Technical University Of Crete

School of Electrical and Computer Engineering

Diploma Thesis

Streaming, High Performance Support Vector Methods

Author:
Periklis Chrysogelos

Supervisor:
Prof. Minos Garofalakis

A thesis submitted in fulfillment of the requirements for the degree of Diploma in Electrical and Computer Engineering

26 August 2016

Abstract

We are in an era where data are constantly being generated and machine learning can benefit from this to produce better models. Support vector machines are a popular machine learning model, which can be adapted and used for various tasks, such as classification, regression and clustering. We study the problem of continuously updating L_2 Support Vector Machines in a distributed environment where new data constantly arrive in remote sites. We approach this as the problem of tracking a convex function's minimum over the convex hull of the union of fully dynamic sets, each one located in one of the sites. We give communication efficient solutions for both the exact and approximate variants of the problem and show that they are applicable in the case of a kernelized SVM trained in an implicit feature space. In our proposed methods, the sites communicate only when it is necessary, that is, every time the model has been truly outdated. Also, in case the sites are forced to communicate, we propose two algorithms, one iterative and one with a single stage of communication.

Contents

Chapter 1

Introduction

Huge amounts of data are constantly generated, from financial series to how we interact with the appliances surrounding us. Their generation speed and size makes processing them challenging. In addition, very ofter they are generated in distributed remote sites, which makes them even harder to process. Thus, it is of big interest to adapt machine learning's methods to handle the rapid nature and huge amount of generated data, in distributed settings.

Support vector machines are a machine learning algorithm which has shown pretty good performance and has a sound theoretical foundation. Thus, in this work, we will study the problem of distributively and continuously training an SMV model. We will focus on the L_2-SVM which, under some conditions, is equivalent to a convex minimization over the convex hull of the data points in the feature space.

1.1 Outline

In the rest of this chapter we will focus on describing related work and describing the computational model we will assume.

In chapter 2 we will study the more general problem of minimizing a convex function over the convex hull of a distributed fully dynamic set of points, which is altered based on streams of events arriving in remote sites. This methods will be expanded in chapter 3 to take advantage of the case of only requiring an approximation to the minimum.

Chapter 4 discusses how this results apply in training L_2 Support Vector Machines, even for feature spaces we prefer not, or can not, explicitly address but we can describe through a kernel function.

Finally, in chapters 5 and 6 we present some experimental results of our work followed by a discussion.

1.2 Related Work

Support vector machines[1] have shown a lot of promising results, but they are characterized by a high training complexity. Many methods have been used to train SVMs locally, such as Sequential Minimal Optimization[8] and some geometric methods. A geometric approach was given by Tsang et al in [13] to train L_2 SVMs in large datasets by showing that L_2-SVMs are equivalent to a minimum enclosing ball problem in the feature space. Mavrofakis and Theodoridis[7] trained a two class SVM using an iterative algorithm which finds the two points which are closer and each one belongs in the reduced convex hull of one class. Cauwenberghs and Poggio[9] give an iterative algorithm for incremental and decremental training of SVMs in a local setting. Zhu et al. introduced PSVM[14], an algorithm for training SVMs in a parallel system and Rai et al. build a one pass SVM in[10].

Processing high speed streams has gained a lot of focus in the last years. Sharfman[12] introduced the Geometric Method to monitor a general threshold function over a convex combination of the distributed vectors. In [5], Lazerson et al. generalize the Geometric Method by using convex decompositions of the inadmissible area. Garofalakis et al.[3] generalized monitoring a threshold function into approximately tracking a function over a convex combination of the distributed vectors and also they extend the above methods to the case of only keeping a sketch of the streams.

1.3 Computational Model

We consider a distributed computing environment comprised of K remote sites, where streams of events arrive and each event is either an addition or a deletion of a point. Each site is responsible for maintaining its set of points and it is assumed that any deletion will arrive in the same site as the corresponding addition, which can be easily handled by a previous stage. We are interested in continuously keeping up-to-date the solution of minimizing a given convex function f over the convex hull of the union of this sets.

Except from such events, queries may also reach any site, which in chapters 2 and 3 may be a query of the minimum value or a point achieving it, while in chapter 4 it may be a query to the Support Vector Machine.

Our main focus will be in developing communication efficient methods without excessive computational cost.

Chapter 2

Minimization

As the first part of our work we will study the problem where each site maintains a fully dynamic set of points and we are interested in tracking the minimum of a convex function f over the convex hull of their union as they evolve.

2.1 Problem Description

In this chapter we consider a distributed system where stream of updates arrive in remote sites. Each update may either be an addition or a deletion of a vector $\mathbf{x_i}$. Given a differentiable function f, we are interested in distributively computing and monitoring its minimum over every convex combination of the data points.

More formally, given a function $f : \mathbb{R}^n \to \mathbb{R}$ we define the problem,

$$\begin{aligned} \underset{\mathbf{a}\in\mathbb{R}^N}{minimize} \quad & f\left(X\mathbf{a}\right) \\ s.t. \quad & \mathbf{0} \le \mathbf{a} \\ & \mathbf{1}^T\mathbf{a} = 1 \end{aligned} \tag{2.1}$$

where X is the matrix whose columns are the data vectors $\mathbf{x_i}$. We are interested in computing the problem's solution and a point achieving it, as well as tracking their values after addition and deletion of columns of X.

As $X\mathbf{a} = \sum_i a_i\mathbf{x_i}$, it is easy to show that problem (2.1) is equivalent to,

$$\underset{\mathbf{e}\in\mathcal{H}}{minimize} \quad f(\mathbf{e}) \tag{2.2}$$

where $\mathcal{H} = ConvexHull(\{\mathbf{x_i}, \forall i\}) = \{\sum_i a_i\mathbf{x_i} | \sum_i a_i = 1, 0 \le a_i \le 1, \forall i\}$.

2.2 Overview

The problem consists of two subproblems, monitoring the correctness of the current solution and updating it when necessary. Initially, in section 2.3 we will

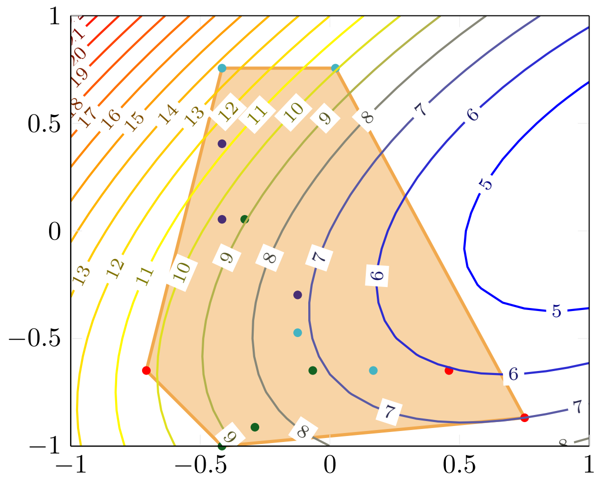


Figure 2.1: Circles are the data points and points of the same color are assigned to the same site. Contours of f are also shown. The convex polygon shown in orange is the convex hull of the points.

set up the common knowledge our remote sites have and define some notation. In sections 2.4, 2.5 we will describe our proposals for solving the two problems.

2.3 Global Knowledge and Notation

In our approach every site has knowledge of the same vector $\hat{\mathbf{e}}$ which yields the minimum of f over the convex hull of vectors $\mathbf{x_i}$. Also, each site stores its set $\mathcal{D}_k$ of vectors $\mathbf{x_i}$ that arrived in its stream and are not deleted. Let $\mathcal{D}$ be representing the union of this sets and $\mathcal{M}_i$ be the points known to site i. Of course $\mathcal{D}_i \subseteq \mathcal{M}_i$, for each site i.

We will call $\hat{\mathbf{e}}$ the global vector. We observe that $\hat{\mathbf{e}} = \sum_i a_i \mathbf{x_i}$. So, in high level, it may be represented in one of two ways, either as a vector or as the convex combination of a subset of $\mathcal{D}$. Having this in mind, for the rest of the document if not explicitly written, when referring to a set containing $\hat{\mathbf{e}}$ it will imply that it either contains $\hat{\mathbf{e}}$ itself, or the vectors with non zero coefficients in the above convex combination, whatever is more appropriate based on $\hat{\mathbf{e}}$'s representation. For example, if $\hat{\mathbf{e}}$ is represented as a convex combination, $\mathcal{M}_i$s will contain every $\mathbf{x_i}$ needed for creating $\hat{\mathbf{e}}$. So, in each case, $\hat{\mathbf{e}} \in \mathit{ConvexHull}\left(\mathcal{M}_i\right)$.

2.4 Monitoring

During the monitoring phase we are interested in distributively detecting when our current global vector $\hat{\mathbf{e}}$ stops being an optimal point over the convex hull. That can happen in two cases, either when the hull is expanded and at least one new point which obtains a lower value for the function under consideration is introduced into it, or when the global vector gets outside the convex hull, due to a change in the hull.

