Suppose the classification function $\bar{f}(\mathbf{x})$ is defined by a set of separating functions $U_0(\mathbf{x}), ..., U_J\left(\mathbf{x}\right)$. We denote by $D_{f_0}\left(U_0, .., U_J\right)$ an integral function measuring deviation of the classification (implied by the separating functions $U_0(\mathbf{x}), ..., U_J\left(\mathbf{x}\right)$) from the true classification defined by $f_0(\mathbf{x})$:

$$D_{f_0}\left(U_0, .., U_J\right) = \int_{R^n} Q\left(U_0\left(\mathbf{x}\right), .., U_J\left(\mathbf{x}\right), f_0\left(\mathbf{x}\right)\right) dF\left(\mathbf{x}\right) \quad , \tag{7}$$

where $Q\left(U_0(\mathbf{x}), .., U_J(\mathbf{x}), f_0(\mathbf{x})\right)$ is a penalty function.

Further, we consider the following penalty function:

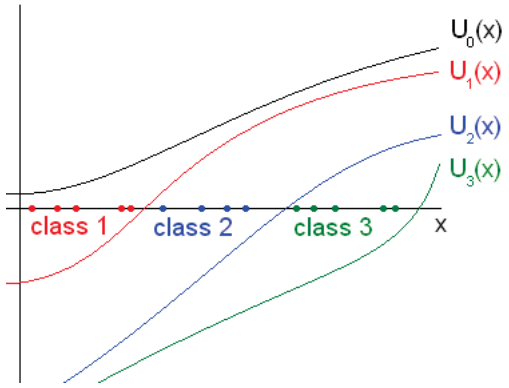

Fig. 1. Classification by separating functions. At each point (object), values of the separating functions with the indices lower than i are positive; the values of the remaining functions are negative or zero. Class of an object is determined by the number of functions with positive values.

$$Q\left(U_0,..,U_J,f_0\left(\mathbf{x}\right)\right)=\sum_{k=0}^{f_0(\mathbf{x})-1}\lambda_k\left(-U_k\left(\mathbf{x}\right)\right)^+ +\sum_{k=f_0(\mathbf{x})}^{J}\lambda_k\left(U_k\left(\mathbf{x}\right)\right)^+, \tag{8}$$

where $(y)^+ = \max(0,y)$, and λ_k, $k=0,..,J$ are positive parameters. The penalty function equals zero if $U_k(\mathbf{x}) \geq 0$ for $k=0,..,f_0(\mathbf{x})-1$, and $U_k(\mathbf{x}) \leq 0$ for $k=f_0(\mathbf{x}),..,J$. If the classification defined by the separating functions $U_0(\mathbf{x}),...,U_J(\mathbf{x})$ according to rule (6) coincides with the true classification $f_0(\mathbf{x})$, then, the value of penalty function (8) equals zero for any object $\mathbf{x}$. If the penalty is positive for an object $\mathbf{x}$ then separating functions misclassify this object. Therefore, the penalty function defined by (8) provides the condition of the correct classification by the separating functions:

$$\begin{gathered}\left\{Q\left(U_0,..,U_J,f_0\left(\mathbf{x}\right)\right)=0\right\}\\ \Updownarrow\\ \left\{\,U_k(\mathbf{x})>0>U_l(\mathbf{x}),\ \forall k=1,..,f_0\left(\mathbf{x}\right)-1,\forall l=f_0\left(\mathbf{x}\right),..,J\,\right\}\,.\end{gathered} \tag{9}$$

The choice of penalty function (8) is motivated by the possibility of building an efficient algorithm for the optimization of integral (7) when the separating functions are linear w.r.t. control parameters. In this case, optimization problem (3) can be reduced to linear programming.

After introducing the separating functions we reformulate optimization problem (3) as follows:

$$\min_{U_0,..,U_J} D_{f_0}\left(U_0,..,U_J\right). \tag{10}$$

This optimization problem finds optimal separating functions. We can approximate the function $D_{f_0}(U_0,..,U_J)$ by sampling objects according to the cumulative distribution function $F(\mathbf{x})$. Assuming that $\mathbf{x}^1,..,\mathbf{x}^m$ are some sample points, the approximation of $D_{f_0}(U_0,..,U_J)$ is given by:

$$\tilde{D}^m_{f_0}(U_0,..,U_J) = \frac{1}{m}\sum_{i=1}^{m} Q\left(U_0\left(\mathbf{x}^i\right),..,U_J\left(\mathbf{x}^i\right),f_0\left(\mathbf{x}^i\right)\right). \tag{11}$$

Therefore, the approximation of deviation function (7) becomes:

$$\begin{aligned}&\tilde{D}^m_{f_0}(U_0,..,U_J) = \\ &= \tfrac{1}{m}\sum_{i=1}^{m}\left(\sum_{k=0}^{f_0(\mathbf{x})-1}\lambda_k\left(-U_k\left(\mathbf{x}^i\right)\right)^+ + \sum_{k=f_0(\mathbf{x})}^{J}\lambda_k\left(U_k\left(\mathbf{x}^i\right)\right)^+\right) \quad .\end{aligned} \tag{12}$$

To avoid possible ambiguity when the value of a separating function equals 0, we introduced a small positive constant δ inside of each term in the penalty function (constant δ has to be chosen small enough in order not to have a significant impact on the final classification):

$$\begin{aligned}&Q(U_0,..,U_J,f_0(\mathbf{x})) = \\ &= \sum_{k=0}^{f_0(\mathbf{x})-1}\lambda_k\left(-U_k(\mathbf{x})+\delta\right)^+ + \sum_{k=f_0(\mathbf{x})}^{J}\lambda_k\left(U_k(\mathbf{x})+\delta\right)^+.\end{aligned} \tag{13}$$

The penalty function equals zero if $U_k(\mathbf{x}) \geq \delta$ for $k = 0,..,f_0(\mathbf{x})-1$, and $U_k(\mathbf{x}) \leq -\delta$ for $k = f_0(\mathbf{x}),..,J$. The approximation of deviation function (7) becomes

$$\begin{aligned}&\tilde{D}^m_{f_0}(U_0,..,U_J) = \\ &= \tfrac{1}{m}\sum_{i=1}^{m}\left(\sum_{k=0}^{f_0(\mathbf{x})-1}\lambda_k\left(-U_k\left(\mathbf{x}^i\right)+\delta\right)^+ + \sum_{k=f_0(\mathbf{x})}^{J}\lambda_k\left(U_k\left(\mathbf{x}^i\right)+\delta\right)^+\right).\end{aligned} \tag{14}$$

Further, we parameterized each separating function by a K-dimensional vector $\boldsymbol{\alpha} \in A \subset \mathrm{R}^K$. Parameter K is defined by the design of the separating functions. Therefore, a set of separating functions $U_{(\boldsymbol{\alpha}^0,..,\boldsymbol{\alpha}^J)}(\mathbf{x}) = \{U_{\boldsymbol{\alpha}^0},..,U_{\boldsymbol{\alpha}^J}\}$ is determined by a set of vectors $\left\{\boldsymbol{\alpha}^0,..,\boldsymbol{\alpha}^J\right\}$. With this parameterization we reformulated the problem (10) as minimization of the convex piece-wise linear functions:

$$\min_{\boldsymbol{\alpha}^0,...,\boldsymbol{\alpha}^J\in A}\sum_{i=1}^{m}\left(\sum_{k=0}^{f_0\left(\mathbf{x}^i\right)-1}\lambda_k\left(-U_{\boldsymbol{\alpha}^k}\left(\mathbf{x}^i\right)+\delta\right)^+ + \sum_{k=f_0\left(\mathbf{x}^i\right)}^{J}\lambda_k\left(U_{\boldsymbol{\alpha}^k}\left(\mathbf{x}^i\right)+\delta\right)^+\right), \tag{15}$$

where $A \subset \mathrm{R}^K$. By introducing new variables σ^j_i, we reduced problem (15) to equivalent mathematical programming problem with the linear objective function:

$$\begin{array}{l} \min\limits_{\alpha,\sigma} \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{J} \lambda_j \sigma_i^j \\ \sigma_i^j \geq -U_{\boldsymbol{\alpha}^j}\left(\mathbf{x}^i\right) + \delta,\, j = 0, .., f_0(\mathbf{x}^i) - 1 \\ \sigma_i^j \geq U_{\boldsymbol{\alpha}^j}\left(\mathbf{x}^i\right) + \delta,\, j = f_0(\mathbf{x}^i), .., J \\ \boldsymbol{\alpha}^0, .., \boldsymbol{\alpha}^J \in A \\ \sigma_i^1, .., \sigma_i^J \geq 0 \end{array} \tag{16}$$

