**Tutorial on Support Vector Machine**

Loc Nguyen

Sunflower Soft Company, Ho Chi Minh City, Vietnam

Email: ng\_phloc@yahoo.com

**Abstract**

Support vector machine is a powerful machine learning method in data classification. Using it for applied researches is easy but comprehending it for further development requires a lot of efforts. This report is a tutorial on support vector machine with full of mathematical proofs and example, which help researchers to understand it by the fastest way from theory to practice. The report focuses on theory of optimization which is the base of support vector machine.

**Keywords:** Support vector machine, SVM, separating hyperplane, sequential minimal optimization, SMO.

**1. Support vector machine**

**Support vector machine (SVM)** (Law, 2006) is a supervised learning algorithm for classification and regression. Given a set of *p-*dimensional vectors in vector space, SVM finds the *separating hyperplane* that splits vector space into sub-set of vectors; each separated sub-set (so-called data set) is assigned by one class. There is the condition for this separating hyperplane: “it must maximize the margin between two sub-sets”. Figure 1.1 (Wikibooks, 2008) shows separating hyperplanes *H*1, *H*2, and *H*3 in which only *H*2 gets maximum margin according to this condition.

Chart, scatter chart

Description automatically generated

**Figure 1.1.** Separating hyperplanes

Suppose we have some *p-*dimensional vectors; each of them belongs to one of two classes. We can find many *p–*1 dimensional hyperplanes that classify such vectors but there is only one hyperplane that maximizes the margin between two classes. In other words, the nearest between one side of this hyperplane and other side of this hyperplane is maximized. Such hyperplane is called *maximum-margin hyperplane* and it is considered as the SVM *classifier*.

Let {*X*1, *X*2,…, *Xn*} be the training set of *n* vectors *Xi* (s) and let *yi* = {+1, –1} be the class label of vector *Xi*. Each *Xi* is also called a data point with attention that *vectors can be identified with data points*. Data point can be called *point*, in brief. It is necessary to determine the maximum-margin hyperplane that separates data points belonging to *yi=+*1 from data points belonging to *yi=*–1 as clear as possible.

According to theory of geometry, arbitrary hyperplane is represented as a set of points satisfying *hyperplane equation* specified by equation 1.1.

|  |  |
| --- | --- |
|  | (1.1) |

Where the sign “” denotes the dot product or scalar product and *W* is *weight vector* perpendicular to hyperplane and *b* is the *bias*. Vector *W* is also called perpendicular vector or normal vector and it is used to specify hyperplane. Suppose *W*=(*w*1, *w*2,…, *wp*) and *Xi=*(*xi*1, *xi*2,…, *xip*), the scalar product is:

Given scalar value *w*, the multiplication of *w* and vector *Xi* denoted *wXi* is a vector as follows:

Please distinguish scalar product and multiplication *wXi*.

The essence of SVM method is to find out weight vector *W* and bias *b* so that the hyperplane equation specified by equation 1.1 expresses the maximum-margin hyperplane that maximizes the margin between two classes of training set.

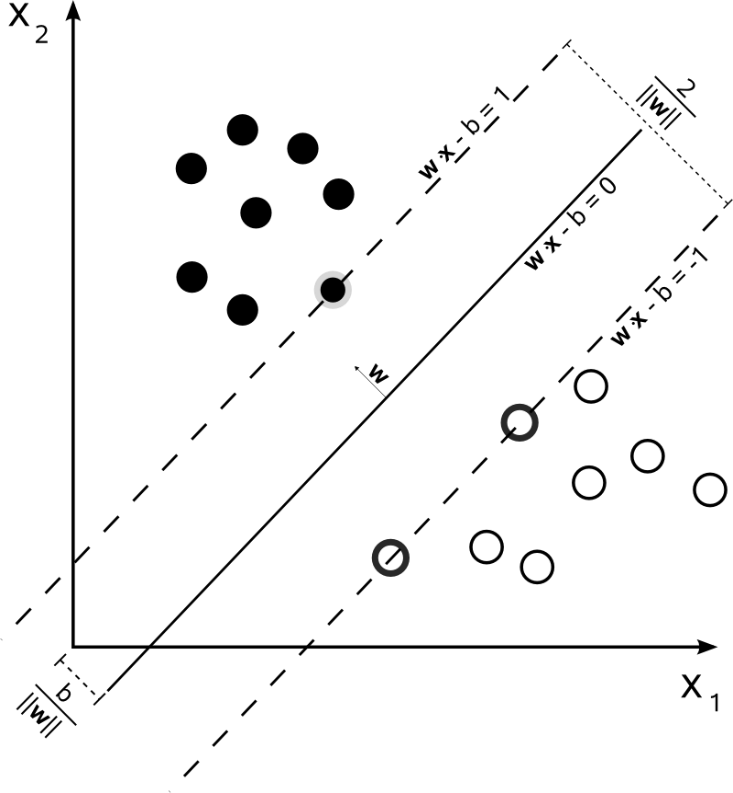
The value is the offset of the (maximum-margin) hyperplane from the origin along the weight vector *W* where |*W*| or ||*W*|| denotes length or module of vector *W*.