The first case only occurs when a point is added. Addition of a point may introduce new points in the convex hull, whereas deletion will produce a hull that is subset of the previous one. So, in this case, the minimum of f over the new convex hull will be greater or equal to the previous one.

On the other hand, our current global vector $\hat{\mathbf{e}}$ may get outside of the hull only in case of a point being deleted. When a new point is added to the hull, the hull will either be expanded or stay unchanged, but will in no case be reduced. So the new hull will be a superset of the previous one, thus $\hat{\mathbf{e}}$ will remain inside the hull.

2.4.1 Detecting Non Optimality

We are now going to develop local constraints to detect when a new optimal vector may have being introduced due to an event on one of the sites. As it has already been mentioned, such an event may only be an addition of a new point to the set of vectors.

More formaly, the task we are interested in this section is to monitor whether a point $\mathbf{e}$ exists in the updated convex hull such that $f(\mathbf{e}) < f(\hat{\mathbf{e}})$. That is, we are interested in creating local constraints that, if none is violated, will provide a certificate that the global constraint $f(\mathbf{e}) < f(\hat{\mathbf{e}})$ is also not violated for any $\mathbf{e} \in \mathcal{H}$.

Lazerson et al.[5] have studied the problem of distributively monitoring such a constraint. They have shown that given a set of point, if the inadmissible area produced by the constraint is convex, then there is a halfspace we can create such that if every point of the set belongs to it, then there is no point in the set's convex hull which violates the global constraint. Given a global vector $\hat{\mathbf{e}}$ belonging to the convex hull and its projection $\hat{\mathbf{e}}^*$ to the boundary of the inadmissible area, the halfspace is defined as the halfspace containing $\hat{\mathbf{e}}$ whose boundary is the hyperplane containing $\hat{\mathbf{e}}^*$ and normal the line segment connecting $\hat{\mathbf{e}}^*$ to $\hat{\mathbf{e}}$.

Building on that idea, we are going to create local constraints for our problem. The inadmissible region in our case is defined as $\bar{\mathcal{A}} = \{\mathbf{z} \in \mathbb{R}^n | f(\mathbf{z}) < f(\hat{\mathbf{e}})\}$, which is convex as it is a $f(\hat{\mathbf{e}})$-sublevel set of the convex function f. We have defined our global vector as $\hat{\mathbf{e}}$, but it already belongs to the boundary of $\bar{\mathcal{A}}$, so its projection is itself. The normal is now defined by $\nabla f(\hat{\mathbf{e}})$, as the distance of $\hat{\mathbf{e}}$ to its projection is 0 and thus the local constraints are $\nabla f(\hat{\mathbf{e}})^T (\mathbf{x_i} - \hat{\mathbf{e}}) \geq 0$ as proven in theorem 1.

Theorem 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex, differentiable function, set $\mathcal{D} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\} \subseteq \mathbb{R}^n$ and $\hat{\mathbf{e}} \in \mathcal{H} \equiv ConvexHull(\mathcal{D})$. Then, $\nabla f(\hat{\mathbf{e}})^T (\mathbf{x} - \hat{\mathbf{e}}) \geq 0, \forall \mathbf{x} \in \mathcal{D} \implies f(\hat{\mathbf{e}}) \leq f(\mathbf{e}), \forall \mathbf{e} \in \mathcal{H}$.*

Proof. For any $\mathbf{e} \in \mathcal{H}$ there exist $a_i \in [0,1], \forall i$ such that $\mathbf{e} = \sum_i a_i \mathbf{x_i}$, and $\sum_i a_i = 1$. Thus, based on theorem's preconditions, $\nabla f(\hat{\mathbf{e}})^T (a_i \mathbf{x_i} - a_i \hat{\mathbf{e}}) \geq$

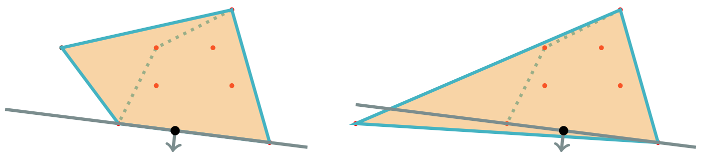

(a) No violation (b) Violation

Figure 2.2: A new point is added to the old convex hull (dotted line). The old global vector is the black dot and the gray line is the hyperplane defined by the local constraint (the arrow is proportional to the negative of the gradient where the global vector is). In the left picture the local constraint for the new point is not violated, while in the right one, it is.

0, $\forall i = 1, \dots, N$. Summing all the inequalities, we get $\nabla f(\hat{\mathbf{e}})^T (\mathbf{e} - \hat{\mathbf{e}}) \geq 0$. And due to convexity, $f(\mathbf{e}) - f(\hat{\mathbf{e}}) \geq \nabla f(\hat{\mathbf{e}})^T (\mathbf{e} - \hat{\mathbf{e}}) \geq 0$. Concluding that, $\left(\nabla f(\hat{\mathbf{e}})^T (\mathbf{x_i} - \hat{\mathbf{e}}) \geq 0, \forall i\right) \implies f(\hat{\mathbf{e}}) \leq f(\mathbf{e}), \forall \mathbf{e} \in \mathcal{H}$ □

Based on theorem 1, if for every i, $\nabla f(\hat{\mathbf{e}})^T (\mathbf{x_i} - \hat{\mathbf{e}}) \geq 0$, then the global vector $\hat{\mathbf{e}}$ is an optimal point of the function over the convex hull. Also, this conditions are local, in the sense that the site in which $\mathbf{x_i}$ is assigned can check this condition without any communication (it already knows the global vector). So, as long as the local constraints hold, no communication is needed.

Theorem 2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex, differentiable function, and $\nabla f(\hat{\mathbf{e}})^T (\mathbf{x_i} - \hat{\mathbf{e}}) < 0$. Then $\exists a \in (0, 1]$ such that $f((1-a)\hat{\mathbf{e}} + a\mathbf{x_i}) < f(\hat{\mathbf{e}})$*

Proof. The function f is differentiable, thus all its directional derivatives in $\hat{\mathbf{e}}$ exist and so does the limit, $\lim_{a\to 0} \frac{f(\hat{\mathbf{e}}+a(\mathbf{x_i}-\hat{\mathbf{e}}))-f(\hat{\mathbf{e}})}{a} = \nabla f(\hat{\mathbf{e}})^T (\mathbf{x_i} - \hat{\mathbf{e}}) < 0$. So $\lim_{a\to 0} \frac{f((1-a)\hat{\mathbf{e}}+a\mathbf{x_i})-f(\hat{\mathbf{e}})}{a} < 0$. Thus, there exists $a > 0$ small enough such that, $\frac{f((1-a)\hat{\mathbf{e}}+a\mathbf{x_i})-f(\hat{\mathbf{e}})}{a} < 0$. So, $\exists a \in (0, 1] : f((1-a)\hat{\mathbf{e}} + a\mathbf{x_i}) < f(\hat{\mathbf{e}})$ □

If $\hat{\mathbf{e}}$ and $\mathbf{x_i}$ belong in a convex hull, so does $(1-a)\hat{\mathbf{e}} + a\mathbf{x_i}$ for all $a \in [0, 1]$. Thus, based on theorem 2 it has also been proven, that if one local constraint has been violated, the global vector must be updated, as it not an optimal point any more.

This lets us conclude that this local constraints are optimal, as a violation will happen if and only if a global violation has happen.

2.4.2 Detecting Drop Out Of The Convex Hull

Now we are going to formulate local conditions to monitor whether the global vector $\hat{\mathbf{e}}$ is still inside the convex hull.

As already mentioned, the global vector may get outside the convex hull only after deleting a point. Initially suppose that every site knows the coefficients of its points used in creating $\hat{\mathbf{e}}$ and that the point being deleted is $\mathbf{x_D}$. If the coefficient a_D of this point is zero, then it is obvious that the global vector $\hat{\mathbf{e}}$

is still inside the convex hull as it is already expressed as a convex combination of the rest of the points. Thus, deleting a point with zero coefficient does not drop the global vector out of the hull.

In case the coefficients are not available an alternative is to check whether for the deleted point we have that $\nabla f(\hat{\mathbf{e}})^T\left(\mathbf{x_D}-\hat{\mathbf{e}}\right) > 0$ which as proven in theorem 3, implies that $a_D = 0$. This condition is less strict, as $a_D = 0$ does not guarantee that $\nabla f(\hat{\mathbf{e}})^T\left(\mathbf{x_D}-\hat{\mathbf{e}}\right) > 0$, as an example consider the case $\hat{\mathbf{e}}$ can be expressed by two different convex combinations of $\mathcal{D}$'s elements. Nevertheless, it is a certificate that the current solution is still valid after the deletion.