Further, we consider that the separating functions are linear in control parameters $\alpha_1, .., \alpha_K$. In this case the separating functions can be represented in the following form:

$$U_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k g_k(\mathbf{x}). \tag{17}$$

In this case, optimization problem (16) for finding optimal separating functions can be reduced to the following linear programming problem:

$$\begin{array}{l} \min\limits_{\alpha,\sigma} \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{J} \lambda_j \sigma_i^j \\ \sigma_i^j + \sum\limits_{k=1}^{K} \alpha_k^j g_k\left(\mathbf{x}^i\right) \geq \delta,\, j = 0, .., f_0(\mathbf{x}^i) - 1 \\ \sigma_i^j - \sum\limits_{k=1}^{K} \alpha_k^j g_k\left(\mathbf{x}^i\right) \geq \delta,\, j = f_0(\mathbf{x}^i), .., J \\ \boldsymbol{\alpha}^0, .., \boldsymbol{\alpha}^J \in A \\ \sigma_i^1, .., \sigma_i^J \geq 0 \end{array} \tag{18}$$

Further, we consider quadratic (in indicator variables) separating functions:

$$U_j(\mathbf{x}) = \sum_{k=1}^{n} \sum_{l=1}^{n} a_{kl}^j x_i x_j + \sum_{k=1}^{n} b_k^j x_k + c^j,\, j = 0, .., J\,. \tag{19}$$

Optimization problem (18) with quadratic separating functions is reformulated as follows:

$$\begin{array}{l} \min\limits_{a,b,c,\sigma} \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{J} \lambda_j \sigma_i^j \\ \sigma_i^j + \sum\limits_{k=1}^{n} \sum\limits_{l=1}^{n} a_{kl}^j x_k^i x_l^i + \sum\limits_{i=1}^{n} b_k^j x_k^i + c^j \geq \delta,\, j = 0, .., f_0(\mathbf{x}^i) - 1 \\ \sigma_i^j - \sum\limits_{k=1}^{n} \sum\limits_{l=1}^{n} a_{kl}^j x_k^i x_l^i - \sum\limits_{k=1}^{n} b_i^j x_k^i - c^j \geq \delta,\, j = f_0(\mathbf{x}^i), .., J \\ \sigma_i^1, .., \sigma_i^J \geq 0 \end{array} \tag{20}$$

Although there are $J+1$ separating functions in problem (20), only $J-1$ functions are essential for the classification. The functions

$$\sum_{k=1}^{n} \sum_{l=1}^{n} a_{kl}^0 x_k x_l + \sum_{k=1}^{n} b_k^0 x_k + c^0$$

and

$$\sum_{k=1}^{n}\sum_{l=1}^{n} a_{kl}^{J} x_k x_l + \sum_{k=1}^{n} b_k^J x_k + c^J$$

are boundary functions. For all the classified objects, the value of the first boundary function is positive, and the value of the second boundary function is negative. This can be easily achieved by setting $c^0 = M$ and $c^J = -M$, where M is a sufficiently large number. Thus, these functions can be removed from optimization problem (20). However, these boundary functions can be used for adjusting flexibility of the classification model. In the next section, we will show how to use these functions for imposing the so-called "squeezing" constraints.

For the case $J = 2$ with only two classes, problem (20) finds a quadratic surface $\sum_{k=1}^{n}\sum_{l=1}^{n} a_{kl}^{1} x_k x_l + \sum_{k=1}^{n} b_k^1 x_k + c^1 = 0$ dividing the object space R^n into two areas. After solving optimization problem (20) we expect that a majority of objects from the first class will belong to the first area. On these points the function $\sum_{k=1}^{n}\sum_{l=1}^{n} a_{kl}^{1} x_k x_l + \sum_{k=1}^{n} b_k^1 x_k + c^1$ is positive. Similar, for a majority of objects from the second class the function $\sum_{k=1}^{n}\sum_{l=1}^{n} a_{kl}^{1} x_k x_l + \sum_{k=1}^{n} b_k^1 x_k + c^1$ is negative.

For the case with $J > 2$, the geometrical interpretation of optimization problem (20) refers to the partition of the object space R^n into J areas by $J-1$ non-intersecting quadratic surfaces

$$\sum_{k=1}^{n}\sum_{l=1}^{n} a_{kl}^{j} x_k x_l + \sum_{k=1}^{n} b_k^j x_k + c^j = 0, \qquad j = 1, .., J-1.$$

Additional feasibility constraints, that assure non-intersection of the surfaces, will be discussed in the following section.

3 Constraints

The considered separating functions, especially the quadratic functions, may be too flexible (have too many degrees of freedom) for datasets with a small number of datapoints. Imposing additional constraints may reduce excessive model flexibility. In this section, we will discuss different types of constraints applied to the model.

Konno and Kobayashi (2000) and Konno et al. (2000) considered convexity constraints on indicator variables of a quadratic utility function. Bugera et al. (2003) imposed monotonicity constraints on the model to incorporate expert preferences.

Constraints play a crucial role in developing the classification model because they reduce excessive flexibility of a model for small training datasets.

Moreover, a classification with multiple separating functions may not be possible for the majority of objects if appropriate constraints are not imposed.

3.1 Feasibility Constraints (F-Constraints)

For classification with multiple separating functions we may potentially come to a possible intersection of separating surfaces. This may lead to inability of the approach to classify some objects. To circumvent this difficulty, we introduce feasibility constraints, that keep the separating functions apart from each other. It makes possible to classify any new point by rule (6). In general, these constraints have the form:

$$U_j(\mathbf{x}) \leq U_{j-1}(\mathbf{x});\ j = 1, .., J;\ \mathbf{x} \in W \subset R^n\,, \tag{21}$$

where W is a set on which we want to achieve the feasibility. We do not consider $W = R^n$, because it leads to "parallel" separating surfaces, which were studied in the previous work by Bugera et al. (2003). In this section, we consider a set W with a finite number of elements. In particular, we consider the set W being the training set plus the set of objects we want to classify (out-of-sample dataset). In this case, constraints (21) can be rewritten as

$$U_j(\mathbf{x}_i) \leq U_{j-1}(\mathbf{x}_i);\ j = 1, .., J;\ i = 1, .., r\,, \tag{22}$$

where r is a number of objects in the set W. The fact that we use out-of-sample points does not cause any problems because we use the data without knowing their class numbers.

