Support Vector Machines (SVM)

• Reading Assignments

C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, Kluwer Academic Publishers, 1998 (on-line).

R. Duda, P. Hart, and D. Stork, *Pattern Classification*, John-Wiley, 2nd edition, 2001 (section 5.11, hard-copy).

S. Gong et al. *Dynamic Vision: From Images to Face Recognition,* Imperial College Pres, 2001 (sections 3.6.2, 3.7.2, hard copy).

• Case Studies

M. Pontil and A. Verri, "Support vector machines for 3D object recognition", *IEEE Transaction on Pattern Analysis and Machine Intelligence,* vol. 20, no. 6, pp. 637-646, 1998 (has nice review of SVM theory, pp. 637-640, 1998 (on-line).

A. Mojan, C. Papageorgiou and T. Poggio, "Example-based object detection in images by components", *IEEE Transaction on Pattern Analysis and Machine Intelligence,* vol. 23, no. 4, pp. 349-361, 2001 (on-line).

B. Moghaddam and M. Yang, "Gender Classification with SVM", *IEEE Conference on Face and Gesture Recognition*, pp. 306-311, 2000 (on-line).

Support Vector Machines (SVM)

• Classification approaches (review)

- Given a set of training patterns from each class, the objective is to establish decision boundaries in the feature space which separate patterns belonging to different classes.

- In the statistical approach, the decision boundaries are determined by the probability distributions of the patterns belonging to each class, which must either be specified or learned.

- In the discriminant-based approach, the decision boundary is constructed explicitly (i.e., knowledge of the form of the probability distribution is not required):

    (1) First a parametric form of the decision boundary (e.g., linear or quadratic) is specified.

    (2) The "best" decision boundary of the specified form is found based on the classification of the training patterns.

• Linear discriminant functions

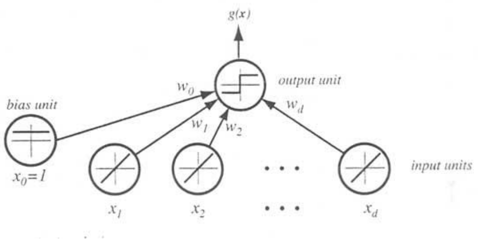
- The problem of finding a discriminant function can be formulated as a problem of minimizing a criterion function (i.e., the sample risk or the training error).

- A linear discriminant function can be written as:

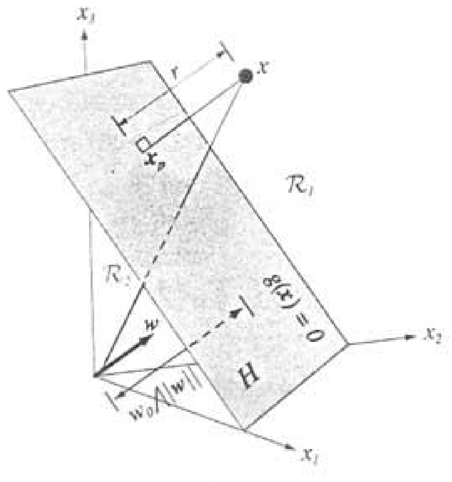
$$g(x) = w^t x + w_0$$

- Assuming two classes, classification is based on the following rule:

Decide ω_1 *if* $g(x)>0$ *and* ω_2 *if* $g(x)<0$



- The decision boundary (i.e., a hyperplane) is defined by the equation $g(x) = 0$.



• **Distance from a point x to the hyperplane**

- Let us express x as follows:

$$x = x_p + r\frac{w}{||w||}$$

- Let's substitute the above expression in $g(x)$

$$g(x) = w^t x + w_0 = w^t(x_p + r\frac{w}{||w||}) + w_0 = w^t x_p + r\frac{w^t w}{||w||} + w_0 = r||w||$$

since $w^t x_p + w_0 = 0$ and $w^t w = ||w||^2$.

- The above expression gives the distance of x from the hyperplane:

$$r = g(x)/||w||$$

- The distance of the origin from the hyperplanes is

$$w_0/||w||$$

• Various types of discriminant functions

Linear discriminant:

$$g(x) = w_0 + \sum_{i=1}^{d} x_i w_i$$

Quadratic discriminant: obtained by adding terms corresponding to products of pairs of components of x

$$g(x) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j w_{ij}$$

Polynomial discriminant: obtained by adding terms such as $x_i x_j x_k w_{ijk}$.