Note that we use two notations and for denoting the length of vector.

Additionally, the value is the width of the margin as seen in figure 1.2. To determine the margin, two parallel hyperplanes are constructed, one on each side of the maximum-margin hyperplane. Such two parallel hyperplanes are represented by two hyperplane equations, as shown in equation 1.2 as follows:

|  |  |
| --- | --- |
|  | (1.2) |

Figure 1.2 (Wikibooks, 2008) illustrates maximum-margin hyperplane, weight vector *W* and two parallel hyperplanes. As seen in the figure 1.2, the margin is limited by such two parallel hyperplanes. Exactly, there are two margins (each one for a parallel hyperplane) but it is convenient for referring both margins as the unified single margin as usual. You can imagine such margin as a road and SVM method aims to maximize the width of such road. Data points lying on (or are very near to) two parallel hyperplanes are called support vectors because they construct mainly the maximum-margin hyperplane in the middle. This is the reason that the classification method is called support vector machine (SVM).



**Figure 1.2.** Maximum-margin hyperplane, parallel hyperplanes and weight vector *W*

To prevent vectors from falling into the margin, all vectors belonging to two classes *yi=*1and *yi=*–1 have two following constraints, respectively:

As seen in figure 1.2, vectors (data points) belonging to classes *yi*=+1 and *yi=*–1 are depicted as black circles and white circles, respectively. Such two constraints are unified into the so-called classification constraint specified by equation 1.3 as follows:

|  |  |
| --- | --- |
|  | (1.3) |

As known, *yi=+*1and *yi=*–1 represent two classes of data points. It is easy to infer that maximum-margin hyperplane which is the result of SVM method is the classifier that aims to determined which class (+1or –1) a given data point *X* belongs to. Your attention please, each data point *Xi* in training set was assigned by a class *yi* before and maximum-margin hyperplane constructed from the training set is used to classify any different data point *X*.

Because maximum-margin hyperplane is defined by weight vector *W*, it is easy to recognize that the essence of constructing maximum-margin hyperplane is to solve the constrained optimization problem as follows:

Where |*W*| is the length of weight vector *W* and is the classification constraint specified by equation 1.3. The reason of minimizing is that distance between two parallel hyperplanes is and we need to maximize such distance in order to maximize the margin for maximum-margin hyperplane. Then maximizing is to minimize . Because it is complex to compute the length |*W*|, we substitute for when is equal to the scalar product as follows:

The constrained optimization problem is re-written, shown in equation 1.4 as below:

|  |  |
| --- | --- |
|  | (1.4) |

Where is called target function with regard to variable *W*. Function is called constraint function with regard to two variables *W*, *b* and it is derived from the classification constraint specified by equation 1.3. There are *n* constraints because training set {*X*1, *X*2,…, *Xn*} has *n* data points *Xi* (s). Constraints inside equation 1.3 implicate the perfect separation in which there is no data point falling into the margin (between two parallel hyperplanes, see figure 1.2). On the other hand, the imperfect separation allows some data points to fall into the margin, which means that each constraint function *gi*(*W*,*b*) is subtracted by an error . The constraints become (Honavar, p. 5):

Which implies that:

We have a *n*-component error vector *ξ=*(*ξ*1, *ξ*2,…, *ξn*) for *n* constraints. The penalty is added to the target function in order to penalize data points falling into the margin. The penalty *C* is a pre-defined constant. Thus, the target function *f*(*W*) becomes:

If the positive penalty is infinity, then, target function may get maximal when all errors *ξi* must be 0, which leads to the perfect separation specified by aforementioned equation 1.4.

Equation 1.5 specifies the general form of constrained optimization originated from equation 1.4.

|  |  |
| --- | --- |
|  | (1.5) |

Where *C* ≥ 0 is the penalty.

The *Lagrangian function* (Boyd & Vandenberghe, 2009, p. 215) is constructed from constrained optimization problem specified by equation 1.5. Let *L*(*W*, *b*, *ξ*, *λ*, *μ*) be Lagrangian function where *λ=*(*λ*1, *λ*2,…, *λn*) and *μ=*(*μ*1, *μ*2,…, *μn*) are *n*-component vectors, *λi* ≥ 0 and *μi* ≥ 0, . We have:

In general, equation 1.6 represents Lagrangian function as follows:

|  |  |
| --- | --- |
|  | (1.6) |

Note that *λ=*(*λ*1, *λ*2,…, *λn*) and *μ=*(*μ*1, *μ*2,…, *μn*) are called Lagrange multipliers or Karush-Kuhn-Tucker multipliers (Wikipedia, Karush–Kuhn–Tucker conditions, 2014) or dual variables. The sign “” denotes scalar product and every training data point *Xi* was assigned by a class *yi* before.