Since classification without feasibility constraints may lead to inability of classifying new objects (especially for small training sets), we will always include the feasibility constraints (22) to classification problem (16). For quadratic separating functions, classification problem (20) with feasibility constraints can be rewritten as

$$\begin{aligned}
&\min_{a,b,c,\sigma} \sum_{i=1}^{m} \sum_{j=1}^{J} \lambda_j \sigma_i^j \\
&\sigma_i^j + \sum_{k=1}^{n} \sum_{l=1}^{n} a_{kl}^j x_k^i x_l^i + \sum_{i=1}^{n} b_k^j x_k^i + c^j \geq \delta,\ j = 0, .., f_0(\mathbf{x}^i) - 1 \\
&\sigma_i^j - \sum_{k=1}^{n} \sum_{l=1}^{n} a_{kl}^j x_k^i x_l^i - \sum_{k=1}^{n} b_i^j x_k^i - c^j \geq \delta,\ j = f_0(\mathbf{x}^i), .., J \\
&\sum_{k=1}^{n} \sum_{l=1}^{n} a_{kl}^{j-1} x_k^i x_l^i + \sum_{i=1}^{n} b_k^{j-1} x_k^i + c^{j-1} - \sum_{k=1}^{n} \sum_{l=1}^{n} a_{kl}^j x_k^i x_l^i + \sum_{i=1}^{n} b_k^j x_k^i + c^j \geq \delta \\
&i = 1, .., r,\ j = 1, .., J \\
&\sigma_i^1, .., \sigma_i^J \geq 0
\end{aligned} \tag{23}$$

3.2 Monotonicity Constraints (M-Constraints)

We use monotonicity constraints to incorporate the preference of greater values of indicator variables. In financial applications, monotonicity with respect

to some indicators follows from "engineering" considerations. For instance, in the bond rating problem, considered in this paper, we have indicators (return on capital, operating-cash-flow-to-debt ratio, total assets), greater values of which lead to the higher ratings of bonds. If we enforce the monotonicity of the separating functions with respect to these indicators, objects with greater values of the indicators will have higher ratings. For a smooth function $h(\mathbf{x})$, the monotonicity constraints can be written as the constraint on the non-negativity of the first partial derivative:

$$\frac{\partial h(\mathbf{x})}{\partial x_i} \geq 0, \quad i \in \{1,..,n\}. \tag{24}$$

For the case of linear separating functions,

$$U_\alpha(\mathbf{x}) = \sum_{k=1}^{K} a_k^\alpha x_k + c^\alpha, \tag{25}$$

the monotonicity constraints are

$$a_k \geq 0,\ k \in K \subset \{1,..,n\}. \tag{26}$$

3.3 Positivity Constraints for Quadratic Separating Functions (P-Constraints)

Unlike the case with the linear functions, imposing exact monotonicity constraints on a quadratic function

$$U_j(\mathbf{x}) = \sum_{k=1}^{n}\sum_{l=1}^{n} a_{kl}^j x_k x_l + \sum_{k=1}^{n} b_k^j x_k + c^j \tag{27}$$

is a more complicated issue (indeed, in general, the quadratic function is not monotonic in the whole space R^n). Instead of imposing exact monotonicity constraints we consider the following constraints (we call them "positivity constraints"):

$$a_{kl}^\gamma \geq 0,\ b_k^\gamma \geq 0,\ k,l = 1,..,n. \tag{28}$$

Bugera et al. (2003) demonstrated that the positivity constraints can be easily included into the linear programming formulation of the classification problem. They do not significantly increase the computational time of the classification procedure, but provide robust results for small training datasets and datasets with missing or erroneous information. These constraints impose monotonicity with respect to variables $x_i,\ i = 1,..,n$ on the positive part $R^+ = \left\{ \mathbf{x} \in R^n \mid x_k \geq 0,\ k = 1,..,n \right\}$ of the object space R^n.

3.4 Gradient Monotonicity Constraints (GM-Constraints)

Another way to enforce monotonicity of quadratic separating functions is to restrict the gradient of separating functions on some set of objects X^* (for example, on a set of objects combining in-sample and out-of-sample points):

$$\left.\frac{\partial U_\alpha(\mathbf{x})}{\partial x^s}\right|_{\mathbf{x}=\mathbf{x}^*} \geq \gamma_s,\ \alpha = 0,..,J,\ s = 1,..,K,\ \mathbf{x}^* \in X^*, \tag{29}$$

where

$$\left.\frac{\partial U_\alpha(\mathbf{x})}{\partial x^s}\right|_{\mathbf{x}=\mathbf{x}^*} = \left.\frac{\partial\left(\sum\limits_{k=1}^{n}\sum\limits_{l=1}^{n} a_{kl}^{\alpha} x_k x_l + \sum\limits_{k=1}^{n} b_k^{\alpha} x_k + c^{\alpha}\right)}{\partial x^s}\right|_{\mathbf{x}=\mathbf{x}^*} = \\ = \sum_{k=1}^{n}\left(a_{ks}^{\alpha} + a_{sk}^{\alpha}\right) x_k^* + b_s^{\alpha}. \tag{30}$$

In constraint (29), γ_s, $s = 1,..,K$ are nonnegative constants.

3.5 Risk Constraints (R-Constraints)

Another important constraint that we apply to the model is the risk constraint. The risk constraint restricts the average value of the penalty function for misclassified objects. For this purpose we use the concept of Conditional Value-at-Risk (CVaR). The optimization approach for CVaR introduced by Rockafellar and Uryasev (2000), was further developed in Rockafellar and Uryasev (2002) and Rockafellar et al. (2002). Suppose X is a training set and for each object from this set the class number is known. In other words, discrete-value function $f(x)$ assigns the class for each object from set X. Let J be a total number of classes. The function $f(x)$ splits the set X into a set of subsets $\{X_1,..,X_J\}$,

$$X_j = \left\{\ x \mid x \in X,\ f(x) = j\ \right\}. \tag{31}$$

Let I_j be the number of elements (objects) in the set X_j. We define the CVaR constraints as follows:

$$\left\{\varsigma_j^+ + \tfrac{1}{\kappa}\tfrac{1}{I_j}\sum_{\mathbf{x}\in X_j}\left(U_j(\mathbf{x}) - \varsigma_j^+\right)^+\right\} \leq \\ \leq \left\{-\varsigma_{j+1}^- - \tfrac{1}{\kappa}\tfrac{1}{I_{j+1}}\sum_{\mathbf{x}\in X_{j+1}}\left(-U_j(\mathbf{x}) - \varsigma_{j+1}^-\right)^+\right\} - \delta \qquad j = 1,..,I-1, \tag{32}$$

where ς_j^+ and ς_j^- are free variables; δ and κ are parameters.

For an optimal solution of the optimization problem (16) with the constraints (32), the left-hand part of the inequality is an average of $U_j(\mathbf{x})$ for the largest $\kappa\%$ of objects from the j^{th} class; the right-hand part of the inequality is an average of $U_j(\mathbf{x})$ for the smallest $\kappa\%$ of objects from the $(j+1)^{th}$ class. We will call these values κ-CVaR largest of the j^{th} class and κ-CVaR smallest of the $(j+1)^{th}$ class, correspondingly. The general sense of the constraints is the following: the κ-CVaR largest of the j^{th} class is smaller at least by δ than the κ-CVaR smallest of the $(j+1)^{th}$ class.

