Erasmus University Rotterdam

Bachelor Thesis

Econometrics & Operations Research

Change point method: an exact line search method for SVMs

Author:
Yegor Troyan
Student number:
386332

Supervisor:
Dr. P.J.F. Groenen
Second assessor:
Dr. D. Fok

July 2, 2017

1 Introduction

Predicting two groups from a set of predictor variables known as binary classification is not a new problem. Various different statistical approaches to binary classification are available in the literature, such as logistic regression, linear or quadratic discriminant analysis and neural networks. Another method which have recently become popular is called Support Vector Machines (SVMs) (Vapnik, 1999). This technique seems to perform better in terms of prediction quality compared to the alternatives mentioned. Its optimization problem is well defined and can be solved through a quadratic programming (Groenen et al., 2008). Moreover, the classification rule SVMs provide is relatively simple and can be immediately used with new samples. However, a disadvantage is that the nonlinear SVM interpretation in terms of predictor variables is not always possible and that the standard dual formulation of SVM may be difficult to comprehend (Groenen et al., 2008).

This paper focuses on linear SVMs, specifically on the primal linear SVM problem. The primal formulation is used as it is easier to interpret, than a standard dual one. Groenen et al. (2008) formulates the SVM in terms of a loss function regularized by a penalty term. This formulation is called an SVM loss function with an absolute hinge error. Other researchers tackled similar SVM formulation using various methods. For example, Zhang (2004) and Bottou (2010) proposed a stochastic gradient descent method. In Collins et al. (2008) the exponential gradient search method is applied. In our paper we discuss an iterative majorization approach to minimizing the SVM loss function introduced by Groenen et al. (2008). Its advantage is that at each iteration of the algorithm we are guaranteed to decrease the loss value until convergence is reached. As the SVM loss function with an absolute hinge error is convex and coercive, iterative majorization converges to a global minimum after a sufficient number of iterations.

The main focus of our paper is the development of an original line search method for optimizing the SVM loss function – the Change Point Method (CPM). The great advantage of the CPM is that it is exact. We also combine the CPM with different search directions (e.g. majorization, coordinate descent) in order to create an efficient optimization method for solving SVM problems. Finally, we compare the performance of different approaches testing them on the seven empirical data sets. The performance measures of interest are the computational efficiency and the loss value reached.

2 Linear SVM

Often in machine learning we need to classify data. Let each object from the dataset be represented by an m-dimensional vector. Suppose, each of these objects belongs to only one of the two classes. Geometrically speaking, we want to find the hyperplane that separates the points in Class_1 from the points in Class_{-1}. There might be infinitely many separations possible. A suitable candidate for the best separating hyperplane is the one for which the distance between it and the nearest point from either group – the margin – is maximal (Figure 1).

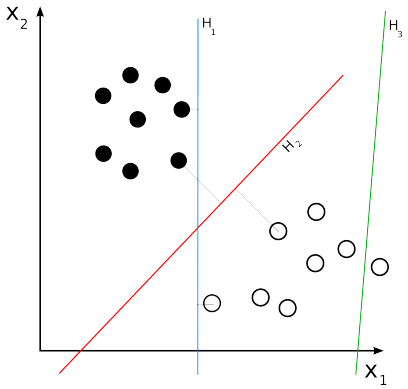


Figure 1: H_1 separates classes with a small margin; H_2 is better than H_1 as the margin is larger; H_3 doesn't separate classes

Let the data points be of the following form:

$$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\},$$

where $\boldsymbol{x}_i$ is an m-dimensional vector of real values and y_i takes values 1 and -1 (these specific numbers are used for convenience only) indicating the class of the object $\boldsymbol{x}_i$:

$$y_i = \begin{cases} 1, \text{if } \boldsymbol{x}_i \in \text{Class}_1 \\ -1, \text{if } \boldsymbol{x}_i \in \text{Class}_{-1}. \end{cases} \tag{1}$$

Let $\boldsymbol{w}$ be the $m \times 1$ vector of weights used to make a linear combination of the elements of $\boldsymbol{x}_i$. Then the prediction q_i for each object $\boldsymbol{x}_i$ is:

$$q_i = c + \boldsymbol{x}_i' \boldsymbol{w}, \tag{2}$$

where c is the intercept.

We want to find a hyperplane which separates the two groups of the objects in the best possible way. Any hyperplane can be defined as a set of points $\boldsymbol{x}$ which satisfies a certain linear relation:

$$\boldsymbol{x}_i' \boldsymbol{w} = c,$$

where $\boldsymbol{w}$ is a normal vector to the hyperplane. If the data set is linearly separable, we can choose two parallel hyperplanes that split the data such that the distance between them is maximized. Mathematically they can be described as:

$$c + \boldsymbol{x}_i' \boldsymbol{w} = 1$$

and

$$c + \boldsymbol{x}_i' \boldsymbol{w} = -1.$$

The distance between these hyperplanes (margin lines) is equal to $\dfrac{2}{\|\boldsymbol{w}\|}$ (Figure 2). The hyperplane we are interested in lies in the middle between them and is called the maximum-margin hyperplane (Boser et al., 1992).

