Learning Boundaries on Military Operational Plans from Simulation Data*

Johan Schubert, Anna Linderhed
Division of Information Systems
Swedish Defence Research Agency
SE-164 90 Stockholm, Sweden
johan.schubert@foi.se, anna.linderhed@foi.se
http://www.foi.se/fusion/

Abstract-**In this paper we learn indicators from simulated data that serve as boundaries on military operational plans of an expeditionary operation. These are boundaries that an operation must not move beyond without risk of drastic failure. We receive simulated and evaluated partial patterns of plan instances from a simulation-based decision support system that are patterns of integer strings. These partial patterns are clustered by an unsupervised neural Potts spin clustering method into clusters where the instances in each cluster have similar characteristics and outcomes. This gives all partial patterns a classification. We use a Dempster-Shafer theory based factor screening method on each pair of clusters, where all activities of the plan are evaluated as to their differentiating capacity between the two sets of partial plan instances. All plan instances are projected from their full integer string representation to a subset of factors with high differentiating capacity. We apply supervised learning by Support Vector Machine using the previous classification to learn support vectors for each pair of clusters given the projected plan instances of these clusters. From these support vectors we derive a lower dimension hyper plane that will serve as one of the indicators. One indicator from each pair of clusters will make up a full set of indicators for this operational plan. This set of indicators can be provided to the intelligence service and used during execution of the plan for assessment of its progress, and serve as a warning bell if the plan approaches an indicator which it should not proceed beyond.**

Keywords-military operational planning; effects-based planning; indicators; partial patterns; clustering; neural network; Potts spin; Dempster-Shafer theory; factor screening, support vector machine; hyper plane.

I. Introduction

In this paper we learn indicators from simulated data that serve as boundaries on military operational plans of an expeditionary operation. These indicators can be provided to the intelligence service for monitoring. We simulate and evaluate alternative plan instances of the overall military plan [1, 2]. This is performed in a simulation-based decision support system that model plans according to the effects-based planning approach. We model the plan and evaluate alternative plan instance on how well they are able to drive the entire state of the simulation model, simulating a large set of actors, towards a predetermined military end state. These plan instances are evaluated as to their performance and clustered by neural Potts spin clustering [3, 4] into clusters where all plan instances have both common characteristics and outcomes [5, 6]. The idea is that these clusters, whenever they contain plan instances of good performance, are a robust set of alternative plans that can be used for minor dynamic replanning whenever necessary.

To differentiate between minor replanning and whenever major replanning becomes necessary in order to avoid drastic negative consequences of plans that begin to deviate substantially from the initial planning, we adopt indicators as warning bells. An indicator is the boundary between two clusters beyond which drastic changes can occur. The indicators are represented as high dimensional hyper planes. We use a support vector machine (SVM) [7, 8] that learn support vectors for each pair of clusters and derive the hyper planes from the support vectors.

In order to reduce the dimensionality of the hyper planes whenever the indicators are provided for human analysis we use Dempster-Shafer theory [9] to screen each activity of the plan. (If the hyper planes are intended for further machine use this may not be necessary.) The idea is to find subsets of alternatives that partition the set of alternative ways to perform the activity, one subset for each cluster. This is done individually for each pair of clusters to find factors of the plan with the highest discriminating capacity between this pair of clusters.

In section II we present a method for clustering all plan instances into clusters with common characteristics and outcomes. In section III we screen all factors of the plan individually for each pair of clusters in order to find the factors with the highest differentiating capacity for this pair of clusters, and reduce the clustered plan instances to these factors. In section IV we use a support vector machine to learn support vectors of each pair of clusters using the reduced plan instances. From these vectors we derive low dimensionality hyper planes that work as the sought after indicators. In section

*This work was supported by the FOI research project "Real-Time Simulation Supporting Effects-Based Planning", which is funded by the R&D programme of the Swedish Armed Forces.

978-1-4577-0653-0/11/$26.00 ©2011 IEEE

V we elaborate on the usage of hyper planes as indicators. Finally, conclusions are drawn (section VI).

II. Clustering partial patterns of plan instances

We cluster the partial patterns of plan instances that are similar in structure and consequences. Similar in structure means that they have more or less carried out similar alternative activities. Similar in consequences means that they travel on average the same distance towards the end state for each carried out activity.

A typical plan instance P_1 is

[1 2 41 42 3 4 43 12 10 54 51 16 47 55 22 26 31 33 58 60 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1902.0 1645.2 256.8],

where all but the last three numbers in this sequence is the number of the selected alternative for each activity in this plan instance. For example, activity number 3 (i.e., position 3 in the sequence) takes alternative number 41. Note, that alternatives for different activities are numbered with running numbers in no particular order, they do not restart at 1 for each new activity, and "0s" are missing values corresponding to future activities that has not yet been simulated for this particular partial plan instance. If all possible plan instances are represented in a tree, a full plan instance (with no missing values) is a path from the root of tree down to one particular leaf. Obviously, the depth of the tree is the length of the sequence minus three (i.e., not counting the *f*, *g* and *h* estimates). Plan instance P_1 above corresponds to a sequence of 20 specific simulations for the first 20 activities where the activities take the numbered alternative listed in the sequence as its input parameter [1].