3.6 Monotonicity of Separating Function with Respect to Class Number (MSF-Constraints)

By introducing these constraints, we move apart values of the separating functions for the objects belonging to different classes. Moreover, the constraints imply monotonicity of the separating functions with respect to the index denoting the class number. The constraint is set for every pair of the objects belonging to the neighboring classes. For every p and q, so that $f_0\left(\mathbf{x}^p\right)+1=f_0\left(\mathbf{x}^q\right)$, the following constraint is imposed on the optimization model:

$$U_i\left(\mathbf{x}^p\right)-U_i\left(\mathbf{x}^q\right)\geq\delta_{ipq},\, i=0,..,I, \tag{33}$$

where δ_{ipq} are non-negative constants, and I is the number of separating functions. Another way to impose monotonicity on the separating functions with respect to the number of class is to consider δ_{ipq} as variables, and include these variables into the objective function:

$$\sum_{i=1}^{m}\sum_{j=1}^{J}\lambda_j\sigma_i^j-\sum_{i,p,q}\delta_{ipq}. \tag{34}$$

3.7 Model Squeezing Constraints (MSQ-Constraints)

Squeezing constraints (implemented together with feasibility constraints) efficiently adjust the flexibility of the model and control the out-of-sample versus in-sample performance of the algorithm. The constraints have the following form:

$$U_0(\mathbf{x}^p)-U_J(\mathbf{x}^p)\leq S_{f(\mathbf{x}^p)},\, p=1,..,m\,. \tag{35}$$

With these constraints we bound variation of values of different separating functions on each object. Another way to squeeze the spread of the separating functions is to introduce a penalty coefficient for the difference of the functions in (35). In this case the objective function of problem (16) can be rewritten as:

$$\sum_{i=1}^{m}\sum_{j=1}^{J}\lambda_j\sigma_i^j+\sum_{p=1}^{m}\left(m_{f(\mathbf{x}^p)}\left(U_0(\mathbf{x}^p)-U_J(\mathbf{x}^p)\right)\right). \tag{36}$$

The advantage of the squeezing constraints is that it is easy to implement in the linear programming framework.

3.8 Level Squeezing Constraints (LSQ-Constraints)

This type of constraints is very similar to the model squeezing constraints. The difference is that instead of squeezing the boundary functions $U_0(\mathbf{x})$ and $U_J(\mathbf{x})$, we bound the absolute deviation of values of the separating functions from their mean values on each class of objects. The constraints have the following form:

$$\sum_{p:f_0(\mathbf{x}^p)=k} \left| U_j\left(\mathbf{x}^p\right) - \frac{1}{I_k} \sum_{q:f_0(\mathbf{x}^q)=k} U_j\left(\mathbf{x}^q\right) \right| \leq \gamma_{jk}, \tag{37}$$

where γ_{jk} are positive constants, and I_k is the number of objects $\mathbf{x}$ in class K. Similar to constraints (35), constants γ_{jk} can be considered as variables and be included into the objective function:

$$\sum_{i=1}^{m} \sum_{j=1}^{J} \lambda_j \sigma_i^j + \sum_{j,k} \gamma_{jk}. \tag{38}$$

4 Choosing Model Flexibility

This section explains our approach to adjusting flexibility of classification models without strict definitions and mathematical details.

The considered constraints form various models based on the optimization of quadratic separating functions. The models differ in the type of constraints imposed on the separating functions.

The major characteristics of a particular classification model are in-sample and out-of-sample errors. To find a classification model we solve optimization problem (16) constructed for a training dataset. The error of misclassification of a classification model on the training dataset is called the "in-sample" error. The error achieved by the constructed classification model on the objects outside of the training set is called the "out-of-sample" error. The misclassification error can be expressed in various ways. We measure the misclassification (if it is not specified otherwise) by the percentage of the objects on which the computed model gives wrong classification.

Theoretically, classification models demonstrate the following characteristics (see Figure 2). For small training sets, the model fits the data with zero in-sample error, whereas the expected out-of sample error is large. As the size of the training set increases, the expected in-sample error diverges from zero (the model cannot exactly fit the training data). Increasing the size of the training set leads to a larger expected in-sample error and a smaller expected out-of-sample error. For sufficiently large datasets, the in-sample and out-of-sample errors are quite close. In this case, we say that the model is saturated with data.

We say that the class of models A is more flexible than the class of models B if the class A includes the class B. Imposing constraints reduces the flexibility of the model. Figure 3 illustrates theoretical in/out-of-sample characteristics for two classes of models with different flexibilities. For small training sets, the less flexible model gives a smaller expected out-of-sample error (compared to the more flexible model) and, consequently, predicts better than the more flexible model. However, the more flexible models outperform the less flexible models for large training sets (more flexible models have a smaller expected

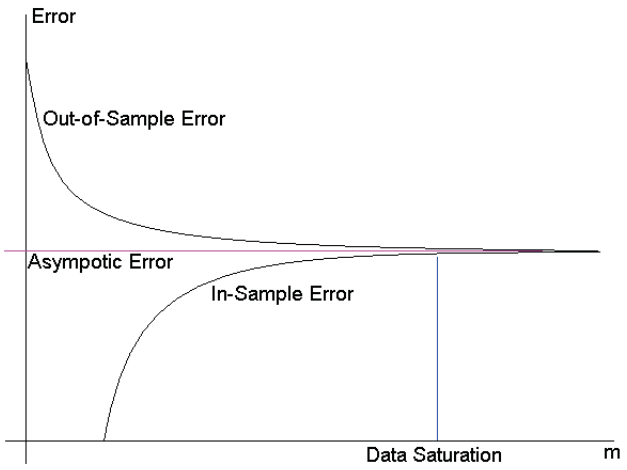

Fig. 2. In-sample and out-of-sample performance vs. size of training dataset.

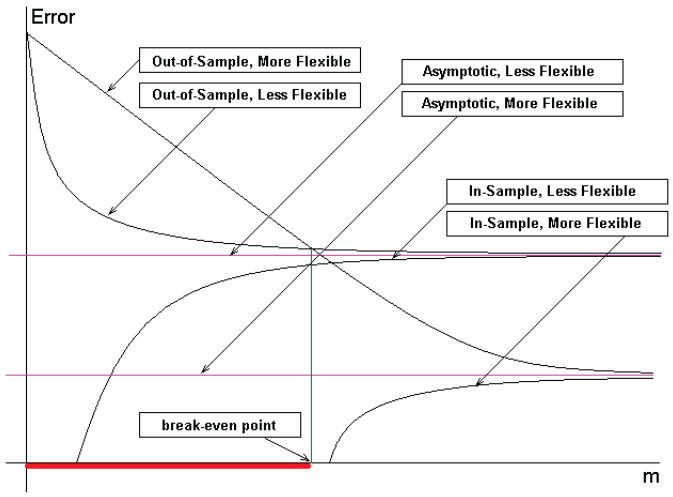

Fig. 3. In-sample and out-of-sample performance for different models.

out-of sample error compared to less flexible models). This is because the less flexible model requires less data for saturation than the more flexible model.

A more flexible model may need a large training dataset to provide a good prediction, while a less flexible model is saturated with data more quickly than a more flexible model. Therefore, for small datasets, less flexible models

tend to outperform (out-of sample) more flexible models. However, for large datasets, more flexible models outperform (out-of-sample) less flexible ones.

We demonstrate these considerations with the following example. For illustration purposes we consider four different models (A, B, C and D) with different levels of flexibility. We applied these models to the classification of a dataset containing 278 in-sample datapoints and 128 out-of-sample datapoints. The in-sample dataset is used to construct linear programming problem (20). The solution of this problem is a set of separating quadratic functions (19). The in-sample and out-of-sample classification is done using the obtained separating functions.

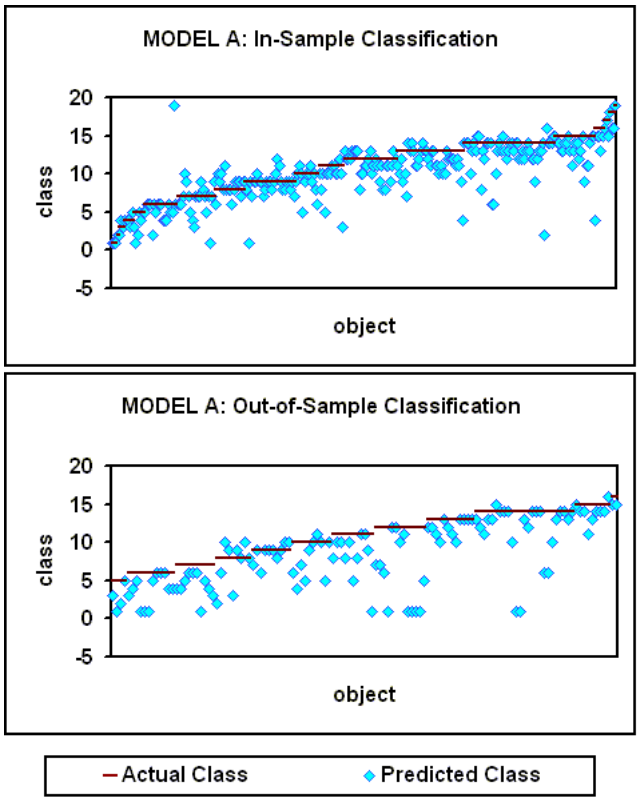

Fig. 4. In-sample and out-of-sample errors for model A. Model A uses parallel separating functions, and is the least flexible model among considered.

