# Reliability Evaluation Using Monte Carlo Simulation and Support Vector Machine

**C.M. Rocco Sanseverino and J.A. Moreno**

Universidad Central, Facultad de Ingeniería, Apartado 47937, Caracas 1040A, Venezuela
e-mail: {rocco,jose}@neurona.ciens.ucv.ve

**Abstract.** In this paper we propose the use of Support Vector Machine (SVM) to evaluate system reliability. The main idea is to develop an estimation algorithm by training a SVM on a restricted data set, replacing the system performance model evaluation by a simpler calculation. The proposed approach is illustrated by an example. System reliability is properly emulated by training a SVM with a small amount of information.

## 1 Introduction

Reliability evaluation of real engineering systems is often performed using simulation tools. Indeed, reliability indices of a system can be seen as the *expected value* of a test function applied to a system state $\mathbf{X}_i$ (vector representing the state of each element) in order to assess whether that specific configuration corresponds to an operating or failed state [1].

For example, in a *s-t* network, to assess if a selected state $\mathbf{X}_i$ corresponds to an operating or failed state, we need to determine its connectivity, which requires knowledge of the cut sets or path sets of the system [2] or to use a depth-first procedure [3].

In other cases, such as telecommunication networks and other real systems, the success of the network requires that a given state is capable of transporting a required flow. To evaluate a state, for example, the max-flow min-cut algorithm [4-5] can be used.

In general, to determine the state of the system (operating or failed) as a function of the state of its components, it is necessary to evaluate a function (one or more) that is called System Function [6] or Structure Function (SF) [7].

In a Monte Carlo simulation, system reliability is evaluated by generating several systems states and evaluating the SF. Since a large number of SF evaluations are required, a fast, approximated algorithm substitutes its evaluation.

There are several approaches that have been used to address the definition of these approximated algorithms [8-12]. In this work, an empirical model built by training a Support Vector Machine (SVM) is presented. SVM provides a novel approach to the

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 147–155, 2002.

two-category classification problem (operating or failed)[13]. Nowadays, SVM has reached a high level of maturity, with algorithms that are simple, easy to implement, faster and with good performance [14].

The organization of the paper is as follow: Section 2 contains an overview of Monte Carlo approach. The SVM technique is presented in Section 3. Finally, section 4 presents the proposed approach and the results of an example.

## 2 Monte Carlo Approach

Monte Carlo is one of the methods used to evaluate system reliability. The basic idea is to estimate reliability by a relatively simple sampling plan that requires little information about the system under study [15].

For example, a system state depends on the combination of all component states and each component state can be determined by sampling the probability that the component appears in that state [5].

If it is assumed that each component has two states (failure and success) and that component failures are independent events, then the state of the ith component ($x_i$) can be evaluated using its failure probability $Pf_i$ and a random number $U_i$, distributed uniformly between [0,1], as [5]:

$$\mathbf{x_i} = \begin{Bmatrix} 1 & \textbf{(success state) if } \mathbf{U_i \geq Pf_i} \\ -1 & \textbf{(failure state) if } \mathbf{0 \leq U_i < Pf_i} \end{Bmatrix}$$

The jth-state of the system containing NC components is expressed by the vector $\mathbf{X}_j = (x_1, x_2, \ldots, x_{NC})$

In general, to evaluate if $\mathbf{X}_j$ is an operating or failure state, a SF has to be defined and evaluated at $\mathbf{X}_j$. The SF should have the form:

$$\mathbf{SF(X_j)} = \begin{Bmatrix} \mathbf{1} & \textbf{if system is operational in this state} \\ \mathbf{-1} & \textbf{if system is failed in this state} \end{Bmatrix}$$

In the Monte Carlo approach the conceptual scheme for reliability evaluation consist on [1]:

1. Select a system state $\mathbf{X}_j$
2. *Calculate the System Function for the selected state* $\mathbf{X}_j$
3. Update the estimate of the expected value of $SF(\mathbf{X}_j)$ ($E(SF(\mathbf{X}_j))$
4. Calculate the uncertainty of the estimate
5. If the uncertainty is acceptable (within a target tolerance), stop; otherwise return to step 1)

Other sample techniques exist which are more effective [15]. But, in general, all Monte Carlo Methods require SF evaluations. As previously mentioned, the SF depends on the type of reliability evaluation required. In this paper, the emulation of the SF using a SVM is considered.