The last three parameters are different evaluation measures called *f*, *g* and *h* ($f = g + h$). They are distance measures calculated from changes in the scenario state and used in an A*-search algorithm.

During simulation an assessment is made of how well each activity is performed. This is done by the function *g*, a function that measures the consequence of all performed activities as a distance from the initial state $S_{0,0}$ to the current simulated state S_{x,y_x} [1]. Function *h* is a heuristic estimate of the remaining distance from S_{x,y_x} to the end state. It is not used in clustering. We have,

$$g(y_x) = \sum_{i=0}^{x-1} \Delta(S_{i,y_i}, S_{i+1,y_{i+1}}) . \quad (1)$$

We observe the difference in consequences between two plans. We compare the incremental changes of *g* called Δg as each plan P_i and P_j progresses down the sequence of additional activities A_k, where

$$\Delta g(P_i.A_k) = g(P_i.A_k) - g(P_i.A_{k-1}) \quad (2)$$

and *i* and *j* are indices for different plan instances and *k* is the index for activities.

In addition, we need to measure the structural distance between two plans. This is done by the Hamming [10] distance *Ha* which measures the structural distance between P_i and P_j. We have,

$$Ha(P_i.A_k, P_j.A_k) = \begin{cases} 0, P_i.A_k = P_j.A_k \\ 1, P_i.A_k \neq P_j.A_k \end{cases} \quad (3)$$

when both activities $P_i.A_k$ and $P_j.A_k$ exists within the simulated sequences P_i and P_j, otherwise 0 by definition.

Using this measure, we compare each activity in two different plans to calculate the structural distance between the plans. For each activity we observe the alternative chosen in both plans.

We put these two measures together into an interaction functions that measures the overall distance between plan P_i and P_j [1].

We have,

$$J_{ij}^{-} = 1 - \left[1 - \sum_k Ha(P_i.A_k, P_j.A_k)\right] \times \left[1 - \sum_k \left|\Delta g(P_i.A_k) - \Delta g(P_j.A_k)\right|\right]. \quad (4)$$

We partition the set of all simulated plans into clusters using the Potts spin model [3] in such a way as to minimize the overall sum of all interactions J_{ij}^{-} within each cluster.

The Potts spin problem consists of minimizing an energy function

$$E = \frac{1}{2} \sum_{i,j=1}^{N} \sum_{a=1}^{q} J_{ij}^{-} W_{ia} W_{ja} \quad (5)$$

by changing the states of the spins W_{ia}'s, where $W_{ia} \in \{0, 1\}$ and $W_{ia} = 1$ means that plan P_i is in cluster *a*. This model serves as a clustering method if J_{ij}^{-} is used as a penalty factor when plan P_i and P_j are in the same cluster.

For computational reasons we use a mean field model, where spins are deterministic with $V_{ia} = \langle W_{ia} \rangle$, $V_{ia} \in [0, 1]$, in order to find the minimum of the energy function. The Potts mean field equations are formulated [4] as

$$V_{ia} = \frac{e^{-H_{ia}[V]/T}}{\sum_{b=1}^{K} e^{-H_{ib}[V]/T}} \quad (6)$$

where

$$H_{ia}[V] = \sum_{j=1}^{N} J_{ij} V_{ja} - \gamma V_{ia} \quad (7)$$

and T is a parameter called the temperature that is used to control the influence of the interaction. This is a system parameter initialized to

$$\frac{1}{K} \cdot max(-\lambda_{min}, \lambda_{max}), \quad (8)$$

where K is the number of clusters, and λ_{min} and λ_{max} are the extreme eigenvalues[1] of M, where

$$M_{ij} = \bar{J}_{ij} - \gamma\delta_{ij}. \quad (9)$$

In order to minimize the energy function (6) and (7) are iterated until a stationary equilibrium state has been reached for each temperature. Then, the temperature is lowered step by step by a constant factor until $\forall i, a.\ V_{ia} = 0, 1$ in the stationary equilibrium state, Fig. 1, [5, 6].

III. Evidential screening of factors for activities with highest differentiating capacity

In this section we investigate which activities of the plan have most differentiating capacity for each pair of clusters using Dempster-Shafer theory. These are the activities that should be part of an indicator projected from $(\mathbb{Z}^+)^{|\{A_k\}|-1}$ to a lower dimension onto the set of these activities. This will reduce, by the same factor, the dimensionality of the support vectors and hyper planes that are learned from all plan instances of reduced dimensionality (section IV) with only the most differentiating activities remaining.

A. Dempster-Shafer theory

In Dempster-Shafer theory belief is assigned to a proposition by a basic belief assignment. The proposition is represented by a subset A of an exhaustive set of mutually exclusive possibilities, a frame of discernment Θ.