Generalized discriminant:

$$g(x) = \sum_{i=0}^{\hat{d}} a_i y_i(x) \quad \text{or} \quad g(x) = a^t y$$

where a is a $\hat{d}$-dimensional weight vector and $y_i(x)$ can be arbitrary functions of x (called $\phi()$ functions, i.e., $y_i = \phi_i(x)$).

(note that w_0 has been absorbed in a, that is, a_0=w_0 and y_0=1)

• Generalized discriminant functions

- Selecting the $y_i(x)$ appropriately and letting $\hat{d}$ be sufficiently large, any discriminant function can be approximated.

- The resulting discriminant function is not linear in x but it is linear in y.

- The $\hat{d}$ functions $y_i(x)$ simply map points in d-dimensional x-space to points in $\hat{d}$-dimensional y-space.

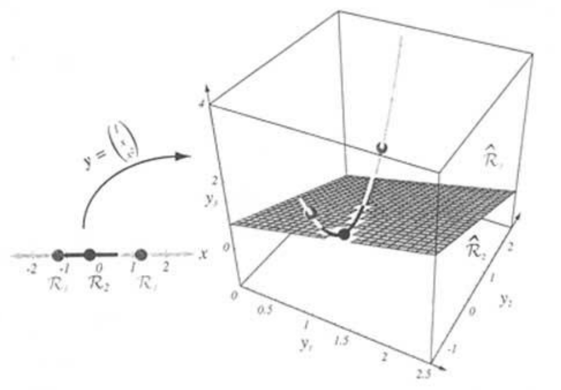
Example: Consider the following quadratic discriminant function:

$$g(x) = a_1 + a_2 x + a_3 x^2 \quad \text{with} \quad y = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}$$

* Maps a line in x-space to a parabola in y-space.

* The plane $g(x) = 0$ or $a^t y = 0$ defined by $a = (-1, 1, 2)$ divides the y-space into two regions.

* Note that the corresponding region R_1 in the x-space is not simply connected.

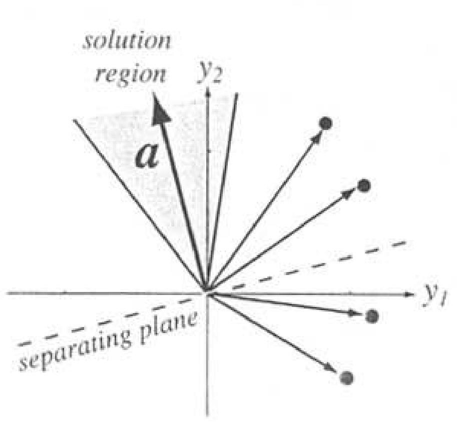


- The main disadvantages of the generalized discriminant are:

(1) It is computationally intensive to compute.
(2) Lots of training examples are required to determine a if $\hat{d}$ is very large (*curse of dimensionality*).

• Solution region

- In general, the solution vector a is not unique (any vector in the solution region satisfies, e.g., $g(x) = a^t y > 0$ for $x \in \omega_1$ and $g(x) = a^t y < 0$ for $x \in \omega_2$)

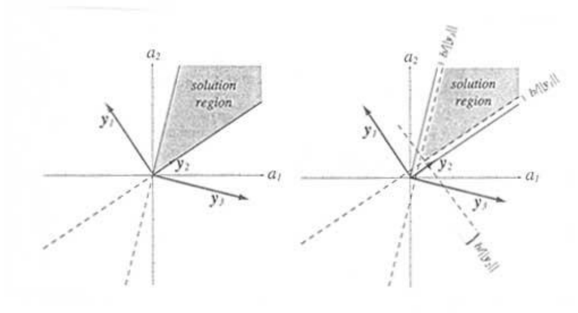


- Additional constraints are necessary to define a uniquely.

find a (i) find the unit-length weight vector that maximizes the minimum distance from the training examples to the separating plane).

(ii) find a minimum length weight vector satisfying $g(x) = a^t y \geq b$ where b is a positive constant.

