VOL5 NO. 1 March 2016 - ISSN 2233 – 1859

UOIuBIH
ORSinBIH
Operations Research Society in Bosnia and Herzegovina

Southeast Europe Journal of Soft Computing

Available online: http://scjournal.ius.edu.ba

IUS Soft Computing Research Group

Support Vector Machines for Predicting Protein Structural Classes via Pseudo Images Derived From Amino Acid Sequences

Betul Akcesme
Mehmet Can
International University of Sarajevo
Faculty of Natural Sciences and Engineering
Hrasnicka Cesta 15, Ilidza 71210 Sarajevo, BIH
bcicek@ius.edu.ba; mcan@ius.edu.ba;

Article Info
Article history:
Article received on February 2015
Received in revised form March 2015

Keywords:
Support vector machines, protein structural classes, class prediction

Abstract
SVM is one of the most widely used and powerful classification algorithms to predict protein structural classes. Via radial base functions, SVM maps the linearly non separable input data to a higher dimensional space where it is almost separable by a hyperplane. First using the training data, six hyperplanes that separate pair wise the four classes training data are constructed. Then the expertise of these six SVMs genuinely aggregated to classify the testing data into four classes. The validation is performed by a boot strap technique. The 33 dimensional data that represents proteins of the data set is derived from pseudo images of proteins that stems from their amino acid sequences. In spite of the simplicity of the features, a Q3 accuracy around 75% is achieved.

1. INTRODUCTION

Support Vector Machine (SVM) is one type of learning machine based on statistical learning theory which has been relatively recently introduced to the field (Vapnik 1995). SVMs have been applied to solve a variety of problems in the field of bioinformatics such as gene function analysis, microarray expression data (Brown et al 2000), protein secondary structure prediction (Hua and Sun 2001), protein fold recognition (Ding and Dubchak 2001), and cancer tissue classification from microarray expression data (Mukherjee et al 1999). Since the first application of SVM for prediction of protein structural classes (Cai et al 2001) it has gained popularity in a wide range of studies. The way of SVM basically performs its function can be explained as following. First step is to map the input vectors (protein sequence data) into a feature space with higher dimension, linearly or non-linearly. The character of the transformation depends on the selection of the kernel function. In the second step, it seeks an optimized linear division within the feature space from the first step to construct a hyperplane which divides the data points into their corresponding classes. Division can be either into two classes or can be extended to multiclass. SVM always looks for a global optimized solution to refrain from over- fitting. Therefore, SVM enables to handle with a large number of features which may causes over-fitting data problem (Vapnik 1998; Cai et al 2003).

2. SUPPORT VECTOR MACHINES

Vapnik is the pioneer of the learning machines tool support vector machines (SVM). The first description of the machine was presented by Boser, Guyon, and Vapnik in 1992 (Boser, et al. 1992).The detailed description of SVMs appeared in Vapnik's 1998 book entitled "Statistical Learning Theory." Cucker and Smale (Cucker, and Smale, 2001), introduced a mathematically rigorous treatment of supervised learning theory, with emphasis on the relationship of approximation to learning and the primary role of inductive inference, Schölkopf, 1997 contributed to the field with the book "Support Vector Learning" (Schölkopf 1997) .Comprehensive treatments of kernel machines, including support vector machines, are presented in the books by Schölkopf and Smola (Schölkopf and Smola, 2002), Herbrich (Herbrich, 2002), Shawe-Taylorand Cristianini (Shawe-Taylorand Cristianini, 2004), and S. Haykin (Haykin, 2009).

To explain how SVM works, we start with the case of separable patterns that arise in the context of pattern classification. The main idea behind the machine may be summed up as follows: Given a training sample, the support vector machine constructs a hyperplane as the decision surface in such a way that the margin of separation between positive and negative examples is maximized. This basic idea has an immediate extension to the more difficult case of nonlinearly separable patterns.

A notion that is central to the development of the support vector learning algorithm is the *inner-product kernel* between a *support vector*$\boldsymbol{x_s}$and a vector **x** drawn from the input data space. Most importantly, the support vectors consist of a small subset of data points extracted by the learning algorithm from the training sample itself. Indeed, it is because of this central property that the learning algorithm, involved in the construction of a support vector machine, is also referred to as a *kernel method.* However, the kernel method basic to the design of a support vector machine is *optimal*, with the optimality being rooted in convex optimization. This highly desirable feature of the machine is achieved at the cost of increased computational complexity.

Support vector machine can be used to solve both pattern-classification and nonlinear-regression problems. However, it is in solving difficult pattern-classification problems where support vector machines have made their most significant impact.

2.1 Optimal Hyperplane for Linearly Separable Patterns

Consider the training sample

$$\{(x_i, s_i)\}, \quad i = 1,2, \dots, N \qquad (1)$$

where $\mathbf{x}_i$ *is* the input feature set for the *i*th protein and s_i is the corresponding desired response (class). In this sub section, we assume that the class represented by the subset $s_i = +1$ and the class represented by the subset $s_i = -1$are linearly separable. That is the decision surface that separates the two classes is a hyperplane of the form

$$\boldsymbol{w^T x + b} = 0, \qquad (2)$$

where **x** is an input vector, **w** is an adjustable weight vector, ***b*** is a bias. Then the decision condition may be written as

$$s_i\,(\boldsymbol{w^T} x_i + b) \geq 1, \quad i = 1, 2, \dots, N \qquad (3)$$

For a given weight vector w and bias b, the separation between the hyperplane defined in Equation (2) and the closest data point is called the margin of separation, denoted by *d*. The goal of a support vector machine is to find the particular hyperplane for which the margin of separation, *d*, is maximized. Under this condition, the decision surface is referred to as the optimal hyperplane.

Figure 1. illustrates the geometric construction of an optimal hyperplane for a two-dimensional input space. Let $w_{opt}$and b_{opt} denote the optimum values of the weight vector and bias, respectively. Then the function

$$g(x) = w_{opt}\boldsymbol{x} + b_{opt} \qquad (4)$$

gives an algebraic measure of the distance from x to the optimal hyperplane (Duda and Hart, 1973).

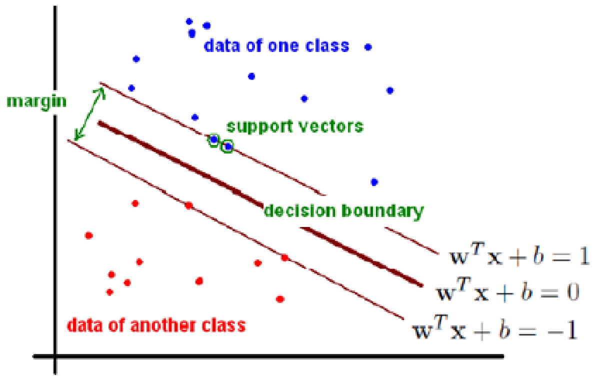
data of one class
margin
support vectors
decision boundary
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$
data of another class

Figure 1.Optimal hyperplane for linearly separable patterns: The data points on the boundaries of the margin are support vectors.

x can be expressed as

$$\boldsymbol{x} = x_p + r\frac{w_{opt}}{\|w_{opt}\|} \qquad (5$$

where x_p is the normal projection of x onto the optimal hyperplane and r is the desired algebraic distance; r is positive if x is on the positive side of the optimal hyperplane and negative if x is on the negative side. Since, by definition, $g(x_p) = 0$, it follows that

$$g(x) = w_{opt}^T\boldsymbol{x} + b_{opt} = r\|w_{opt}\| \qquad (6)$$

or, equivalently,

$$\boldsymbol{r} = \frac{g(x)}{\|w_{opt}\|} \qquad (7)$$

In particular, the distance from the origin (i.e., x = 0) to the optimal hyperplane is given by $b_{opt} = r\|w_{opt}\|$. If

$b_{opt} > 0$, the origin is on the positive side of the optimal hyperplane; if $b_{opt} < 0$,it is on the negative side. If $b_o = 0$, the optimal hyperplane passes through the origin. A geometric interpretation of these algebraic results is given in Figure. 2.