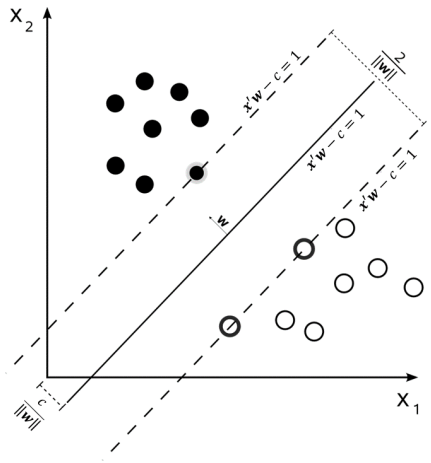


Figure 2: Maximum margin hyperplane and the margins

To extend SVMs to the general case, where the groups are not linearly separable, we introduce a so called absolute hinge error function. Here observations $\boldsymbol{x}_i$ contribute to the error in the following way: if the i^{th} object belongs to Class_1 and the prediction q_i is such that $q_i \geq 1$ then the error for this prediction is zero. On the other hand, if $q_i < 1$ (wrong side of the margin) then the error is linearly accounted for, yielding a value of $1 - q_i$. In the case where the observation falls in Class_{-1} and $q_i \leq -1$, the object is labelled correctly and the associated error is zero. However, when $q_i > 1$ (wrong side of the margin) then the value of the error is also linearly accounted for as $q_i + 1$. Thus, objects that are predicted correctly do not contribute to the error while objects that are incorrectly predicted contribute to the error linearly. A graph of this error function, for a single observation is provided in Figure 3.

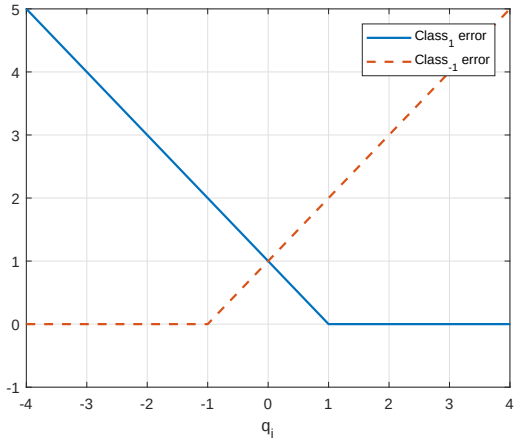


Figure 3: Absolute Hinge Error function for Class_1 objects and Class_{-1} objects.

The distance between the margin lines is inversely proportional to $\|\boldsymbol{w}\|$. The smaller this distance is, the more $\boldsymbol{x_i}$'s will be assigned to the correct class. Therefore it might be beneficial to choose very large $\|\boldsymbol{w}\|$. But at the same time, as mentioned before, we want to maximize the margin. The following function must

then be minimized (Groenen et al., 2008):

$$\begin{aligned} L_{\text{SVM}}(c, \boldsymbol{w}) &= \sum_{i \in Class_1} \max(0, 1 - q_i) + \sum_{i \in Class_{-1}} \max(0, q_i + 1) + \lambda \boldsymbol{w}'\boldsymbol{w}, \\ &= \text{Class}_1 \text{ errors} + \text{Class}_{-1} \text{ errors} + \text{Penalty for nonzero } \boldsymbol{w} \\ &= \sum_{i=1}^{n} \max(0, 1 - y_i q_i) + \lambda \boldsymbol{w}'\boldsymbol{w}, \end{aligned} \tag{3}$$

where parameter λ is added to control the length of $\boldsymbol{w}$. The penalty term also helps to avoid overfitting.

3 Majorization

Groenen et al. (2008) suggest that the SVM problem can be solved with the iterative majorization (IM). Its clear advantage is that in each iteration of the algorithm the SVM loss function value decreases until the convergence criterion is met. As the SVM loss function with the absolute or quadratic hinge error is convex and coercive, iterative majorization converges close to the global minimum (Groenen et al., 2008).

The principle behind the iterative majorization is relatively simple. Let $f(\boldsymbol{q})$ be the function to be minimized. Let $g(\boldsymbol{q}, \overline{\boldsymbol{q}})$ be the auxiliary function called majorizing function, that depends on $\boldsymbol{q}$ and the previous known estimate $\overline{\boldsymbol{q}}$ – a supporting point. The majorizing function has to satisfy the following requirements (Groenen et al., 2008):

1. it should touch f at the supporting point: $f(\overline{\boldsymbol{q}}) = g(\overline{\boldsymbol{q}}, \overline{\boldsymbol{q}})$,
2. it must never be below f: $f(\boldsymbol{q}) \leq g(\boldsymbol{q}, \overline{\boldsymbol{q}})$,
3. $g(\boldsymbol{q}, \overline{\boldsymbol{q}})$ should be simple, preferably linear or quadratic.

Let $\boldsymbol{q}^*$ be such that $g(\boldsymbol{q}^*, \overline{\boldsymbol{q}}) \leq g(\overline{\boldsymbol{q}}, \overline{\boldsymbol{q}})$ by choosing $\boldsymbol{q}^* = \text{argmin}_{\boldsymbol{q}} g(\boldsymbol{q}, \boldsymbol{q}^*)$. As the majorizing function is never below the original function, the so called sandwich inequality is obtained (De Leeuw, 1994):

$$f(\boldsymbol{q}^*) \leq g(\boldsymbol{q}^*, \overline{\boldsymbol{q}}) \leq g(\overline{\boldsymbol{q}}, \overline{\boldsymbol{q}}) = f(\overline{\boldsymbol{q}}). \tag{4}$$

It follows that the update $\boldsymbol{q}^*$ obtained by minimizing the majorizing function must also decrease the value of the original function f. This is how one full iteration of the majorization algorithm is performed. Repeating the process produces a monotonically non-increasing series of loss function values. For convex and coercive f the algorithm reaches the global minimum after sufficiently many iterations (Groenen et al., 2008).