The basic belief assignment (or mass function) is a function from the power set of Θ to [0, 1].

$$m: 2^{\Theta} \rightarrow [0, 1] \quad (10)$$

whenever

$$m(\emptyset) = 0 \quad (11)$$

and

$$\sum_{A \subseteq \Theta} m(A) = 1 \quad (12)$$

where $m(A)$ is called a basic belief number, that is the belief committed exactly to A.

The total belief in a proposition A is obtained from the sum of belief for those propositions that are subsets of the proposition in question and the belief committed exactly to A

[1] In MATLAB a vector of eigenvalues is returned by the function `eig(M)`.

INITIALIZE

K (number of clusters); N (number of plans);

$\bar{J}_{ij} \quad \forall i, j$;

$s = 0$; $t = 0$; $\varepsilon = 0.001$; $\tau = 0.9$; $\gamma = 0.5$;

$T^0 = T_c$ (a critical temperature) $= \frac{1}{K} \cdot max(-\lambda_{min}, \lambda_{max})$, where λ_{min} and λ_{max} are the extreme eigenvalues of M,

where $M_{ij} = \bar{J}_{ij} - \gamma\delta_{ij}$;

$V^0_{ia} = \frac{1}{K} + \varepsilon \cdot rand[0,1] \quad \forall i, a$;

REPEAT

- REPEAT−2
  - $\forall i$ Do:
    - $H^s_{ia} = \sum_{j=1}^{N} \bar{J}_{ij} V_{ja}^{\begin{cases} s+1, & j<i \\ s, & j \geq i \end{cases}} - \gamma V^s_{ia} \quad \forall a$;
    - $F^s_i = \sum_{a=1}^{K} e^{-H^s_{ia}/T^t}$;
    - $V^{s+1}_{ia} = \frac{e^{-H^s_{ia}/T^t}}{F^s_i} + \varepsilon \cdot rand[0,1] \quad \forall a$;
    - $s = s + 1$;
- UNTIL−2
  - $\frac{1}{N}\sum_{i,a} \left| V^s_{ia} - V^{s-1}_{ia} \right| \leq 0.01$;
- $T^{t+1} = \tau \cdot T^t$;
- $t = t + 1$;

UNTIL

$\frac{1}{N}\sum_{i,a} (V^s_{ia})^2 \geq 0.99$;

RETURN

$\left\{ \chi_a \middle| \forall S_i \in \chi_a.\ \forall b \neq a\ V^s_{ia} > V^s_{ib} \right\}$;

Fig. 1. Clustering algorithm.

$$\text{Bel}(A) = \sum_{B \subseteq A} m(B) \quad (13)$$

where Bel(A) is the total belief in A and Bel($\cdot$) is called a belief function

$$\text{Bel}: 2^{\Theta} \rightarrow [0, 1] \quad (14)$$

A subset A of Θ is called a focal element of A if the basic belief number for A is non-zero.

B. Maximum differentiating capacity

The most differentiating activities are found by investigating the maximum differentiating capacity of two disjoint subsets of the frame of discernment $\Theta_k =$

$\{P_i.A_k | \chi_j, A_k\}$ one for each cluster, i.e., the set of possible values of A_k over all clusters χ_j, where i, j and k are indices for different plan instances, clusters and activities, respectively. Note that Θ_k is not dependent on cluster, but varies for each activity.

We develop a method, which for each cluster χ_j calculate histograms for all activities A_k over all partial plan instances that we receive from the simulation-based decision support system.

From all plan instances P_i in each cluster χ_j we build the histogram over all activities A_k. We have,

$$h_{\chi_j}^{A_k}(l) = \sum_i \begin{cases} 1, & P_i.A_k = l \\ 0, & P_i.A_k \neq l \end{cases} \qquad (15)$$

where $l \in \{P_i.A_k | \chi_j, A_k\}_c$ and $l = 0$ is a missing value due to a partial plan instance that provides no information regarding A_k.

In Fig. 2 and Fig. 3 we provide one example of histograms calculated by (15) for activity A_8, the activity with the highest differentiating capacity, for two clusters χ_1 and χ_2, respectively.

The histogram in Fig. 2 is a summation for activity A_8 over all plan instances in cluster χ_1 of how many times each alternative was carried out.

What we are looking for are activities where there are alternatives with very different frequencies for the two clusters χ_1 and χ_2, where some alternatives have much higher frequency for one cluster, and other alternatives have much higher frequency for the other cluster. When this is the case we have an activity with high discriminating capacity.

Comparing Fig. 2 and Fig. 3 we observe directly that A_8 has high discriminating capacity since the frequency of alternative 12 is much higher for χ_2 than for χ_1, i.e., $h_{\chi_1}^{A_8}(\{12\}) \ll h_{\chi_2}^{A_8}(\{12\})$ (first bar in Fig. 2 and Fig. 3) and the frequency of alternative 13 is much higher for χ_1 than the frequency for χ_2, i.e., $h_{\chi_1}^{A_8}(\{13\}) \gg h_{\chi_2}^{A_8}(\{13\})$ (second bar in Fig. 2 and Fig. 3).