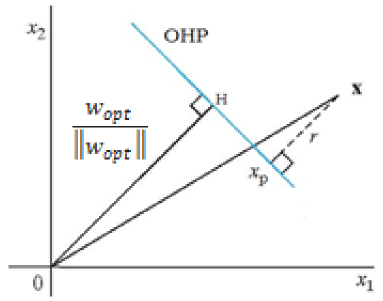

Figure2.Geometric interpretation of algebraic distances of points to the optimal hyperplane for a two-dimensional case.

The issue at hand is to find the parameters $w_{opt}$and $b_{opt}$for the optimal hyperplane, given the training set T = {($\mathbf{x}_i$, s_i), , 1,2 …, N}. In light of the results portrayed in Figure. 2, we see that the pair (w_{opt},b_{opt}) must satisfy the following constraint:

$$\begin{cases} w_{opt}^T\boldsymbol{x} + b_{opt} \geq 1, \ for\, s_i = +1 \\ w_{opt}^T\boldsymbol{x} + b_{opt} \leq -1, \ for\, s_i = -1 \end{cases} \qquad (8)$$

Note that if Equation (2) holds—that is, if the patterns are linearly separable—we can always rescale $w_{opt}$and b_{opt}such that Equation (8) holds; this scaling operation leaves Equation (4) unaffected.

The particular data points $(\boldsymbol{x_i}, s_i)$ for which the first or second line of Equation (8) is satisfied with the equality sign are called *support vectors*—hence the name “support vector machine.”All the remaining examples in the training sample are completely irrelevant. Because of their distinct property, the support vectors play a prominent role in the operation of this class of learning machines. In conceptual terms, the support vectors are those data points that lie closest to the optimal hyperplane and are therefore the most difficult to classify. As such, they have a direct bearing on the optimum location of the decision surface.

Consider a support vector $\boldsymbol{x_{sv}}$for which $\boldsymbol{s_{sv}} = +1$. Then, by definition, we have

$$g(x_{sv}) = w_{opt}^T x_{sv} + b_{opt} = \begin{cases} +1\, for\ s_{sv} = +1 \\ -1\, for\ s_{sv} = -1 \end{cases} \qquad \textbf{(9)}$$

From Equation (7), the algebraic distance from the support vector $\boldsymbol{x_{sv}}$ to the optimal hyperplane is

$$r = \frac{g(x_{sv})}{\|w_{opt}\|} = \begin{cases} \frac{1}{\|w_{opt}\|} \ for\ s_{sv} = +1 \\ \frac{-1}{\|w_{opt}\|} \ for\ s_{sv} = -1 \end{cases} \qquad (10)$$

where the plus sign indicates that$\boldsymbol{x_{sv}}$lies on the positive side of the optimal hyperplane and the minus sign indicates that $\boldsymbol{x_{sv}}$lies on the negative side of the optimal hyperplane. Let q denote the optimum value of the *margin of separation* between the two classes that constitute the training sample t. Then, from Equation (10), it follows that

$$q = 2r = \frac{2}{\|w_{opt}\|} \qquad (11)$$

Equation (11) states the following: Maximizing the margin of separation between binary classes is equivalent to minimizing the Euclidean norm of the weight vector w.

In summary, the optimal hyperplane defined by Equation (4) is unique in the sense that the optimum weight vector w_{opt} provides the maximum possible separation between positive and negative examples. This optimum condition is attained by minimizing the Euclidean norm of the weight vector **w**.

2.1.1 Quadratic Optimization for Finding the Optimal Hyperplane

The support vector machine is formulated in the realm of convex optimization (Boyd and Vandenbergh 2004; Bertsekas et al. 2003), hence the well-defined optimality of the machine is guaranteed. Proceeds goes along four main steps:

1. Statement of the problem in the primal weight space as a constrained-optimization problem is set.
2. The Lagrangian function of the problem is constructed.
3. The conditions for optimality of the machine are derived.
4. The optimization problem in the dual space of Lagrange multipliers is formulated.

The Primal Problem

Let us formally state the constrained-optimization problem as follows: Given the training sample $T = \{(\boldsymbol{x_i}, s_i), \quad i = 1,2, \dots, N\}$, find the optimum values of the weight vector *w* and bias *b* such that they satisfy the constraints in (8),and the weight vector *w* minimizes the cost function

$$\mathcal{H}(w) = w^T w/2 \qquad (12)$$

This constrained-optimization problem is called the *primal problem.* It is basically characterized as follows:

- The cost function $\mathcal{H}(w)$ is a *convex* function of **w**.
- The constraints are *linear* in **w**.

Therefore, we may solve the constrained-optimization problem by using the *method of Lagrange multipliers* (Bertsekas, 1995). First, we construct the *Lagrangian function*

$$\mathfrak{H}(w, b, \alpha) = \frac{w^T w}{2} - \sum_{i=1}^{N} \alpha_i [s_i(\boldsymbol{w}^T x_i + b) - 1] \qquad (13)$$

where the auxiliary nonnegative variables $\alpha_i$are called *Lagrange multipliers.* The solution to the constrained-optimization problem is determined by the *saddle point* of the Lagrangian function $\mathfrak{H}(w, b, \alpha)$. A saddle point of a Lagrangian is a point where the roots are real, but of opposite signs; such a singularity is always unstable. The saddle point has to be *minimized* with respect to **w** and *b*; it also has to be *maximized* with respect to *a*. Thus, computing the gradient of$\mathfrak{H}(w,b,\alpha)$with respect to *(**w**,b)*and setting the results equal to zero, we get the following *conditions of optimality*

$$\nabla\mathfrak{H}(w,b,\alpha)=0 \qquad (14)$$

Application of optimality condition to the Lagrangian function of Equation (13) yields the following equality:

$$w=\sum_{i=1}^{N_s}\alpha_i s_i x_i \qquad (15)$$

Application of optimality condition to the Lagrangian function of Equation (13) yields another constraint

$$\sum_{i=1}^{N}\alpha_i s_i=0 \qquad (16)$$

The solution vector $\boldsymbol{w}$ is defined in terms of an expansion that involves the *N* training examples. Note, however, that although this solution is unique by virtue of the convexity of the Lagrangian, the same cannot be said about the Lagrange multipliersα_i.

It is also important to note that for all the constraints that are not satisfied as equalities, the corresponding multiplier $\alpha_i$must be zero. In other words, only those multipliers that exactly satisfy the condition

$$\alpha_i[s_i(\boldsymbol{w}^T x_i + b)-1]=0 \qquad (17)$$

can assume nonzero values. This property is a statement of the *Karush–Kuhn–Tucker condition*(Fletcher, 1987; Bertsekas, 1995;Karush, 1939; Kuhn and Tucker, 1951; Kuhn1976).

