SUPPORT VECTOR MACHINE REGRESSION AND CLASSIFICATION

Historical Note. Support vector machine (SVM) analysis is a machine learning tool for regression and classification. It was first proposed by Vladimir Vapnik in his book "The Nature of Statistical Learning Theory", Springer-Verlag, New York, 1995.

Support Vector Regression

The goal of Support Vector Regression is to find a function $f(x_1, ..., x_k)$ that deviates from the observed response y by a value not greater than a pre-specified ε for each training point, and at the same time is as flat as possible.

Let $x = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ & & \cdots & \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}$ be the $n \times k$ matrix of predictor values in the training set. Let $\beta = (\beta_1, \dots, \beta_k)'$ be the column-vector of slopes, and $b = (b_1, \dots, b_n)'$ be the column-vector of intercepts. To find a linear function $f(x) = x\beta + b$ and ensure that it is as flat as possible, we need to find $f(x)$ with the minimal norm $J(\beta) = \frac{1}{2}\beta'\beta$. We also need to observe the constraint that all residuals do not exceed ε, that is, $\left|y_i - \mathrm{x}_i\beta - b\right| \le \varepsilon, \;\; i = 1, ..., n$, where $\mathrm{x}_i = (x_{i1}, ..., x_{ik})$.

It is possible that no such function f exists. To deal with the infeasible constraints, two non-negative **slack variables** ξ_i and ξ_i^* are introduced for each data point. The objective now is to minimize (the expression is termed the **primal formula**) $J(\beta) = \frac{1}{2}\beta'\beta + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$ such that $y_i - \mathrm{x}_i\beta - b \le \varepsilon + \xi_i$, and $\mathrm{x}_i\beta + b - y_i \le \varepsilon + \xi_i^*$, for all $i = 1, \dots, n$. Here the constant C is the **box constraint**, a positive numeric value that controls the penalty imposed on observations that lie outside the ε-margin and helps to prevent overfitting (it is also known as **regularization parameter**). This value determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated.

The optimization problem is computationally simpler if formulated in terms of **Lagrange multipliers** α_i and α_i^* for the ith individual, $i = 1, \dots, n$. This leads to minimization of **Lagrangian** (the expression is termed the **dual formula**):

$$L(\alpha) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\mathrm{x}_i\mathrm{x}_j' \; + \; \varepsilon\sum_{i=1}^{n}\left(\alpha_i + \alpha_i^*\right) \; + \; \sum_{i=1}^{n}\left(\alpha_i^* - \alpha_i\right)y_i,$$

subject to the constraints:

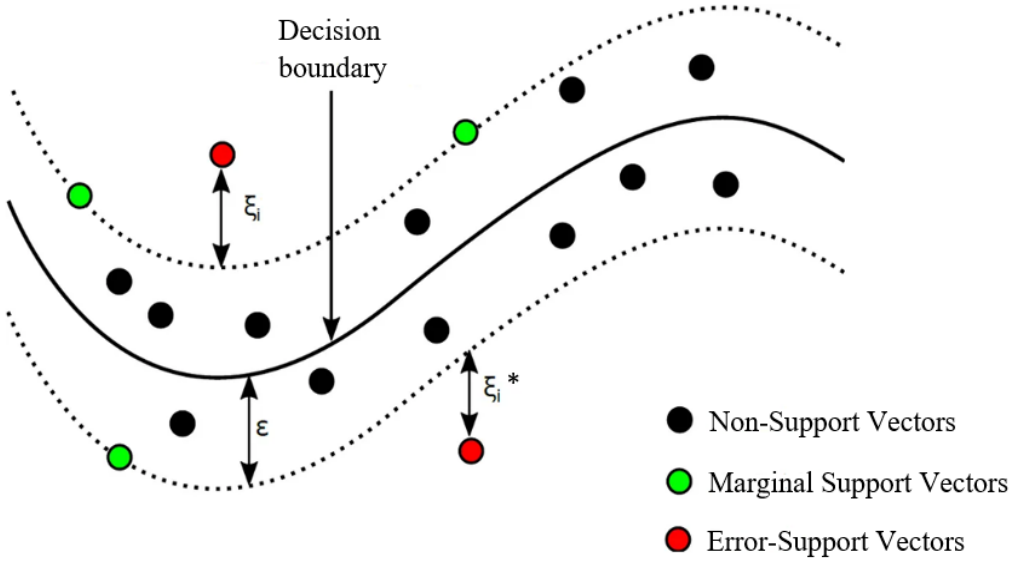
$$\sum_{i=1}^{n}\left(\alpha_i - \alpha_i^*\right) = 0, \;\; \text{and} \;\; 0 \le \alpha_i, \; \alpha_i^* \le C.$$