Figures 4, 5, 6, and 7 demonstrate the in-sample and out-of-sample performances of the considered models. Although the figures are based on actual computational results related to classification of bonds, they are presented in this paper for illustration purposes only. The horizontal axis of each graph

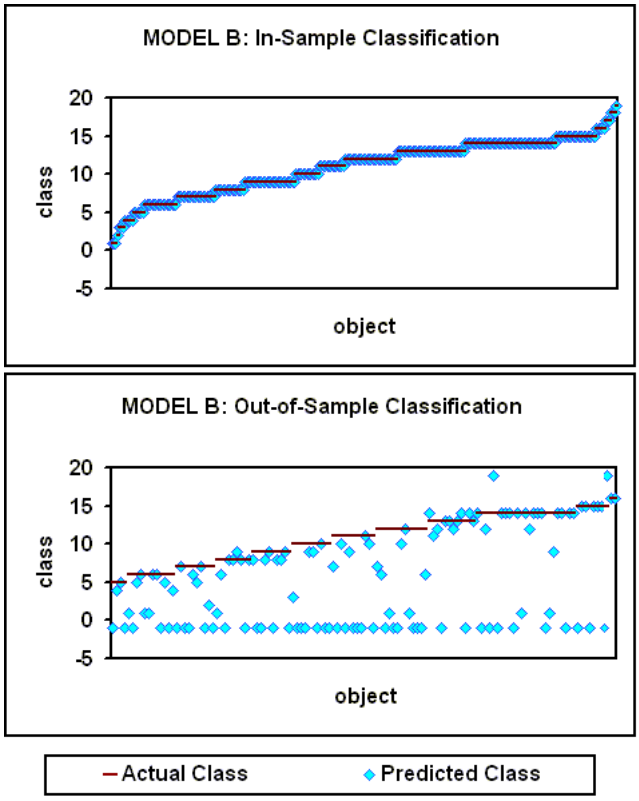

Fig. 5. In-sample and out-of-sample errors for model B. Model B uses separating functions without any constraints, and is the most flexible model. It has the best in-sample performance than Model A, but fails to assign classes for most of out-of-sample objects.

corresponds to the object number, which is ordered according to the object actual class number. The vertical line corresponds to the calculated by the model class number. The solid line on the graph represents the actual class number of an object. The round-point line represents the calculated class. The left graphs show in-sample classification: the class is computed by the separating functions for the objects from the in-sample dataset. The right graphs show out-of-sample classification: the class is computed by the separating functions for the objects from the out-of-sample dataset.

Model A uses "parallel" quadratic separating functions. This case corresponds to the classification model considered by Bugera et al. (2003), where instead of multiple separating functions one utility function was used. The classification with the utility function can be interpreted as a classification with separating functions (15) with the following constraints on the functions

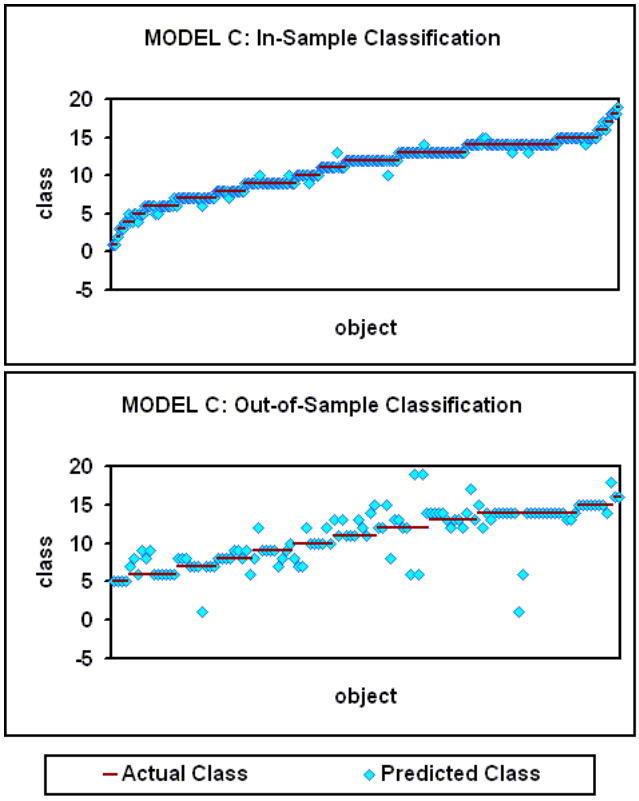

Fig. 6. In-sample and out-of-sample errors for model C. Model C is obtained from Model B by imposing feasibility constraints. This makes the out-of-sample prediction possible for all out-of-sample objects.

$U_\alpha(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{\alpha} x_i x_j + \sum_{i=1}^{n} b_i^{\alpha} x_i + c^{\alpha}$:

$$\forall \alpha_1, \alpha_2 \in \{0, .., J\} : \forall i, j \in \{0, .., n\} : a_{ij}^{\alpha_1} = a_{ij}^{\alpha_2}, b_i^{\alpha_1} = b_i^{\alpha_2} \qquad (39)$$

Constraints (39) make the separating functions parallel in the sense that the difference between the functions remains the same for all the points of the object space.

Figure 4 shows that Model A has large errors for both in-sample and out-of-sample calculations. According to Figure 8, when a model has approximately the same in-sample and out-of-sample errors, the data saturation occurs. Therefore, the out-of-sample performance cannot be improved by reducing flexibility of the model. A more flexible model should be considered.

Model B uses quadratic separating functions without any constraints. This model is more flexible than Model A, because it has more control variables.

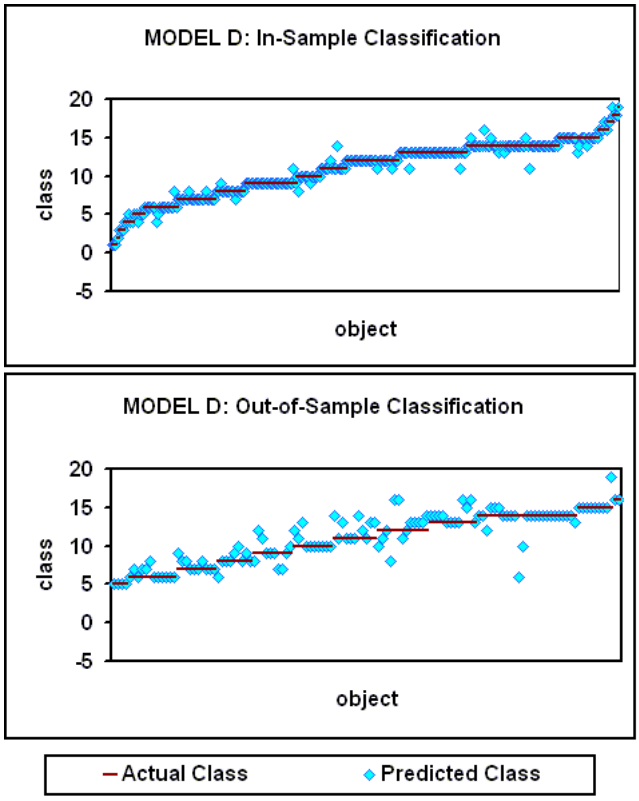

Fig. 7. In-sample and out-of-sample errors for model D. Model D is obtained from Model C by adding CVaR-risk constraints. Model D has the best out-of-sample performance among all the considered models (A, B, C, and D).