(the new solution region lies inside the previous solution region, being insulated by the old boundaries by the distance $b/||y_i||$)



• Learning and risk minimization

- The aim of any learning machine is to estimate $g(x)$ from a finite set of observations by minimizing the empirical risk (i.e., some kind of an error function).

Example: The least-squares method minimizes the empirical risk shown below:

$$R_{emp}(w, w_0) = \frac{1}{n} \sum_{k=1}^{n} [z_k - g(x_k, w, w_0)]^2$$

where z_k is the desired classification for pattern k (e.g., $z_k = \pm 1$ according to whether pattern k is in ω_1 or ω_2)

- The conventional empirical risk minimization over training data does not imply good generalization to novel test data.

(1) There could be a number of different functions which all give a good approximation to the training data set.

(2) It is difficult to determine a function which best captures the the true underlying structure of the data distribution.

• Structural risk minimization

- To guarantee an "upper bound on generalization error", statistical learning theory says that the *capacity* of the learned functions must be controlled (i.e., functions with large capacity are able to represent many dichotomies for a given data set).

- Structural risk minimization aims to address this problem and provides a well defined quantitative measure of the *capacity* of a learned function to generalize over unknown test data.

- The Vapnik-Chervonenkis (VC) dimension has been adopted as one of the most popular measures for such a capacity.

- According to the structural risk minimization principle, a function that describes the training data well (i.e., minimizes the empirical risk) and belongs to a set of functions with lowest VC dimension will generalize well **regardless of the dimensionality of the input space**.

$$err_{true} \leq err_{training} \sqrt{\frac{VC(log(2m/VC)+1) - log(\delta/4)}{n}}$$

with probability $(1-\delta)$ (Vapnik, 1995)

(Structural Minimization Principle)

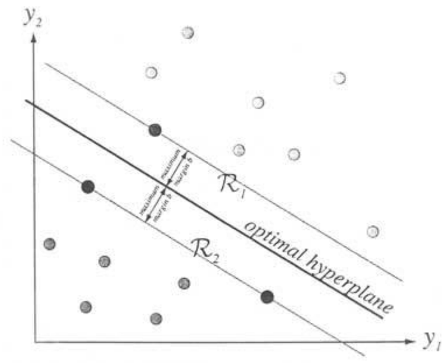
• Optimal hyperplane and support vectors

- It has been shown (Vapnik, 1995) that maximizing the margin distance between the classes is equivalent to minimizing the VC dimension.

- This optimal hyperplane is the one giving the largest margin of separation between the classes (i.e., bisects the shortest line between the convex hulls of the two classes).

- A relatively small subset of the patterns (*support vectors*) lie exactly on the margin (the closest patterns to the hyperplane and the most difficult to classify).

- The optimal hyperplane is completely determined by these support vectors.



• Overview of SVM

- SVM are primarily two-class classifiers with the distinct characteristic that they aim to find the optimal hyperplane such that the expected generalization error (i.e., error for the unseen test patterns) is minimized.

- Instead of directly minimizing the empirical risk calculated from the training data, SVMs perform *structural risk minimization* to achieve good generalization (i.e., minimize an upper bound on expected generalization error).

- The optimization criterion is the width of the margin between the classes (i.e., the empty area around the decision boundary defined by the distance to the nearest training patterns).

• Positives/Negatives

- (Pos) Appears to avoid overfitting in high dimensional spaces and generalize well using a small training set (the complexity of SVM is characterized by the number of support vectors rather than the dimensionality of the transformed space -- no formal theory to justify this).

- (Pos) Global optimization method, no local optima (SVM are based on exact optimization, not approximate methods).

- (Neg) Applying trained classifiers can be expensive.

• SVM training

- The goal is to find the separating plane with the largest margin (i.e., find the support vectors).

- Training a SVM is equivalent to solving a quadratic programming problem with linear constraints (the number of variables is equal to the number of training data).