The β parameter is found as

$$\beta = \sum_{i=1}^{n} \left(\alpha_i - \alpha_i^*\right) \mathrm{x}_i'.$$

The function f is calculated according to the formula:

$$f(x) = \sum_{i=1}^{n} \left(\alpha_i - \alpha_i^*\right) \mathrm{x}_i \, x' + b.$$



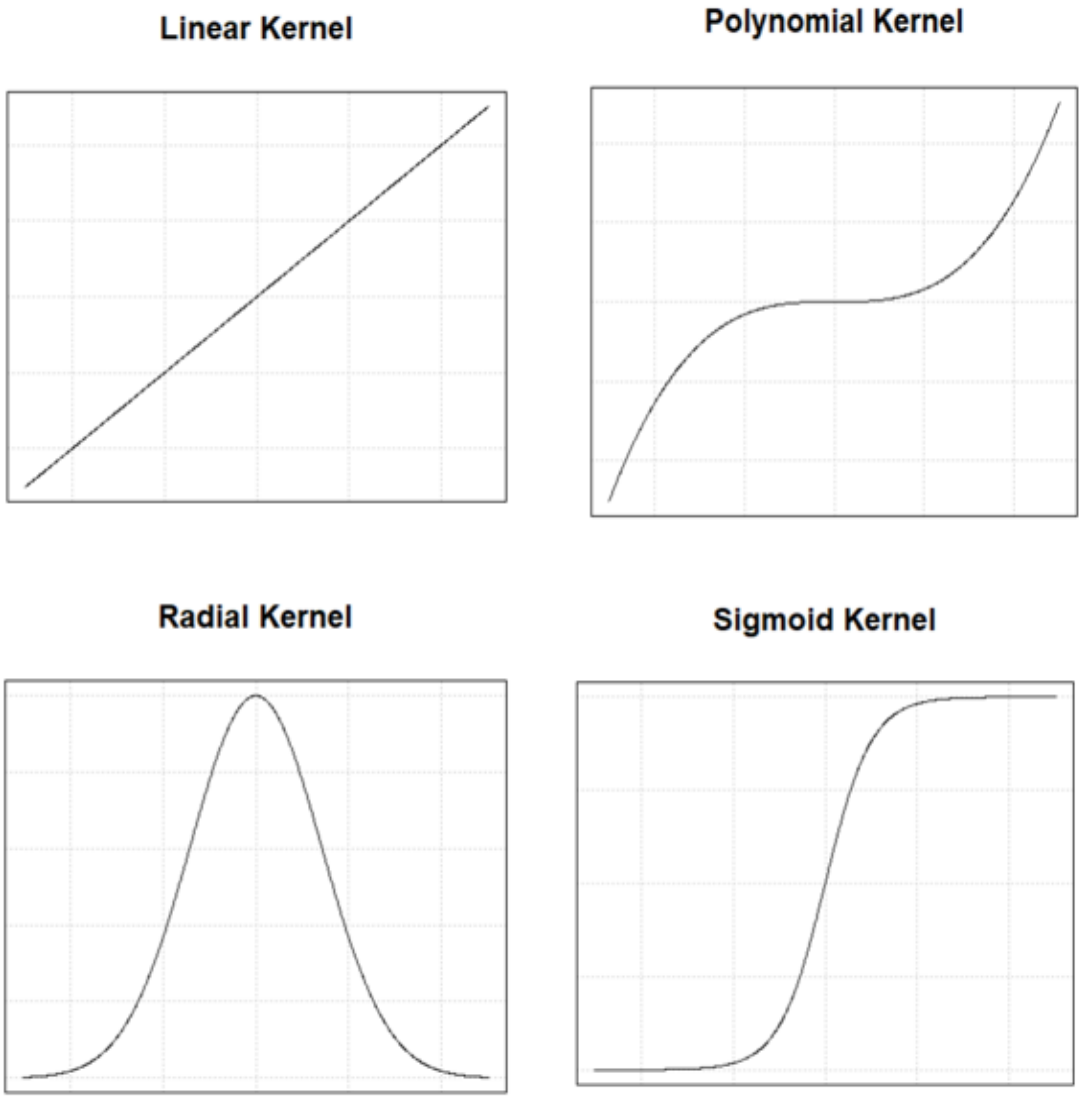
To obtain the optimal solution, the **Karush-Kuhn-Tucker (KKT) complementarity conditions** are used as optimization constraints. For linear support vector regression they are as follows: $\alpha_i(\varepsilon + \xi_i - y_i + \mathrm{x}_i\beta + b) = 0$, $\alpha_i^*(\varepsilon + \xi_i^* + y_i - \mathrm{x}_i\beta - b) = 0$, $\xi_i(C - \alpha_i) = 0$, and $\xi_i^*(C - \alpha_i^*) = 0$, for any $i = 1, \ldots, n$. These conditions indicate that all observations strictly inside the epsilon tube have Lagrange multipliers $\alpha_i = \alpha_i^* = 0$. Those observations for which Lagrange multipliers are non-zero (observations on the boundary of the epsilon tube) are called **support vectors**.