Theorem 3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex, differentiable function, $\mathcal{D} \subseteq \mathbb{R}$, $\hat{\mathbf{e}} = \sum_i a_i \mathbf{x_i}$, $\sum_i a_i = 1$, $a_i \geq 0, \forall \mathbf{x_i} \in \mathcal{D}$ and $f\left(\hat{\mathbf{e}}\right) \leq f\left(\mathbf{e}\right), \forall \mathbf{e} \in ConvexHull\left(\mathcal{D}\right)$. If $\nabla f(\hat{\mathbf{e}})^T\left(\mathbf{x_j}-\hat{\mathbf{e}}\right) > 0$, then $a_j = 0$.*

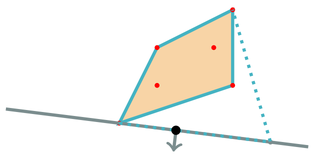
Proof. Suppose $a_j \neq 0$. So $a_j > 0$ and $\nabla f(\hat{\mathbf{e}})^T\left(\mathbf{x_j}-\hat{\mathbf{e}}\right) > 0$ implies $\nabla f(\hat{\mathbf{e}})^T\left(a_j\mathbf{x_j}-a_j\hat{\mathbf{e}}\right) > 0$. But, $\nabla f(\hat{\mathbf{e}})^T\left(a_i\mathbf{x_i}-a_i\hat{\mathbf{e}}\right) \geq 0, \forall i \neq j$. Summing all the inequalities, $\nabla f(\hat{\mathbf{e}})^T\left(\sum_i a_i\mathbf{x_j}-\sum_i a_i\hat{\mathbf{e}}\right) > 0$, which implies $\nabla f(\hat{\mathbf{e}})^T\left(\hat{\mathbf{e}}-\hat{\mathbf{e}}\right) = 0 > 0$ which is a contradiction. Thus $a_j = 0$. □

When we can not obtain one of the above certificates, we can distinguish between two cases. Either the global vector is not included in the updated convex hull or it can be written as a convex combination of the rest of the elements. In the first case, our current solution is outdated and the only thing we can do is start the update process as it will be described in section 2.5.2. In the other case no action is needed. Unfortunately, in the general case the site where the delete event arrived does not have the information needed to detect it. So we can either move straight to updating our solution, and possible converge in the same one if it is still in the hull, or we can first check if the solution is still in the hull and proceed in updating only if necessary.

From theorem 3 it follows that only vectors $\mathbf{x_i}$ satisfying $\nabla f(\hat{\mathbf{e}})^T\left(\mathbf{x_i}-\hat{\mathbf{e}}\right) = 0$ may have non zero coefficients in any representation of $\hat{\mathbf{e}}$ as a convex combination of $(\mathcal{D} \setminus \{\mathbf{x_D}\})$'s elements. Thus, to check whether $\hat{\mathbf{e}}$ is contained in the updated convex hull, it is sufficient for the site to collect all $\mathbf{x_i} \in \mathcal{D} \setminus \{\mathbf{x_D}\}$ satisfying $\nabla f(\hat{\mathbf{e}})^T\left(\mathbf{x_i}-\hat{\mathbf{e}}\right) = 0$ and try to express it as their convex combination. If it succeeds, then no further action, rather than possibly transmitting the new combination, is needed. Otherwise it has proven that $\hat{\mathbf{e}}$ is not contained in the updated convex hull and an update is necessary. Condition $\nabla f(\hat{\mathbf{e}})^T\left(\mathbf{x_i}-\hat{\mathbf{e}}\right) = 0$ can of course be checked locally in $\mathbf{x_i}$'s site.

2.5 Updating

When the monitoring phase detects that the global vector $\hat{\mathbf{e}}$ may have been outdated, a replacement, let it be $\hat{\mathbf{e}}_+$, must be selected.


(a) Dropped

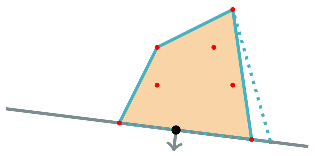

(b) Not dropped

Figure 2.3: A old point is removed from the old convex hull (dotted line). The old global vector is the black dot and the gray line is the hyperplane defined by the local constraint (the arrow is proportional to the negative of the gradient where the global vector is). In the left picture the global vector has been dropped out of the hull, in contrast to the one in the right.

2.5.1 Updating Global Vector After Expansion

Let us first focus on the case of the global vector being outdated due to the addition of a point which violated the local constraint. To update the global vector in this case we propose two methods for the description of which we will denote the site where the new point arrived as r and the point as $\mathbf{x_N}$. Note that $\mathbf{x_N} \in \mathcal{M}_r$.

Update Method 1

As part of the first proposed update method the following convex minimization problem is solved locally in site r, for $s = r$,

$$\begin{aligned} \underset{a_j, \forall \mathbf{y_j} \in \mathcal{M}_s}{\textit{minimize}} \quad & f\left(\sum_{\mathbf{y_j} \in \mathcal{M}_s} a_j \mathbf{y_j}\right) \\ \textit{s.t.} \quad & 0 \leq a_j, \qquad\qquad \forall \mathbf{y_j} \in \mathcal{M}_s \\ & \sum_{\mathbf{y_j} \in \mathcal{M}_s} a_j = 1 \end{aligned} \tag{2.3}$$

The point achieving the minimum is then broadcasted to all other sites and labeled as the new global vector, replacing the old one. After that, each site has to check if any of its points is now violating the local constraint (due to changing the global vector). If no site has a violation, then the update process has finished and all the sites have knowledge of the new and updated global vector $\mathbf{\hat{e}_+}$ due to the last broadcast. Otherwise, one of the sites having at least one violation is selected to locally solve (2.3) with itself as s and broadcast the new solution, forcing every other site on rechecking the local constraint on its points. This process is repeated until no violation is found.

Proof Of Correctness We will now go ahead and prove that the above process converges to the correct point. It is sufficient to show that this method makes an improvement in each step without ever dropping below the minimum we are seeking and that it will terminate if and only if a point achieving the minimum over the updated convex hull has been found.

Denote $\mathbf{e_i}$ and $\mathcal{M}^i$ as the solution and the set of vectors, respectively, of the i-th minimization problem solved while updating the global vector using the above method. Also let $\mathbf{e_0} = \hat{\mathbf{e}}$ and $\mathcal{M}^0 = \mathcal{M}_r$. As already noted, the global vector belongs to every sites' $ConvexHull\left(\mathcal{M}_j\right)$, so $\mathbf{e_{i-1}} \in ConvexHull\left(\mathcal{M}^i\right)$ before solving the i-th minimization problem. Thus, $f\left(\mathbf{e_i}\right) \leq f\left(\mathbf{e_{i-1}}\right)$. Also, the site that solves the problem at each step has a vector which violates the local constraint, thus from theorem 2 it is obvious that the inequality is strict, that is, $f\left(\mathbf{e_i}\right) < f\left(\mathbf{e_{i-1}}\right)$. In addition, $f\left(\mathbf{e_i}\right) \geq f\left(\hat{\mathbf{e}}_+\right)$ as $\bigcup_s ConvexHull\left(\mathcal{M}_s\right) \subseteq ConvexHull\left(\bigcup_s \mathcal{M}_s\right)$, so the true minimum will never be surpassed.

Now it has only been left to show that this method will stop only when such a point has been found. This is a direct consequence of the terminating condition and theorem 1.

Update Method 2

The second method is based on the fact that $\mathbf{x_N}$ is the only point in $\mathcal{D}$ violating the local constraints. That is, $\mathbf{x_N} \in \mathcal{S}^- \equiv \left\{\mathbf{x} \in \mathbb{R}^n \middle| \nabla f(\hat{\mathbf{e}})^T \left(\mathbf{x} - \hat{\mathbf{e}}\right) < 0\right\}$ whereas $\mathcal{D} \setminus \{\mathbf{x_N}\} \subseteq \mathcal{S}^+ \cup \partial\mathcal{S}$, where $\partial\mathcal{S} \equiv \left\{\mathbf{x} \in \mathbb{R}^n \middle| \nabla f(\hat{\mathbf{e}})^T \left(\mathbf{x} - \hat{\mathbf{e}}\right) = 0\right\}$ and $\mathcal{S}^+ \equiv \left\{\mathbf{x} \in \mathbb{R}^n \middle| \nabla f(\hat{\mathbf{e}})^T \left(\mathbf{x} - \hat{\mathbf{e}}\right) > 0\right\}$. Also we know that $\hat{\mathbf{e}}_+ \in \mathcal{S}^-$, as $0 > f(\hat{\mathbf{e}}_+) - f(\hat{\mathbf{e}}) \geq \nabla f(\hat{\mathbf{e}})^T \left(\hat{\mathbf{e}}_+ - \hat{\mathbf{e}}\right)$. But, by definition $\hat{\mathbf{e}}_+ \in ConvexHull(\mathcal{D})$, thus $\hat{\mathbf{e}}_+ \in ConvexHull(\mathcal{D}) \cap \mathcal{S}^-$. This means that we only need to solve the minimization over this set. As this set is open, we will extend it to contain its boundary to make the analysis simpler. Let $\mathcal{H}^- \equiv ConvexHull(\mathcal{D}) \cap (\mathcal{S}^- \cup \partial\mathcal{S})$. Minimization can now be solved over $\mathcal{H}^-$.