• Linear SVM: The separable case

- As we have seen, a linear discriminant satisfies the following equation:

$$g(x_k) = w^t x_k + w_0 = \begin{cases} > 0 & \text{if } x_k \in \omega_1 \\ < 0 & \text{if } x_k \in \omega_2 \end{cases}, \quad k = 1, 2, .., n$$

- For each pattern x_k, $k = 1, 2, .., n$ let's define $z_k = \pm 1$, according to whether pattern k is in ω_1 or ω_2, then we can combine the above inequalities into one set of inequalities:

$$z_k g(x_k) > 0 \ or \ z_k(w^t x_k + w_0) > 0, \ k = 1, 2, .., n$$

- Since the data is separable, there exist a hyperplane that separates the positive from the negative examples; the distance from a point x_k to the hyperplane (i.e., $g(x_k)/||w||$) should satisfy the constrain:

$$\frac{z_k g(x_k)}{||w||} \geq b, \quad b > 0 \text{ (margin).}$$

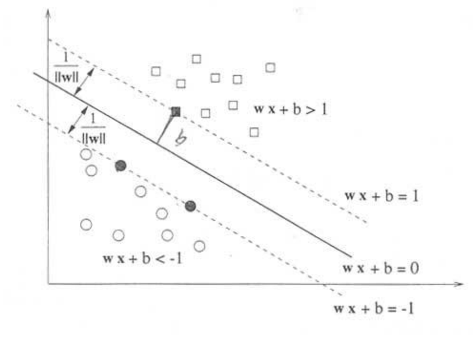
- To ensure uniqueness, we impose the constraint $b\,||w|| = 1$ (i.e., the solution vector w can be scaled arbitrarily and still preserve the above constrain). - Using the above constraint, $g(x)$ should satisfy the following inequality:

$$z_k g(x_k) \geq 1, \quad with \ b = \frac{1}{||w||} \ (1) \quad \text{(margin)}$$

- The goal of the SVM is to maximize $1/||w||$ subject to the constraint imposed by Eq. (1), or, equivalently:

Problem 1: Minimize $\frac{1}{2} ||w||^2$

subject to $z_k(w^t x_k + w_0) \geq 1, \quad k = 1, 2, .., n$



• **Solving "Problem 1"**

- First, we form the Lagrange function:

$$L(w, w_0, \lambda) = \frac{1}{2} \|w\|^2 - \sum_{k=1}^{n} \lambda_k [z_k (w^t x_k + w_0) - 1], \quad \lambda_k \geq 0$$

- We want to minimize $L()$ with respect to (w, w_0) and maximize it with respect to λ_k (i.e., determine the saddle point of $L()$).

- We can reformulate "Problem 1" as maximizing the following problem (*dual problem*):

$$\textbf{Problem 2}\text{: Maximize } \sum_{k=1}^{n} \lambda_k - \frac{1}{2} \sum_{k,j}^{n} \lambda_k \lambda_j z_k z_j x_j^t x_k$$

$$\text{subject to } \sum_{k=1}^{n} z_k \lambda_k = 0, \quad \lambda_k \geq 0, \, k = 1, 2, \ldots, n$$

- During optimization, the values of all λ_k become 0, except for the support vectors.

- The solution for w is given as a linear combination of the support vectors:

$$w = \sum_{k=1}^{n} z_k \lambda_k x_k \quad (\lambda_k \neq 0 \text{ only if } x_k \text{ is a support vector})$$

- The solution for w_0 can be determined using any support vector x_k:

$$w^t x_k + w_0 = z_k \;\text{ or }\; w_0 = z_k - w^t x_k$$

- The decision function for the optimal hyperplane is given by

$$g(x) = \sum_{k=1}^{n} z_k \lambda_k (x^t x_k) + w_0 \;\text{ or }\; g(x) = \sum_{k=1}^{n} z_k \lambda_k (x . x_k) + w_0$$

- The decision rule is

decide ω_1 if g(x)>0 and ω_2 if g(x)<0

• Linear SVM: The non-separable case

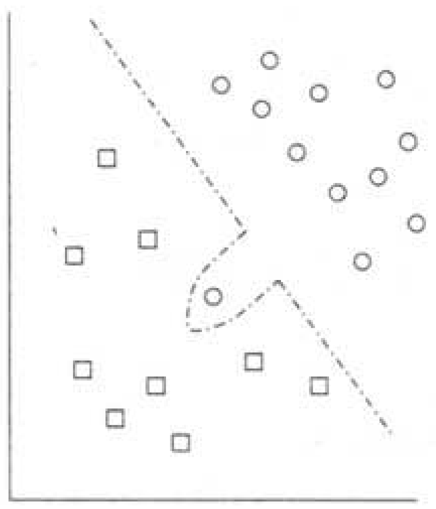
- When the data is not linearly separable, we can either use the non-linear SVM (see next section) or modify the problem to allow misclassified data by introducing error variables ψ_k:

$$\textbf{Problem 3}\text{: Minimize } \frac{1}{2}\,||w||^2 + c\sum_{k=1}^{n}\psi_k$$

$$\text{subject to } z_k(w^t x_k + w_0) \geq 1 - \psi_k, \quad k = 1, 2, .., n$$

- The result is a hyperplane that minimizes the sum of errors ψ_k while maximizing the margin for the correctly classified data.