Some regression problems, however, cannot be adequately described using a linear model. In such a case, the Lagrange dual formulation allows it to be extended to nonlinear functions, using kernels. Nonlinear support vector regression finds the coefficients that minimize the Lagrangian

$$L(\alpha) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) G\left(\mathrm{x}_i, \mathrm{x}_j\right) + \varepsilon \sum_{i=1}^{n} \left(\alpha_i + \alpha_i^*\right) + \sum_{i=1}^{n} \left(\alpha_i^* - \alpha_i\right) y_i,$$

where $G\left(\mathrm{x}_i, \mathrm{x}_j\right)$ is the kernel function. Several types of kernels are used: $G\left(\mathrm{x}_i, \mathrm{x}_j\right) = \mathrm{x}_i \mathrm{x}_j'$ is a **linear kernel**, $G\left(\mathrm{x}_i, \mathrm{x}_j\right) = \left(1 + \mathrm{x}_i \mathrm{x}_j'\right)^d$ is a **polynomial** kernel of degree d, $G\left(\mathrm{x}_i, \mathrm{x}_j\right) = \exp\left(-\gamma \left|\left|\mathrm{x}_i - \mathrm{x}_j\right|\right|^2\right)$

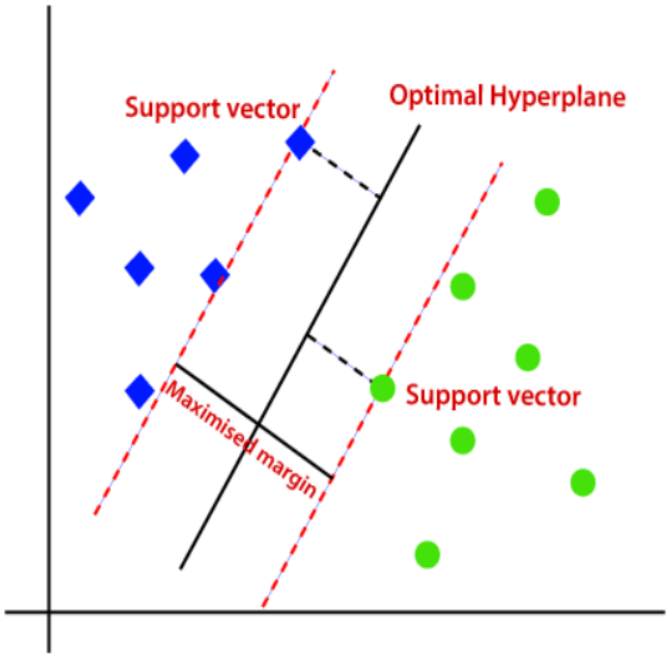
is a **radial basis function (RBF)** or **radial** or **Gaussian** kernel, and $G(\mathrm{x}_i, \mathrm{x}_j) = tanh(\mathrm{x}_i\mathrm{x}'_j)$ is a **sigmoid** kernel, where $tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ is the hyperbolic function (see the illustrations below).