Iterative majorization also has a useful property which allows to apply the algorithm to optimizing the $L_{\text{SVM}}(c, \boldsymbol{w})$ – an additivity rule. Suppose we have two functions $f_1(\boldsymbol{q})$ and $f_2(\boldsymbol{q})$ and both can be majorized with $g_1(\boldsymbol{q}, \overline{\boldsymbol{q}})$ and $g_2(\boldsymbol{q}, \overline{\boldsymbol{q}})$ respectively. Then the following majorizing inequality holds (Groenen et al., 2008):

$$f(q) = f_1(\boldsymbol{q}) + f_2(\boldsymbol{q}) \leq g_1(\boldsymbol{q}, \overline{\boldsymbol{q}}) + g_2(\boldsymbol{q}, \overline{\boldsymbol{q}}) = g(\boldsymbol{q}, \overline{\boldsymbol{q}}).$$

To apply the IM to our problem we need to find a majorizing function for (3). Assume the majorizing function exists for each individual error term of the form

$$f_{-1}(q_i) \leq a_{-1i} q_i^2 - 2b_{-1i} q_i + c_{-1i} = g_{-1}(q_i), \tag{5}$$

$$f_1(q_i) \leq a_{1i} q_i^2 - 2b_{1i} q_i + c_{1i} = g_1(q_i), \tag{6}$$

where a_i, b_i and c_i are specific and known for certain hinge loss function (A.1). Let

$$a_i = \begin{cases} \max(\delta, a_{-1i}), & \text{if } i \in G_{-1}, \\ \max(\delta, a_{1i}), & \text{if } i \in G_1, \end{cases} \tag{7}$$

$$b_i = \begin{cases} b_{-1i}, & \text{if } i \in G_{-1}, \\ b_{1i}, & \text{if } i \in G_1, \end{cases} \tag{8}$$

$$c_i = \begin{cases} c_{-1i}, & \text{if } i \in G_{-1}, \\ c_{1i}, & \text{if } i \in G_1, \end{cases} \tag{9}$$

where δ replaces a_{-i1} or a_{i1} for the respective class when $\overline{q} = -1$ or $\overline{q} = 1$. Summing all the terms leads to the total majorizing inequality, quadratic in c and $\boldsymbol{w}$:

$$L_{\text{SVM}}(c, \boldsymbol{w}) \leq \sum_{i=1}^{n} a_i q_i^2 - 2\sum_{i=1}^{n} b_i q_i + \sum_{i=1}^{n} c_i + \lambda \sum_{j=1}^{m} w_j^2. \tag{10}$$

If we add a column of ones as the first column of the $\boldsymbol{X}$ matrix and let $\boldsymbol{v}' = \begin{bmatrix} c & \boldsymbol{w}' \end{bmatrix}$, $q_i = c + \boldsymbol{x}'\boldsymbol{w}$ can be expressed as $\boldsymbol{q} = \boldsymbol{X}\boldsymbol{v}$. Thus (10) can be rewritten as

$$\begin{aligned} L_{\text{SVM}}(\boldsymbol{v}) &\leq \sum_{i=1}^{n} a_i (\boldsymbol{x}_i'\boldsymbol{v})^2 - 2\sum_{i=1}^{n} b_i \boldsymbol{x}_i'\boldsymbol{v} + \sum_{i=1}^{n} c_i + \lambda \sum_{j=2}^{m+1} v_j^2 \\ &= \boldsymbol{v}'\boldsymbol{X}'\boldsymbol{A}\boldsymbol{X}\boldsymbol{v} - 2\boldsymbol{v}'\boldsymbol{X}'\boldsymbol{b} + c_m + \lambda \boldsymbol{v}'\boldsymbol{P}\boldsymbol{v} \\ &= \boldsymbol{v}'(\boldsymbol{X}'\boldsymbol{A}\boldsymbol{X} + \lambda \boldsymbol{P})\boldsymbol{v} - 2\boldsymbol{v}'\boldsymbol{X}'\boldsymbol{b} + c_m, \end{aligned} \tag{11}$$

where $\boldsymbol{A}$ is a diagonal matrix with elements a_i on the diagonal, $\boldsymbol{b}$ is a vector with elements b_i and $\boldsymbol{P}$ is an identity matrix except for the element p_{11} that is equal to zero. Differentiating (11) with respect to $\boldsymbol{v}$ yields

$$(\boldsymbol{X}'\boldsymbol{A}\boldsymbol{X} + \lambda \boldsymbol{P})\boldsymbol{v} = \boldsymbol{X}'\boldsymbol{b}. \tag{12}$$

Solving the set of linear equations will result in an update $\boldsymbol{v}^+$:

$$\boldsymbol{v}^+ = (\boldsymbol{X}'\boldsymbol{A}\boldsymbol{X} + \lambda \boldsymbol{P})^{-1}\boldsymbol{X}'\boldsymbol{b}. \tag{13}$$

The pseudocode of SVM-Maj for the absolute hinge errors is as follows (Groenen et al., 2008):

Input: $\boldsymbol{y}$, $\boldsymbol{X}$, λ, ϵ
Output: c_t, $\boldsymbol{w}_t$
$t = 0$;
Set ϵ to a small positive value;
Set $\boldsymbol{w}_0$ and c_0 to random initial value;
Compute $L_{\text{SVM}}(c_0, \boldsymbol{w}_0)$;
while $t = 0$ *or* $(L_{t-1} - L_{SVM}(c_t, \boldsymbol{w}_t))/L_{t-1} > \epsilon$ **do**
    t = t + 1;
    $L_{t-1} = L_{\text{SVM}}(c_{t-1}, \boldsymbol{w}_{t-1})$;
    $a_i = \max(\delta, \frac{1}{4}|y_i q_i + 1|^{-1})$ and $b_i = y_i a_i - \frac{1}{4}$;
    Make diagonal matrix $\boldsymbol{A}$ with elements a_i;
    Find $\boldsymbol{v}$ that solves
$$(\boldsymbol{X}'\boldsymbol{A}\boldsymbol{X} + \lambda \boldsymbol{P})\boldsymbol{v} = \boldsymbol{X}'\boldsymbol{b}$$
    Set $c_t = v_1$ and $w_{tj} = v_{j+1}$ for $j = 1, ..., m$;
end

Algorithm 1: The majorization algorithm for the absolute hinge error.