This method builds on the fact that if we can, without a lot of communication, create a representation of $\mathcal{H}^-$ on a site, we can then solve the rest of the problem locally. One obvious way to do it would be to gather all the points in this site and solve the problem, this would not even require explicitly computing $\mathcal{H}^-$, but it would imply huge amounts of data transfers on every update.

Fortunately, there is a more efficient way. Based on theorem 4, $\mathcal{H}^-$ is equivalent to the convex hull of $\{\mathbf{x_N}\} \cup \hat{\mathcal{D}}$, where $\hat{\mathcal{D}}$ is the set of the points where rays emitted from $\mathbf{x_N}$ toward every other point of $\mathcal{D}$ intersect $\partial\mathcal{S}$. Strictly speaking, $\hat{\mathcal{D}}$ is defined as $\{\mathbf{z_i} | \mathbf{z_i} = \lambda_i \mathbf{x_i} + (1 - \lambda_i)\,\mathbf{x_N} \in \partial\mathcal{S}, \mathbf{x_i} \in \mathcal{D} \setminus \{\mathbf{x_N}\}, \lambda_i \in [0,1]\}$. We still have make no improvement over sending all the data points, but we can make one using the fact that we do not need all the points of $\hat{\mathcal{D}}$ to create the convex hull, but only its extremes, that is, points that can not be written as a convex combination of the rest of the points. It worths commenting here that while we could just send the extreme points of $\mathcal{D}$ to one site to do the computation, this would imply a higher communicational cost, as an extreme point of $\hat{\mathcal{D}}$ has been generated by an extreme point of $\mathcal{D}$, but an extreme of $\mathcal{D}$ does not necessarily produce an extreme on $\hat{\mathcal{D}}$.

While each site can locally compute the part of $\hat{\mathcal{D}}$ corresponding to its points (let it be $\hat{\mathcal{D}}_i$), $\hat{\mathcal{D}}$ is a distributed set and thus we can not compute its extremes without extra cost. So we propose using an heuristic based on the fact that an

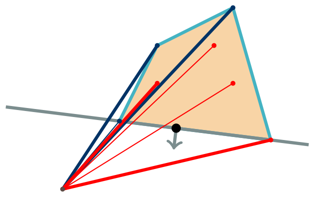


Figure 2.4: A point which violated the local constraint and the rays emitted from it to each other point. If red and blue are the two sites, with each one knowing only the points of its color, then only the points corresponding to bold rays will be transmitted.

extreme of a partitioned set will be an extreme in one of its subsets. Thus, each site can compute the extremes of its $\hat{\mathcal{D}}_i$ and send them. The site collecting the extremes can optionally discard any non extremes before proceeding in solving the minimization.

Finally, we can reduce the number of non extremes transmitted by having each site to compute its $\hat{\mathcal{D}}_i$ not only based on its points, but based on all the points it has knowledge of $(\mathcal{M}_i \backslash \{\mathbf{x_N}\})$. In this case, $\hat{\mathcal{D}}_i$s are not disjoint, so each extreme point will be detected by at least one site. Forcing each site to transmit only points corresponding to its stream will prevent multiple transmissions of the same point.

Now we are ready to summarize our second proposed method for updating the solution. Initially the site which detected the violation broadcasts the violator $(\mathbf{x_N})$. Then each site locally computes $\hat{\mathcal{D}}_i$, as described, and finds its extremes using any one of the known local algorithms for this tasks. Each site transmits the extremes it detected that correspond to points originating from its stream to a node assigned to do the final computation. This site collects $\mathbf{x_N}$ and all the candidate extreme points in a set, let it be $\mathcal{C}$, and optionally discards any non extremes, deleteing them from $\mathcal{C}$. This site then locally solves the following convex optimization problem,

$$\begin{aligned} \underset{a_j, \forall \mathbf{x_j} \in \mathcal{C}}{\mathit{minimize}} \quad & f\left(\sum_{\mathbf{x_j} \in \mathcal{C}} a_j \mathbf{x_j}\right) \\ \mathit{s.t.} \quad & 0 \leq a_j, \qquad\qquad \forall \mathbf{x_j} \in \mathcal{C} \\ & \sum_{\mathbf{x_j} \in \mathcal{C}} a_j = 1 \end{aligned} \tag{2.4}$$

The solution of this problems is the updated solution and a point achieving it is the updated global vector $\hat{\mathbf{e}}_+$, so it is broadcasted to all the other sites.

Theorem 4. *Let $\mathcal{A}$ be a set, $\partial\mathcal{B}$ be a hyperplane and $\mathcal{B}^+$, $\mathcal{B}^-$ the two open half-spaces created by subtracting $\partial\mathcal{B}$ from the space. If $\mathbf{y}$ is a point in $\mathcal{B}^-$, $\mathcal{A} \subseteq \mathcal{B}^+ \cup \partial\mathcal{B}$ and $\hat{\mathcal{A}} \equiv \{\mathbf{z_i} | \mathbf{z_i} = \lambda_i \mathbf{x_i} + (1-\lambda_i)\, \mathbf{y} \in \partial\mathcal{B}, \mathbf{x_i} \in \mathcal{A}, \lambda_i \in [0,1]\}$, then $ConvexHull\,(\mathcal{A} \cup \{\mathbf{y}\}) \cap (\mathcal{B}^- \cup \partial\mathcal{B}) \equiv ConvexHull(\hat{\mathcal{A}} \cup \{\mathbf{y}\})$.*

Proof. We will start by proving that $\mathit{ConvexHull}\left(\mathcal{A} \cup \{\mathbf{y}\}\right) \cap \left(\mathcal{B}^{-} \cup \partial\mathcal{B}\right) \supseteq \mathit{ConvexHull}\left(\hat{\mathcal{A}} \cup \{\mathbf{y}\}\right)$. Observe that $\hat{\mathcal{A}} \cup \{\mathbf{y}\} \subseteq \mathit{ConvexHull}\left(\mathcal{A} \cup \{\mathbf{y}\}\right)$, so $\mathit{ConvexHull}\left(\hat{\mathcal{A}} \cup \{\mathbf{y}\}\right) \subseteq \mathit{ConvexHull}\left(\mathcal{A} \cup \{\mathbf{y}\}\right)$. In addition $\mathcal{B}^{-} \cup \partial\mathcal{B}$ is convex, so $\hat{\mathcal{A}} \cup \{\mathbf{y}\} \subseteq \mathcal{B}^{-} \cup \partial\mathcal{B}$ implies $\mathit{ConvexHull}\left(\hat{\mathcal{A}} \cup \{\mathbf{y}\}\right) \subseteq \mathcal{B}^{-} \cup \partial\mathcal{B}$. Combining this two relations, we have concluded the first part of the proof, having proven that $\mathit{ConvexHull}\left(\mathcal{A} \cup \{\mathbf{y}\}\right) \cap \left(\mathcal{B}^{-} \cup \partial\mathcal{B}\right) \supseteq \mathit{ConvexHull}\left(\hat{\mathcal{A}} \cup \{\mathbf{y}\}\right)$.

Now we must also prove that the opposite relation also holds. Assume that $\mathcal{B}^{-} \cup \partial\mathcal{B} \equiv \left\{\mathbf{x} \in \mathbb{R}^n \middle| \mathbf{p}^T\mathbf{x} \leq b\right\}$. For every $\mathbf{x_i} \in \mathcal{A}$, $\mathbf{p}^T\mathbf{x_i} \geq b$ and $\mathbf{p}^T\mathbf{y} < b$. Let us also show that any element $\mathbf{x_i}$ of $\mathcal{A}$ is mapped into one element of $\hat{\mathcal{A}}$. A $\mathbf{x_i} \in \mathcal{A}$ will be mapped to $\mathbf{z_i} = \lambda_i\mathbf{x_i} + (1-\lambda_i)\mathbf{y}$ for $\lambda_i = \frac{\mathbf{p}^T\mathbf{x_i} - b}{\mathbf{p}^T\mathbf{x_i} - \mathbf{p}^T\mathbf{y}}$. We observe that $\lambda_i \in [0,1)$ due to space's partition and thus $\mathbf{z_i}$ belongs in $\hat{\mathcal{A}}$.