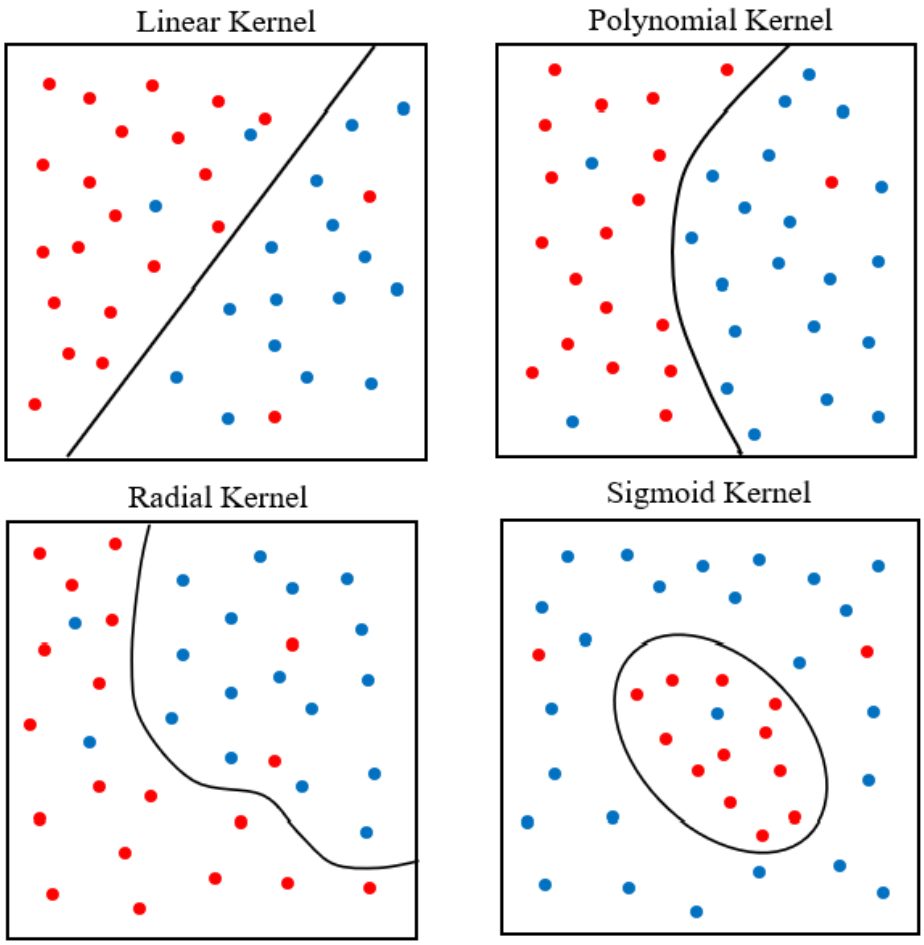
SAS Enterprise Miner doesn't handle this method, so we use R and Python only.

Support Vector Machine for Binary Classifier

For binary response, a support vector machine classifies data by finding the best hyperplane that separates data points of one class from those of the other class. The best hyperplane for an SVM is the one with the largest margin between the two classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points. The **support vectors** are the data points that are closest to the separating hyperplane; these points are on the boundary of the slab.