However, our interest is in finding different subsets of alternatives with maximum differentiating capacity. We must also handle the situation with missing values "0". In order to handle this situation we need to represent the histograms as basic belief assignments within Dempster-Shafer theory.

From each histogram we construct a basic belief assignment where the frequency of missing values "0" is assigned to Θ_k. This is a mass function where all focal elements except one are singleton subsets of the frame $[\{l\}, m_{\chi_j}(\{l\})]$ (i.e., activities of the plan). The exception being the support of Θ_k $[\Theta_k, m_{\chi_j}(\Theta_k)]$ as the only non-singleton focal element.

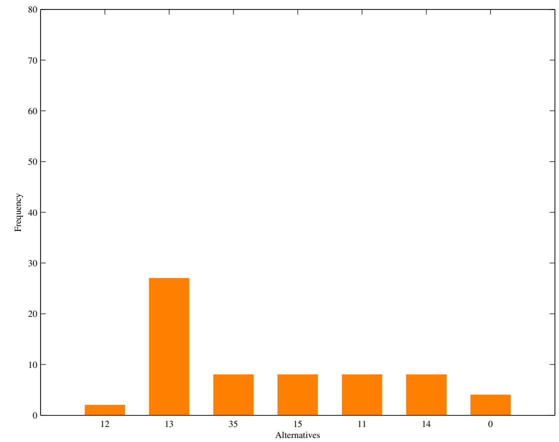

Fig. 2. Histogram $h_{\chi_1}^{A_8}(l)$ over alternatives for activity A_8 of all plan instances in cluster χ_1, where $l \in \{0, 11, 12, 13, 14, 15, 35\}$.

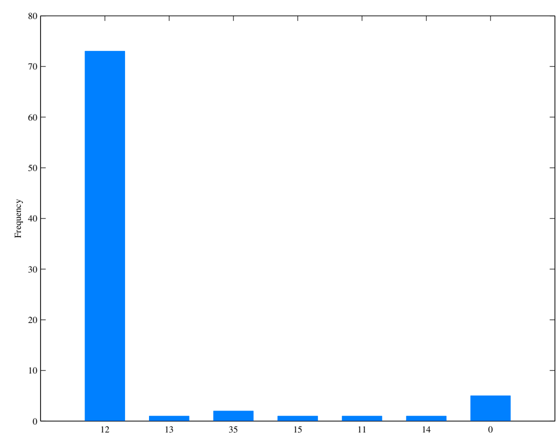

Fig. 3. Histogram $h_{\chi_2}^{A_8}(l)$ over alternatives for activity A_8 of all plan instances in cluster χ_2.

For all subsets of Θ_k we construct m_{χ_j} for χ_j. We get

$$\begin{aligned} m_{\chi_j}^{A_k}(\{l\}) &= \frac{1}{N} \cdot h_{\chi_j}^{A_k}(l), && l \in \{P_i.A_k | \chi_j, A_k\} \\ m_{\chi_j}^{A_k}(\Theta_k) &= 1 - \sum_{k=1}^{|\Theta_k|} m_{\chi_j}^{A_k}(\{k\}), && l = 0 \\ m_{\chi_j}^{A_k}(B) &= 0, && 1 < |B| < |\Theta_k|, \end{aligned} \qquad (16)$$

where N are the number of plan instances. Note, that all subsets B with cardinality $1 < |B| < |\Theta_k|$ receive zero support. This is equivalent to a discounted Bayesian belief function [9], whenever there are some missing values.

In order to evaluate the discriminating capacity of a particular activity A_k for a pair of clusters χ_i and χ_j we investigate the separation of all disjoint subsets. We find the maximum separation for two disjoint subsets where we measure the difference in belief for one subset X between χ_i and χ_j for A_k and for another disjoint subset Y, the difference in belief for this subset between χ_j and χ_i. Here, we have $X \cap Y = \emptyset$ and $X \cup Y \subseteq \Theta_k$, i.e., *not* necessarily $X \cup Y = \Theta_k$.

We calculate the discriminating capacity $DC(A_k)$ of activity A_k as a difference of subsets of the frame Θ

$$DC(A_k) = \max_{\substack{X, Y \subseteq \Theta_k \\ X \cap Y = \emptyset}} \left[\mathrm{Bel}_{\chi_i}(X) - \mathrm{Bel}_{\chi_j}(X) + \mathrm{Bel}_{\chi_j}(Y) - \mathrm{Bel}_{\chi_i}(Y) \right] \tag{17}$$

where $0 \le DC(A_k) \le 2$. The maximum in (17) is found by evaluating $DC(A_k)$ for all $X, Y \subseteq \Theta_k$ where $X \cap Y = \emptyset$. This is of course a problem of exponential computational complexity, but easy to do since Θ_k is usually very small, often $|\Theta_k| \le 5$.

For activity A_8 we get two belief functions for clusters χ_1 and χ_2, respectively, over all focal elements. In Fig. 4, Fig. 5 and Fig. 6 all focal elements are in numerical order.