According to Figure 5, the in-sample error equals zero for Model B. However, in contrast to the in-sample behavior, the out-of-sample error is larger for Model B than for Model A. Moreover, Model B is not able to assign a class to many of the out-of-sample objects (on the picture the unpredictable result corresponds to the negative number -1 of the class). The separating functions may intersect, and some areas of the object are impossible to classify. For objects from these areas, the condition

$$U_0(\mathbf{x}^*), ..., U_{j-1}(\mathbf{x}^*) > 0 \tag{40}$$

and

$$U_j(\mathbf{x}^*), ..., U_J(\mathbf{x}^*) \leq 0 \tag{41}$$

are not satisfied for any j. It makes the result of the out-of-sample classification uninterpretable.

Model C is obtained by adding the feasibility constraints

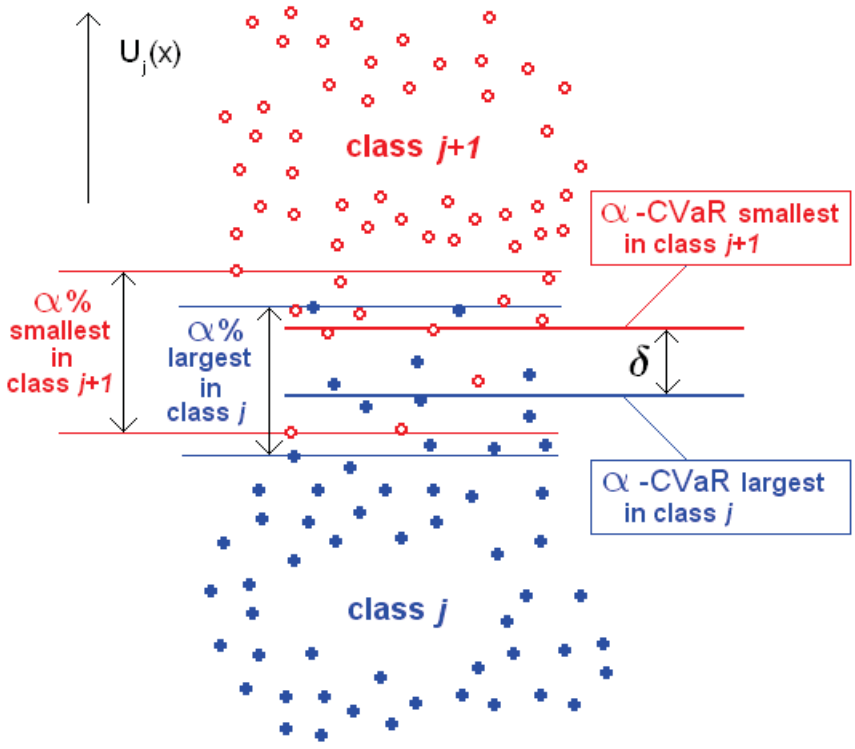

Fig. 8. Nature of risk constraint. For an optimal solution of the optimization problem with the CVaR constraints, the α-CVaR largest of the j^{th} class (the average of $U_j(\mathbf{x})$ for the largest $\alpha\%$ of objects from the j^{th} class) is smaller at least by δ than the α-CVaR smallest of the $(j+1)^{th}$ class (the average of $U_{j+1}(\mathbf{x})$ for the smallest $\alpha\%$ of objects from the $(j+1)^{th}$ class).

$$U_{j-1}(\mathbf{x}_i) \leq U_j(\mathbf{x}_i),\ j \in \{1,..,J\},\ i = 1,..,m\,. \tag{42}$$

to Model B. Since Model C is less flexible then Model B, the in-sample error becomes greater than in Model B (see Figure 6). On the other hand, the model has a better out-of-sample performance than Models A and B. Moreover, a feasibility constraint makes the classification possible for any out-of-sample object. Since there is a significant discrepancy between in-sample and out-of-sample performances of the model, it is reasonable to impose more constraints on the model.

Model D is obtained by adding CVaR-risk constraints to Model C:

$$\left\{\varsigma_j^+ + \frac{1}{\alpha}\frac{1}{I_j}\sum_{\mathbf{x}\in X_j}\left(U_j(\mathbf{x})-\varsigma_j^+\right)^+\right\} \leq \left\{-\varsigma_{j+1}^- - \frac{1}{\alpha}\frac{1}{I_{j+1}}\sum_{\mathbf{x}\in X_{j+1}}\left(-U_j(\mathbf{x})-\varsigma_{j+1}^-\right)^+\right\} - \delta \quad j=1,..,I-1. \tag{43}$$

Figure 7 shows that this model has higher in-sample error compared to Model B, but the out-of-sample performance is the best among the considered models. Moreover, the risk constraint reduces the number of many-class misclassification jumps. Whereas Model C has 8 out-of-sample objects for which the value of misprediction is more than 5 classes, Model D has only one object with a 5-class misprediction. So, Model D has the best out-of-sample performance among all the considered models. This has been achieved by choosing appropriate flexibility of the model.

A model with low flexibility may fit an in-sample dataset well. For models with a high in-sample error (such as Model A), more flexibility can be added by introducing more control variables to the model. In classification models with separating functions, feasibility constraints play a crucial role because they make classification always possible for out-of-sample objects. An excessive flexibility of the model may lead to "overfitting" and poor prediction characteristics of the model. To remove excessive flexibility of the considered models various types of constraints (such as risk constraints) can be imposed. The choice of the types of constraints, as well as the choice of the class of separating functions, plays a critical role for good out-of-sample performance of the algorithm.

5 Error Estimation

To estimate the performance of the considered models, we use the "leave-one-out" cross validation scheme. For the description of this scheme and other cross validation approaches, see, for instance, Efron and Tibshirani (1994). Let us denote by m a number of objects in the considered dataset. For each model (defined by the classes of the constraints imposed on optimization problem (20)), we performed m experiments. By excluding objects $\mathbf{x}^i$ one by one from the set X, we constructed m training sets,

$$Y_i = X \setminus \left\{\mathbf{x}^i\right\}, \, i=1,..,m\,.$$

For each set Y_i, we solved (20) optimization problems with the appropriate set of constraints and found the optimal parameters of the separating functions $\{U_{\alpha^0}(\mathbf{x}),..,U_{\alpha^J}(\mathbf{x})\}$. Further, we computed the number of misclassified objects M_i from the set Y_i. Let us introduce the variable

$$P_i = \begin{cases} 0, \, if \;\; \{U_{\alpha^0}(\mathbf{x}),..,U_{\alpha^J}(\mathbf{x})\} \;\; correctly \;\; classifies \;\; \mathbf{x}^i, \\ 1, \, otherwise\,. \end{cases} \tag{44}$$

The in-sample error estimate $E_{in-sample}$ is calculated by the following formula:

$$E_{in-sample} = \frac{1}{m}\sum_{i=1}^{m}\frac{M_i}{(m-1)} = \frac{1}{m\,(m-1)}\sum_{i=1}^{m} M_i, \tag{45}$$

where M_i is the number of misclassified objects in the set Y_i. In the last formula, the ratio $\frac{M_i}{m-1}$ estimates the probability of an in-sample misclassification in the set Y_i. We calculated the average of these probabilities to estimate the in-sample error.