The Dual Problem

The *primal problem* deals with a convex cost function and linear constraints. Given such a constrained-optimization problem, it is possible to construct another problem called the *dual problem.* This second problem has the same optimal value as the primal problem, but with the Lagrange multipliers providing the optimal solution. In particular, we may state the following *duality theorem* (Bertsekas, 1995):

(a) If the primal problem has an optimal solution, the dual problem also has an optimal solution, and the corresponding optimal values are equal.

(b) In order for $w_{opt}$to be an optimal primal solution and $\alpha_{opt}$to be an optimal dual solution, it is necessary and sufficient that $w_{opt}$is feasible for the primal problem, and

$$\Phi(w_{opt})=\psi(w_{opt},b_{opt},\alpha_{opt})=\min_w\psi(w,b,\alpha)(18)$$

To postulate the dual problem for our primal problem, we first expand Equation (13), term by term, obtaining

$$\psi(w,b,\alpha)=\frac{w^T w}{2}-\sum_{i=1}^{N}\alpha_i s_i\boldsymbol{w}^T x_i-b\sum_{i=1}^{N}\alpha_i s_i+\sum_{i=1}^{N}\alpha_i \qquad (19)$$

The third term on the right-hand side of Equation (19) is zero by virtue of the optimality condition of Equation (16). Furthermore, from Equation (15), we have

$$w^T w=\sum_{i=1}^{N}\alpha_i s_i\boldsymbol{w}^T x_i=\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j s_i s_j x_i^T x_j \qquad (20)$$

Accordingly, setting the objective function $\psi(w,b,\alpha)=Q(\alpha)$, we may reformulate Equation (19) as

$$Q(\alpha)=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j s_i s_j x_i^T x_j \qquad (21)$$

where the $\alpha_i$are all nonnegative. Note that we have changed the notation from $\psi(w,b,\alpha)$ to $Q(\alpha)$, so as to reflect the transformation from the *primal* optimization problem to its *dual.*

We may now state the *dual problem* as follows:

Given the training sample $T=\{(x_i,s_i),\ \ i=1,2,\dots,N\}$, find the Lagrange multipliers $\{\alpha_i,\ \ i=1,2,\dots,N\}$which constitutes a solution to the following constrained optimization problem:

$Max\ Q(\alpha)$

subject to the constraints

1) $\sum_{i=1}^{N}\alpha_i s_i=0$

2) $\alpha_i\geq 0,\ \ i=1,2,\dots,N \qquad (22)$

Unlike the primal optimization problem based on the Lagrangian of Equation (13), the dual problem defined in Equation (20) is formulated entirely in terms of the training data. Moreover, the function Q(a) to be maximized depends only on the input patterns in the form of a set of dot products

$$x_i^T x_j, i,j=1,2,\dots,N \qquad (23)$$

The support vectors constitute a subset of the training sample, which means that the solution vector is sparse (Girosi, 1998; Vapnik 1998; Steinwart, 2003; Suykens, et al. 2002). That is to say, constraint (2) (Equation 22) of the dual problem is satisfied with the inequality sign for all the support vectors for which the α's are nonzero, and with the equality sign for all the other data points in the training sample, for which the α's are all zero. Accordingly, having determined the optimum Lagrange multipliers, denoted by $\alpha_{opt,i}$, we may compute the optimum weight vector w by using Equation (15) as

$$\sum_{i=1}^{N_{sv}}\alpha_{opt,i}s_i x_i \qquad (24)$$

where $N_{sv}$is the number of support vectors for which the Lagrange multipliers $\alpha_{opt,i}$ are all nonzero. To compute the optimum bias b_{opt} we may use the w_{opt}thus obtained and take advantage of Equation (9), which pertains to a positive support vector:

$$b_{opt}=1-w_{opt}^T x_{sv}=1-\sum_{i=1}^{N_{sv}}\alpha_{opt,i}s_i x_i^T x_{sv}\,,\ \ for\ s_{sv}=1 \qquad (25)$$

Recall that the support vector x corresponds to any point *(x,s)* in the training sample for which the Lagrange multiplier α is nonzero. From a numerical (practical) perspective, it is better to average Equation (23) over all the support vectors—that is, over all the nonzero Lagrange multipliers.

2.2 Statistical Properties of the Optimal Hyperplane

In a support vector machine, a structure is imposed on the set of separating hyperplanes by constraining the Euclidean norm of the weight vector **w**. Specifically, we may state the following theorem (Vapnik, 1995, 1998):

Let D denote the diameter of the smallest ball containing all the input vectors$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$. The set of optimal hyperplanes described by the equation

$$w_{opt}^T x + b_{opt} = 0 \qquad (26)$$

has a VC dimension, h, bounded from above as

$$h \leq min\left\{\left\lceil\frac{D^2}{\rho^2}\right\rceil, m_0\right\} + 1 \qquad (27)$$

where the ceiling sign $\lceil\ \ \rceil$means the smallest integer greater than or equal to the number enclosed within the sign, $\rho$is the margin of separation equal to $\rho = \frac{2}{\|w\|}$, and $m_0$is the dimensionality of the input space.

The VC dimension, short for *Vapnik– Chervonenkis dimension*, provides a measure of the complexity of a space of functions (Vapnik– Chervonenkis, 1964). The theorem just stated tells us that we may exercise control over the VC dimension, that is the complexity of the optimal hyperplane, independently of the dimensionality $m_0$of the input space, by properly choosing the margin of separation p.

Suppose, then, we have a nested structure made up of separating hyperplanes described by

$$S_k = \{\boldsymbol{w}^T x + b \colon \|w\|^2 \leq c_k\,,\ \ k = 1,2,\ldots\} \quad (28)$$

By virtue of the upper bound on the VC dimension *h* defined in Equation (27), the nested structure described in Equation (28) may be reformulated in terms of the margin of separation in the equivalent form

$$S_k = \left\{\left\lceil\frac{r^2}{\rho^2}\right\rceil + 1 \colon \rho^2 \geq \alpha_k,\ \ k = 1,2,\ldots\right\} \qquad (29)$$

The $\alpha_k$and $c_k$in Equations (28) and (29) are constants.

Equation (22) states that the optimal hyperplane is a hyperplane for which the margin of separation between the positive and negative examples is the largest possible. Equivalently, Equation (28) states that construction of the optimal hyperplane is realized by making the squared Euclidean norm of the weight vector **w** the smallest possible. In a sense, these two equations reinforce the statement we made previously in light of Equation (11).

2.3 Optimal Hyperplane for Nonseparable Patterns

The discussion thus far has focused on linearly separable patterns. In this section, we consider the more difficult case of nonseparable patterns. Given such a sample of training data, it is not possible to construct a separating hyperplane without encountering classification errors. Nevertheless, we would like to find an optimal hyperplane that minimizes the probability of classification error, averaged over the training sample.

The margin of separation between classes is said to be *soft* if a data point $(\boldsymbol{x}_i, s_i)$ violates the following condition:

$$s_i\,(\boldsymbol{w}^T x_i\ +\ b\,) \geq 1,\quad i = 1, 2, \ldots, N \qquad (30)$$

This violation can arise in one of two ways:

• The data point $(\boldsymbol{x}_i, s_i)$ falls inside the region of separation, but on the correct side of the decision surface.
• The data point $(\boldsymbol{x}_i, s_i)$ falls on the wrong side of the decision surface.