- The constant c controls the tradeoff between margin and misclassification errors (aims to prevent outliers from affecting the optimal hyperplane).



- We can reformulate "Problem 3" as maximizing the following problem (*dual problem*):

$$\textbf{Problem 4}\text{: Maximize } \sum_{k=1}^{n}\lambda_k - \frac{1}{2}\sum_{k,j}^{n}\lambda_k\lambda_j z_k z_j x_j^t x_k$$

$$\text{subject to } \sum_{k=1}^{n} z_k\lambda_k = 0 \text{ and } 0 \leq \lambda_k \leq c,\ k = 1, 2, .., n$$

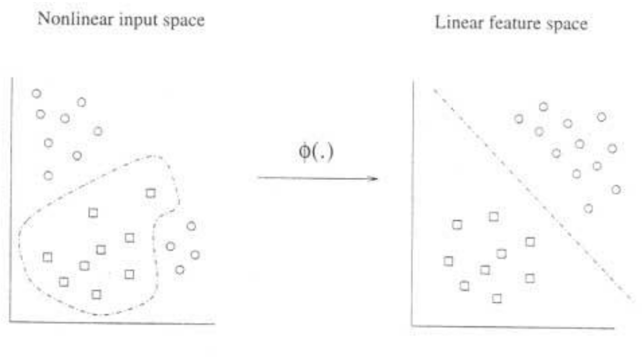
where the use of error variables ψ_k constraint the range of the Lagrange coefficients from 0 to c.

• Nonlinear SVM

- Extending the above concepts to the non-linear vase relies on preprocessing the data to represent them in a much higher dimensionality space.

$$x_k \rightarrow \Phi(x_k)$$

- Using an appropriate nonlinear mapping Φ() to a sufficiently high dimensional space, data from two classes can always be separated by a hyperplane.



- The decision function for the optimal hyperplane is given by

$$g(x) = \sum_{k=1}^{n} z_k \lambda_k (\Phi(x).\,\Phi(x_k)) + w_0$$

- The decision rule is the same as before:

$$decide\ \omega_1\ if\ g(x)>0\ and\ \omega_2\ if\ g(x)<0$$

- The disadvantage of this approach is that the mapping $x_k \rightarrow \Phi(x_k)$ might be very computationally intensive to compute.

• The kernel trick

- If there were a "kernel function" $K(x, x_k) = \Phi(x).\Phi(x_k$ we would only need to use $K()$ and would never need to explicitly even know what $\Phi()$ is.

- The decision function for the optimal hyperplane is then given by

$$g(x) = \sum_{k=1}^{n} z_k \lambda_k K(x, x_k) + w_0$$

Example: consider $x \in R^2$, $\Phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix} \in R^3$, and $K(x, y) = (x.y)^2$

$$(x.y)^2 = (x_1y_1 + x_2y_2)^2$$

$$\Phi(x).\Phi(y) = x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2 = (x_1y_1 + x_2y_2)^2$$

- Note that neither the mapping $\Phi()$ nor the high dimensional space are unique.

$$\Phi(x) = \frac{1}{\sqrt{2}} \begin{pmatrix} (x_1^2 - x_2^2) \\ 2x_1x_2 \\ (x_1^2 + x_2^2) \end{pmatrix} \in R^3 \quad \text{or} \quad \Phi(x) = \begin{pmatrix} x_1^2 \\ x_1x_2 \\ x_1x_2 \\ x_2^2 \end{pmatrix} \in R^4$$

• Suitable kernel functions

- Kernel functions which can be expressed as a dot product in some space satisfy the *Mercer's* condition (see Burges' paper).

- The *Mercer's* condition does not tell us how to construct $\Phi()$ or even what the high dimensional space is.