Suppose (*W\**, *b\**) is solution of constrained optimization problem specified by equation 1.5 then, the pair (*W\**, *b\**) is minimum point of target function *f*(*W*) or target function *f*(*W*) gets minimum at (*W\**, *b\**) with all constraints . Note that *W\** is called *optimal weight vector* and *b\** is called *optimal bias*. It is easy to infer that the pair (*W\**, *b\**) represents the maximum-margin hyperplane and it is possible to identify (*W\**, *b\**) with the maximum-margin hyperplane. The ultimate goal of SVM method is to find out *W\** and *b\**. According to Lagrangian duality theorem (Boyd & Vandenberghe, 2009, p. 216) (Jia, 2013, p. 8), the pair (*W\**, *b\**) is the extreme point of Lagrangian function as follows:

|  |  |
| --- | --- |
|  | (1.7) |

Where Lagrangian function *L*(*W*, *b*, *ξ*, *λ*, *μ*) is specified by equation 1.6.

Now it is necessary to solve the Lagrangian duality problem represented by equation 1.7 to find out *W\**. Thus, the Lagrangian function *L*(*W*, *b*, *ξ*, *λ*, *μ*) is minimized with respect to the primal variables *W*, *b* and maximized with respect to the dual variables *λ=*(*λ*1, *λ*2,…, *λn*) and *μ=*(*μ*1, *μ*2,…, *μn*), in turn. If gradient of *L*(*W*, *b*, *ξ*, *λ*, *μ*) is equal to zero then, *L*(*W*, *b*, *ξ*, *λ*, *μ*) will gets minimum value with note that gradient of a multi-variable function is the vector whose components are first-order partial derivative of such function. Thus, setting the gradient of *L*(*W*, *b*, *ξ*, *λ*, *μ*) with respect to *W*, *b*, and *ξ* to zero, we have:

In general, *W\** is determined by equation 1.8 as follows:

|  |  |
| --- | --- |
|  | (1.8) |

It is required to determine Lagrange multipliers *λ=*(*λ*1, *λ*2,…, *λn*) in order to evaluate *W\**. Substituting equation 1.8 into Lagrangian function *L*(*W*, *b*, *ξ*, *λ*, *μ*) specified by equation 1.6, we have:

(according to equation 1.8, *L*(*W*,*b*,*ξ*,*λ*,*μ*) gets minimum at and and )

Where *l*(*λ*) is called *dual function* represented by equation 1.9.

|  |  |
| --- | --- |
|  | (1.9) |

According to Lagrangian duality problem represented by equation 1.7, *λ=*(*λ*1, *λ*2,…, *λn*) is calculated as the maximum point *λ*\**=*(*λ*1\*, *λ*2\*,…, *λn*\*) of dual function *l*(*λ*). In conclusion, maximizing *l*(*λ*) is the main task of SVM method because the optimal weight vector *W\** is calculated based on the optimal point *λ\** of dual function *l*(*λ*) according to equation 1.8.

Maximizing *l*(*λ*) is quadratic programming (QP) problem, specified by equation 1.10.

|  |  |
| --- | --- |
|  | (1.10) |

The constraints are implied from the equations when . The QP problem specified by equation 1.10 is also known as Wolfe problem (Honavar, p. 42).

There are some methods to solve this QP problem but this report introduces a so-called Sequential Minimal Optimization (SMO) developed by author Platt (Platt, 1998). The SMO algorithm is very effective method to find out the optimal (maximum) point *λ\** of dual function *l*(*λ*).

Moreover SMO algorithm also finds out the optimal bias *b\**, which means that SVM classifier (*W\**, *b\**) is totally determined by SMO algorithm. The next section described SMO algorithm in detail.

**2. Sequential minimal optimization**

The ideology of SMO algorithm is to divide the whole QP problem into many smallest optimization problems. Each smallest problem relates to only two Lagrange multipliers. For solving each smallest optimization problem, SMO algorithm includes two nested loops as shown in table 2.1 (Platt, 1998, pp. 8-9):

|  |
| --- |
| SMO algorithm solves each smallest optimization problem via two nested loops:   * The outer loop finds out the first Lagrange multiplier *λi* whose associated data point *Xi* violates KKT condition (Wikipedia, Karush–Kuhn–Tucker conditions, 2014). Violating KKT condition is known as the first choice heuristic. * The inner loop finds out the second Lagrange multiplier *λj* according to the second choice heuristic. The second choice heuristic that maximizes optimization step will be described later. * Two Lagrange multipliers *λi* and *λj* are optimized jointly according to QP problem specified by equation 1.10.   SMO algorithm continues to solve another smallest optimization problem. SMO algorithm stops when there is convergence in which no data point violating KKT condition is found; consequently, all Lagrange multipliers *λ*1, *λ*2,…, *λn* are optimized. |