# 3 Support Vector Machine

Support Vector Machines provide a new approach to the two-category classification problem [13].

SVMs have been successfully applied to a number of applications ranging from particle identification, face identification and text categorization to engine detection, bioinformatics and data base marketing. The approach is systematic and properly motivated by statistical learning theory [16].

SVM is an estimation algorithm ("learning machine") based on [13]:

- Parameter estimation procedure ("Training") from a data set
- Computation of the function value ("Testing")
- Generalization accuracy ("Performance")

Training involves optimization of a convex cost function: there are no local minima to complicate the learning process. Testing is based on the model evaluation using the most informative patterns in the data (the support vectors). Performance is based on error rate determination as test set size tends to infinity [17].

Suppose $\mathbf{X}_i$ is a system state and $y_i$ is the result of applying the SF to $\mathbf{X}_i$: $y_i = SF(\mathbf{X}_i)$.

Consider a set of N training data points $\{(\mathbf{X}_1,y_1), \ldots. (\mathbf{X}_N,y_N)\}$. The main idea is to obtain a hyperplane that separates failed from non-failed in this space, that is, to construct the hyperplane H: $y = \mathbf{w} \cdot \mathbf{X}\text{-}b = 0$ and two hyperplanes parallel to it:

$$H_1: y = \mathbf{w} \cdot \mathbf{X}\text{-}b = +1 \text{ and}$$
$$H_2: y = \mathbf{w} \cdot \mathbf{X}\text{-}b = -1$$

with the condition, that there are no data points between $H_1$ and $H_2$, and the distance between $H_1$ and $H_2$ (the margin) is maximized. Figure 1 shows the situation [18].

The quantities **w** and b are the parameters that control the function and are referred as the weight vector and bias [16].

The problem can be formulated as:

$$\underset{\mathbf{w},b}{\text{Min}} \; \tfrac{1}{2}\, \mathbf{w}^T\mathbf{w}$$
$$\text{s.t} \quad y_i\,(\mathbf{w} \cdot \mathbf{X}\text{-}b) \geq 1$$

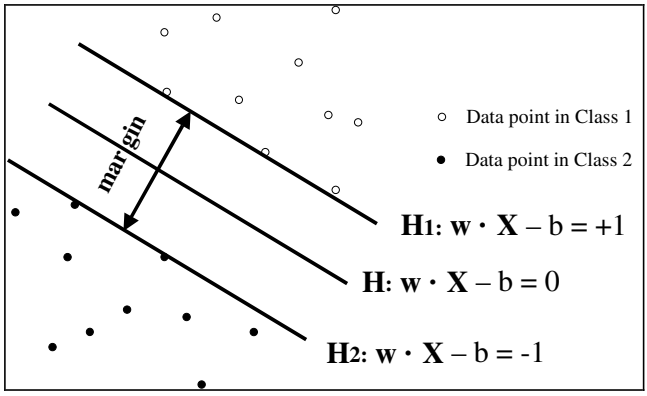


**Fig. 1.** Decision hyperplanes generated by a linear SVM [18]

This is a convex, quadratic programming problem in (**w**, b), in a convex set. Once a SVM has been trained, it is simple to determine on which side of the decision boundary a given test pattern $\mathbf{X}^*$ lies and assign the corresponding class label, using sgn ($\mathbf{w} \cdot \mathbf{X}^* + b$).

When the maximal margin hyperplane is found, only those points which lie closest to the hyperplane have $\alpha_i > 0$ and these points are the support vectors, that is, the critical elements of the training set. All other points have $\alpha_i = 0$. This means that if all other training points were removed and training was repeated, the same separating hyperplane would be found [13]. In figure 2, the points a, b, c, d and e are examples of support vectors [18].