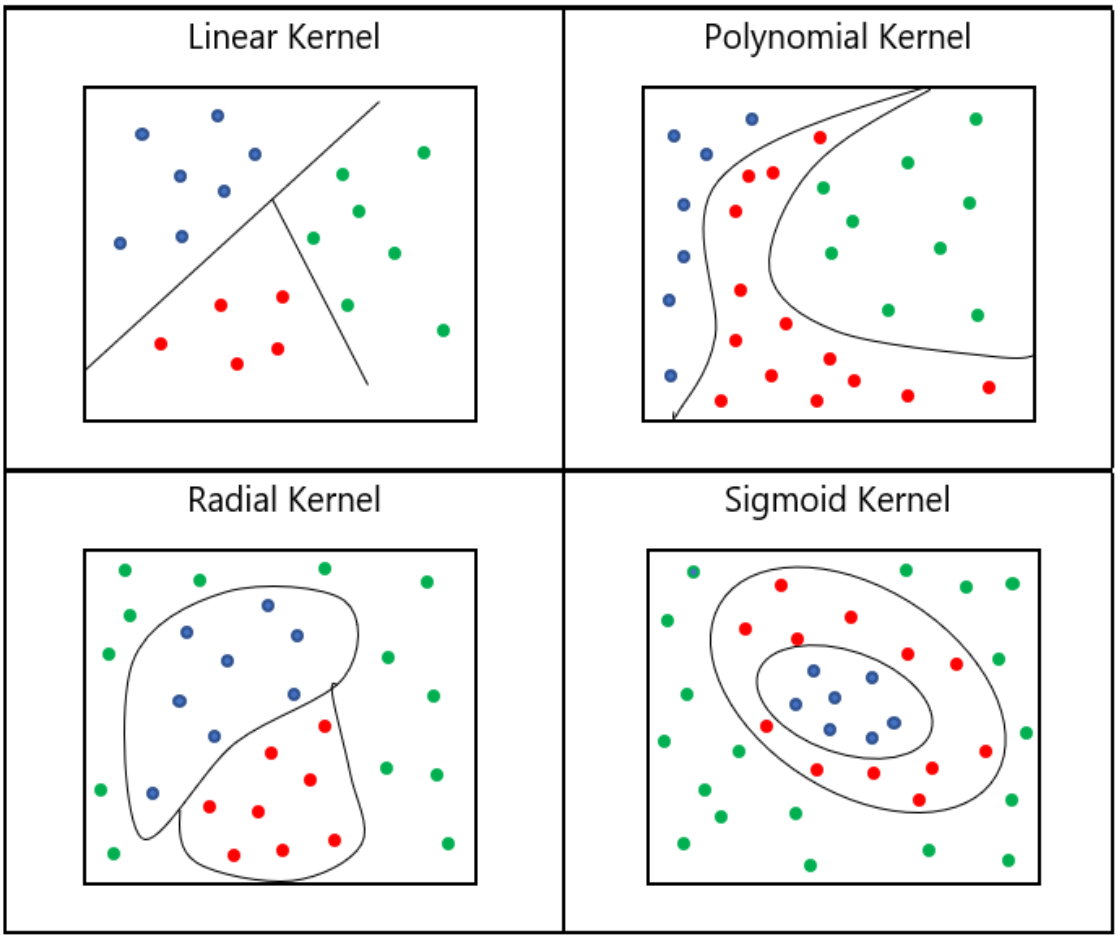
In mathematical terms, the response variable $y_i = \pm 1$ represents the category that the ith individual belongs to. The hyperplane has the equation $f(x) = x\beta + b = 0$. To find the best separating hyperplane, we find β and b that minimize $\beta'\beta$ such that for all $i = 1, \ldots, n$, $y_i f(x_i) \geq 1$. The **support vectors** are the points on the boundary, that is, those for which $y_i f(x_i) = 1$. The dual formulation in this setting is to maximize with respect to α $\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \, y_i \, y_j x_i x_j'$, subject to the constraints $\sum_{i=1}^{n} y_i \alpha_i = 0$, and $0 \leq \alpha_i \leq C, \;\; i = 1, \ldots, n$. Some binary classification problems can't be solved with a simple hyperplane. In this case, kernels (polynomial, radial, or sigmoid) are used. The resulting separating borders are schematically depicted in the figure below. Note different shapes for different kernels.