Note that we have correct classification in the first case, but misclassification in the second. To set the stage for a formal treatment of nonseparable data points, we introduce a new set of nonnegative scalar variables, ξ_i, $i = 1, 2, \ldots, N$ into the definition of the separating hyperplane (i.e., decision surface), as shown here:

$$s_i\,(\boldsymbol{w}^T x_i\ +\ b\,) \geq 1 - \xi_i,\quad i = 1, 2, \ldots, N \qquad (31)$$

$\xi_i$are called *slack variables*; they measure the deviation of a data point from the ideal condition of pattern separability. For ξ_i< 1, the data point falls inside the region of separation, but on the correct side of the decision surface. Forξ_i> 1, it falls on the wrong side of the separating hyperplane. The support vectors are those particular data points that satisfy Equation (31) precisely even if ξ_i> 0. Moreover, there can be support vectors satisfying the condition ξ_i = 0. Note that if an example with ξ_i> 0 is left out of the training sample, the decision surface will change. The support vectors are thus defined in exactly the same way for both linearly separable and nonseparable cases.

The problem is to find a separating hyperplane for which the misclassification error, averaged over the training sample, is minimized. We may do this by minimizing the functional

$$\Gamma(\xi) = \sum_{i=1}^N \mathcal{U}(\xi_i - 1) \qquad (32)$$

with respect to the weight vector **w**, subject to the constraint described in Equation (31) and the constraint on $\|\boldsymbol{w}\|$. The function $\mathcal{U}(\ \ )$ is the *unit step function.*

Minimization of $\Gamma(\xi)$ with respect to **w** is a nonconvex optimization problem (Cook, 1971; Garey and Johnson 1979; Cormen et al. 1990). To make the optimization problem mathematically tractable, we approximate the functional $\Gamma(\xi)$ by writing

$$\Gamma(\xi) = \sum_{i=1}^N \xi_i \qquad (33)$$

Moreover, we simplify the computation by formulating the functional to be minimized with respect to the weight vector **w** as follows:

$$\psi(w,\xi) = \frac{w^T w}{2} + C\sum_{i=1}^{N}\xi_i \quad (34)$$

As before, minimizing the first term in Equation (34) is related to the support vector machine. As for the second term, $\sum_{i=1}^{N}\xi_i$, it is an upper bound on the number of test errors.

The parameter *C* controls the tradeoff between complexity of the machine and the number of nonseparable points; it may therefore be viewed as the *reciprocal* of a parameter commonly referred to as the "regularization" parameter.When the parameter *C* is assigned a large value, the implication is that the designer of the support vector machine has high confidence in the quality of the training sample t. Conversely, when *C* is assigned a small value, the training sample t is considered to be noisy, and less emphasis should therefore be placed on it. The parameter *C* may be determined *experimentally.*

The functional $\psi(w,\xi)$ is optimized with respect to **w** and ξ_i, $i = 1,2,\ldots,N$, subject to the constraint described in Equation (31), andξ_i> 0. In so doing, the squared norm of **w** is treated as a quantity to be jointly minimized with respect to the nonseparable points rather than as a constraint imposed on the minimization of the number of nonseparable points.

The optimization problem for nonseparable patterns just stated includes the optimization problem for linearly separable patterns as a special case. Specifically, setting ξ_i= 0 for all i in both Eqs. (31) and (34) reduces them to the corresponding forms for the linearly separable case.

We may now formally state the *primal problem* for the nonseparable case as follows:

Given the training sample

$$T = \{(\boldsymbol{x_i}, s_i),\quad i = 1,2,\ldots,N\}$$

find the optimum values of the weight vector **w** and bias b such that they satisfy the constraint

$$s_i\,(\boldsymbol{w^T}x_i + b) \geq 1 - \xi_i,\quad \xi_i \geq 0,\ i = 1,2,\ldots,N \quad (35)$$

and such that the weight vector **w** and the slack variables ξ_i minimize the cost functional

$$\psi(w,\xi) = \frac{w^T w}{2} + C\sum_{i=1}^{N}\xi_i \quad (36)$$

where C is a user-specified positive parameter.

Using the method of Lagrange multipliers and proceeding in a manner similar to that described in Section 2, we may formulate the dual problem for nonseparable patterns as follows: Given the training sample $T = \{(\boldsymbol{x_i}, s_i),\ i = 1,2,\ldots,N\}$, find the Lagrange multipliers $\{\alpha_i,\ i = 1,2,\ldots,N\}$that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j s_i s_j x_i^T x_j \quad (37)$$

subject to the two constraints

$$\begin{cases} \sum_{i=1}^{N}\alpha_i s_i = 0, \\ 0 \leq \alpha_i \leq C, i = 1,2,\ldots,N \end{cases} \quad (38)$$

where C is a user-specified positive parameter.

Note that neither the slack variables $\xi_i$nor their own Lagrange multipliers appear in the dual problem. The dual problem for the case of nonseparable patterns is thus similar to that for the simple case of linearly separable patterns, except for a minor, but important difference. The objective function $Q(\alpha)$ to be maximized is the same in both cases. The nonseparable case differs from the separable case in that the constraint $0 \leq \alpha_i$ is replaced with the more stringent constraint $0 \leq \alpha_i \leq C$. Except for this modification, the constrained optimization for the nonseparable case and computations of the optimum values of the weight vector **w** and bias *b* proceed in the same way as in the linearly separable case. Note also that the support vectors are defined in exactly the same way as before.

Unbounded Support Vectors

For a prescribed parameter *C*, a data point $(\boldsymbol{x_i}, s_i)$ for which the condition $0 \leq a_i \leq C$holds is said to be an *unbounded*, or *free support vector*. When $a_i = C$, we find that

$$s_i F(\boldsymbol{x_i}) < 1,\ a_i = C \quad (39)$$

where $F(\mathbf{x}_i)$ is the approximating function realized by the support vector machine for the input $\mathbf{x}_i$. On the other hand, when $a_i = 0$, we find that

$$s_i F(\boldsymbol{x_i}) > 1, a_i = 0 \quad (40)$$

In light of these two arguments, it follows that for unbounded support vectors, we have

$$s_i F(\boldsymbol{x_i}) = 1 \quad (41)$$

Consequently, there is a distinct possibility of degeneracy (i.e., reduced optimality conditions) in the solution to a pattern-classification problem computed by the support vector machine. By this statement, we mean that a point $(\boldsymbol{x_i}, s_i)$ that satisfies the margin requirement exactly has no constraint on the possible value of the associated a_i.

In Rifkin (2002), it is argued that the number of unbounded support vectors is the primary reason for how difficult, in a computational sense, the training of a support vector machine can be.

Support Vector Machine for Pattern Classification

With the material on how to find the optimal hyperplane for nonseparable patterns at hand, we are now in a position to formally describe the construction of a support vector machine for a pattern-recognition task.

Basically, the idea of a support vector machine hinges on two mathematical operations summarized here and illustrated in Figure. 3.:

1. Nonlinear mapping of an input vector into a high-dimensional feature space that is hidden from both the input and output;
2. Construction of an optimal hyperplane for separating the features discovered in step 1.

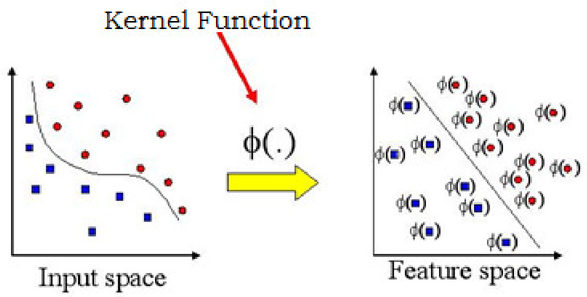

Figure 3.Nonlinear mapping from the input space to the feature space

The number of features constituting the hidden space in Figure 3 is determined by the number of support vectors. Thus, SVM theory provides an analytic approach for determining the optimum size of the feature (hidden) space, thereby assuring optimality of the classification task.