In Fig. 4 and Fig. 5 we find the belief (13) for activity A_8 for all subsets of Θ of the mass functions constructed in (16) for clusters χ_1 and χ_2, respectively. What we are looking for are two disjoint subsets of Θ with maximum difference of belief between χ_1 and χ_2. In Fig. 6 we observe the difference in belief for all subsets of Θ. We notice that there are several subset with large differences in belief between clusters χ_1 and χ_2. Using the results of Fig. 4 and Fig. 5 and (17) we can calculate the discriminating capacity of activity A_8; $DC(A_8)$.

With this measure we can rank all activities of the plan as to their discriminating capacity for each pair of clusters. Using a threshold we can project all partial plan instances onto a smaller number of screened factors with high discriminating capacity.

In Fig. 7 we return to the example a show $DC(A_k)$ for all 54 activities of the plan in this example. From this result we can select a subset of activities that has the highest discriminating capacity ranked by $DC(A_k)$ as a lower dimension projection.

In addition to the alternatives for all activities of plans, each plan instance also consist of three real values (f, g, h) describing the consequence of the plan instance as evaluated by the simulation-based decision support system, these three values are always included in the projected plan instance.

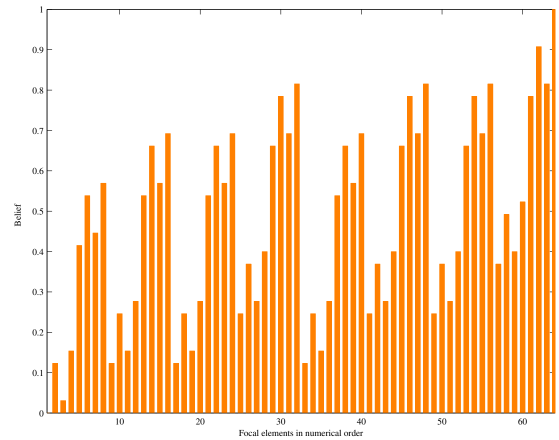

Fig. 4. Belief function over all subsets of alternatives for activity A_8 and cluster χ_1.

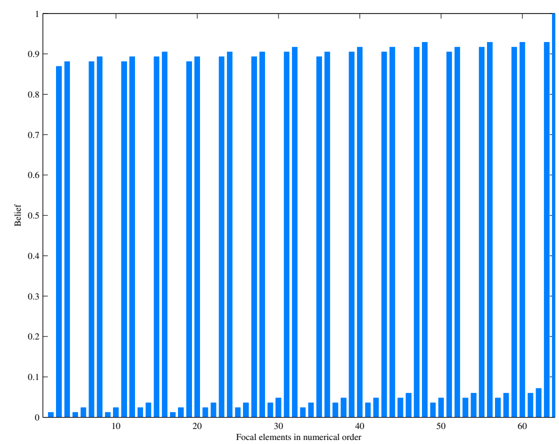

Fig. 5. Belief function over all subsets of alternatives for activity A_8 and cluster χ_2.

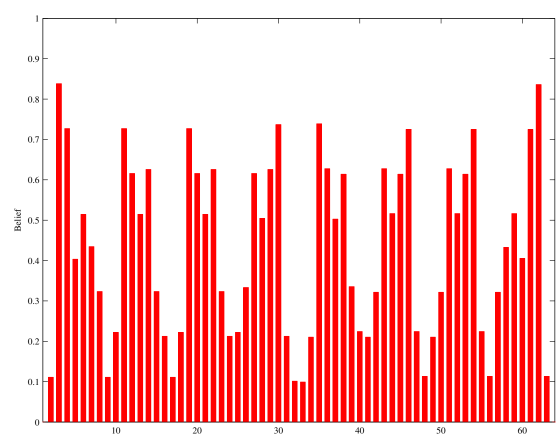

Fig. 6. Absolute difference between Fig. 4 and Fig. 5.

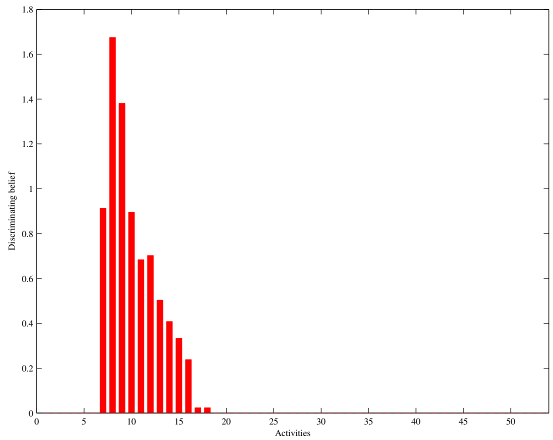

Fig. 7. Discriminating capacity for each activity.