**Table 2.1.** Ideology of SMO algorithm

The ideology of violating KKT condition as the first choice heuristic is similar to the event that wrong things need to be fixed priorly. Before describing SMO algorithm in detailed, the KKT condition with subject to SVM is mentioned firstly because violating KKT condition is known as the first choice heuristic of SMO algorithm. KKT condition indicates both partial derivatives of Lagrangian function and complementary slackness are zero (Wikipedia, Karush–Kuhn–Tucker conditions, 2014). Referring equations 1.8 and 1.4, the KKT function of SVM is summarized as equation 2.1:

|  |  |
| --- | --- |
|  | (2.1) |

Note, the complementary slackness is *λi*(1 – *yi* (*W*∘*Xi* – *b*) – *ξi*) = 0 and –*μiξi* = 0 for all *i* because of the two conditions 1 – *yi* (*W*∘*Xi* – *b*) – *ξi* ≤ 0 and –*μi* ≤ 0 for all *i*. When we understand deeply convex optimization, the KKT condition is the same to the QP problem specified by equation 1.10 if target function and constraint sets are convex. Thus, the solution (*W\**, *λ\**) is saddle point of Lagrangian function.

KKT condition is analyzed into three following cases (Honavar, p. 7):

1. If *λi=*0 then, *μi* = *C* – *λi* = *C*. It implies *ξi=*0 from equation . Then, from equation we have:
2. If 0 < *λi* < *C* then, we have . Due to *μi* = *C* – *λi* > 0, it implies *ξi=*0 from equation . It is easy to infer that:
3. If *λi=C* then, we have *μi* = *C* – *λi* = 0 and . Due to *μi* = 0, it implies *ξi* ≥ 0 from equation . Given *ξi* ≥ 0 the equation leads to:

Let be prediction error, we have:

The KKT condition implies:

|  |  |
| --- | --- |
|  | (2.2) |

Equation 2.2 expresses directed corollaries from KKT condition. It is commented on equation 2.2 that if *Ei*=0, the KKT condition is always satisfied. Data points *Xi* satisfying equation *yiEi=*0 lie on the margin (lie on the two parallel hyperplanes). These points are called *support vectors*. According to KKT corollary, support vectors are always associated with non-zero Lagrange multipliers such that 0<*λi*<*C*. Note, such Lagrange multipliers 0<*λi*<*C* are also called non-boundary multipliers because they are not bounds such as 0 and *C*. So, support vectors are also known as *non-boundary* data points. It easy to infer from equation 1.8.

that support vectors along with their non-zero Lagrange multipliers form mainly the optimal weight vector *W\** representing the maximum-margin hyperplane – the SVM classifier. This is the reason that this classification approach is called support vector machine (SVM). Figure 2.1 (Moore, 2001, p. 5) illustrates an example of support vectors.

Chart

Description automatically generated

**Figure 2.1.** Support vectors

Violating KKT condition is the first choice heuristic of SMO algorithm. By negating three corollaries specified by equation 2.2, KKT condition is violated in three following cases:

By logic induction, these cases are reduced into two cases specified by equation 2.3.

|  |  |
| --- | --- |
|  | (2.3) |

Equation 2.3 is used to check whether given data point *Xi* violates KKT condition. For explaining the logic induction, from (*λ*i = 0 and *yiEi* > 0) and (0 < *λi* < *C* and *yiEi* ≠ 0) obtaining (*λi* < *C* and *yiEi* > 0), from (*λi* = *C* and *yiEi* < 0) and (0 < *λi* < *C* and *yiEi* ≠ 0) obtaining (*λi* > 0 and *yiEi* < 0). Conversely, from (*λi* < *C* and *yiEi* > 0) and (*λi* > 0 and *yiEi* < 0) we obtain (*λ*i = 0 and *yiEi* > 0), (0 < *λi* < *C* and *yiEi* ≠ 0), and (*λi* = *C* and *yiEi* < 0).

The main task of SMO algorithm (see table 2.1) is to optimize jointly two Lagrange multipliers in order to solve each smallest optimization problem, which maximizes the dual function *l*(*λ*).

Where,

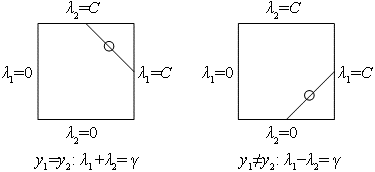
Without loss of generality, two Lagrange multipliers *λi* and *λj* that will be optimized are *λ*1 and *λ*2 while all other multipliers *λ*3, *λ*4,…, *λn* are fixed. Old values of *λ*1 and *λ*2 are denoted and . Your attention please, old values are known as current values. Thus, *λ*1 and *λ*2 are optimized based on the set: , , *λ*2, *λ*3,…, *λn*. The old values and are initialized by zero (Honavar, p. 9). From the condition , we have:

and

It implies following equation of line with regard to two variables *λ*1 and *λ*2:

|  |  |
| --- | --- |
|  | (2.4) |

Equation 2.4 specifies the linear constraint of two Lagrange multipliers *λ*1 and *λ*2. This constraint is drawn as diagonal lines in figure 2.2 (Honavar, p. 9).