2.4 Inner-product Kernel for Support Vector Machines

Let $\boldsymbol{x}$be a vector from the input space of dimension *n*. Let $\{\boldsymbol{\varphi}_i(x),\ \ i = 1,2,\dots,\infty\}$ be a set of nonlinear functions that, transform the input space of dimension *n* to a feature space of infinite dimensionality. Given this transformation, we may define a hyperplane acting as the decision surface in accordance with the formula

$$\sum_{i=1}^{\infty} w_i\ \varphi_i(\boldsymbol{x}) = 0 \qquad (42)$$

in matrix notation

$$\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{\Phi}(\boldsymbol{x}) = 0 \qquad (43)$$

where $\boldsymbol{\Phi}(\boldsymbol{x})$ is the *feature vector* and **w** is the corresponding *weight vector* of weights $\{w_i,\ \ i = 1,2,\dots,\infty\}$that transforms the feature space to the output space. In the output space, decision is made whether the input vector $\boldsymbol{x}$ belongs to one of two possible classes, positive or negative. Bias is included in *w*in Equation (42), and in (43).

As in Section 2.3, we seek linear separability of the transformed patterns in the feature space. For this, we may adapt Equation (15) to our present situation by expressing the weight vector as

$$w = \sum_{i=1}^{N_s} \alpha_i s_i \Phi(x_i) \qquad (44)$$

where

$$\Phi(x_i) = [\,\varphi_1(x_i), \varphi_2(x_i), \dots]^T \qquad (45)$$

and N_s is the number of support vectors. Hence, substituting Equation (44) into Equation (43), we may express the decision surface in the output space as

$$w = \sum_{i=1}^{N_s} \alpha_i s_i \Phi^{\mathrm{T}}(x_i)\Phi(x) \qquad (46)$$

We now immediately see that the scalar term $\Phi^{\mathrm{T}}(x_i)\Phi(x)$ in Equation (46) represents an *inner product.* Accordingly, let this inner-product term be denoted as the scalar

$$\mathrm{K}(x_i, x) = \Phi^{\mathrm{T}}(x_i)\Phi(x) = \sum_{i=1}^{\infty} \varphi_j^T(x_i)\varphi_j(x), \qquad i, 1,2,..,N_s \qquad (47)$$

Correspondingly, we may express the optimal decision surface (hyperplane) in the output space as

$$\sum_{i=1}^{N_s} \alpha_i s_i \mathrm{K}(x_i, x) = 0 \qquad (48)$$

The function $\mathrm{K}(x_i, x)$ is called the *inner-product kernel* which is formally defined as follows (Shawe-Taylor and Cristianini, 2004; Aizerman et al. 1964a, 1964b, Vapnik and Chervonenkis 1964; Boser et al. 1992).

The kernel $\mathrm{K}(x_i, x)$is a function that computes the inner product of the images produced in the feature space under the embedding $\Phi$of two data points in the input space.

Hence the kernel $\mathrm{K}(x_i, x)$is a function that has two basic properties (Schölkopf and Smola 2002; Herbrich, 2002; Shawe-Taylor and Cristianini 2004):

*Property 1.*It is symmetric about the center point $\boldsymbol{x}_i$, that is,

$$K(\boldsymbol{x}, \boldsymbol{x}_i) = K(\boldsymbol{x}_i, \boldsymbol{x}) \text{ for all } \boldsymbol{x}_i,\quad i = 1,2,\dots,N \qquad (49)$$

and it attains its maximum value at the point $\mathbf{x} = \boldsymbol{x}_i$.

*Property 2***.** The total volume under the surface of the function $K(\boldsymbol{x}, \boldsymbol{x}_i)$ is a constant.

2.5 Design of Support Vector Machines

The expansion of the kernel $K(\boldsymbol{x}, \boldsymbol{x}_i)$in Equation (47) permits us to construct a decision surface that is nonlinear in the input space, but whose image in the feature space is linear. With this expansion at hand, we may now state the dual form for the constrained optimization of a support vector machine as follows:

Given the training sample $\{(\boldsymbol{x}_i, \boldsymbol{s}_i),\ \ i = 1,2,\dots,N\}$, find the Lagrange multipliers $\{\alpha_i,\ \ i = 1,2,\dots,N\}$that maximize the objective function

$$\mathrm{Q}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j\, s_i s_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \qquad (50)$$

subject to the constraints

$$\sum_{i=1}^{N} s_i\alpha_i = 0, \qquad (51)$$

$$0 \le \alpha_i \le C,\quad i = 1,2,\dots,N$$

where C is a user-specified positive parameter.

Constraint (1) arises from optimization of the Lagrangian

Q(a) with respect to the bias *b*, which is a rewrite of Equation (13). The dual problem just stated is of the same form as that for the case of nonseparable patterns considered in Section 2.3, except for the fact that the inner product $x_i^T x_j$has been replaced by the kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

SVMs have been used in a wide range of problems including drug design (Robert et al. 2000), image recognition and text classification (Joachims, 1998), microarray gene expression data analysis (Brown et. al. 2000), and protein fold recognition, predicting protein structural class (Cai, et. al. 2001, Dinubhai , and Shah, 2013; Akcesme, 2015), protein structure prediction (Mandle, at al, 2012; Hua, and Sun, 2001; Anjum, 2007; Ward, 2003). In this paper, we apply Vapnik's Support Vector Machine (Ding, and Dubchak 2001) for the classification of proteins into four structural classes all-alpha, all-beta, alpha+beta, alpha/beta.

3. A TEAM OF SUPPORT VECTOR MACHINES FOR PREDICTING PROTEIN STRUCTURAL CLASSES

SVM is one of the most widely used and powerful classification algorithms to predict protein structural class (Zhang 2013; Liu and Jia 2010; Zhang 2011; Ding 2012; Kurgan and Homeian 2006). SVM maps the input data to a higher dimensional space where it looks for a hyperplane to separate the training protein samples by their classes.

3.1 Support Vector Machines in Use

Support vector machines, are better performs in two class pair wise classification then multiple classification. Therefore we train six support vector machines to classify

types = {{1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4}, {3, 4}} (52)

where type 1 is all-α, type 2 is all-β, type 3 is α+β, and type 4 α/β. To maximize objective function of the support vector machine, built-in function *NMaximize* of Wolfram MATHEMATICA is employed. Because of limitations of this function, the sizes of training sets are limited to 40, and 80.

To classify proteins, each protein is represented by a vector of dimension 33. To transform linearly non separable data to a higher dimensional space where transformed data is linearly separable, Gaussian type radial base functions are employed.

$$\varphi(x) = \mathrm{Exp}[-\|x - \mathrm{xtr}\|\hat{}2/(2\mathrm{sigma}\hat{}2)] \qquad (53)$$

The objective function of the maximization problem (50) is constructed. *NMaximize* function returns the values of weights $\alpha_i, i = 1, \dots, 2m$ where *m* is the size of the training set from each of the two classes.