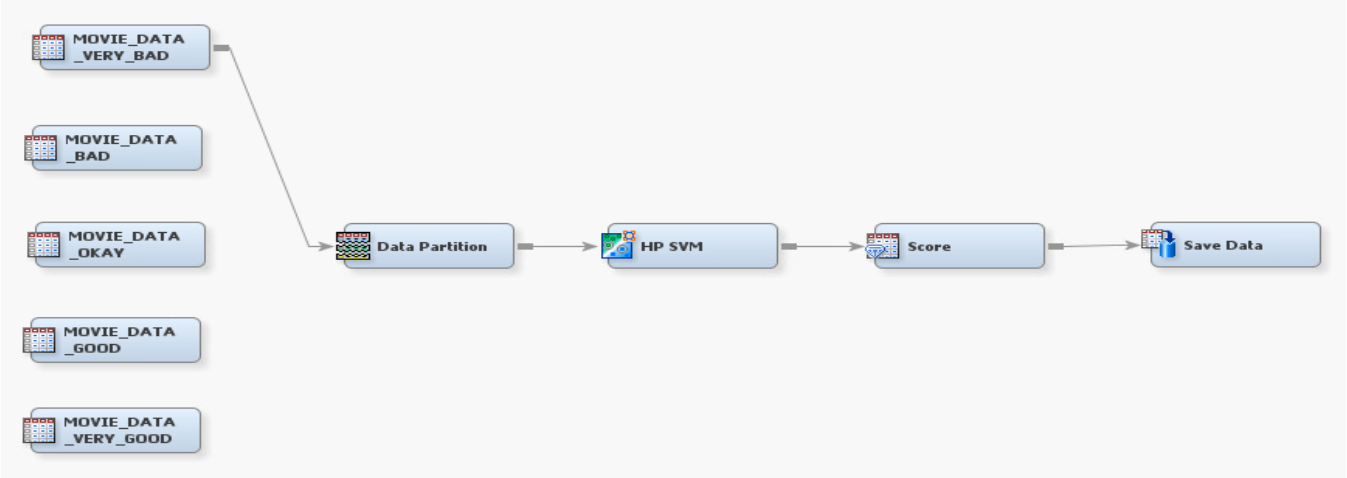
Support Vector Machine for Multinomial Classifier

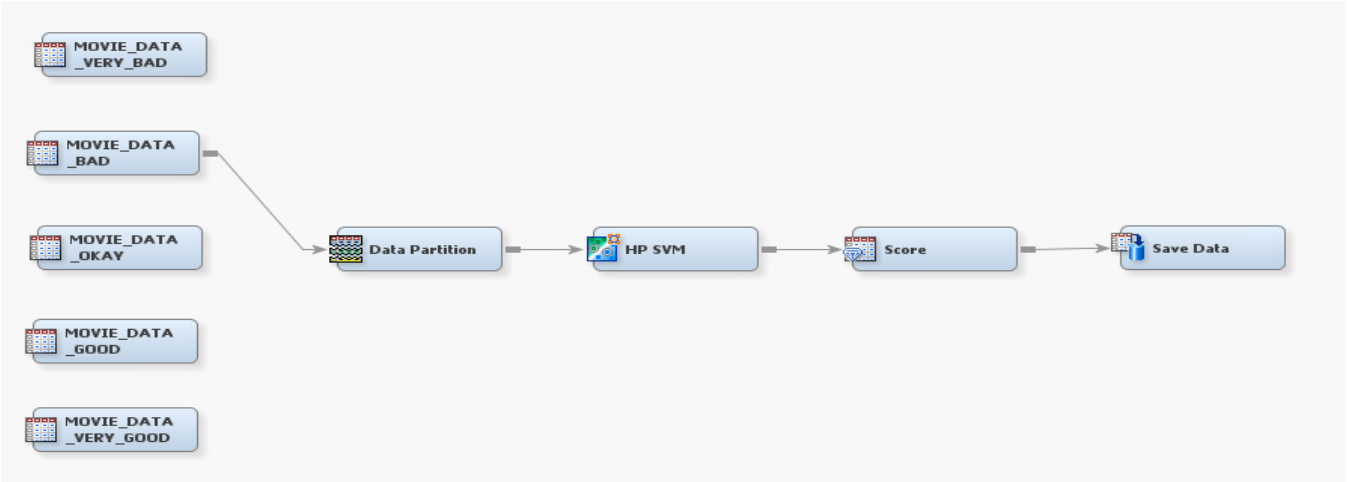
One-versus-all is the most practical approach in the case of multinomial classification problems. Suppose there are c classes. The approach dictates the creation of c indicator variables for the classes and running c separate support vector machines for binary classification. For each class, output the predicted probability and choose the class with the highest value as the predicted class.