4 Change point method

In this report we optimize the primal loss function with the absolute hinge error using a newly developed exact linesearch method – Change Point Method (CPM). Recall that this loss function can be formulated as follows:

$$L_{\text{SVM}}(c, \boldsymbol{w}) = \sum_{i=1}^{n} \max(0, 1 - y_i q_i) + \lambda \boldsymbol{w}'\boldsymbol{w}.$$

Assume for simplicity, that

$$q_i = \boldsymbol{x}_i'\boldsymbol{w}, \tag{14}$$

where q_i is the prediction and $\boldsymbol{x}_i$ is the vector of predictor variables for the i^{th} observation; no intercept. Prediction q_i varies as $\boldsymbol{w}$ does. Assume also we have a search direction $\boldsymbol{s}$. The change of $\boldsymbol{w}$ in the direction $\boldsymbol{s}$ by the distance h leads to

$$q_i = \boldsymbol{x}_i'(\boldsymbol{w} + h\boldsymbol{s}). \tag{15}$$

As we move along $\boldsymbol{s}$, q_i increases in one direction and decreases in the opposite one. For a certain value of h, q_i is exactly equal to y_i. We call this place a change point – p_i. For the values of h below (above) p_i, prediction q_i is correct. It changes to incorrect for h above (below) the change point. Incorrectly predicted observations linearly contribute to the overall error of prediction: with each incorrect prediction q_i, the gradient of the loss function changes by

$$\frac{\partial}{\partial h}(1 - y_i q_i) = -y_i \boldsymbol{x}_i'\boldsymbol{s}. \tag{16}$$

It is important to note that the change from the correct to the incorrect prediction can happen as h steps over the change point from left to right as well as from right to left. The direction of change depends on the combination of signs of y_i and $\boldsymbol{x}_i'\boldsymbol{s}$. Namely, if $y_i > 0$ & $\boldsymbol{x}_i'\boldsymbol{s} > 0$ or $y_i < 0$ & $\boldsymbol{x}_i'\boldsymbol{s} < 0$ then it follows from (15) and the definition of the absolute hinge error that the prediction q_i is correct on the right of corresponding the change point. On the other hand, if $y_i > 0$ & $\boldsymbol{x}_i'\boldsymbol{s} < 0$ or $y_i < 0$ & $\boldsymbol{x}_i'\boldsymbol{s} > 0$ then q_i is correct to the left of p_i.

The initial step of the CPM is the derivation of the change points for each observation i. The following expressions must be solved for h $\forall i$:

$$\begin{aligned} q_i &= y_i \\ \boldsymbol{x}_i'(\boldsymbol{w} + h\boldsymbol{s}) &= y_i \\ h = p_i &= \frac{y_i - \boldsymbol{x}_i'\boldsymbol{w}}{\boldsymbol{x}_i'\boldsymbol{s}}. \end{aligned} \tag{17}$$

The n calculated change points are sorted in ascending order (from now on $p_i \leq p_{i+1} \leq ... \leq p_{n-1} \leq p_n$) and, splitting the number line, provide us with the $n + 1$ intervals of interest.

The SVM loss function is convex and coercive. Then, according to Fermats theorem (interior extremum theorem), the optimal h is found at the point where the gradient of this function is equal to zero. As the loss function is a sum of individual elements, we can calculate its gradient as the sum of the gradients of its elements:

$$\frac{\partial L_{\text{SVM}}}{\partial h} = \sum_{i}^{n} \{0, -y_i \boldsymbol{x}_i'\boldsymbol{s}\} + 2\lambda(h\|\boldsymbol{s}\|^2 + \boldsymbol{w}'\boldsymbol{s}). \tag{18}$$

This is done for each interval made by the change points. As it was mentioned before, depending on the combination of the signs of y_i and $\boldsymbol{x}_i'\boldsymbol{s}$, the observation is predicted incorrectly for the values of h either to the left or to the right of the change point. Grouping the observations based on this criterion, we can sum each group up cumulatively (from right to left or from left to right depending on the group) to efficiently get the $\sum_{i=1}^{n} \max(0, 1 - y_i q_i)$ for each interval.

At each change point p_i L_{SVM} is non-differentiable, but it is easy to calculate the lower and upper bounds of the subdifferential there. As p_i is both the ending point of the interval i and the starting point of the interval $i+1$, the gradient of the loss function is calculated for each point twice, resulting in two vectors: $\boldsymbol{E}$, containing the lower bounds of subdifferentials at each changepoint, and $\boldsymbol{S}$, containing the upper bounds.

Using these two vectors we can calculate the optimal value of h, considering all its possible positions with respect to the change points (Figure 4).

Figure 4a represents the scenario where the optimum lies between the two consecutive change points. In this case, the signs of the gradients at the start and the end of one of the intervals must be different:

$$sign(\boldsymbol{S}_i) \neq sign(\boldsymbol{E}_i).$$

The optimal h can then be found by a linear interpolation:

$$h = -(p_{i+1} - p_i)\frac{\boldsymbol{S}_i}{\boldsymbol{E}_{i+1} - \boldsymbol{S}_i} + p_i.$$

Figure 4b pictures the situation where none of the intervals crosses zero. This is true if the signs of the gradients at the end of interval i and at the start of the interval $i+1$ are different:

$$sign(\boldsymbol{E}_i) \neq sign(\boldsymbol{S}_{i+1}).$$

The value of the change point i is the optimal h.

Figures 4c and 4d are special cases of the first considered scenario. The gradient of the loss function is equal to zero exactly at one of the change points. The optimal h is equal to the value of the change point at which the element of either $\boldsymbol{S}$ or $\boldsymbol{E}$ is zero.