The out-of-sample error estimate $E_{out-of-samle}$ is defined by the ratio of the total number of the misclassified out-of-sample objects in m experiments to the number of experiments:

$$E_{out-of-sample} = \frac{1}{m}\sum_{i=1}^{m} P_i \quad . \tag{46}$$

The considered leave-one-out cross-validation scheme provides the highest confidence level (by increasing the number of experiments) while effectively improving the predicting power of the model (by increasing the size of the in-sample datasets)

6 Bond Classification Problem

We have tested the approach with a bond classification problem. Bonds represent the most liquid class of the fixed-income securities. A bond is an obligation of the bond issuer to pay cash flow to the bond holder according to the rules specified at the time the bond is issued. A bond pays a specific cash flow (face value) at the time of maturity. In addition, a bond may pay periodic coupon payments.

Although a bond generates a prespecified cash flow stream, it may default, if an issuer gets into financial difficulties or becomes a bankrupt. To characterize this risk, bonds are rated by several rating organizations (*Standard & Poor's* and *Moody's* are major rating companies). Bond rating evaluates the possibility of default of a bond issuer based on the issuer's financial condition and profits potential. The assignment of a rating class is mostly based on the issuer's financial status. A rating organization evaluates the status using expert opinions and formal models based on various factors including financial ratios, such as the ratio of debt to equity, the ratio of current assets to current liabilities, and the ratio of cash flow to outstanding debt.

According to *Standard and Poor's*, bond ratings start at AAA for bonds having the highest investment quality, and end at D for bonds in payment default. The rating may be modified by a plus or minus to show relative standing within the category. Typically, a bond with a lower rating has a lower price than a bond generating the same cash flow, but having a higher rating.

In this study, we have replicated Standard & Poor's ratings of bonds using the suggested classification methodology. The rating replication problem is reduced to a classification problem with 20 classes.

7 Description of Data

For the computational verification of the proposed methodology we used several datasets (A, W, X, Y, Z) provided by the research group of the RiskSolutions branch of Standard and Poor's, Inc. The datasets contain quantitative information about several hundred companies rated by Standard and Poor's, Inc. Each entry in the datasets has fourteen fields that correspond to certain parameters of a specific firm in a specific year. The first two fields are the company name, and the year when the rating was calculated. We used these fields as identifiers of objects for classification. The next eleven fields contain quantitative information about financial performance of the considered company. These fields are used for the decision-making in the classification process. The last field is the credit rating of the company assigned by Standard and Poor's, Inc. Table 1 represents the information used for classification.

Table 1. Data format of an entry of the dataset.

Identifier	Quantitative characteristics	Class
1) Company Name	1) Industry Sector	1) Credit rating
2) Year	2) EBIT interests coverage (times)	
	3) EBITDA interest coverage (times)	
	4) Return on capital (%)	
	5) Operating income/sales (%)	
	6) Free operating cash flow/total debt (%)	
	7) Funds form operations/total debt (%)	
	8) Total debt/capital (%)	
	9) Sales (mil)	
	10) Total equity (mil)	
	11) Total assets (mil)	

We preprocessed the data and rescaled all quantitative characteristics into [-1,1] intervals: all quantitative characteristics are monotonic. Since the total number of rating classes equals 20, we used integer numbers from 1 to 20 to represent the credit rating of an object. The rating is arranged according to credit quality of the objects: the greater value corresponds to the better credit rating.

In the case study, we considered 5 different datasets (A, W, X, Y, Z) to verify the proposed methodology for different sizes of the input data. Each set was split into an in-sample set and an out-of-sample set. The first one was

used for developing a model, and the second one was used for the verification of the out-of-sample performance of the developed model. Table 2 contains information about the sizes of the considered datasets.

Table 2. Sizes of the considered datasets.

Dataset	A	W	X	Y	Z
Size of Set	406	205	373	187	108
Size of In-Sample	278	172	315	157	89
Size of Out-of-Sample	128	33	58	30	19

8 Numerical Experiments

For dataset A we performed computational experiments with 16 models generated by all possible combinations of four different types of constraints. For each dataset W, X, Y, and Z we found a model with the best out-of-sample performance.

For dataset A we applied optimization problem (20) with Feasibility Constraints (F). Besides, we applied all possible combinations of the following four types of constraints: Gradient Monotonicity Constraints (GM), Monotonicity of Separation Function w.r.t. Class Constraints (MSF), Risk Constraints (R), and Level Squeezing Constraints (LSQ). Each combination of the constraints corresponds to one of 16 different models, see Table 3. Although we considered other types of constraints (see Section 3.3), we report the results only for the models with F-, GM-, MSF-, R- and LSQ-constraints.

The numerical experiments were conducted with Pentium III , 1.2GHz in C/C++ environment. The linear programming problems were solved by CPLEX 7.0 package. The calculation results for 16 models for dataset A are presented in Table 4 and Figure 9.

Figure 9 represents in-sample and out-of-sample errors for different models. For each model there are three columns representing the computational results. The first column corresponds to the percentage of in-sample classification error, the second column corresponds to the percentage of out-of-sample classification error, and the third column represents the average out-of-sample classification expressed in percents (100% misprediction corresponds to the case when the difference between actual and calculated classes equals 1).

On Figure 9, the models are arranged according to an average of out-of-sample error (see Table 4, column AVR). Models 0000 and 0010 have zero in-sample error. These two models are the most "flexible" – they fit the data with zero objective in (20). The smallest average out-of-sample error is obtained by 0101- model with MSF and LSQ constraints. Table 4 shows that this model has the lowest maximal misprediction. Models 0100 and 0110 have the

Table 3. List of the considered models.

Model Identificator	Types of Constraints Applied			
	GM	MSF	R	LSQ
0000				
0001				YES
0010			YES	
0011			YES	YES
0100		YES		
0101		YES		YES
0110		YES	YES	
0111		YES	YES	YES
1000	YES			
1001	YES			YES
1010	YES		YES	
1011	YES		YES	YES
1100	YES	YES		
1101	YES	YES		YES
1110	YES	YES	YES	
1111	YES	YES	YES	YES

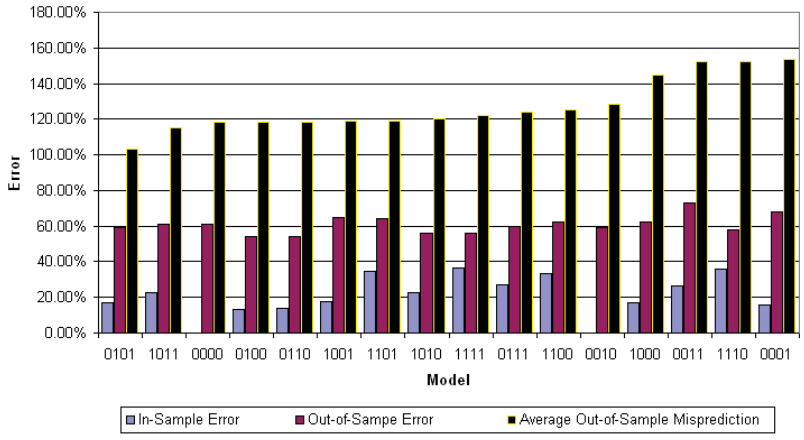

Fig. 9. In-sample and out-of-sample errors for various models.

minimal number of out-of-sample mispredictions (column Error(1)), but the maximal mispredictions (column MAX) for these models are high. It is worth mentioning that imposing constraints on the separating functions increases the computing time. However, for all the considered optimization problems, the classification procedure did not take more than 5 minutes: The CPLEX LP solver, that was used for this study, computed classification model 1111 (all constraints were imposed) in about 5 minutes. Pentium III 1.2Ghz was

used for the study. As for all the other models, it took CPLEX LP solver less than 5 minutes.