**Figure 2.2.** Linear constraint of two Lagrange multipliers

In figure 2.2, the box is bounded by the interval [0, *C*] of Lagrange multipliers, . The left box and right box in figure 2.2 imply that *λ*1 and *λ*2 are proportional and inversely proportional, respectively. SMO algorithm moves *λ*1 and *λ*2 along diagonal lines so as to maximize the dual function *l*(*λ*). Multiplying two sides of equation by *y*1, we have:

Where *s=y*1*y*2. Let,

We have equation 2.5 as a variant of the linear constraint of two Lagrange multipliers *λ*1 and *λ*2 (Honavar, p. 9):

|  |  |
| --- | --- |
| Where, | (2.5) |

By fixing multipliers *λ*3, *λ*4,…, *λn*, all arithmetic combinations of , , *λ*3, *λ*4,…, *λn* are constants denoted by term “*const*”. The dual function *l*(*λ*) is re-written (Honavar, pp. 9-11):

Let,

Let be the optimal weight vector based on old values of two aforementioned Lagrange multipliers. Following linear constraint of two Lagrange multipliers specified by equation 2.4, we have:

Let,

We have (Honavar, p. 10):

Let and assessing the coefficient of *λ*2, we have (Honavar, p. 11):

(Where is the old value of the bias *b*)

According to equation 2.3, and are old prediction errors on *X*2 and *X*1, respectively:

Recall that we had:

Thus, equation 2.6 specifies dual function with subject to the second Lagrange multiplier *λ*2 that is optimized in conjunction with the first one *λ*1 by SMO algorithm.

|  |  |
| --- | --- |
|  | (2.6) |

Where,

The first and second derivatives of dual function *l*(*λ*2) with regard to *λ*2 are:

The quantity *η* is always non-negative due to:

Recall that the goal of QP problem is to maximize the dual function *l*(*λ*2) so as to find out the optimal multiplier (maximum point) . The second derivative of *l*(*λ*2) is always non-negative and so, *l*(*λ*2) is concave function and there always exists the maximum point . The function *l*(*λ*2) gets maximal if its first derivative is equal to zero:

Therefore, the new values of *λ*1 and *λ*2 that are solutions of the smallest optimization problem of SMO algorithm are:

Obviously, is totally determined in accordance with , thus we should focus on . Because multipliers *λi* are bounded, , it is required to find out the range of . Let *L* and *U* be lower bound and upper bound of , respectively. We have (Honavar, pp. 11-13):

1. If *s =* 1, then *λ*1 + *λ*2 = *γ*. There are two sub-cases (see figure 2.3) as follows (Honavar, p. 11):
   * If *γ* ≥ *C* then *L* = *γ* – *C* and *U* = *C*.
   * If *γ* < *C* then *L* = 0 and *U* = *γ*.
2. If *s = –*1, then *λ*1 *–* *λ*2 = *γ*. There are two sub-cases (see figure 2.4) as follows (Honavar, pp. 11-12):
   * If *γ* ≥ 0 then *L* = 0 and *U* = *C* – *γ*.
   * If *γ* < 0 then *L* = –*γ* and *U* = *C*.

Diagram, schematic

Description automatically generated

**Figure 2.3.** Lower bound and upper bound of two new multipliers in case *s =* 1

(Honavar, p. 12)

Diagram, schematic

Description automatically generated

**Figure 2.4.** Lower bound and upper bound of two new multipliers in case *s* = –1

(Honavar, p. 13)

The value is clipped as follows (Honavar, p. 12):

In the case *η=*0 that is undetermined, is assigned by which bound (*L* or *U*) maximizes the dual function *l*(*λ*2).

In general, table 2.2 summarizes how SMO algorithm optimizes jointly two Lagrange multipliers.

|  |
| --- |
| If *η* > 0:  If *η* = 0:  Where prediction errors and dual function *l*(*λ*2) are specified by equation 2.6. Lower bound *L* and upper bound *U* are described as follows:   * If *s=*1 and *γ* > *C* then *L* = *γ* – *C* and *U* = *C*. * If *s=*1 and *γ* < *C* then *L* = 0 and *U* = *γ*. * If *s = –*1 and *γ* > 0 then *L* = 0 and *U* = *C* – *γ*. * If *s = –*1 and *γ* < 0 then *L* = –*γ* and *U* = *C*.   Where according to equation 2.5.  Let Δ*λ*1 and Δ*λ*2 represent the changes in multipliers *λ*1 and *λ*2, respectively.  The new value of the first multiplier *λ*1 is re-written in accordance with the change Δ*λ*1. |

**Table 2.2.** SMO algorithm optimizes jointly two Lagrange multipliers

Basic tasks of SMO algorithm to optimize jointly two Lagrange multipliers are now described in detailed. The ultimate goal of SVM method is to determine the classifier (*W\**, *b\**). Thus, SMO algorithm updates optimal weight *W\** and optimal bias *b\** based on the new values and at each optimization step.