Figure 4e illustrates the case where $\boldsymbol{E}_1$ and $\boldsymbol{S}_n$ are both negative. The optimal h can be expressed as:

$$h = p_n + \left|\frac{\boldsymbol{S}_n}{2\lambda\|\boldsymbol{s}\|^2}\right|.$$

An opposite scenario is shown in Figure 4f: $\boldsymbol{E}_1$ and $\boldsymbol{S}_n$ are both positive. The optimal h can be expressed as:

$$h = p_n - \left|\frac{\boldsymbol{E}_1}{2\lambda\|\boldsymbol{s}\|^2}\right|.$$

After the optimal h is found, the new vector of parameters $\boldsymbol{w}^+$ is calculated as $\boldsymbol{w}^+ = \boldsymbol{w} + h\boldsymbol{s}$. The intercept can be easily incorporated in the algorithm by fixing all the elements of $\boldsymbol{w}$ and optimizing the loss function without the penalty term, as the magnitude of the intercept should not be penalized.

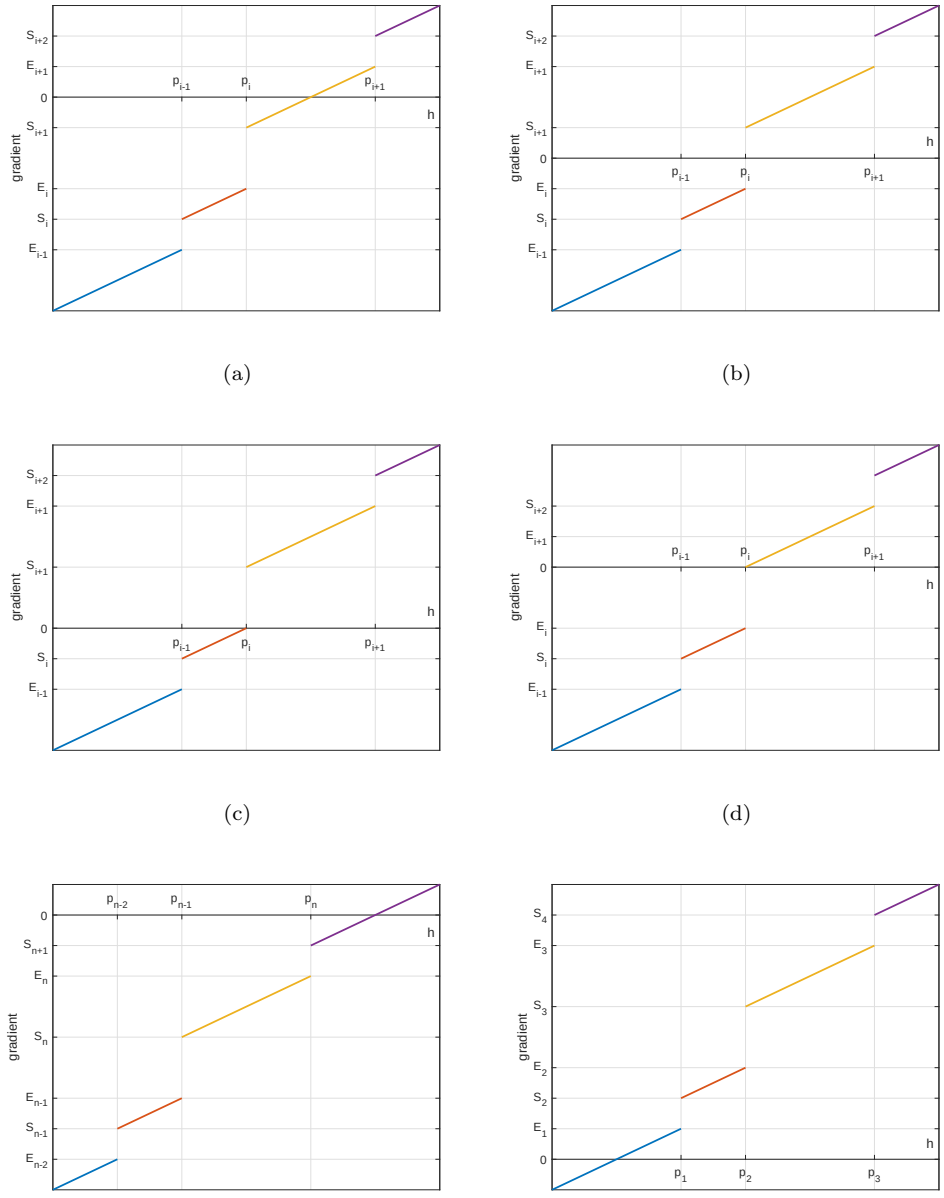


Figure 4: The six possible positions of the optimum with respect to the changepoints

5 Experiments

5.1 Methodology

The CPM was developed as an auxiliary step on the way to creating a new efficient approach to solving linear SVM problems. To test the CPM, we used an SVM-Maj algorithm as a benchmark for three different implemented approaches:

- coordinate descent,
- gradient descent (steepest descent),
- αMajorization.

Coordinate descent is a derivative-free algorithm. The idea behind it is, instead of calculating a search direction, to make use of the axes and search for the optimum cyclically along them. The pseudocode for the CD approach we used is as follows:

Input: $\boldsymbol{y}$, $\boldsymbol{X}$, λ, ϵ
Output: c_t, $\boldsymbol{v}$
$t = 0$;
Set ϵ to a small positive value;
Set $\boldsymbol{v}$ to random initial value;
Compute L_{SVM};
while $t = 0$ *or* $(L_{t-1} - L_{SVM}(\boldsymbol{v}))/L_{t-1} > \epsilon$ **do**
    t = t + 1;
    $L_{t-1} = L_{\text{SVM}}(\boldsymbol{v}_t)$;
    for $j = 1 : m + 1$ **do**
        Fix all the elements of $\boldsymbol{v}$ except of v_j and find an optimal value of v_j along this direction using the CPM. Set v_j to this optimum. Do NOT include the penalty in the loss function for v_1 as the magnitude of the constant should not be penalized.
    end
    Compute L_{SVM};
end
$c = \boldsymbol{v}(1)$;
$\boldsymbol{w} = \boldsymbol{v}(2 : \text{end})$;

Algorithm 2: The pseudocode for the coordinate descent for the absolute hinge error.