- By using different kernel functions, SVM implement a variety of learning machines, some of which coincide with classical architectures (see below).

polynomial: $K(x, x_k) = (x.x_k)^d$

sigmoidal: $K(x, x_k) = tanh(v_k(x.x_k) + c_k)$
(corresponds to a two-layer sigmoidal neural network)

$$\textit{Gaussian: } K(x, x_k) = exp(\frac{-||x - x_k||^2}{2\sigma_k^2})$$

(corresponds to a radial basis function (RBF) neural network)

- The kernel trick implies that the computation remains feasible even if the feature space has very high dimensionality.

  * It can be shown for the case of polynomial kernels that the data is mapped to a space of dimension $h = \binom{p+d-1}{d}$ where p is the original dimensionality.

  * Suppose p=256 and d = 4, then h=183,181,376 !!

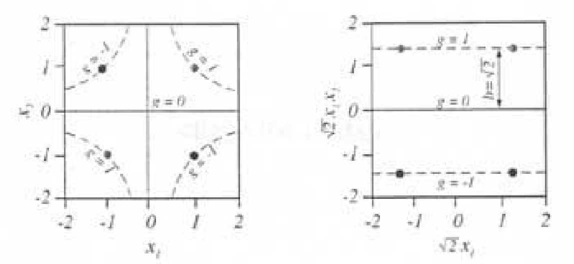
  * A dot product in the high dimensional space would require $O(h)$ computations while the kernel requires only $O(p)$ computations.

• An example

- Consider the XOR problem which is non-linearly separable:

(1,1) and (-1, -1) belong to ω_1

(1,-1) and (-1, 1) belong to ω_2



- Consider the following mapping (many other mappings could be used too):

$$y = \Phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_2 \\ x_2^2 \\ 1 \end{pmatrix}$$

- The above transformation maps x_k to a 6-dimensional space:

$$y_1 = \Phi(x_1) = \begin{pmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix} \quad y_3 = \Phi(x_3) = \begin{pmatrix} 1 \\ -\sqrt{2} \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix}$$

$$y_2 = \Phi(x_2) = \begin{pmatrix} 1 \\ \sqrt{2} \\ -\sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix} \quad y_4 = \Phi(x_4) = \begin{pmatrix} 1 \\ -\sqrt{2} \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix}$$

- We seek to maximize:

$$\sum_{k=1}^{4} \lambda_k - \frac{1}{2} \sum_{k,j}^{4} \lambda_k \lambda_j z_k z_j \Phi(x_j^t) \Phi(x_k)$$

$$\text{subject to } \sum_{k=1}^{4} z_k \lambda_k = 0, \;\; \lambda_k \geq 0, \; k = 1, 2, \ldots, 4$$

- The solution turns out to be:

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{8}$$

- Since all $\lambda_k \neq 0$, all x_k are support vectors !

- We can now compute w:

$$w = \sum_{k=1}^{4} z_k \lambda_k \Phi(x_k) = \frac{1}{8}\begin{pmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix} - \frac{1}{8}\begin{pmatrix} 1 \\ \sqrt{2} \\ -\sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix} + \frac{1}{8}\begin{pmatrix} 1 \\ -\sqrt{2} \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix} - \frac{1}{8}\begin{pmatrix} 1 \\ -\sqrt{2} \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 0 \\ 0 \\ \sqrt{2} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

- The solution for w_0 can be determined using any support vector, e.g., x_1:

$$w^t \Phi(x_1) + w_0 = z_1 \;\text{ or }\; w_0 = z_1 - w^t x_1 = 0$$

- The margin b is computed as follows:

$$b = \frac{1}{||w||} = \sqrt{2}$$

- The decision function is the following:

$$g(x) = w^t \Phi(x) + w_0 = x_1 x_2$$

where we decide ω_1 if $g(x) > 0$ and ω_2 if $g(x) < 0$

• Limitations of SVM

- The biggest limitation of SVM lies in the choice of the kernel (the best choice of kernel for a given problem is still a research problem).

- A second limitation is speed and size (mostly in training - for large training sets, it typically selects a small number of support vectors, therby minimizing the computational requirements during testing).

- The optimal design for multiclass SVM classifiers is also a research area.