Let be the new (optimal) weight vector, according equation 2.1 we have:

Let be the old weight vector:

It implies:

Let be the new prediction error on *X*2:

The new (optimal) bias is determining by setting with reason that the optimal classifier (*W\**, *b\**) has zero error.

In general, equation 2.7 specifies the optimal classifier (*W\**, *b\**) resulted from each optimization step of SMO algorithm.

|  |  |
| --- | --- |
| Where is the old value of weight vector, of course we have: | (2.7) |

By extending the ideology shown in table 2.1, SMO algorithm is described particularly in table 2.3 (Platt, 1998, pp. 8-9) (Honavar, p. 14).

|  |
| --- |
| All multipliers *λi* (s), weight vector *W*, and bias *b* are initialized by zero. SMO algorithm divides the whole QP problem into many smallest optimization problems. Each smallest optimization problem focuses on optimizing two joint multipliers. SMO algorithm solves each smallest optimization problem via two nested loops:   1. The outer loop alternates one sweep through all data points and as many sweeps as possible through non-boundary data points (support vectors) so as to find out the data point *Xi* that violates KKT condition according to equation 2.3. *The Lagrange multiplier λi associated with such Xi is selected as the first multiplier aforementioned as* ***λ*1**. Violating KKT condition is known as the first choice heuristic of SMO algorithm. 2. The inner loop browses all data points at the first sweep and non-boundary ones at later sweeps so as to find out the data point *Xj* that maximizes the deviation where and are prediction errors on *Xi* and *Xj*, respectively, as seen in equation 2.6. *The Lagrange multiplier λj associated with such Xj is selected as the second multiplier aforementioned as* ***λ*2**. Maximizing the deviation is known as the second choice heuristic of SMO algorithm.    1. *Two Lagrange multipliers λ*1 *and λ*2 *are optimized jointly*, which results optimal multipliers and , as seen in table 2.2.    2. *SMO algorithm updates optimal weight W\* and optimal bias b\** based on the new values and according to equation 2.7.   SMO algorithm continues to solve another smallest optimization problem. SMO algorithm stops when there is convergence in which no data point violating KKT condition is found. Consequently, all Lagrange multipliers *λ*1, *λ*2,…, *λn* are optimized and the optimal SVM classifier (*W\**, *b\**) is totally determined. |

**Table 2.3.** SMO algorithm

When both optimal weight vector *W\** and optimal bias *b\** are determined by SMO algorithm or other methods, the maximum-margin hyperplane known as SVM classifier is totally determined. According to equation 1.1, the equation of maximum-margin hyperplane is expressed in equation 2.8 as follows:

|  |  |
| --- | --- |
|  | (2.8) |

For any data point *X*, classification rule derived from maximum-margin hyperplane (SVM classifier) is used to classify such data point *X*. Let *R* be the classification rule, equation 2.9 specifies the classification rule as the sign function of point *X*.

|  |  |
| --- | --- |
|  | (2.9) |

After evaluating *R* with regard to *X*, if *R*(*X*)=1 then, *X* belongs to class +1; otherwise, *X* belongs to class –1. This is the simple process of data classification. The next section illustrates how to apply SMO into classifying data points where such data points are documents.

**3. An example of data classification by SVM**

It is necessary to have an example for illustrating how to classify documents by SVM. Given a set of classes *C* = {*computer science*, *math*}, a set of terms *T* = {*computer*, *derivative*} and the corpus = {*doc*1*.txt*, *doc*2*.txt*, *doc*3*.txt*, *doc*4*.txt*}. The training corpus (training data) is shown in following table 3.1 in which cell (*i*, *j*) indicates the number of times that term *j* (column *j*) occurs in document *i* (row *i*); in other words, each cell represents a term frequency and each row represents a document vector. Note that there are four documents and each document in this section belongs to only one class such as computer science or math.

|  |  |  |  |
| --- | --- | --- | --- |
|  | *computer* | *derivative* | ***class*** |
| *doc*1*.txt* | 20 | 55 | math |
| *doc*2*.txt* | 20 | 20 | computer science |
| *doc*3*.txt* | 15 | 30 | math |
| *doc*4*.txt* | 35 | 10 | computer science |

**Table 3.1.** Term frequencies of documents (SVM)

Let *Xi* be data points representing documents *doc*1*.txt*, *doc*2*.txt*, *doc*3*.txt*, *doc*4*.txt*, *doc*5*.txt*. We have *X*1=(20,55), *X*2=(20,20), *X*3=(15,30), and *X*4=(35,10). Let *yi*=+1 and *yi*=–1 represent classes “*math*” and “*computer science*”, respectively. Let *x* and *y* represent terms “*computer*” and “*derivative*”, respectively and so, for example, it is interpreted that the data point *X*1=(20,55) has abscissa *x*=20 and ordinate *y*=55. Therefore, term frequencies from table 3.1 is interpreted as SVM input training corpus shown in table 3.2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | *x* | *y* | *yi* |
| *X*1 | 20 | 55 | +1 |
| *X*2 | 20 | 20 | –1 |
| *X*3 | 15 | 30 | +1 |
| *X*4 | 35 | 10 | –1 |