IV. Learning support vectors and hyperplanes as boundaries on military plan

Finding indicators is necessary in order to find a way to check if a plan is good or not without simulation. Support Vector Machine (SVM) is a method that can be used to summarize the information contained in a data set by the Support Vector (SV) produced. Ongoing work is three-folded. First, find the best way to represent training data for use in SVM. Secondly, analyze the problem of finding optimal SVM-parameters and kernel. Finally, find out how to present the SV information for use as indicators. An SVM analysis finds the line (or, in general, hyper plane) that is oriented so that the margin between the support vectors is maximized.

The first moment is to adapt the plans to the SVM machinery. SVM requires that each data instance is represented as a vector of real numbers. A plan with R activities combined in N different ways generate N number of R-dimensional vectors. From section II we have the plans clustered into different classes to be used as training targets y_i. The clusters are represented as classes which in turn are represented as +1 or -1. Training plans are represented by vectors $\boldsymbol{x}_i = \{x_{i1}, \ldots, x_{il}\}$. Initially they are of high dimensionality but the dimensionality can be reduced by the techniques presented in section III. The plan vectors $\boldsymbol{x}_i$ are normalized. Scaling them before applying SVM is very important. This is to avoid that attributes in greater numeric ranges dominate those in smaller numeric ranges.

The concept of treating the objects to be classified as points in a high-dimensional space and finding a line that separates them is not unique to the SVM. The SVM, however, is different from other hyper plane-based classifiers in how the hyper plane is chosen. If we define the distance from the separating hyper plane to the nearest data point as the margin of the hyper plane, then the SVM selects the maximum margin separating hyper plane. Selecting this hyper plane maximizes the SVM's capability to calculate the correct classification of up to that time unseen plan instances.

C. Principles of SVM

The basic idea of SVM is to find a function $f(x)$ that has at most deviation from the actually obtained targets y_i for all the training data $\{(x_1, y_1), \ldots, (x_l, y_l)\} \subset \boldsymbol{X} \times \boldsymbol{R}$ where $\boldsymbol{X}$ denotes the space of the input plans.

In the case of linear functions f, a separating hyper plane, written in terms of a weight vector $\boldsymbol{w}$ and a threshold b takes the form $f(x) = (\boldsymbol{x},\boldsymbol{w}) + b$ with $\boldsymbol{w} \in \boldsymbol{X}, b \in \boldsymbol{R}$ where $(\ ,\ )$ denotes the dot product in $\boldsymbol{X}$. We want to minimize the norm $\|\boldsymbol{w}\|^2 = (\boldsymbol{w},\boldsymbol{w})$ as shown in Fig. 8. This can be formulated as a convex optimization problem.

Minimize

$$\frac{1}{2}\|\boldsymbol{w}\|^2, \tag{18}$$

subject to

$$y_i - (\boldsymbol{x}_i,\boldsymbol{w}) - b \geq 1 \qquad i = 1, \ldots, l. \tag{19}$$

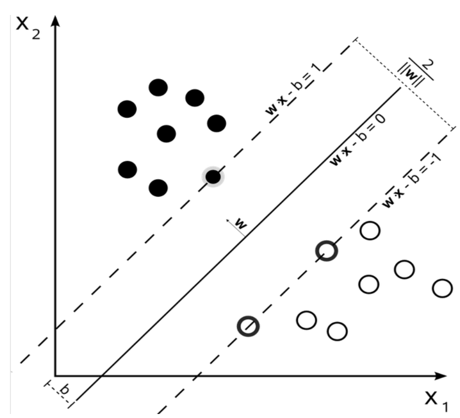

Fig. 8. Optimal linear divider of two separate classes.

For all points from the hyper plane $HP[(\boldsymbol{x}_i,\boldsymbol{w}) + b = 0]$, the distance between origin and the hyper plane HP is $\frac{b}{\|\boldsymbol{w}\|}$. We consider the plans from the class -1 that satisfy the equality $(\boldsymbol{x}_i,\boldsymbol{w}) + b = -1$, and determine the hyper plane HP_1; the distance between origin and the hyper plane HP_1 is equal to $\frac{|-1-b|}{\|\boldsymbol{w}\|}$. Similarly, the plans from the class +1 satisfy the equality $(\boldsymbol{x}_i,\boldsymbol{w}) + b = 1$, and determine the hyper plane HP_2; the distance between origin and the hyper plane HP_2 is equal to $\frac{|1-b|}{\|\boldsymbol{w}\|}$. Hyper planes HP, HP_1, and HP_2 are parallel and no training plans are located between hyper planes HP_1 and HP_2. Based on the above considerations, the distance between hyper planes HP_1 and HP_2 is $\frac{2}{\|\boldsymbol{w}\|}$.

The standard way to train an SVM is to introduce Lagrange multipliers α_i and optimize them by solving a dual problem. We construct a Lagrange function L from the primal function,

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \alpha_i \{ y_i [(\boldsymbol{x}_i, \boldsymbol{w}) + b] - 1 \} \quad (20)$$

where α_i are the Lagrangian multipliers.