Gradient or steepest descent is a first-order optimization method. The idea is to search for the optimum iteratively in the direction of the negative of the gradient. Therefore to solve our SVM problem we simply apply the CPM with the search direction $\boldsymbol{s}_k = -\nabla L_{\text{SVM}}(\boldsymbol{w}_k)$ for each iteration k until the convergence criterion is met.

αMajorization was brought to life from the idea that the iterative majorization process could be accelerated by optimizing the step length in the majorization direction. The idea is based on the fact that, even though in each iteration k of MM algorithm the majorizing function is minimized, this doesn't imply that there is no better solution for minimizing the objective function in the direction $\boldsymbol{v}_{k+1} - \boldsymbol{v}_k$. This fact is used in the SVM-Maj algorithm by Groenen et al. (2008), where, from a certain iteration on, the step length is doubled which improves the convergence speed. Doubling the step is the simplest way to accelerate the MM algorithm and in practice it usually halves the number of interations until convergence (Wu et al., 2010). However this improvement is not guaranteed. In the αMajorization we use the direction provided by an iteration of the SVM-Maj algorithm and search for the minimum along it with the CPM. As a result we receive an exact argument of the minimum of the function in the given direction – $\boldsymbol{v}_k + \alpha(\boldsymbol{v}_{k+1} - \boldsymbol{v}_k)$ – where α is an optimal step length multiplier. Achieved point is then used in the next iteration as a new supporting point (Algorithm 3). This approach is expected to significantly decrease the number of iterations until convergence and therefore to increase the speed of convergence compared to the SVM-Maj algorithm.

Input: $\boldsymbol{y}$, $\boldsymbol{X}$, λ, ϵ
Output: c_t, $\boldsymbol{w}_t$
$t = 0$;
Set ϵ to a small positive value;
Set $\boldsymbol{w}_0$ and c_0 to random initial value;
Compute $L_{\text{SVM}}(c_0, \boldsymbol{w}_0)$;
while $t = 0$ *or* $(L_{t-1} - L_{SVM}(c_t, \boldsymbol{w}_t))/L_{t-1} > \epsilon$ **do**
    t = t + 1;
    $\boldsymbol{v}_{prev} = \boldsymbol{v}$;
    $L_{t-1} = L_{\text{SVM}}(c_{t-1}, \boldsymbol{w}_{t-1})$;
    $a_i = \frac{1}{4}|y_i q_i + 1|^{-1}$ and $b_i = y_i a_i - \frac{1}{4}$;
    Make diagonal matrix $\boldsymbol{A}$ with elements a_i;
    Find $\boldsymbol{v}$ that solves

$$(\boldsymbol{X}'\boldsymbol{A}\boldsymbol{X} + \lambda \boldsymbol{P})\boldsymbol{v} = \boldsymbol{X}'\boldsymbol{b}$$

    Use $\boldsymbol{s} = \boldsymbol{v}_{prev} - \boldsymbol{v}$ as a search direction;
    Optimize L_{SVM} in the direction $\boldsymbol{s}$ using the Change point method;
    Set $\boldsymbol{v} = \boldsymbol{v}_{prev} + \alpha \boldsymbol{s}$;
    Set $c_t = v_1$ and $w_{tj} = v_{j+1}$ for $j = 1, ..., m$;
end

Algorithm 3: The αMaj algorithm for the absolute hinge error.

We are interested in comparing the speed of convergence and the loss values attained by the algorithms. The comparison is carried out in Matlab 2017a, on a 2.6 Ghz Intel processor with 8 GB of memory under Windows 10. All the approaches are tested with the optimal λ's and p's for each dataset and a stopping criterion $(L_{t-1} - L_{\text{SVM}}(c_t, \boldsymbol{w}_t))/L_{t-1} < \epsilon = 10^{-7}$ taken from Groenen et al. (2008). Seven different data sets from UCI repository (Newman et al. 1998) and LibSVM sources (Chang & Lin 2006) are used. Table 1 shows the data sets, where n is the total number of observations, n_1 and n_{-1} are the number of observations for the Class$_1$ and Class$_{-1}$ respectively and m is the number of variables.

Data set	Source	n	n_1	n_{-1}	m	Sparsity
Australian	UCI	690	307	383	14	20.04
Breast.cancer.w	UCI	699	458	241	9	00.00
Diabetes	LibSVM	768	500	268	8	00.00
Heart.statlog	UCI	270	120	150	13	00.00
Hepatitis	UCI	155	123	32	19	39.86
Sonar	UCI	208	97	111	60	00.07
Voting	UCI	434	167	267	16	45.32

Table 1: Data sets

5.2 Results

Tables 2 and 3 below show the CPU times and the achieved loss values for the five approaches applied to the seven data sets. As the optimized function is convex and coercive all the algorithms should find the global minimum of the loss function. Ideally, this means they should give the same loss value under our pre-set conditions.

First important observation is that only SVM-Maj (standard and with the doubled step length) and αMajorization algorithms do consistently converge to the real global minimum. αMajorization in general provides as good as or better loss function values than SVM-Maj but the difference is less than 10^{-3}. Coordinate and gradient descents behave worse and in general do not converge, stopping away from the global optimum. Both are suffering from zigzagging closer to the minimum. Moreover gradient descent's performance strongly depends on the starting point: e.g. as seen from the Table 3, with the given starting

points, GD failed to converge for datasets 1, 4 and 5, however in general the results are always different and are not predictable.