**Table 3.2.** Training corpus (SVM)

Data points *X*1, *X*2, *X*3, and *X*4 are depicted in figure 3.1 in which classes “*math*” (*yi*=+1) and “*computer*” (*yi* = –1) are represented by shading and hollow circles, respectively. Note that some figures (like figure 3.1) in this research are drawn by the software *Graph* http://www.padowan.dk developed by author Ivan Johansen (Johansen, 2012). I express my deep gratitude to the author Ivan Johansen for providing the great software Graph.

Timeline

Description automatically generated

**Figure 3.1.** Data points in training data (SVM)

By applying SMO algorithm described in table 2.3 into training corpus shown in table 3.1, it is easy to calculate optimal multiplier *λ\**, optimal weight vector *W\** and optimal bias *b\**. Firstly, all multipliers *λi* (s), weight vector *W*, and bias *b* are initialized by zero. This example focuses on perfect separation and so, .

**At the first sweep:**

The outer loop of SMO algorithm searches for a data point *Xi* that violates KKT condition according to equation 2.3 through all data points so as to select two multipliers that will be optimized jointly. We have:

Due to *λ*1=0 < *C=*+∞ and *y*1*E*1=1\*1=1 > 0, point *X*1 violates KKT condition according to equation 2.3. Then, *λ*1 is selected as the first multiplier. The inner loop finds out the data point *Xj* that maximizes the deviation . We have:

Because the deviation is maximal, the multiplier *λ*2 associated with *X*2 is selected as the second multiplier. Now *λ*1 and *λ*2 are optimized jointly according to table 2.2.

Optimal classifier (*W\**, *b\**) is updated according to equation 2.7.

Now we have:

The outer loop of SMO algorithm continues to search for another data point *Xi* that violates KKT condition according to equation 2.3 through all data points so as to select two other multipliers that will be optimized jointly. We have:

Due to *λ*3=0 < *C* and *y*3*E*3=1\*(10/7) > 0, point *X*3 violates KKT condition according to equation 2.3. Then, *λ*3 is selected as the first multiplier. The inner loop finds out the data point *Xj* that maximizes the deviation . We have:

Because both deviations and are maximal, the multiplier *λ*2 associated with *X*2 is selected randomly among {*λ*1, *λ*2} as the second multiplier. Now *λ*3 and *λ*2 are optimized jointly according to table 2.2.

(*L* and *U* are lower bound and upper bound of )

Optimal classifier (*W\**, *b\**) is updated according to equation 2.7.

Now we have:

The outer loop of SMO algorithm continues to search for another data point *Xi* that violates KKT condition according to equation 2.3 through all data points so as to select two other multipliers that will be optimized jointly. We have:

Due to *λ*4=0 < *C* and *y*4*E*4=(–1)\*(18/7) < 0, point *X*4 does not violate KKT condition according to equation 2.3. Then, the first sweep of outer loop stops with the results as follows:

Note that *λ*1 is approximated to 0 because it is very small.

**At the second sweep:**

The outer loop of SMO algorithm searches for a data point *Xi* that violates KKT condition according to equation 2.3 through non-boundary data points so as to select two multipliers that will be optimized jointly. Recall that non-boundary data points (support vectors) are ones whose associated multipliers are not bounds 0 and *C* (0<*λi*<*C*). At the second sweep, there are three non-boundary data points *X*1, *X*2 and *X*3. We have:

Due to and *y*1*E*1 = 1\*(–4) < 0, point *X*1 violates KKT condition according to equation 2.3. Then, *λ*1 is selected as the first multiplier. The inner loop finds out the data point *Xj* among non-boundary data points (0<*λi*<*C*) that maximizes the deviation . We have:

Because the deviation is maximal, the multiplier *λ*2 associated with *X*2 is selected as the second multiplier. Now *λ*1 and *λ*2 are optimized jointly according to table 2.2.

(*L* and *U* are lower bound and upper bound of )

Optimal classifier (*W\**, *b\**) is updated according to equation 2.7.

The second sweep stops with results as follows:

**At the third sweep:**

The outer loop of SMO algorithm searches for a data point *Xi* that violates KKT condition according to equation 2.3 through non-boundary data points so as to select two multipliers that will be optimized jointly. Recall that non-boundary data points (support vectors) are ones whose associated multipliers are not bounds 0 and *C* (0<*λi*<*C*). At the third sweep, there are only two non-boundary data points *X*2 and *X*3. We have:

Due to and *y*3*E*3 = 1\*(4/7) > 0, point *X*3 violates KKT condition according to equation 2.3. Then, *λ*3 is selected as the first multiplier. Because there are only two non-boundary data points *X*2 and *X*3, the second multiplier is *λ*2. Now *λ*3 and *λ*2 are optimized jointly according to table 2.2.