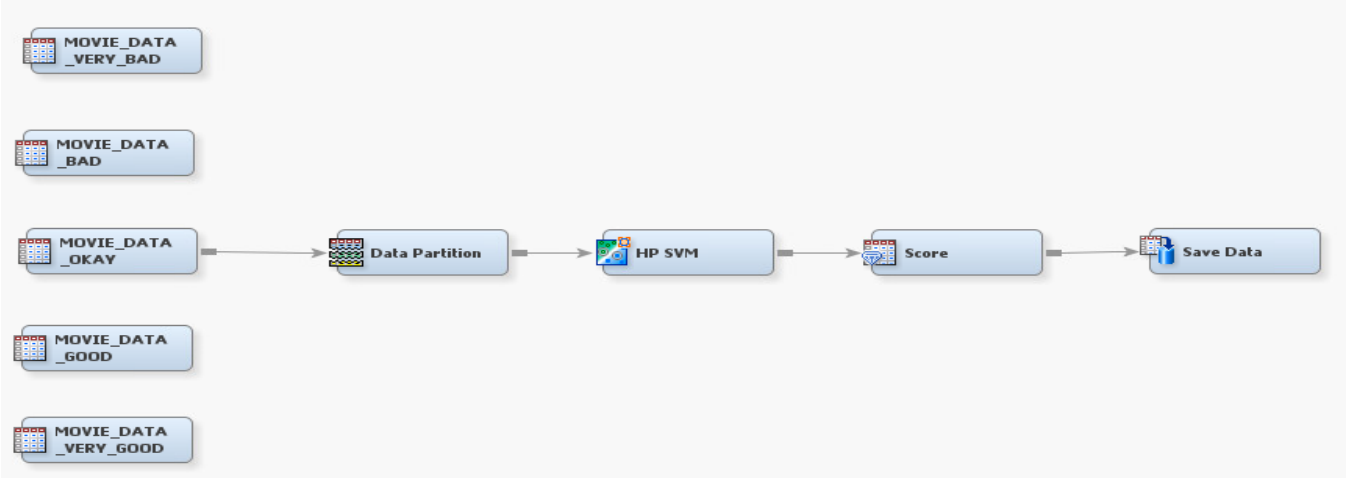
In case of three classes, the separating borders of hyperplanes corresponding to linear, polynomial, radial, and sigmoid kernels are schematically depicted below.