It can be shown that this function has a saddle point with respect to the primal and dual variables at the solution, and it follows from the saddle point condition that the partial derivatives of L with respect to the primal variables have to vanish for optimality. Then we can write

$$\boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i , \quad (21)$$

i.e., $\boldsymbol{w}$ can be completely described as a linear combination of the training plans $\boldsymbol{x}_i$. The plans $\boldsymbol{x}_i$ for which $\alpha_i > 0$ are called Support Vectors, they lie exactly at the margin. SVs do lie on the boundary of the convex hulls of the two classes, thus they possess supporting hyper planes. The Support Vector optimal hyper plane is the hyper plane which lies in the middle of the two parallel supporting hyper planes (of the two classes) with maximum distance. We have the decision function,

$$sign(\boldsymbol{wx} + b) . \quad (22)$$

The complexity of a function's representation by Support Vectors is independent of the dimensionality of the input space X, and depends only on the number of Support Vectors.

Normally, data is not separable because the target function is essentially nonlinear. But it can be separable in higher dimensional space. In the case where a linear boundary is inappropriate we map our input vectors nonlinearly into a high dimensional feature space and perform the separation there. One can first do a nonlinearly transform on the set of input vectors $\boldsymbol{x}_i, \ldots, \boldsymbol{x}_l$ into a high-dimensional feature space by a map $\phi : \boldsymbol{x}_i \rightarrow \boldsymbol{z}_l$ and then do a linear separation. This feature expansion has the same optimization as before. This classifier is nonlinear in original features, but linear in expanded feature space. We have replaced $\boldsymbol{x}$ by $\phi(\boldsymbol{x})$ for some nonlinear ϕ so the decision boundary is some nonlinear surface

$$\boldsymbol{w}\phi(\boldsymbol{x}) + b = 0 \quad (23)$$

and the decision function

$$sign[\boldsymbol{w}\phi(\boldsymbol{x}) + b] . \quad (24)$$

Since $\boldsymbol{w}$ is a linear combination of (signed) training examples, $\boldsymbol{w}$ has a finite representation even if there are infinitely many features.

Using the kernel trick [11] we can represent the decision function in higher dimensions without using $\boldsymbol{w}$.

When used in feature space $\boldsymbol{x}$ is created by applying a nonlinear feature expansion function ϕ to some original vector q_i. We have

$$K(q_i, q_j) = \phi(q_i)\phi(q_j) . \quad (25)$$

K is called a kernel function and satisfies a condition analogous to nonnegative definiteness for a matrix.

In many cases there is a simple expression for K even if there is none for ϕ.

One choice of kernel used in this work is a Gaussian, which has a single parameter $\gamma = \frac{1}{2\sigma^2}$. For this kernel we have

$$K(q_i, q_j) = \exp\left(-\frac{\|q_i, q_j\|^2}{\sigma^2}\right) . \quad (26)$$

This means that even if we do calculations in feature space we use the original input variables and the decision function becomes

$$sign\sum_i K(q, q_i) y_i \alpha_i + b . \quad (27)$$

D. Choosing parameters

For the use in an SVM algorithm the training data must be normalized. Target values included in the vector y are set to -1 or 1 depending on which class they represent. The plan instance vectors $\boldsymbol{x}_i$ are normalized. We transform the data to corresponding values between 0 and 1. In this example we use some lower dimensional training set created using filter 3 from section III in which the four most significant activities and the resulting values of g to form a five dimensional vector. The g value is normalized only with respect to other g values and not to the rest of the vector. A typical plan instance P_1 is [1 2 41 42 3 4 43 12 10 54 51 16 47 55 22 26 31 33 58 60 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1902.0 1645.2 256.8], and a corresponding training vector then would be [43 12 10 54 1645.2] and when normalized [0.7963 0.2222 0.1852 1.0000 0.9900].

The accuracy of an SVM model is largely dependent on the selection of the model parameters. Some flexibility in separating the categories is needed. SVM implementation have a cost parameter, *C,* that controls the trade off between allowing training errors and forcing rigid margins. This parameter gives the model a soft margin that permits some misclassifications [8]. Increasing C increases the cost of misclassifying plans and forces a more accurate model to be crated. A search can be used to find the optimal value of C and γ.

Best combination of C and γ is selected by an exhaustive search with growing sequences of C and γ. This is the most simple brute force method to find optimal parameters and is used in this paper. Values tested in this initial work are:

```
C = [0.1 1 10 100 1000 inf];
γ = [0.3 0.8 1 10];
```

We also tried different values for the conditioning parameter for solving the quadratic programming problem [12] included in the algorithm;

```
ε = [0.001 0.01 0.05 0.1 1 10];
```

The final model is the one with parameters C, γ, ε, such that $\|w\|^2$ is minimized. Based on the first coarse search we did a finer search and the resulting parameters for our example is

$C = 0.1$, $\gamma = 0.3$, $\varepsilon = 1$;

More advanced methods typically check each combination of parameter choices using cross validation, and the parameters with best cross-validation accuracy are chosen. The final model, which is used for testing and for classifying $\Phi(\boldsymbol{x})$ new data, is then trained on the whole training set using the selected parameters. Cross validation will be studied in future work.