Let $\mathbf{z}$ be a point in $\mathit{ConvexHull}\left(\mathcal{A} \cup \{\mathbf{y}\}\right) \cap \left(\mathcal{B}^{-} \cup \partial\mathcal{B}\right)$. As it is contained in the convex hull of $\mathcal{A} \cup \{\mathbf{y}\}$, there exist $\mu_i \geq 0$ and $\mu_{\mathbf{y}} \geq 0$ such that $\sum_i \mu_i + \mu_{\mathbf{y}} = 1$ and $\mathbf{z} = \sum_i \mu_i\mathbf{x_i} + \mu_{\mathbf{y}}\mathbf{y}$. Manipulating this expression we can get that $\mathbf{z} = \sum_i \frac{\mu_i}{1-\lambda_i}\left(1-\lambda_i\right)\mathbf{x_i} + \mu_{\mathbf{y}}\mathbf{y} = \sum_i \frac{\mu_i}{1-\lambda_i}\mathbf{z_i} + \left(\mu_{\mathbf{y}} - \sum_i \frac{\mu_i}{1-\lambda_i}\lambda_i\right)\mathbf{y}$. For each i, $\frac{\mu_i}{1-\lambda_i} \geq 0$. In addition, $\mu_{\mathbf{y}} - \sum_i \frac{\mu_i}{1-\lambda_i}\lambda_i = 1 - \sum_i \frac{1}{1-\lambda_i}\mu_i$. We are only interesting in showing that the coefficient of $\mathbf{y}$ is also nonnegative, thus it is sufficient to show that $1 \geq \sum_i \frac{\mu_i}{1-\lambda_i} = \sum_i \frac{\mathbf{p}^T\mathbf{x_i} - \mathbf{p}^T\mathbf{y}}{b - \mathbf{p}^T\mathbf{y}}\mu_i = \frac{\mathbf{p}^T\mathbf{z} - \mu_{\mathbf{y}}\mathbf{p}^T\mathbf{y} - \sum_i \mu_i\mathbf{p}^T\mathbf{y}}{b - \mathbf{p}^T\mathbf{y}}$. Which is equivalent to $b - \mathbf{p}^T\mathbf{y} \geq \mathbf{p}^T\mathbf{z} - \mathbf{p}^T\mathbf{y}$, which is true as $\mathbf{z} \in \mathcal{B}^{-} \cup \partial\mathcal{B}$, thus $b \geq \mathbf{p}^T\mathbf{z}$. So we can conclude that $\mathbf{z}$ belongs in the convex hull of $\hat{\mathcal{A}} \cup \{\mathbf{y}\}$. Thus, $\mathit{ConvexHull}\left(\mathcal{A} \cup \{\mathbf{y}\}\right) \cap \left(\mathcal{B}^{-} \cup \partial\mathcal{B}\right) \subseteq \mathit{ConvexHull}\left(\hat{\mathcal{A}} \cup \{\mathbf{y}\}\right)$.

We have shown that each of the two sets is subset of the other, thus they are indeed equivalent. □

2.5.2 Update after contraction

Update method 1 is also applicable in case a vector has been deleted. The only difference is that the first time problem 2.3 is solved, it may return a solution greater to the previous minimum. To be exact, it can not return a lower value and if the same one is returned, then no extra iterations will be needed. Except from that, the rest of the method stays the same and the proof still holds, proving that this method converges. Note that $\mathbf{x_D}$ should be deleted from every site's list of known points prior to using the method.

Chapter 3

Approximate Minimization

Many tasks do not need the exact solution of the minimization problem but only an ϵ-approximation. Thus, in this section we will extend our methods to handle approximate minimization. More formally, if at any time a point $\mathbf{e}^* \in \mathcal{H} \equiv \mathit{ConvexHull}\,(\mathcal{D})$ minimizes f over $\mathcal{H}$, we will be interested in computing and updating a global vector $\hat{\mathbf{e}}$ satisfying $f(\mathbf{e}^*) \leq f(\hat{\mathbf{e}}) \leq f(\mathbf{e}^*) + \epsilon$. Of course ϵ is assumed to be positive.

3.1 Approximate Monitoring

We will start relaxing our methods by discussing how the monitoring phase changes.

3.1.1 Detecting Outdated Approximation

We will now extend theorem 1 to the approximate case.

Theorem 5. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a convex, differentiable function, $\mathcal{D} = \{\mathbf{x_i} | \forall i\}$ and $\hat{\mathbf{e}} \in \mathcal{H} \equiv \mathit{ConvexHull}\,(\mathcal{D})$. Then, $\left(\nabla f(\hat{\mathbf{e}})^T (\mathbf{x_i} - \hat{\mathbf{e}}) \geq -\epsilon, \quad \forall i\right) \implies f(\hat{\mathbf{e}}) \leq f(\mathbf{e}) + \epsilon, \forall \mathbf{e} \in \mathcal{H}$.*

Proof. For any $\mathbf{e} \in \mathcal{H}$ there exist $a_i \in [0,1], \forall i$ such that $\mathbf{e} = \sum_i a_i \mathbf{x_i}$, with $\sum_i a_i = 1$. Thus, based on theorem's preconditions, $\nabla f(\hat{\mathbf{e}})^T (a_i \mathbf{x_i} - a_i \hat{\mathbf{e}}) \geq -a_i \epsilon, \quad \forall i = 1, \ldots, N$. Summing all the inequalities, we get $\nabla f(\hat{\mathbf{e}})^T (\mathbf{e} - \hat{\mathbf{e}}) \geq -\epsilon$. And due to convexity, $f(\mathbf{e}) - f(\hat{\mathbf{e}}) \geq \nabla f(\hat{\mathbf{e}})^T (\mathbf{e} - \hat{\mathbf{e}}) \geq -\epsilon$. So, $\left(\nabla f(\hat{\mathbf{e}})^T (\mathbf{x_i} - \hat{\mathbf{e}}) \geq -\epsilon, \forall i\right) \implies f(\hat{\mathbf{e}}) \leq f(\mathbf{e}) + \epsilon, \forall \mathbf{e} \in \mathcal{H}$ □

We will call $\nabla f(\hat{\mathbf{e}})^T (\mathbf{x_i} - \hat{\mathbf{e}}) \geq -\epsilon$ relaxed local constraint on $\mathbf{x_i}$. Theorem 5 states that if no local constraint is violated, $f(\hat{\mathbf{e}}) \leq f(\mathbf{e}) + \epsilon, \forall \mathbf{e} \in \mathcal{H}$, but $\mathbf{e}^* \in \mathcal{H}$, thus $f(\hat{\mathbf{e}}) \leq f(\mathbf{e}^*) + \epsilon$.

Theorem 2 still holds for the relaxed local constrained, so using one of the existing update strategies guarantees a descent step.

So, to make our approach support ϵ-approximate minimization, the only change needed is to replace the local constraints by their relaxed counterparts.

3.1.2 Detecting Drop Out In The Approximate Case

In case the coefficient a_D of the deleted point $\mathbf{x_D}$ in a convex combination of the global vector is zero or $\nabla f(\hat{\mathbf{e}})^T (\mathbf{x_D} - \hat{\mathbf{e}}) > 0$, then the global vector is still inside the hull, as in the exact case. Unfortunately, if both of this are false, testing if the global vector is inside the hull will require knowledge of much more points than in the exact case. Thus a simple alternative is to just move straight forward to the update stage and let it return the same solution, if it still is inside the convex hull.

3.2 Approximate Update

The update methods can as well be adjusted to the approximate case.

3.2.1 Approximate Update After Expansion

Both methods used in the exact case can be adapted to the approximate one.

Update Method 1

The first update method needs only two changes. First of all, when a site checks its points for violators after an intermediate update of the global vector, it is sufficient to check if there is any point violating the relaxed constraint. This guarantees that the algorithm will terminate with an ϵ-approximate solution, as a consequence of theorem 5.

Also, it is sufficient for problem 2.3 to be approximately solved. The precision should be at least ϵ and small enough for the solver to return a solution smaller than the previous one (with the exception of the first iteration). That guarantees progress of the method in each step. Such a small precision is guaranteed to exist, as it has already been proven that in case violators exist a descend step always exists.

Update Method 2

For the second method, instead of sending the points that the rays intersect the hyperplane $\partial\mathcal{S}$, now we send the points where the rays meet the hyperplane $\partial\mathcal{S}_\epsilon \equiv \left\{\mathbf{x} \middle| \nabla f(\hat{\mathbf{e}})^T (\mathbf{x} - \hat{\mathbf{e}}) = -\epsilon\right\}$. The point $\mathbf{x_N}$ which triggered the update is the only one contained in $\mathcal{S}_\epsilon^- \equiv \left\{\mathbf{x} \middle| \nabla f(\hat{\mathbf{e}})^T (\mathbf{x} - \hat{\mathbf{e}}) < -\epsilon\right\}$, thus theorem 4 still holds with $\partial\mathcal{S}_\epsilon$ as $\partial\mathcal{B}$, $\mathcal{S}_\epsilon^-$ as $\mathcal{B}^-$ and the rest of the space as $\mathcal{B}^+$. Based on this, we know that the node receiving the points can fully describe and minimize over the intersection of the updated convex hull with $\mathcal{S}_\epsilon^- \cup \partial\mathcal{S}_\epsilon$. The result of the minimization, let it be $\mathbf{e_c}$, will either lie on $\partial\mathcal{S}_\epsilon$ or on $\mathcal{S}_\epsilon^-$. We shall discuss this two cases separately.