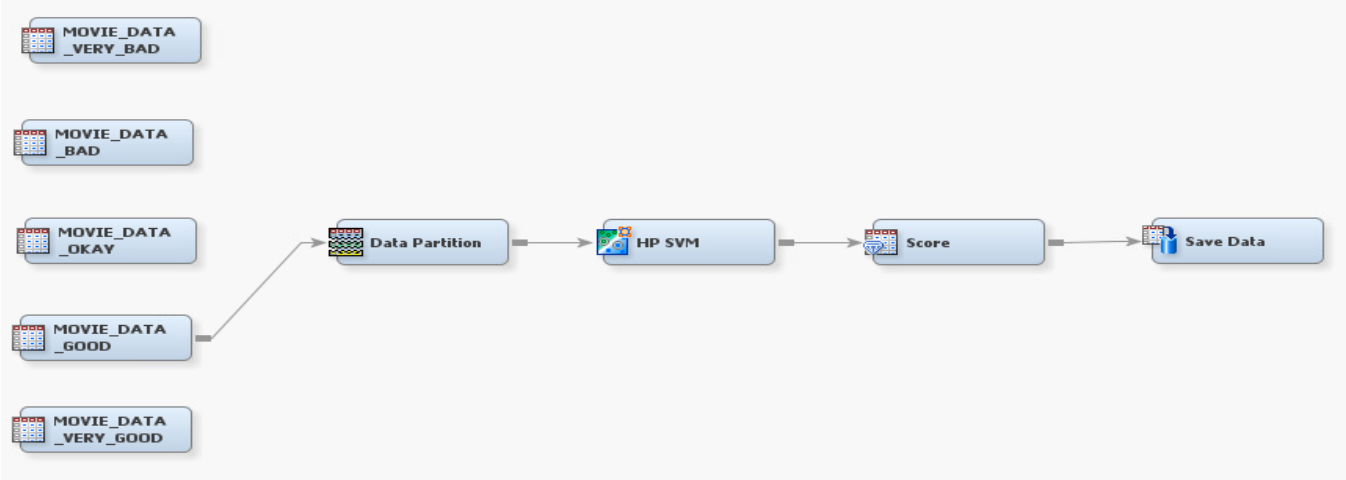


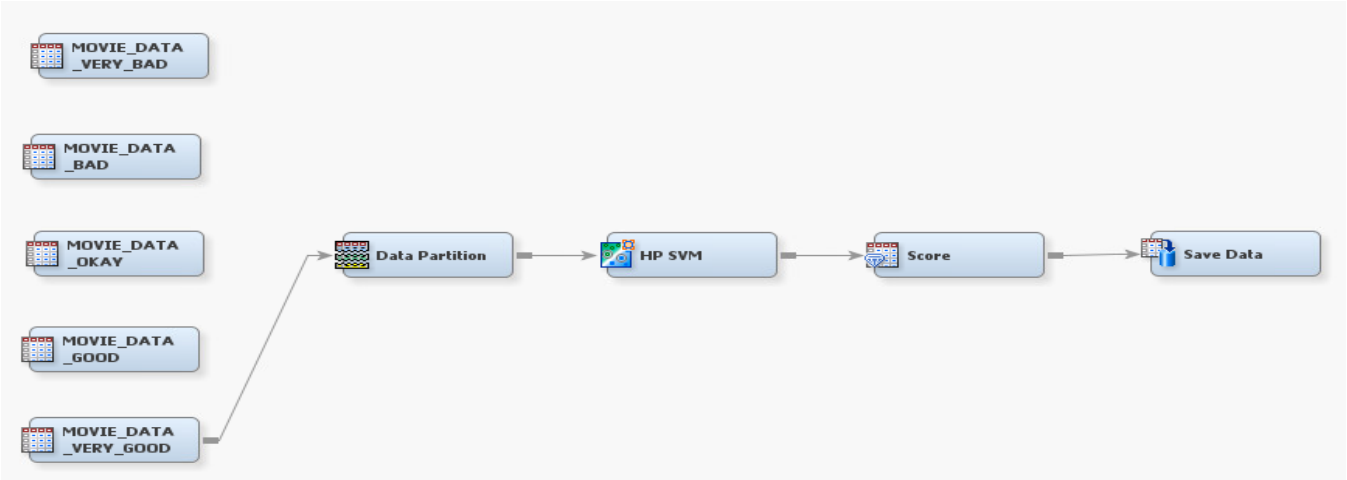
Example. We use the data in the file "movie_data.csv". In SAS Enterprise Miner:
We first create indicator variables for the classes "very bad", "bad", "okay", "good", and "very good" and save the data into the file "movie_data_ind.csv", removing the rating variable. Then in Enterprise Miner, we run the given path with polynomial (quadratic), radial, and sigmoid kernels for each of the five classes separately. We specify each indicator variable as the binary target variable and rename the data set to reflect what class is being modeled. We run the five path diagrams depicted below.


MOVIE_DATA _VERY_BAD
MOVIE_DATA _BAD
MOVIE_DATA _OKAY
MOVIE_DATA _GOOD
MOVIE_DATA _VERY_GOOD
Data Partition
HP SVM
Score
Save Data


MOVIE_DATA _VERY_BAD
MOVIE_DATA _BAD
MOVIE_DATA _OKAY
MOVIE_DATA _GOOD
MOVIE_DATA _VERY_GOOD
Data Partition
HP SVM
Score
Save Data


MOVIE_DATA _VERY_BAD
MOVIE_DATA _BAD
MOVIE_DATA _OKAY
Data Partition
HP SVM
Score
Save Data
MOVIE_DATA _GOOD
MOVIE_DATA _VERY_GOOD


MOVIE_DATA _VERY_BAD
MOVIE_DATA _BAD
MOVIE_DATA _OKAY
Data Partition
HP SVM
Score
Save Data
MOVIE_DATA _GOOD
MOVIE_DATA _VERY_GOOD



Then we collect all the scored data and identified the class with the highest predicted probability. Finally, we compute the accuracy of prediction. □