Fig. 9 show the examples from using a lower dimensional training set from section III. The figure show a projection down to 2 dimensions, x showing the positive vectors and + showing the negative, and o represents the support vectors. The visualization is very hard to do in two dimensions for a 5-dimensional problem.

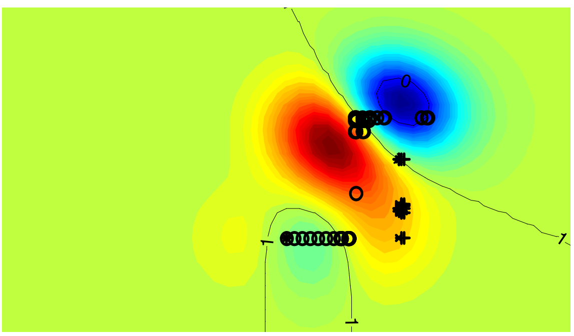

Fig. 9. Learning hyper plan boundaries. Examples from using some lower dimensional training set filter.

E. Classification with more than two classes

Using a hyper plane to separate the feature vectors into two classes' works when there are only two target categories, but how do we handle the case where we have more than two classes? The two most used methods are: (1) "one against many" where each category is split out and all of the other categories are merged; and, (2) "one against one" where $k(k-1)/2$ models are constructed where k is the number of categories. The case of many classes is left to future work.

V. Using hyperplanes as indicators

When representing the classification border by the SVM optimal hyper plane, each dimension has a bound for the corresponding action in the plan. Using the SVM decision function in (27), each activity can be evaluated by its presence in the tested plans presented to the decision function. Based on equation (28) a plan P will be classified as A or B;

$$P = \begin{cases} A, & \sum \exp\left(-\frac{\|q_i, q_j\|^2}{\sigma^2}\right) y_i \alpha_i + b > 0 \\ B, & \sum \exp\left(-\frac{\|q_i, q_j\|^2}{\sigma^2}\right) y_i \alpha_i + b < 0 \end{cases} \quad (28)$$

This way we can correct our bad plans to be good plans by simply change the bad activities.

Thus, the hyper plane will serve as a warning bell when the execution of an operational plan approach the boundary beyond which its performance can deteriorate drastically, and where radical dynamic replanning may become necessary.

VI. Conclusions

In this paper we conclude that it is possible to learn indicators from simulated data of partial plan instances that describe a military operational plan, by using a series of computational processing steps, such as

- calculating distances between all pairs of partial plan instances,
- clustering plan instances with Potts spin neural clustering,
- projecting plan instances to the most differentiating factors using evidential screening of factors,
- learning support vectors from clusters of projected classified plan instances,
- deriving hyper plans from support vectors as indicators.

Before we have a useful tool, a thorough parameter study is needed for the SVM analysis. This is important for reliability.

VII. References

[1] J. Schubert, F. Moradi, H. Asadi, P. Hörling, and E. Sjöberg, "Simulation-based decision support for effects-based planning," in Proceedings of the 2010 IEEE International Conference on Systems, Man and Cybernetics, October 2010. Piscataway, NJ: IEEE, 2010, pp. 636–645.

[2] F. Moradi and J. Schubert, "Modelling a simulation-based decision support system for effects-based planning," in Proceedings of the NATO Symposium on Use of M&S in: Support to Operations, Irregular Warfare, Defence Against Terrorism and Coalition Tactical Force Integration (MSG-069), paper 11, pp. 1–14, October 2009.

[3] F. Y. Wu, "The Potts model," Reviews of Modern Physics, vol. 54, pp. 235–268, January 1982.

[4] C. Peterson and B. Söderberg, "A new method for mapping optimization problems onto neural networks," International Journal of Neural Systems, vol. 1, pp. 3–22, May 1989.

[5] M. Bengtsson and J. Schubert, "Dempster-Shafer clustering using Potts spin mean field theory," Soft Computing, vol. 5, pp. 215–228, June 2001.

[6] J. Schubert, "Clustering belief functions based on attracting and conflicting metalevel evidence using Potts spin mean field theory," Information Fusion, vol. 5, pp. 309–318, December 2004.

[7] V. Vapnik, The Nature of Statistical Learning Theory. New York: Springer, 1995.

[8] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, pp. 273–297, September 1995.

[9] G. Shafer, A Mathematical Theory of Evidence. Princeton, NJ: Princeton University Press, 1976.

[10] R. W. Hamming, "Error detecting and error correcting codes," The Bell Systems Technical Journal, vol. 29, pp. 147–160, April 1950.

[11] M. Aizerman, É. M. Braverman, and L. I. Rozonoér, "Theoretical foundations of the potential function method in pattern recognition learning," Automation and Remote Control, vol. 25, pp. 821–837, 1964.

[12] P. Wolfe, "The simplex method for quadratic programming," Econometrica, vol. 27, pp. 382–398, July 1959.