In case $\mathbf{e_c} \in \mathcal{S}_\epsilon^-$, we have computed our new global vector and no point is violating the updated relaxed constraint. Broadcasting it will end the update phase. To prove this, observe that $\mathbf{e_c}$ is a point achieving the minimum over $\mathcal{C} \equiv ConvexHull\left(D\right) \cap \left(\mathcal{S}_\epsilon^- \cup \partial\mathcal{S}_\epsilon\right)$, thus for every point $\mathbf{z} \in \mathcal{C}$, it holds that $\nabla f(\mathbf{e_c})^T\left(\mathbf{z} - \mathbf{e_c}\right) \geq 0$. Also, if $\mathbf{y_i}$ is the intersection of the ray from $\mathbf{x_N}$ to $\mathbf{x_i} \neq \mathbf{x_N}$, then $\nabla f(\mathbf{e_c})^T\left(\mathbf{y_i} - \mathbf{e_c}\right) \geq 0$ and $\mathbf{y_i} = a_i\mathbf{x_i} + \left(1 - a_i\right)\mathbf{x_N}$ for some $a_i \in (0, 1]$. In addition, for any point $\mathbf{w} \in ConvexHull\left(D\right) \cap \mathcal{S}_\epsilon^+$, there exist $\lambda_i \in [0, 1]$, $\lambda_N \in [0, 1]$ with $\sum \lambda_i + \lambda_N = 1$ and $\mathbf{w} = \sum \lambda_i \mathbf{x_i} + \lambda_N \mathbf{x_N}$. Substituting, $\mathbf{w} = \sum \lambda_i \left(\frac{1}{a_i}\mathbf{y_i} + \left(1 - \frac{1}{a_i}\right)\mathbf{x_N}\right) + \lambda_N \mathbf{x_N} = \beta \sum \mu_i \mathbf{y_i} + \left(1 - \beta\right)\mathbf{x_N}$, where $\beta = \sum_i \frac{\lambda_i}{a_i} \geq 0$ and $\mu_i = \frac{\lambda_i}{a_i\beta}$. As $\mu_i \geq 0$ and $\sum_i \mu_i = 1$, $\sum \mu_i \mathbf{y_i} \in \mathcal{C}$, thus $\beta\nabla f(\mathbf{e_c})^T\left(\sum \mu_i \mathbf{y_i} - \mathbf{e_c}\right) \geq 0$. But, $\mathbf{e_c} \in \mathcal{S}_\epsilon^-$, thus any convex coefficient representing $\mathbf{e_c}$ has a non zero coefficient for $\mathbf{x_N}$ so by theorem 3, $\nabla f(\mathbf{e_c})^T\left(\mathbf{x_N} - \mathbf{e_c}\right) = 0$. Summing this two results, we can now prove that $\nabla f(\mathbf{e_c})^T\left(\mathbf{w} - \mathbf{e_c}\right) \geq 0$, concluding a proof that any point of $ConvexHull\left(D\right)$ is satisfying not only the relaxed constraints, but the original ones as well and $\mathbf{e_c}$ is fulfilling our requirements for the global vector.

Unfortunately on the other case, where $\mathbf{e_c} \in \partial\mathcal{S}_\epsilon$, no such guarantee can be given. Because of $\mathbf{e_c} \in \partial\mathcal{S}_\epsilon$ we know that no point achieving the minimum can be in $\mathcal{S}_\epsilon^-$. Thus it must be in $\left\{\mathbf{x}\middle|\nabla f(\hat{\mathbf{e}})^T\left(\mathbf{x} - \hat{\mathbf{e}}\right) \in [-\epsilon, 0)\right\}$. Obviously, this means that our old solution was already an ϵ-approximation to the one on the updated convex hull. But keeping the old one breaks our monitoring conditions, which requires that no point is violating the relaxed constrain when a new one arrives. Thus we must fall back to the first method to reach a state we can keep monitoring the streams in a communication efficient way.

3.2.2 Approximate update after contraction

Update method 1 for the approximate case can again be adjusted in the same way the exact one was adjusted for the exact minimization case.

3.3 Relative Error Guarantees

In the previous sections we have assumed that ϵ is a known constant and thus the error is absolute. Still, by substituting $\epsilon = \frac{\hat{\epsilon}}{1+\hat{\epsilon}} f\left(\hat{\mathbf{e}}\right)$, we can convert the bound into a $\hat{\epsilon}$ relative one, that is $f(\mathbf{e}^*) \leq f\left(\hat{\mathbf{e}}\right) \leq \left(1 + \hat{\epsilon}\right) f\left(\mathbf{e^*}\right)$. But, now, ϵ should change every time a new global vector is transmitted, even if it is a temporary one.

Chapter 4

L_2 Support Vector Machines

In this chapter we will initially give a small introduction to Support Vector Machines in 4.1 and then a discussion on applying our methods to L_2 Support Vector Machines will follow.

4.1 Support Vector Machines

Support vector machines[1] are a popular machine learning algorithm initially introduced for two class classification. In the case of two class classification they try to find a maximal separating hyperplane, that is a hyperplane which separates the two classes with a gap as big as possible. This can only be accomplished when the data of the two classes are linearly separable. To handle the case of non linearly separable data, a penalty factor is introduced. Every point which is located in the wrong side of the plane is introducing a penalty, based on some loss function. Traditionally, this loss function is the hinge loss but for the L_2 SVMs we will use, it is the squared hinge loss function. In addition, usually the points are mapped in to a higher dimensional space through a function $\boldsymbol{\phi}(\mathbf{x})$ and separated there. Let $\mathbf{x_i}$ be the initial data points and $y_i \in \{\pm 1\}$ the label of $\mathbf{x_i}$, then, following [13], the formulation of the L_2-SVM is,

$$\begin{aligned} \underset{\mathbf{w},b,\rho,\xi_i}{minimize} \quad & \|w\|^2 + b^2 - 2\rho + C\sum_i \xi_i^2 \\ s.t. \quad & y_i\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_i}) + b\right) \geq \rho - \xi_i \end{aligned}$$

The dual of this problem can be shown to be,

$$\begin{aligned} \underset{\mathbf{a}}{minimize} \quad & \mathbf{a}^T\hat{\Phi}^T\hat{\Phi}\mathbf{a} + diag\left(\hat{\Phi}^T\hat{\Phi}\right)\mathbf{a} \\ s.t. \quad & \mathbf{a} \geq \mathbf{0} \\ & \mathbf{1}^T\mathbf{a} = 1 \end{aligned}$$

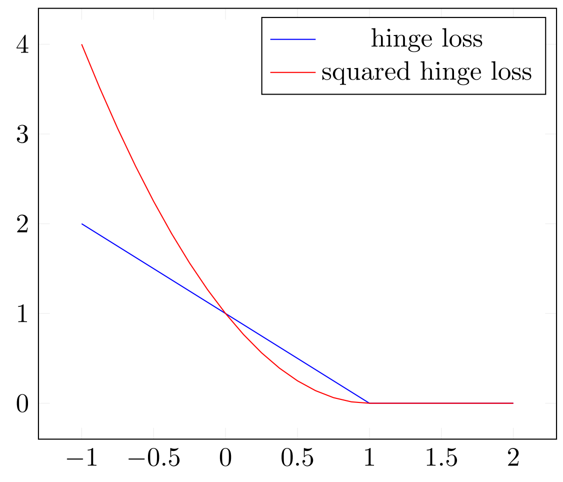


Figure 4.1: The two loss functions. The horizontal axis is proportional to the distance of the point from the separating hyperplane, with positives been in its classes half-space.

where $\hat{\mathbf{\Phi}}$ is a matrix whose i-th column is $\hat{\boldsymbol{\phi}}(\mathbf{x_i}) = [y_i \boldsymbol{\phi}(\mathbf{x_i})^T \quad y_i \quad \frac{1}{\sqrt{C}} \mathbf{e_j}^T]^T$. The corresponding kernel function is $\hat{k}(\mathbf{x_i}, \mathbf{x_j}) = y_i y_j \left(k(\mathbf{x_i}, \mathbf{x_j}) + 1\right) + \frac{1}{C}\delta_{ij}$, where $\delta_{ij} = 1$ if $i = j$, $\delta_{ij} = 0$ otherwise.