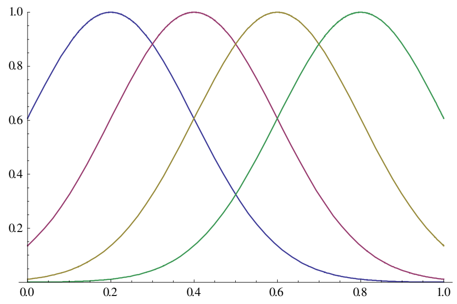
Figure 4.Gaussian radial base functions$\varphi(x)$.

According to the decision function (51), a test set that consists of proteins from classes for which the support vector machine is trained is classified.

3.1 A Team of Competing Support Vector Machines

To run six support vector machines in (52) in tandem, that are trained to classify classes pair wise, we benefit from the fact that a SVM which is trained to distinguish between class *i*, and class j can also be used to distinguish between class *j*, and class *i*. We also invent four dummy SVMs to distinguish between classes *i*, and class *i* which always vote *zero*. Therefore we create a classifier that is a combination of sixteen support vector machines.

SVM(1,1)	SVM(1,2)	SVM(1,3)	SVM(1,4)
SVM(2,1)	SVM(2,2)	SVM(2,3)	SVM(2,4)
SVM(3,1)	SVM(3,2)	SVM(3,3)	SVM(3,4)
SVM(4,1)	SVM(4,2)	SVM(4,3)	SVM(4,4)

When a test data enters into the classifier, a 4×4 decision matrix is created. Assume the test data was from class all-β. If they succeed correctly classifying this data, support vector machines of the second row would vote +1, while machines on the second column vote -1. The other experts in the classifier would vote mixed votes since they don’t have expertise in classifying this type of a protein.

0	-1	1	-1
1	0	1	1
-1	-1	0	-1
1	-1	1	0

When we reorganize votes in Figure 6 columns after rows we get:

$$\{0,-1,1,-1,0,1,-1,1\}$$
$$\{1,0,1,1,-1,0,-1,-1\}$$
$$\{-1,-1,0,-1,1,1,0,1\}$$
$$\{1,-1,1,0,-1,1,-1,0\}$$

Assume an unknown protein is entered into classifier, and the decision of experts are as follows:

$$\{0,-1,1,-1,0,1,-1,1\}$$
$$\{1,0,-1,1,-1,0,1,-1\}$$
$$\{1,1,0,-1,-1,-1,0,-1\}$$
$$\{1,-1,1,0,-1,1,-1,0\}$$

Hamming distances of these vectors to their ideal vectors are 3, 2, 1, and 2. Then classifier concludes that the unknown protein is of class three, that is the class α+β.

4. FEATURES DERIVED FROM PSEUDO IMAGES OFAMINO ACID SEQUENCES

The major goal of this section may be summarized as follows: take the sequence of a protein, attaching a gray level to twenty amino acids, transform the sequence into a sequence of gray levels, then, and generate the features that will subsequently be fed to a classifier in order to classify the image in one of the possible classes.

4.1 Transform Sequence into a Sequence of Gray Levels

Consider the protein with PDB code 1A1W (Eberstadt, M et.al. 1998). It consists of amino acids

MDPFLVLLHSVSSSLSSSELTELKYLCLGRVGKRKL
ERVQSGLDLFSMLLEQNDLEPGHTELLRELLASLRR
HDLLRRVDDFELEHHHHHH

When the symbols for amino acids are transformed into integers from 1 to 20 according their positions in the

letteralphabet = "ARNDCQHGEILKMFPSTWYV";

we obtain the sequence of integers

$$K = \{1, 2, 3, \ldots, 20\} \quad (54)$$

When each integer in (54) is replaced by its reciprocal, then the sequence of gray levels

$$I = \{1, 1/2, 1/3, \ldots, 1/20\} \quad (55)$$

is obtained.

When each letter in the amino acid sequence of the protein is replaced by its integer label in (54), a sequence of integers we get

{13, 4, 15, 14, 11, 20, 11, 11, 7, 16, 20, 16, 16, 16, 11, 16, 16, 16, 9, 11, 17, 9, 11, 12, 19, 11, 5, 11, 8, 2, 20, 8, 12, 2, 12, 11, 9, 2, 20, 6, 16, 8, 11, 4, 11, 14, 16, 13, 11, 11, 9, 6, 3, 4, 11, 9, 15, 8, 7, 17, 9, 11, 11, 2, 9, 11, 11, 1, 16, 11, 2, 2, 7, 4, 11, 11, 2, 2, 20, 4, 4, 14, 9, 11, 9, 7, 7, 7, 7, 7, 7}

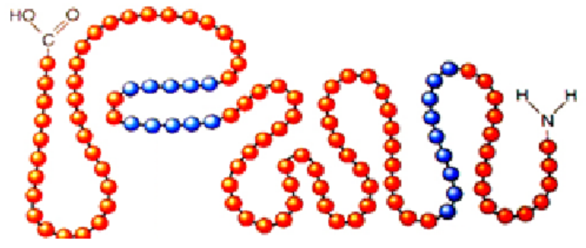

Figure 5.Sequence of a Protein: a chain of amino acids.

Then each integer in the above is replaced by its reciprocal to obtain a sequence of gray levels. Only to make it apparent we repeat it fifty times. The gray image of the result is in the below:

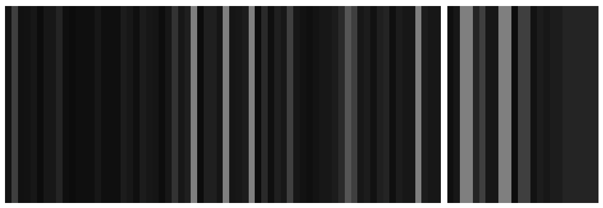

Figure 6. Gray image that represents the protein with PDB code 1A1W.

This image is stored in the computer as a one-dimensional array

$$S(z_i), i = 1, \ldots, N, \; z_i \in I \quad (56)$$

where N is the number of residues of the protein. *ith* element of the array corresponds to a pixel of the image, whose gray level is equal to z_i. Gray levels $z_i \in I$ are quantized in Ng=20 discrete gray levels and Ng is known as the depth of the image.

4.1 Drive Features from Gray Images

The need for feature generation stems from our inability to use the raw data in classification. For our classification task the number of the pixels is the same as the length of sequence which is the number of the residues in amino acid chain.

Feature generation is a procedure that computes new variables that in one way or another originate from the stored values of the image array *S*. The goal is to generate features that exhibit high information-packing properties, from the class separability point of view. Because we cannot use the raw data *S* directly, the features should encode efficiently the relevant information residing in the original data.

4.1.1 Features for Texture Characterization

The distribution of the gray levels over the pixels of a region of an image, determines its texture. By the appearance we describe an image as
fine or coarse,
smooth or irregular,
homogeneous or inhomogeneous.

Our goal in this subsection is to generate appropriate features that, somehow, quantify these properties of an image region. These features will emerge by the use of space relations underlying the gray level distribution.

First-Order Statistics Features

1)The first-order histogram:

Let z be a variable representing the gray levels in the region of interest.

P(z) = number of pixels with gray level z / total number of pixels in the region (57)

That is, P(z) is the fraction of pixels with gray level z.

2) Moments:

$m_i = E[z^i] = \sum_{j=1}^{20} z_j^i\, P(z_j),\ \ i = 1,2,\ldots$(58)

It is clear that$m_0 = 1$ and $m_1 = E[z]$, the mean value of *z*.