Small problems can be solved by any general-purpose optimization package that solves linearly constrained convex quadratic programs. For larger problems, a range of existing techniques can be used [16].

If the surface separating the two classes is not linear, the data points can be transformed to another high dimensional feature space where the problem is linearly separable. Figure 3 is an example of such transformation [16].

The algorithm that finds a separating hyperplane in the feature space can be obtained in terms of vector in input space and a transformation function $\Phi(\cdot)$. It is not necessary to be explicit about the transformation $\Phi(\cdot)$ as long as it is known that a *kernel function* $\mathbf{K}(\mathbf{X}_i,\mathbf{X}_j)$ is equivalent to a dot product in some other high dimensional feature space [13,16-19].

There are many kernel functions that can be used this way, for example [13,16]:

$\mathbf{K}(\mathbf{X}_i,\mathbf{X}_j) = e^{-||\mathbf{X}_i-\mathbf{X}_j||^2/2\sigma^2}$ the Gaussian radial basis function kernel

$\mathbf{K}(\mathbf{X}_i,\mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j + m)^p$ the polynomial kernel

The Mercer's theorem is applied to determine if a function can be used as kernel function [19].

With a suitable kernel, SVM can separate in the feature space the data that in the original input space was non-separable. This property means that we can obtain nonlinear algorithms by using proven methods to handle linearly separable data sets [19].

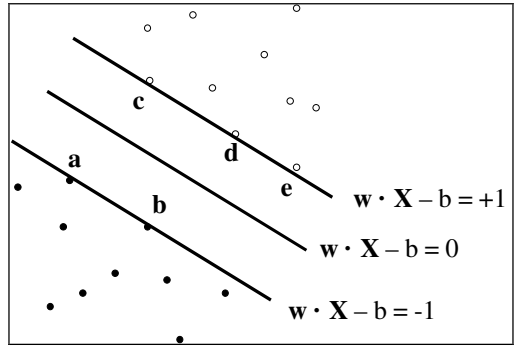


**Fig. 2.** Example of support vectors [18]

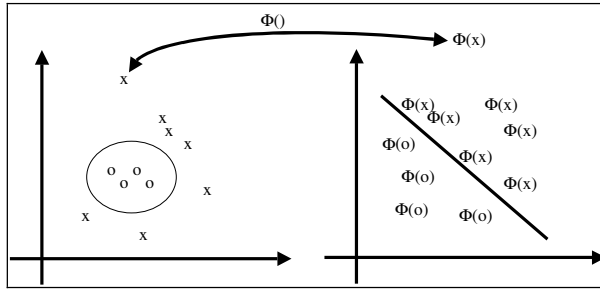


**Fig. 3.** A non-linear separating region transformed in to a linear one [16]

The choice of the kernel is a limitation of the support vector approach. Some work has been done on limiting kernels using prior knowledge [19]. However, it has been noticed that when different kernel functions are used in SVM, they empirically lead to very similar classification accuracy [19]. In these cases, the SVM with lower complexity should be selected.

The performance of a binary classifier is measured using sensitivity, specificity and accuracy [20]:

$$\text{sensitivity} = \frac{TP}{TP + FN}$$

$$\text{specificity} = \frac{TN}{TN + FP}$$

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

TP=Number of True Positive classified cases (SVM correctly classifies)

TN=Number of True Negative classified cases (SVM correctly classifies)

FP= Number of False Positive classified cases (SVM labels a case as positive while it is a negative)

FN= Number of False Negative classified cases (SVM labels a case as negative while it is a positive)

For reliability evaluation, *sensitivity* gives the percentage of correctly classified operational states and the *specificity* the percentage of correctly classified failed states.

## 4 Proposed Approach

In order to apply SVM in Monte Carlo reliability evaluations, a data set using the system state vector $\mathbf{X}_i$ and $y_i = SF(\mathbf{X}_i)$ is built to train the SVM. Since a SVM is going to replace the SF, the data set is built by sampling the configuration space. A system state is randomly generated and then its SF is evaluated. During the system state generation only different states are selected, that is, there are no replicated states in the training data set.