Again, following the assumption made in [13], that $\boldsymbol{\phi}(\mathbf{x_i})^T \boldsymbol{\phi}(\mathbf{x_i}) = k$, for all $\mathbf{x_i}$ and k is a constant, we can solve

$$\begin{aligned} \underset{\mathbf{a}}{\textit{minimize}} \quad & \mathbf{a}^T \hat{\Phi}^T \hat{\Phi} \mathbf{a} = \left\| \hat{\Phi} \mathbf{a} \right\|^2 \\ \textit{s.t.} \quad & \mathbf{a} \geq \mathbf{0} \\ & \mathbf{1}^T \mathbf{a} = 1 \end{aligned}$$

instead of the dual, as if a minimum is found to be attained by a point, then this point will be a point minimizing the dual as well. This is clearly a minimization of the convex squared L_2 norm over the convex hull of $\hat{\boldsymbol{\phi}}(\mathbf{x_i})$s.

4.1.1 One class L_2 Support Vector Machine

Extending to the one class L_2 SVM, the problem now is[13],

$$\begin{aligned} \underset{\mathbf{w}, \rho, \xi_i}{\textit{minimize}} \quad & \|w\|^2 - 2\rho + C \sum_i \xi_i^2 \\ \textit{s.t.} \quad & \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_i}) \geq \rho - \xi_i \end{aligned}$$

and under the same assumption, the dual is the same, but with $\hat{\boldsymbol{\phi}}(\mathbf{x_i}) = [\boldsymbol{\phi}(\mathbf{x_i})^T \quad \frac{1}{\sqrt{C}} \mathbf{e_j}^T]^T$. The corresponding kernel function is $\hat{k}(\mathbf{x_i}, \mathbf{x_j}) = k(\mathbf{x_i}, \mathbf{x_j}) + \frac{1}{C}\delta_{ij}$, where $\delta_{ij} = 1$ if $i = j$, $\delta_{ij} = 0$ otherwise.

4.2 L_2-SVMs as Minimization Over the Convex Hull

The dual can be solved using the previously presented methods with $f(\mathbf{x}) = \|\mathbf{x}\|^2$ to train the L_2-SVM by firstly computing $\hat{\boldsymbol{\phi}}(\mathbf{x_i})$ and using this as input to the rest of the algorithm.

4.3 Minimization Using Kernels

The problem with the above approach is that $\hat{\boldsymbol{\phi}}(\mathbf{x_i})$ may be of a very high dimension, even infinitive. Also, it is pretty common not to define $\boldsymbol{\phi}(\mathbf{x_i})$ explicitly. Thus, we shall show that our methods can be used in the case we want to train our SVM through a kernel function $k(\mathbf{x_i}, \mathbf{x_j}) = \boldsymbol{\phi}(\mathbf{x_i})^T\boldsymbol{\phi}(\mathbf{x_j})$.

To show the applicability of our methods, we shall show that all the computations can be done using the kernel function.

First of all, we can no longer represent the global vector $\hat{\mathbf{e}}$ as a vector, but only as a convex combination. Note that we shall consider part of the convex combination only the vectors with non zero coefficients. Let $\hat{\mathbf{e}} = \sum_i a_i\hat{\boldsymbol{\phi}}(\mathbf{x_i})$. Also, when we need to transmit a vector, for example when broadcasting the convex combination of $\hat{\mathbf{e}}$ or during update, we can not transmit $\hat{\boldsymbol{\phi}}(\mathbf{x_i})$, so we transmit $\mathbf{x_i}$.

4.3.1 Monitoring with Kernels

For the monitoring phase we only need to test the coefficients in the convex combination of $\hat{\mathbf{e}}$ and compute $\nabla f(\hat{\mathbf{e}})^T\left(\hat{\boldsymbol{\phi}}(\mathbf{x_i}) - \hat{\mathbf{e}}\right)$.

The coefficients are already known as they are part of the knowledge needed to represent $\hat{\mathbf{e}}$ as a convex combination.

To compute $\nabla f(\hat{\mathbf{e}})^T\left(\hat{\boldsymbol{\phi}}(\mathbf{x_i}) - \hat{\mathbf{e}}\right)$, we shall observe that $\nabla f(\hat{\mathbf{e}}) = 2\hat{\mathbf{e}}$, so

$$\nabla f(\hat{\mathbf{e}})^T\left(\hat{\boldsymbol{\phi}}(\mathbf{x_i}) - \hat{\mathbf{e}}\right) = \sum_j a_j\hat{\boldsymbol{\phi}}(\mathbf{x_j})^T\hat{\boldsymbol{\phi}}(\mathbf{x_i}) - 2f(\hat{\mathbf{e}}) = \sum_j a_j\hat{k}(\mathbf{x_j}, \mathbf{x_i}) - 2f(\hat{\mathbf{e}}) \tag{4.1}$$

where $f(\hat{\mathbf{e}})$ can also be computed as $\sum_j\sum_i a_j\hat{k}(\mathbf{x_j}, \mathbf{x_i})a_i$ if not available.

A node can still compute this quantity locally as it will known all $\mathbf{x_j}$s participating in the convex combination of $\hat{\mathbf{e}}$.

4.3.2 Updating with Kernels

To handle updating we need to support three operations, ray intersection, convex hull membership and solving problems 2.3, 2.4.

Ray Intersection

In the second update method we need the point $\hat{\boldsymbol{\phi}}(\mathbf{y_i})$ where the ray from $\hat{\boldsymbol{\phi}}(\mathbf{x_N})$ to $\hat{\boldsymbol{\phi}}(\mathbf{x_i})$ intersects a hyperplane of the form $\left\{\boldsymbol{\psi}\middle|\nabla f(\hat{\mathbf{e}})^T(\boldsymbol{\psi}-\hat{\mathbf{e}})=b\right\}$ for some $b \leq 0$. But as $\hat{\boldsymbol{\phi}}(\mathbf{y_i})$ belongs to the ray, $\hat{\boldsymbol{\phi}}(\mathbf{y_i}) = \lambda_i\hat{\boldsymbol{\phi}}(\mathbf{x_i}) + (1-\lambda_i)\hat{\boldsymbol{\phi}}(\mathbf{x_N})$ for some $\lambda_i \in (0,1]$. Substituting in hyperplane's equation we get that,

$$\lambda_i = \frac{\sum_j a_j\hat{k}(\mathbf{x_j},\mathbf{x_N}) - f(\hat{\mathbf{e}}) - \frac{1}{2}b}{\sum_j a_j\hat{k}(\mathbf{x_j},\mathbf{x_N}) - \sum_j a_j\hat{k}(\mathbf{x_j},\mathbf{x_i})} \tag{4.2}$$

which can be computed locally for $\mathbf{x_i}$, as the site will have received $\mathbf{x_N}$ and it knows $\hat{\mathbf{e}}$. The kernel function now has as follows,

$$\hat{k}(\mathbf{y_i},\mathbf{x_j}) = \hat{k}(\mathbf{x_j},\mathbf{y_i}) = \lambda_i\hat{k}(\mathbf{x_i},\mathbf{x_j}) + (1-\lambda_i)\hat{k}(\mathbf{x_N},\mathbf{x_j}) \tag{4.3a}$$

$$\begin{aligned}\hat{k}(\mathbf{y_i},\mathbf{y_j}) = \hat{k}(\mathbf{y_j},\mathbf{y_i}) &= \lambda_i\lambda_j\hat{k}(\mathbf{x_i},\mathbf{x_j}) + (1-\lambda_i)(1-\lambda_j)\hat{k}(\mathbf{x_N},\mathbf{x_N}) \\ &\quad + (1-\lambda_i)\lambda_j\hat{k}(\mathbf{x_N},\mathbf{x_j}) + (1-\lambda_j)\lambda_i\hat{k}(\mathbf{x_N},\mathbf{x_i})\end{aligned} \tag{4.3b}$$

Convex Hull Membership

Most of the algorithms that can compute the extremes points of a set $\mathcal{C}$ require a method to test if a given point $\hat{\boldsymbol{\phi}}(\mathbf{x}) \in \mathcal{C}$ is contained in the convex hull of another set $\mathcal{V} \subseteq \mathcal{C}$.

In the kernelized formulation of the problem, there are at least two ways to do accomplish it. The first one is by computing the distance of the point from the convex hull of $\mathcal{V}$ by solving the quadratic minimization problem,

$$\begin{aligned}\underset{\mathbf{a}}{minimize} \quad & \left\|\hat{\Xi}\mathbf{a} - \hat{\boldsymbol{\phi}}(\mathbf{x})\right\|^2 \\ s.t. \quad & \mathbf{a} \geq \mathbf{0} \\ & \mathbf{1}^T\mathbf{a} = 1\end{aligned}$$

where $\hat{\Xi}$ is the matrix whose columns are the vectors of $\mathcal{V}$. This problem is equivalent to,

$$\begin{aligned}\underset{\mathbf{a}}{minimize} \quad & \sum_{i:\hat{\boldsymbol{\phi}}(\mathbf{x_i})\in\mathcal{V}}\ \sum_{j:\hat{\boldsymbol{\phi}}(\mathbf{x_j})\in\mathcal{V}} a_j\hat{k}(\mathbf{x_j},\mathbf{x_i})a_i - 2\sum_{j:\hat{\boldsymbol{\phi}}(\mathbf{x_j})\in\mathcal{V}} a_j\hat{k}(\mathbf{x_j},\mathbf{x}) + \hat{k}(\mathbf{x},\mathbf{x}) \\ s.t. \quad & \mathbf{a} \geq \mathbf{0} \\ & \mathbf{1}^T\mathbf{a} = 1\end{aligned}$$

If its solution is greater than zero, then $\hat{\boldsymbol{\phi}}(\mathbf{x})$ does not belong in the convex hull of $\mathcal{V}$. Otherwise, it is a convex combination of $\mathcal{V}$'s elements.