3)Central Moments:

$$\mu_i = E[(z - E[z])^i] = \sum_{j=1}^{20} (z_j - m_1)^i\, P(z_j),\ \ i = 1,2,\ldots \tag{59}$$

The most frequently used central moments are μ_2, which is the variance, and μ_3 which is known as the skewness and μ_4, the kurtosis of the image.
The variance is a measure of the image spread, that is, a measure of how much the gray levels differ from the mean. Skewness is a measure of the degree of histogram asymmetry around the mean, and kurtosis is a measure of the image sharpness.

4)Absolute Moments:

$$\mu_i = E[|z - E[z]|^i] = \sum_{j=1}^{20} |z_j - m_1|^i\, P(z_j),\ \ i = 1,2,\ldots \tag{60}$$

5) Entropy:

Entropy is a measure of image uniformity.

$H = -E[\log_2 P(z)] = \sum_{j=1}^{20} P(z_j)\log_2 P(z_j)$ (61)

Image is uniform if $P(z)$is constant. That is H is higher.

Second-Order Statistics

6) Features-Co-occurrence Matrices

First-order statistics provide information related to the gray level distribution in the image, they do not give any information about the relative positions of the various gray levels within the image. All low-value gray levels may position together, or they may interchange with the high-value ones. This type of information can be extracted from the second-order histograms, where the pixels are considered in pairs.

For each combination value of d a one-dimensional histogram is defined:

$P(S(i) = z_1, S(i \pm d) = z_2) =$ *The number of pixels at distance d with values* (z_1, z_2) */ total number of possible pairs.* (62)

7) Angular second moment

This feature is a measure of the smoothness of the image. If all pixels are of the same gray level z = k , then

$P\,(k,k) = 1\ and$

$P(i,j) = 0, i \neq k\ or\ j \neq k\ , and$ (63)

$ASM = 1.$

At the other extreme, if we could have all possible pairs of gray levels with equal probability 1/R, then ASM = R/R^2 = 1/R.

The less smooth the image is, the more uniformly distributed P(i,j) and the lower the ASM.

$ASM = \sum_{i=1}^{20}\sum_{j=1}^{20} P(i,j)^2$ (64)

8) Contrast

This is a measure of local gray level variations hence it is the image contrast.

$CON = \sum_{n=1}^{20} n^2 \left\{\sum_{i=1}^{20}\sum_{j=1}^{20} P(i,j)\right\}_{|i-j|=n}$ (65)

Indeed, inside of the bracelet is the percentage of pixel pairs whose intensity differs by n. The factor n^2weighs the big differences more; thus, CON takes high values for images of high contrast.

9) Inverse Difference Moment

$IDF = \sum_{i=1}^{20}\sum_{j=1}^{20} P(i,j)/(1 + (i-j)^2)$ (66)

Because of the $1/(i-j)^2$ dependence, this feature takes high values for low-contrast images.

10) Second-Order Entropy

$H_{xy} = -\sum_{j=1}^{20} P(i,j)\log_2 P(i,j)$ (67)

Second-order entropy is a measure of randomness of the distribution of gray levels, and takes low values for smooth images as the first order entropy.

Features Using Gray Level Run Lengths

Length of a set of consecutive pixels having the same gray level value is a gray level run. Run length features encode textural information related to the number of times each gray level occurs. *Gray Level Run Lengths matrix* $Q(i,j)$is such that the element (i,j) represents a pixel with the gray level$z_i$appears in the image by itself, j=1, the number of times it appears in pairs, j=2, and so on. Its $(i\,,j\,)$th element gives the number of times a gray level $z_i,\ i = 1,2,\ \ldots, 20$, appears in the image with run length$r_j\ , j = l\,, 2, \ldots, N_r$.Considering the length of proteins in datasets, we have adopted *Gray Level Run Lengths matrices* $Q(i,j)$of size20×50.

11) Short Run Emphasis

This feature emphasizes small run lengths, due to the division byj^2.

$$SRE = \frac{\sum_{i=1}^{20}\sum_{j=1}^{Nr} Q_{RL}(i,j)/j^2}{\sum_{i=1}^{20}\sum_{j=1}^{Nr} Q_{RL}(i,j)} \quad (68)$$

The denominator is the total number of run lengths in the matrix.

12) Long Run Emphasis

This feature emphasizes long run lengths, due to the multiplication byj^2.

$$LRE = \frac{\sum_{i=1}^{20}\sum_{j=1}^{Nr} Q_{RL}(i,j)j^2}{\sum_{i=1}^{20}\sum_{j=1}^{Nr} Q_{RL}(i,j)} \quad (69)$$

SRE will be large for coarser and LRE will be large for smoother images.

13) Gray Level Nonuniformity

The term in the brackets is the total number of run lengths for each gray level.

$$GLNU = \frac{\sum_{i=1}^{20}\left[\sum_{j=1}^{Nr} Q_{RL}(i,j)\right]^2}{\sum_{i=1}^{20}\sum_{j=1}^{Nr} Q_{RL}(i,j)} \quad (70)$$

Large run length values contribute a great deal because of the square. When runs are uniformly distributed among the gray levels, GLNU takes small values.

14) Run Length Nonuniformity

$$RLN = \frac{\sum_{i=1}^{Nr}\left[\sum_{j=1}^{20} Q_{RL}(i,j)\right]^2}{\sum_{i=1}^{20}\sum_{j=1}^{Nr} Q_{RL}(i,j)} \quad (71)$$

RLN is a measure of run length nonuniformity

15) Run Percentage

$$RP = \frac{\sum_{i=1}^{20}\sum_{j=1}^{Nr} Q_{RL}(i,j)}{L} \quad (72)$$

RP takes low values for smooth images.

Summary of Features

First-Order Statistics Features

Features 1-20 come from the first-order histogram: That is, the fraction of pixels with gray level $z_i, i = 1, \ldots, N$. Moments 21, central moments 22, absolute moments 23, entropy 24.

Second-Order Statistics

Angular second moment 25, contrast 26, inverse difference moment 27, second-order entropy 28.

Features Using Gray Level Run Lengths

Shortrun emphasis 29, longrun emphasis 30, gray level nonuniformity 31, run length nonuniformity 32, run percentage 33.

Each protein in the dataset represented by these 33 features derived from the pseudo images of proteins using their amino acid sequences. Then this data is used in training six machine teams of support vector machines.

4. RESULTS AND DISCUSSION

For the proteins in 25PDB database, pseudo images are created from their amino acid sequences. Then 33 features are derived from these images by the techniques of digital image processing (Akcesme, B., Thesis 2016). Using these features, proteins of 25PDB dataset are classified. Classification accuracies are as in the confusion matrix below:

$$\begin{bmatrix} 85.5 & 3. & 3. & 8.5 \\ 4. & 68.5 & 10.5 & 17. \\ 7. & 2.5 & 88. & 2.5 \\ 12. & 19.5 & 1. & 67.5 \end{bmatrix}$$

It is seen that Q3 accuracy around 75% is achieved, and 88% accuracy in the classification of the class α+β is remarkable.

In the same thesis work, other feature sets are also derived from pseudo images of PSSM matrices, and predicted secondary structures of proteins. Especially features derived from the pseudo images of secondary structures supplied Q3 classification accuracy around 90%.

REFERENCES

Aizerman, A., Braverman, E. M., & Rozoner, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning.Automation and remote control, 25, 821-837.

Aizerman, M. A., Braverman,E.M., and Rozonoer, L.I. (1964). The probability problem of patternrecognition learning and the method of potential functions, Automation and RemoteControl, vol. 25, pp. 1175–1193.