Table 4. Calculation results for dataset A

Model	Error(In)	AVR	Correct	Error(1)	Error(2)	Error(3)	Error(4)	MAX
0000	0.00%	1.18	37.37%	62.63%	23.23%	9.09%	3.03%	18
0001	16.00%	1.54	30.30%	67.68%	35.35%	22.22%	13.13%	13
0010	0.00%	1.28	39.39%	60.61%	24.24%	10.10%	5.05%	17
0011	26.70%	1.53	25.25%	74.75%	37.37%	18.18%	12.12%	7
0100	13.50%	1.18	44.44%	55.56%	25.25%	12.12%	7.07%	18
0101	17.00%	1.03	39.39%	60.61%	25.25%	12.12%	4.04%	5
0110	13.60%	1.18	44.44%	55.56%	25.25%	12.12%	7.07%	18
0111	27.30%	1.24	38.38%	61.62%	30.30%	13.13%	5.05%	16
1000	16.90%	1.44	36.36%	63.64%	26.26%	11.11%	10.10%	16
1001	17.50%	1.19	33.33%	66.67%	29.29%	11.11%	7.07%	6
1010	22.60%	1.20	42.42%	57.58%	26.26%	12.12%	8.08%	14
1011	22.50%	1.15	37.37%	62.63%	29.29%	11.11%	7.07%	7
1100	33.10%	1.25	36.36%	63.64%	24.24%	14.14%	7.07%	18
1101	34.50%	1.19	34.34%	65.66%	24.24%	14.14%	7.07%	10
1110	35.70%	1.53	40.40%	59.60%	30.30%	19.19%	9.09%	18
1111	36.70%	1.22	42.42%	57.58%	28.28%	18.18%	8.08%	10

Error(In) = percentage of in-sample misclassifications; AVR = an average error (in number of classes); Correct = percentage of correct out-of-sample predictions; Error(1) = percentage of out-of-sample mispredictions; Error(i) = percentage of out-of-sample prediction errors for more than or equal to i classes; MAX = maximal error for out-of-sample classification.

For the datasets W, X, Y, and Z we imposed various constraints, and selected the models with best performance. For each dataset we choose the best performing models and compared performances of the models with reference models used by *RiskSolutions group* at *Standard and Poor's*, see Table 5. For each data set the table contains the following performance information about the best chosen and the reference model: the average error (expressed in number of classes); the standard deviation; the percentage of correct out-of-sample predictions, the distribution of out-of-sample prediction (expressed in percentage of out-of-sample predictions with a deviation less than or equal to i classes for i=1,2,3); and the description of the constraints included in the best model.

The results of this analysis show that the proposed methodology of classification with the separating functions is competitive with the reference model available from *RiskSolutions group* at *Standard and Poor's*. Although the proposed models do not provide better results compared to the reference model in all the cases, the proposed algorithms have a better or at least comparable performance, for small datasets Z and W.

Table 5. Comparison of best found model with reference model for sets W, X, Y and Z.

Model	W		X		Y		Z	
In-Sample	172		315		157		89	
Out-of-Sample	33		58		30		19	
Model	Best	Refer.	Best	Refer.	Best	Refer.	Best	Refer.
AVR	1.82	1.18	1.67	1.02	1.93	1.13	2.21	2.32
STDV	1.72	0.92	2.45	1.07	2.60	1.50	2.20	2.50
Correct	33.3%	21.2%	24.1%	37.9%	30.0%	50.0%	15.8%	21.1%
1-Class area	42.4%	72.7%	53.4%	70.7%	60.0%	70.0%	42.1%	63.2%
2-Class area	69.7%	87.9%	86.2%	94.8%	80.0%	83.3%	78.9%	68.4%
3-Class area	84.8%	100.0%	94.8%	96.6%	86.7%	86.7%	84.2%	68.4%
4-Class area	93.9%	100.0%	96.6%	98.3%	86.7%	96.7%	84.2%	78.9%
Constraints	Feasibility		Feasibility		Feasibility		Feasibility	
	MSF		MSF		MSF		CVaR	MSF

For each dataset the table has two columns. The first column corresponds to the best found model; the second column corresponds to the reference model from RiskSolutions. Each column contains information about out-of-sample performance of the corresponding model.
AVR = an average error (in number of classes); STDV = a standard deviation; Correct = percentage of correct out-of-sample predictions; i-Class area = percentage of out-of-sample predictions with a deviation less than or equal to i classes; Constraints = description of the constraints included in the best model

9 Concluding Remarks

We extended the methodology discussed in previous work (Bugera et al., 2003) to the case of multi-class classification. The extended approach is based on finding "optimal" separating functions belonging to a prespecified class of functions. We have considered the models with quadratic separating functions (in indicator parameters) with different types of constraints. As before, selecting a proper class of constraints plays a crucial role in the success of the suggested methodology: by imposing constraints, we adjust the flexibility of a model to the size of the training dataset. The advantage of the considered methodology is that the optimal classification model can be found by linear programming techniques, which can be efficiently used for large datasets.

We have applied the developed methodology to a bond rating problem. We have found that for the considered dataset, the best out-of-sample characteristics (the smallest out of sample error) are delivered by the model with MSF and LSQ constraints on the control variables (coefficients of the separating function).

The study was focused on the evaluation of computational characteristics of the suggested approach. As before, despite the fact that the obtained results

are data specific, we conclude that the developed methodology is a robust classification technique suitable for a variety of engineering applications. The advantage of the proposed classification is its simplicity and consistency if compared to the proprietary models which are based on the combination of various techniques, such as expert models, neural networks, classification trees etc.

References

Bugera, V., Konno, H. and Uryasev, S.: 2003, Credit cards scoring with quadratic utility function, *Journal of Multi-Criteria Decision Analysis* **11**(4-5), 197–211.

Efron, B. and Tibshirani, R.J.: 1994, *An Introduction to the Bootstrap*, Chapman & Hall, New York.

Konno, H., Gotoh, J. and Uho, T.: 2000, A cutting plane algorithm for semi-definite programming problems with applications to failure discrimination and cancer diagnosis, *Working Paper 00-5*, Tokyo Institute of Technology, Center for research in Advanced Financial Technology, Tokyo.

Konno, H. and Kobayashi, H.: 2000, Failure discrimination and rating of enterprises by semi-definite programming, *Asia-Pacific Financial Markets* **7**, 261–273.

Mangasarian, O., Street, W. and Wolberg, W.: 1995, Breast cancer diagnosis and prognosis via linear programming, *Operations Research* **43**, 570–577.

Pardalos, P.M., Michalopoulos, M. and Zopounidis, C.: 1997, On the use of multi-criteria methods for the evaluation of insurance companies in greece, *in* C. Zopounidis (ed.), *New Operational Approaches for Financial Modeling*, Physica, Berlin-Heidelberg, pp. 271–283.

Rockafellar, R.T. and Uryasev, S.: 2000, Optimization of conditional value-at-risk, *The Journal of Risk* **2**(3), 21–41.

Rockafellar, R.T. and Uryasev, S.: 2002, Conditional value-at-risk for general loss distributions, *Journal of Banking and Finance* **26**(7), 1443–1471.

Rockafellar, R.T., Uryasev, S. and Zabarankin, M.: 2002, Generalized deviations in risk analysis, *Finance and Stochastics* **10**, 51–74.

Vapnik, V.: 1998, *Statistical Learning Theory*, John Wiley & Sons, New York.

Zopounidis, C. and Doumpos, M.: 1997, Preference desegregation methodology in segmentation problems: The case of financial distress, *in* Zopounidis C. (ed.), *New Operational Approaches for Financial Modeling*, Physica, Berlin-Heidelberg, pp. 417–439.

Zopounidis, C., Pardalos, P., Doumpos, M. and Mavridou, T.: 1998, Multicriteria decision aid in credit cards assessment, *in* C. Zopounidis and P. Pardalos (eds), *Managing in Uncertainty: Theory and Practice*, Kluwer Academic Publishers, New York, pp. 163–178.