(*L* and *U* are lower bound and upper bound of )

Optimal classifier (*W\**, *b\**) is updated according to equation 2.7.

The third sweep stops with results as follows:

After the third sweep, two non-boundary multipliers were optimized jointly. You can sweep more times to get more optimal results because data point *X*3 still violates KKT condition as follows:

Due to and *y*3*E*3 = 1\*(–32/35) < 0, point *X*3 violates KKT condition according to equation 2.3. But it takes a lot of sweeps so that SMO algorithm reaches absolute convergence (*E*3=0 and hence, no KKT violation) because the penalty *C* is set to be +∞, which implicates the perfect separation. This is the reason that we can stop the SMO algorithm at the third sweep in this example. In general, you can totally stop the SMO algorithm after optimizing two last multipliers which implies that all multipliers were optimized.

As a result, *W\** and *b\** were determined:

The maximum-margin hyperplane (SVM classifier) is totally determined as below:

The SVM classifier is depicted in figure 3.2.

A picture containing timeline

Description automatically generated

**Figure 3.2.** An example of maximum-margin hyperplane

Where the maximum-margin hyperplane is draw as bold line. Data points *X*2 and *X*3 are support vectors because their associated multipliers *λ*2 and *λ*3 are non-zero (0<*λ*2=2/125<*C*=+∞, 0<*λ*3=2/125<*C*=+∞). Your attention please, that weight vector *W* is depicted as an arrow indicates mainly its direction. The scale of weight vector in this figure is very small.

Derived from the above classifier , the classification rule is:

Now we apply classification rule into document classification. Suppose the numbers of times that terms “*computer*” and“*derivative*” occur in document *D* are 40 and 10, respectively. We need to determine which class document *D*=(40, 10) is belongs to. We have:

Hence, it is easy to infer that document *D* belongs to class “*computer science*” (*yi* = –1).

**4. Conclusions**

In general, the main ideology of SVM is to determine the separating hyperplane that maximizes the margin between two classes of training data. Based on theory of optimization, such optimal hyperplane is specified by the weight vector *W\** and the bias *b\** which are solutions of constrained optimization problem. It is proved that there always exist these solutions but the main issue of SVM is how to find out them when the constrained optimization problem is transformed into quadratic programming (QP) problem. SMO which is the most effective algorithm divides the whole QP problem into many smallest optimization problems. Each smallest optimization problem focuses on optimizing two joint multipliers. It is possible to state that SMO is the best implementation version of the “architecture” SVM.

SVM is extended by concept of kernel function. The dot product in separating hyperplane equation is the simplest kernel function. Kernel function is useful in case of requirement of data transformation (Law, 2006, p. 21). There are many pre-defined kernel functions available for SVM. Readers are recommended to research more about kernel functions (Cristianini, 2001).

**References**

Boyd, S., & Vandenberghe, L. (2009). *Convex Optimization.* New York, NY, USA: Cambridge University Press.

Cristianini, N. (2001). Support Vector and Kernel Machines. *The 28th International Conference on Machine Learning (ICML).* Bellevue, Washington, USA: Website for the book "Support Vector Machines": http://www.support-vector.net. Retrieved 2015, from http://www.cis.upenn.edu/~mkearns/teaching/cis700/icml-tutorial.pdf

Honavar, V. G. (n.d.). *Sequential Minimal Optimization for SVM.* Iowa State University, Department of Computer Science. Ames, Iowa, USA: Vasant Honavar homepage.

Jia, Y.-B. (2013). *Lagrange Multipliers.* Lecture notes on course “Problem Solving Techniques for Applied Computer Science”, Iowa State University of Science and Technology, USA.

Johansen, I. (2012, December 29). Graph software. *Graph version 4.4.2 build 543(4.4.2 build 543)*. GNU General Public License.

Law, M. (2006). *A Simple Introduction to Support Vector Machines.* Lecture Notes for CSE 802 course, Michigan State University, USA, Department of Computer Science and Engineering.

Moore, A. W. (2001). *Support Vector Machines.* Carnegie Mellon University, USA, School of Computer Science. Available at http://www. cs. cmu. edu/~awm/tutorials.

Platt, J. C. (1998). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.* Microsoft Research.

Wikibooks. (2008, January 1). *Support Vector Machines.* Retrieved 2008, from Wikibooks website: http://en.wikibooks.org/wiki/Support\_Vector\_Machines

Wikipedia. (2014, August 4). *Karush–Kuhn–Tucker conditions*. (Wikimedia Foundation) Retrieved November 16, 2014, from Wikipedia website: http://en.wikipedia.org/wiki/Karush–Kuhn–Tucker\_conditions