The SVM is trained using a data set with N data while NT data are used to evaluate the SVM performance.

Once a suitable SVM is selected, the system reliability is evaluated by generating a random system state $\mathbf{X}^*_i$ and by evaluating sgn $(\mathbf{w} \cdot \mathbf{X}^*_i + b)$. The process is repeated NM times. At the end, the system reliability can be estimated as:

$$\text{Reliability} = \frac{\text{Number of Operating States}}{NM}$$

### 4.1 Example

Figure 4 shows the system to be evaluated [21]. Each link has reliability $r_i$ and capacity of 100 units. The goal is to evaluate the reliability between the source node *s* and the terminal node *t*. A system failure is defined when the flow at terminal node is less than 200 unit. Using a pure Monte Carlo approach and a max-flow min-cut algorithm as the SF, the estimated reliability is 0.93794.

In this case there are $2^{21}$ possible combinations. The state space is randomly sampled and a training data set with 500 different states is generated. Different kernels were tried and it was found that the best SVM has a second order polynomial kernel, with only 90 support vectors.

Once the SVM is trained, it is used to evaluate the system reliability. Table 1 shows the size of the test data set, the system reliability based on the number of operating states obtained evaluating the SF, the system reliability using SVM and the relative error:

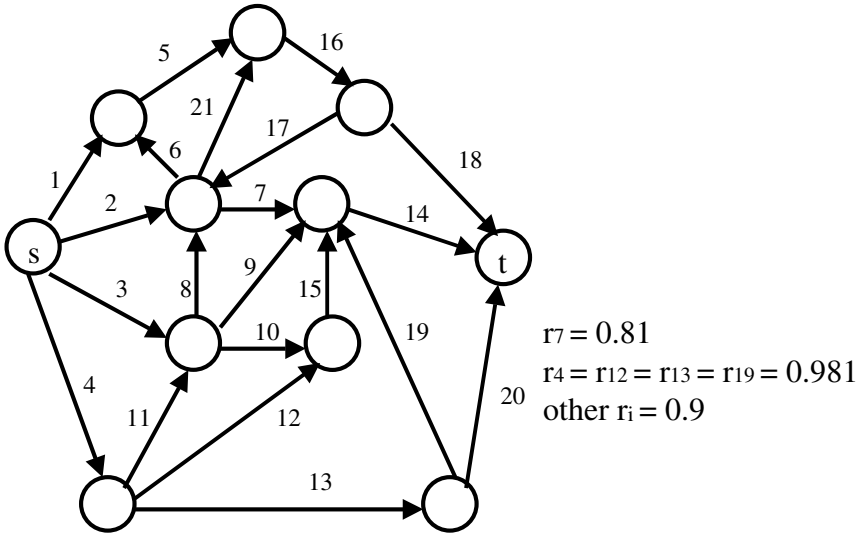


**Fig. 4.** Network for example 4.1 [21]

$$\text{Relative Error } (\%) = \frac{(\text{SF Evaluation} - \text{SVM Evaluation})}{\text{SF Evaluation}} \text{ x } 100$$

System reliability obtained using SF or SVM are very close. It is interesting to note that the system reliability is properly emulated using only a little fraction of the state space (90 support vectors/$2^{21}$ states = 0.0043 %).

The complexity of a SVM is O(NC·NSV) [13], where NC is the number of components in the system and NSV is the number of support vector, while the complexity of the max flow algorithm is $O(N^3)$ [4], where N is the number of nodes. An additional speed up factor can be obtained, using the virtual support vector method or the reduced set method. Description of those techniques can be found in [13].