The speed of convergence is much more interesting. Both coordinate and gradient descent show some great potential – based on our experiments they can be up to 20-25 times faster than SVM-Maj – but as convergence is not guaranteed, they cannot be applied safely to solving the problem. αMajorization, however, besides consistently reaching the smallest loss values, is also much faster than the standard iterative majorization and it's accelerated version with the double step length. The average improvement for the 7 datasets is 3.4 times over the SVM-Maj and 1.8 – over the doubled step majorization (which, as expected, is around twice faster than the basic version). This is already a feasible advantage, but moreover it is known (Groenen et al., 2008) that iterative majorization becomes slower for the large number of variables m and the number of objects n, as each its iteration becomes slower. During the testing αMajorization showed up to 8-9 times decrease in the number of iterations needed to converge to the global minimum. Indeed on the larger datasets the CPU time decrease is consistently higher than on average reaches up to 6.3 times.

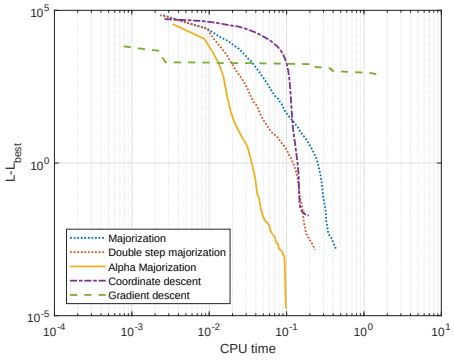
The graphical comparison of the performance of the algorithms is shown on the Figure 5. Here the evolution of $L - L_{best}$ is plotted against the CPU time, where L is the loss value after a certain CPU time used and L_{best} is the minimal loss value achieved by any of the five methods. The graph further confirms that the majorization approaches are better than the other two, and that αMajorization is a great improvement over the SVM-Maj algorithm. It is also interesting to see that sometimes CD or GD clearly show a very fast convergence, which again leads to the idea of their potential.

Data set	p	**CPU time**				
		Maj	2Maj	αMaj	CD	GD
Australian	-0.5	0.4417	0.2338	0.1003	0.1908	1.4037
Breast.cancer.w	7.5	0.2432	0.1285	0.0512	0.0130	0.0128
Diabetes	1	0.2274	0.0818	0.0481	0.0478	0.0434
Heart.statlog	0	0.0136	0.0102	0.0125	0.0213	0.0639
Hepatitis	0	0.0101	0.0076	0.0069	0.0426	0.0095
Sonar	0.5	0.0234	0.0233	0.0198	0.0274	0.0280
Voting	-5.5	0.5063	0.2217	0.0903	0.0635	0.0156

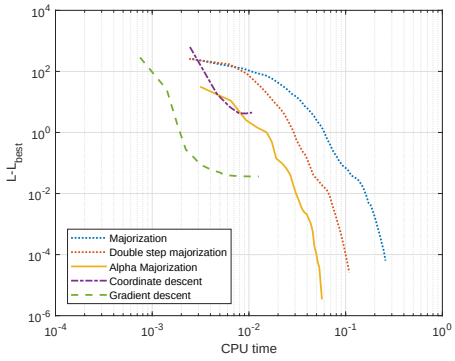
Table 2: CPU time comparison

Data set	p	**Loss value**				
		Maj	2Maj	αMaj	CD	GD
Australian	-0.5	199.3379	199.3379	199.3365	199.3556	889.8765
Breast.cancer.w	7.5	68.5778	68.5778	68.5777	73.0607	68.6139
Diabetes	1	396.5750	396.5758	396.5751	396.5952	397.7805
Heart.statlog	0	92.1256	92.1249	92.1240	92.2704	156.8364
Hepatitis	0	101.3649	101.3650	101.3651	102.1506	157.3018
Sonar	0.5	121.5664	121.5669	121.5664	134.8765	122.1448
Voting	-5.5	25.3687	25.3686	25.3684	26.0129	26.3085