Akcesme, B. (2015) Prediction of Protein Structural Classes for Low-Similarity Sequences Based On Predicted Secondary Structure, Southeast Europe Journal of Soft Computing, Vol 4 No. 1, 24-31.

Akcesme, B., (2016) Predicting Protein Structural Classes, PhD thesis in preparation, International University of Sarajevo, Faculty of Natural Sciences and Engineering, Department of Genetics and Bionegineering.

Anjum B.R.A.(2007) Protein Secondary Structure Prediction Using Support Vector Machines, Neural Networks and GeneticAlgorithms. Thesis, Georgia State University.

Bertsekas,D.P. (1995)*Nonlinear Programming*. Belmont, MA:Athenas Scientific.

Bertsekas, D.P., with Nedich, A., and Ozdaglar, A.E.(2003)*Convex Analysis and Optimization*,Nashua, NH:Athena Scientific.

Boser, B., GuyonI., and Vapnik, V.N. (1992) A training algorithm for optimal margin classifiers,Fifth Annual Workshop on Computational Learning Theory, pp. 144–152. San Mateo, CA:Morgan Kaufmann.

Boyd, S., and Vandenberghe, L.(2004)*Convex Optimization*. Cambridge, U.K., and New York:Cambridge University Press.

Brown MPS, Grundy WN, Lin D, Cristianini N, Sugnet C, Ares JM,Haussler D. (2000) Knowledge-based Analysis of Microarray GeneExpression Data by using Support Vector Machines Proc. Natl.Acad. Sci. 97:262-267

Cai, Y. D., Liu, X. J., Xu, X. B., & Chou, K. C. (2003). Support vector machines for prediction of protein domain structural class. Journal of theoretical biology, 221(1), 115-120.

Cai, YD., Liu, XJ., Xu, XB. and Zhou, GP. (2001)Support Vector Machines for predicting protein structural class, BMC Bioinformatics, 2:3

Cook,A.S. (1971) The complexity of theorem-proving procedures, Proceedings of the 3rd AnnualACM Symposium on Theory of Computing, pp. 151–158, New York.

Cormen,T.H.,C.E. Leiserson, and R.R. Rivest, 1990. *Introduction to Algorithms*. Cambridge,MA:MIT Press.

Cortes, C. , and Vapnik, V. (1995). Support vector networks. Mach. Learning 20, 273–293.

Cucker, F., andSmale, S. (2001)On the Mathematical Foundations of Learning, Bulletin (New Series) Of The American Mathematical Society, Volume 39, Number 1, Pages 1–49

Ding CHQ, Dubchak, I. (2001) Multi-class Protein Fold Recognition Using Support Vector Machines and Neural Networks Bioinformatics, 4(17):349-358

Dinubhai , PM., Shah, HB. (2013) Comparative Study of Multi-class Protein Structure Prediction Using Advanced Soft computing Techniques, International Journal of Engineering Science and Innovative Technology (IJESIT)Volume 2, Issue 2, March 2013

Duda, R.O., and Hart,P.E. (1973)*Pattern Classification and Scene Analysis*, New York:Wiley.

Eberstadt, M., Huang, B., Chen, Z., Meadows, R. P., Ng, S. C., Zheng, L., ... & Fesik, S. W. (1998). NMR structure and mutagenesis of the FADD (Mort1) death-effector domain. Nature, 392(6679), 941-945.

Fletcher, R., (1987)*Practical Methods of Optimization*, 2d ed., New York:Wiley.

Garey, M.R., and Johnson, D.S. (1979)*Computers and Intractability*, New York:W.H. Freeman.

Girosi, F. (1998) An equivalence between sparse approximation and support vector machines,Neural Computation, vol. 10, pp. 1455–1480.

Haykin, S. (2009)*Neural Networks and Learning Machines*, Third Edition, Pearson Education, Inc., Upper Saddle River, New Jersey 07458.

Herbrich, R., (2002)*Learning Kernel Classifiers:Theory and Algorithms*, Cambridge,MA:MIT Press.

Hua, S., and Sun, Z. (2001). A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. Journal of molecular biology, 308(2), 397-407.

Joachims T. (1998) Text Categorization with Support Vector Machines:Learning with Many Relevant Features, Proceedings ofthe European Conference on Machine Learning, Springer, 1998

Karush,W., (1939) Minima of functions of several variables with inequalities as side conditions,master's thesis, Department of Mathematics, University of Chicago.

Kuhn, H.W., and Tucker, A.W. (1951) Nonlinear programming, in J. Neyman, ed., Proceedingsof the 2nd Berkley Symposium on Mathematical Statistics and Probabilities, pp. 481–492,Monterey CA: University of California Press.

Kuhn, H.W., (1976) Nonlinear programming: A historical view, in R.N. Cottle and C.E. Lemke,eds., SIAM-AMS Proceedings, vol. IX, American Mathematical Society, pp. 1–26.

Mandle, AK., Jain, P. and Shrivastava, SK. (2012) Protein Structure Prediction UsingSupport Vector Machine, International Journal on Soft Computing (IJSC) Vol. 3, No.1, 67-78.

Mukherjee, S., Tamayo, P., Slonim, D., Verri, A., Golub, T., Mesirov, J., & Poggio, T. (1999). Support vector machine classification of microarray data.

Rifkin, R.M., 2002. Everything old is new again: A fresh look at historical approaches in machinelearning, Ph.D. thesis, MIT.

Robert B., Matthew T., Sean H., Bernard B. (2000) Drug Design by MachineLearning: Support Vector Machine for Pharmaceutical DataAnalysis, Proceedings of the AISB'00 Symposium on Artificial Intelligence in Bioinformatics. 1-4

Shao, XJ., Tian, YJ., Deng, NY. (2007) SVM-based Automatic Classification for Protein Structural Domain, The First International Symposium on Optimization and Systems Biology (OSB'07)Beijing, China, August 8–10.

Schölkopf, B., (1997)*Support Vector Learning*, Munich, Germany: R. Oldenbourg Verlag

Schölkopf, B., and Smola, A.J. (2002)*Learning with Kernels: Support Vector Machines, Regularization,Optimization, and Beyond*, Cambridge, MA: MIT Press.

Shawe-Taylor, J., and Cristianini, N.(2004)*Kernel Methods for Pattern Analysis*, Cambridge, U.K., and New York: Cambridge University Press.

Steinwart, I. (2003) Sparseness of support vector machines, J. Machine Learning Research, vol. 4,pp. 1071–1105.

Sugnet, C. W., Furey, T. S., and Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. Proceedings of the National Academy of Sciences, 97(1), 262-267.

Suykens, J.A.,Van Gestel, T., DeBrabanter, J., DeMoor, B., and Vanderwalle, J. (2002)*Least-Squares Support Vector Machines*, River Edge, NJ:World Scientific.

Vapnik VN: Statistical Learning Theory Wiley-Interscience, New York, 1998

Vapnik, V.N.(1998)*Statistical Learning Theory*, John Wiley & Sons, Inc.

Vapnik,V.N., and Chervonenkis, A.Ya. (1964) A note on a class of perceptrons,Automation andRemote Control, vol. 25, pp. 103–109.

Vapnik,V.N.(1995)*The Nature of Statistical Learning Theory*, New York: Springer-Verlag.

Ward,J. J., McGuffin, L. J., Buxton, B. F. and Jones, D. T. (2003) Secondary structure prediction with supportvector machines, Bioinformatics, Vol. 19 no. 1, pages 1650–1655.