**Table 1.** SF and SVM Reliability results

| Testing data set size | SF Reliability | SVM Reliability | Relative Error (%) |
|---|---|---|---|
| 1000 | 0.9390 | 0.9480 | -0,96% |
| 5000 | 0.9390 | 0.9410 | -0,21% |
| 10000 | 0.9367 | 0.9382 | -0,16% |
| 20000 | 0.9382 | 0.9398 | -0,17% |

## 5 Conclusions

This paper has presented an approach to evaluate system reliability based on SVM. The excellent results obtained in the example show that the method could be used to evaluate the reliability of a system by emulating the SF with a SVM. In the example presented the SVM, built from a little fraction of the total state space, produces very close reliability estimation with relative error less than 1 %

Additionally the model based on SVM takes the most informative patterns in the data (the support vectors) which can be used to evaluate approximate reliability importance of the components.

## References

1. Pereira M.V.F, Pinto L.M.V.G.: "A New Computational Tool for Composite Reliability Evaluation", *IEEE Power System Engineering Society Summer Meeting*, 1991, 91SM443-2
2. Billinton, R. Allan R.N: *Reliability Evaluation of Engineering Systems, Concepts and Techniques*. Second Edition. Plenum Press. 1992
3. Reingold E., Nievergelt J., Deo N.: *Combinatorial Algorithms: Theory and Practice*, Prentice Hall, New Jersey, 1977
4. Papadimitriou C. H., Steiglitz K.: *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, New Jersey, 1982
5. Billinton, R. Li W.: *Reliability Assessment of Electric Power System Using Monte Carlo Methods*. Plenum Press. 1994
6. Dubi A.: "Modeling of Realistic System with the Monte Carlo Method: A Unified System Engineering Approach", *Proc. Annual Reliability and Maintainability Symposium*, Tutorial Notes, 2001
7. Pohl E.A., Mykyta E.F.: "Simulation Modeling for Reliability Analysis", *Proc. Annual Reliability and Maintainability Symposium*, Tutorial Notes, 2000
8. Marseguerra M., Masine R., Zio E., Cojazzi G.: "Using Neural Networks as Nonlinear Model Interpolators in Variance Decomposition-Based Sensitivity Analysis", *Third International Symposium on Sensitivity Analysis of Model Output*, Madrid, June 2001.
9. Merrill H., Schweppe F.C., "Energy Strategic Planning for Electric Utilities Part I and Part II, Smarte Methodology", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS - 101, No. 2, February 1982
10. Sapio B.: "SEARCH (Scenario Evaluation and Analysis through Repeated Cross- impact Handling): A new method for scenario analysis with an application to the Videotel service in Italy", *International Journal of Forecasting*, (11) 113-131, 1995
11. Mukerji R., Lovell B.: "Creating Data Bases for Power Systems Planning Using High Order Linear Interpolation", *IEEE Transactions on Power Systems*, Vol.3, No.4, November 1988
12. Rahman S., Shrestha G.: "Analysis of Inconsistent Data in Power Planning", *IEEE Transactions on Power Systems*, Vol.6, No.1, February 1991

13. Burges C.: "A tutorial on Support Vector Machines for Patter Recognition", http://www.kernel-machines.org/
14. Platt J.:"Fast Training of Support Vector Machines using Sequential Minimal Optimization", http://www.research.microsoft.com/~jplatt
15. Fishman G.: "A Comparison of Four Monte Carlo Methods for Estimating the Probability of s-t Connectedness", *IEEE Transaction on Reliability*, Vol. R-35, No. 2, June 1986
16. Cristianini N., Shawe-Taylor J.: "*An introduction to Support Vector Machines*", Cambridge University Press, 2000
17. Campbell C.: "Kernel Methods: A survey of Current Techniques", http://www.kernel-machines.org/
18. http://www.ics.uci.edu/~xge/svm/
19. Campbell C.: "An Introduction to Kernel Methods", In R.J. Howlett and L.C. Jain, editors, *Radial Basis Function Networks: Design and Applications*, page 31. Springer Verlag, Berlin, 2000
20. Veropoulos K., Campbell C., Cristianini N.: "Controlling the Sensitivity of Support Vector Machines", *Proceedings of the International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 1999 (IJCAI99)*, Workshop ML3, p. 55-60.
21. Yoo Y.B., Deo N.: "A Comparison of Algorithm for Terminal-Pair Reliability", *IEEE Transaction on Reliability*, Vol. 37, No. 2, June 1988