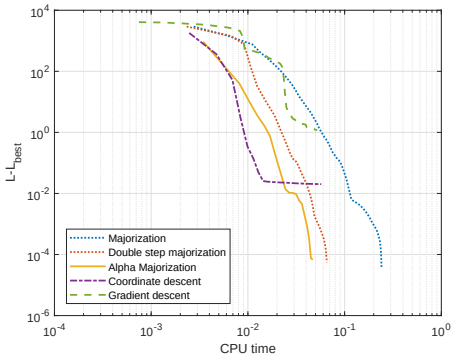
Table 3: Loss value comparison



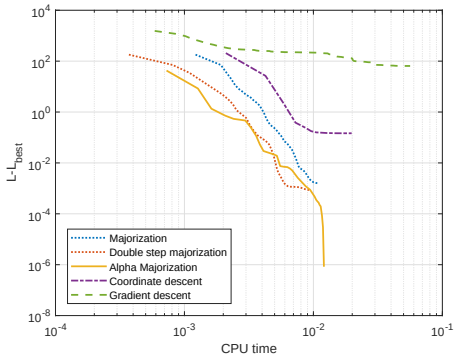
(a) Australian



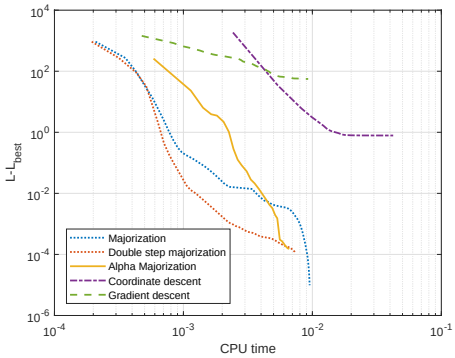
(b) Breast.cancer.w



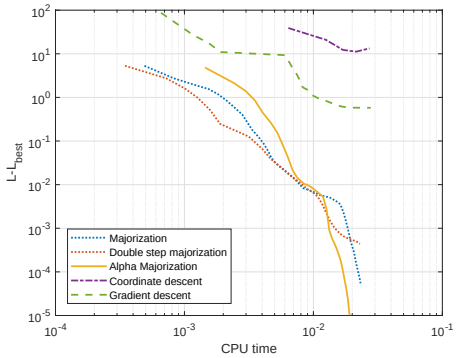
(c) Diabetes



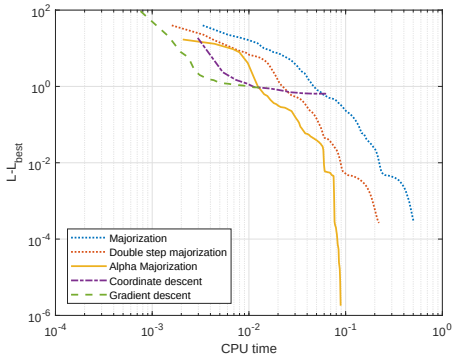
(d) Heart.statlog



(e) Hepatitis



(f) Sonar



(g) Voting

Figure 5: The evolution of $L - L_{best}$ for five methods for each dataset

6 Conclusion

In this paper, we developed an exact line search method which works in the framework of SVMs – the CPM. It was tested in combination with several different search directions in application to the primal linear SVM loss function with an absolute hinge error defined in the paper of Groenen et al. (2008).

The numerical experiments exhibited some great results. First of all, we found out that coordinate and gradient descent methods do not in general converge to the global minimum. The problems of the gradient descent methods were expected as its convergence for non-differentiable functions is ill-defined. Secondly, combining CPM with the SVM-Maj algorithm we created the αMajorization algorithm and were able to achieve major performance improvements, especially on larger datasets. This fact makes our algorithm comparable to the best ones out there.

The main limitation of our method is that it only works with the absolute hinge error. Nevertheless, there are no visible obstacles to adapt it to different hinges e.g. quadratic. Research in this direction is very interesting: implementing the quadratic hinge would presumably allow for the problem to be tackled with the steepest descent method, as the loss function becomes everywhere differentiable. In combination with the CPM we expect the minimization to converge fast and accurately.

Finally, it is important to note, that there clearly is a room for code optimizations. It might lead to improvements in computational time of the majorization step relative to the time used by the CPM search or vice versa. For example, one could find an efficient solution to the set of linear equations which is required in each iteration of the majorization algorithm or use a sophisticated sorting algorithm to speed up the CPM. However, the number of iterations decreases significantly enough to state that αMajorization will generally outperform the SVM-Maj algorithm.

A Appendix

A.1 Majorization with absolute hinge error

Here we show the quadratic majorizing function for the absolute hinge error derived in Groenen et al. (2008). The majorizing function for Class_{-1} is as follows:

$$g_{-1}(q) = a_{-1}q^2 - 2b_{-1}q + c_{-1}.$$

The formal requirements for it are:

$$\begin{aligned} f_{-1}(q,\overline{q}) &= g_{-1}(q,\overline{q}),\\ f'_{-1}(q,\overline{q}) &= g'_{-1}(q,\overline{q}),\\ f_{-1}(-2-\overline{q}) &= g_{-1}(-2-\overline{q}),\\ f'_{-1}(-2-\overline{q}) &= g'_{-1}(-2-\overline{q}),\\ f_{-1}(q) &\leq g_{-1}(q). \end{aligned}$$

It can be verified that the choice of

$$a_{-1} = \frac{1}{4}|\overline{q}+1|^{-1}, \tag{19}$$

$$b_{-1} = -a_{-1} - \frac{1}{4}, \tag{20}$$

$$c_{-1} = a_{-1} + \frac{1}{2} + \frac{1}{4}|\overline{q}+1|, \tag{21}$$

satisfies these requirements.
The process is similar for Class_1. The formal requirements for the majorizing function

$$g_1(q) = a_1q^2 - 2b_1q + c_1.$$

are now:

$$\begin{aligned} f_1(q,\overline{q}) &= g_1(q,\overline{q}),\\ f'_1(q,\overline{q}) &= g'_1(q,\overline{q}),\\ f_1(2-\overline{q}) &= g_1(2-\overline{q}),\\ f'_1(2-\overline{q}) &= g'_1(2-\overline{q}),\\ f_1(q) &\leq g_1(q). \end{aligned}$$

The appropriate choice for parameters is:

$$a_1 = \frac{1}{4}|\overline{q}+1|^{-1}, \tag{22}$$

$$b_1 = a_1 + \frac{1}{4}, \tag{23}$$

$$c_1 = a_1 + \frac{1}{2} + \frac{1}{4}|\overline{q}+1|, \tag{24}$$

References

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on computational learning theory* (pp. 144–152).

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of compstat'2010* (pp. 177–186). Springer.

Chang, C.-C., & Lin, C.-J. (2006). Libsvm: a library for support vector machines, 2001. software available at http.

Collins, M., Globerson, A., Koo, T., Carreras, X., & Bartlett, P. L. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, *9*(Aug), 1775–1822.

De Leeuw, J. (1994). Block-relaxation algorithms in statistics. In *Information systems and data analysis* (pp. 308–324). Springer.

Groenen, P., Nalbantov, G., & Bioch, C. (2008, April). Svm-maj: A majorization approach to linear support vector machines with different hinge errors. *Advances in Data Analysis and Classification*, *2*(1), 17–43. doi: 10.1007/s11634-008-0020-9

Newman, D. J., Hettich, S., Blake, C. L., & Merz, C. J. (1998). {UCI} repository of machine learning databases.

Vapnik, V. (1999). *The nature of statistical learning theory (information science and statistics)*. Springer.

Wu, T. T., Lange, K., et al. (2010). The mm alternative to em. *Statistical Science*, *25*(4), 492–505.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on machine learning* (p. 116).