An alternative is based on theorem 6. Based on this theorem we can solve the problem

$$\begin{aligned} \underset{\mathbf{a}}{minimize} \quad & \hat{\boldsymbol{\phi}}(\mathbf{x})^T \left(\hat{\Xi}\mathbf{a} - \hat{\boldsymbol{\phi}}(\mathbf{x})\right) - \beta \\ s.t. \quad & \mathbf{a} \geq \mathbf{0} \\ & \mathbf{1}^T\mathbf{a} = 1 \\ & \hat{\Xi}^T \left(\hat{\Xi}\mathbf{a} - \hat{\boldsymbol{\phi}}(\mathbf{x})\right) \geq \beta \end{aligned}$$

which is equal to,

$$\begin{aligned} \underset{\mathbf{a}}{minimize} \quad & \sum_{i:\hat{\boldsymbol{\phi}}(\mathbf{x_i})\in\mathcal{V}} \hat{k}(\mathbf{x},\mathbf{x_i})a_i - \hat{k}(\mathbf{x},\mathbf{x}) - \beta \\ s.t. \quad & \mathbf{a} \geq \mathbf{0} \\ & \mathbf{1}^T\mathbf{a} = 1 \\ & \sum_{i:\hat{\boldsymbol{\phi}}(\mathbf{x_i})\in\mathcal{V}} \hat{k}(\mathbf{x_j},\mathbf{x_i})a_i - \hat{k}(\mathbf{x_j},\mathbf{x}) \geq \beta, \qquad \forall j : \hat{\boldsymbol{\phi}}(\mathbf{x_j}) \in \mathcal{V} \end{aligned}$$

$\hat{\boldsymbol{\phi}}(\mathbf{x})$ is inside the convex hull if and only if this problem is not unbounded and has a non-negative solution. Note that this problem is a linear program.

Theorem 6. *Let* $\mathbf{p}$ *be a vector and* $\mathcal{X}$ *be a set of vector whose vectors compromise the column of* B*. Then, exactly one of the following holds,*

1. $\mathbf{p} \in ConvexHull\,(\mathcal{X})$
2. $\exists \mathbf{x} \in ConvexHull\,(\mathcal{X})$ *and* $\exists \beta \in \mathbb{R}$ *such that both* $\mathbf{p}^T(\mathbf{x}-\mathbf{p}) < \beta$ *and* $\forall \hat{\mathbf{x}} \in \mathcal{X}$ *it holds that* $\hat{\mathbf{x}}^T(\mathbf{x}-\mathbf{p}) \geq \beta$

Proof. Let $A = \begin{bmatrix} B \\ \mathbf{1}^T \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$. Then, by Farkas' lemma, either $\exists \boldsymbol{\lambda} \geq 0$ such that $A\boldsymbol{\lambda} = \mathbf{b}$ or $\exists \mathbf{y}$ such that $\mathbf{y}^T A \geq 0$ and $\mathbf{y}^T\mathbf{b} < 0$.

The first case is equivalent to our first case.

In the second case, equivalently we have that $\exists \mathbf{y}, y_s$ such that $\mathbf{y}^T B + y_s \geq 0$ and $\mathbf{y}^T\mathbf{p} + y_s < 0$.Let, $\mathbf{z}$ be the point of the convex hull closer to $\mathbf{p}$. Then

$$\mathbf{z} = \underset{\hat{\mathbf{x}}\in ConvexHull(\mathcal{X})}{argmin} \|\hat{\mathbf{x}} - \mathbf{p}\|^2 = f(\hat{\mathbf{x}})$$

and $\nabla f(\mathbf{x}) = 2\mathbf{x} - 2\mathbf{p}$, thus $(\mathbf{z}-\mathbf{p})^T(\hat{\mathbf{x}}-\mathbf{z}) \geq 0$, or equivalently, $(\mathbf{z}-\mathbf{p})^T\hat{\mathbf{x}} - (\mathbf{z}-\mathbf{p})^T\mathbf{z} \geq 0$. Also, $(\mathbf{z}-\mathbf{p})^T\mathbf{p} - (\mathbf{z}-\mathbf{p})^T\mathbf{z} = -\|\mathbf{z}-\mathbf{p}\|^2 < 0$ as $\mathbf{p}$ is not in $ConvexHull\,(\mathcal{X})$. So, both of the inequalities are satisfied for $\mathbf{y} = \mathbf{z}$ and $y_s = -(\mathbf{z}-\mathbf{p})^T\mathbf{z}$. Thus, if $\mathbf{p} \notin ConvexHull\,(\mathcal{X})$, then $\exists \mathbf{x} \in ConvexHull\,(\mathcal{X})$ and $\beta \in \mathbb{R}$ such that $\mathbf{p}^T(\mathbf{x}-\mathbf{p}) < \beta$ and $\hat{\mathbf{x}}^T(\mathbf{x}-\mathbf{p}) \geq \beta$, $\forall \hat{\mathbf{x}} \in \mathcal{X}$. □

Local problems

The last thing we should show, is that we can solve the local problems 2.3, 2.4. They are of the form,

$$\begin{aligned} \underset{a_j, \forall \hat{\boldsymbol{\phi}}(\mathbf{x_j}) \in \mathcal{C}}{\text{minimize}} \quad & \left\| \left(\sum_{\hat{\boldsymbol{\phi}}(\mathbf{x_j}) \in \mathcal{C}} a_j \hat{\boldsymbol{\phi}}(\mathbf{x_j}) \right) \right\|^2 \\ s.t. \quad & 0 \leq a_j, \qquad\qquad \forall \mathbf{x_j} \in \mathcal{C} \\ & \sum_{\hat{\boldsymbol{\phi}}(\mathbf{x_j}) \in \mathcal{C}} a_j = 1 \end{aligned} \tag{4.4}$$

which is equivalent to

$$\begin{aligned} \underset{a_j, \forall \hat{\boldsymbol{\phi}}(\mathbf{x_j}) \in \mathcal{C}}{\text{minimize}} \quad & \sum_{\hat{\boldsymbol{\phi}}(\mathbf{x_i}) \in \mathcal{C}} \sum_{\hat{\boldsymbol{\phi}}(\mathbf{x_j}) \in \mathcal{C}} a_j \hat{\boldsymbol{\phi}}(\mathbf{x_j})^T \hat{\boldsymbol{\phi}}(\mathbf{x_i}) a_i = \sum_{\hat{\boldsymbol{\phi}}(\mathbf{x_i}) \in \mathcal{C}} \sum_{\hat{\boldsymbol{\phi}}(\mathbf{x_j}) \in \mathcal{C}} a_j \hat{k}(\mathbf{x_j}, \mathbf{x_i}) a_i \\ s.t. \quad & 0 \leq a_j, \qquad\qquad \forall \mathbf{x_j} \in \mathcal{C} \\ & \sum_{\hat{\boldsymbol{\phi}}(\mathbf{x_j}) \in \mathcal{C}} a_j = 1 \end{aligned} \tag{4.5}$$

which is a quadratic program.

4.4 Answering queries

Queries to the L_2-SVM can now be answered by any node, as the global vector and the minimum are the only information needed.

Chapter 5

Experimental Study

We simulated training a one class L_2 SVM with an RBF kernel, on a chessboard dataset where each point is uniformly at random selected in $\mathbb{R}^{10}$ such that projecting it into any of the 5 pairs of coordinates (1-2, 3-4, 5-6, 7-8, 9-10) will give a point belonging in a white square of a chessboard. Each point is randomly assigned into one of the sites and the model start complete untrained, that is, the first point is used as initialization of the model.

The global vector was represented as a convex combination, thus any component with a non-zero coefficient in this convex combination was known to every site. Also, every time the convex combination was updated only the new components where transmitted.

In figure 5.1 the number of iterations before convergence of the update method 1 is plotted. Observe that in most cases the number of iterations is pretty small and that as the model gets trained, update is triggered less often.

In figure 5.2 the number of vector broadcasts needed for each addition is plotted. Observe that this is also small enough. This partially happens because in the settings used the support vectors are a significant part of the population of data vectors, see figure 5.3.

In figure 5.4 the moving windows average (of length 100) of vector broadcasts needed for each addition is plotted. Observe that it is pretty small for this dataset.

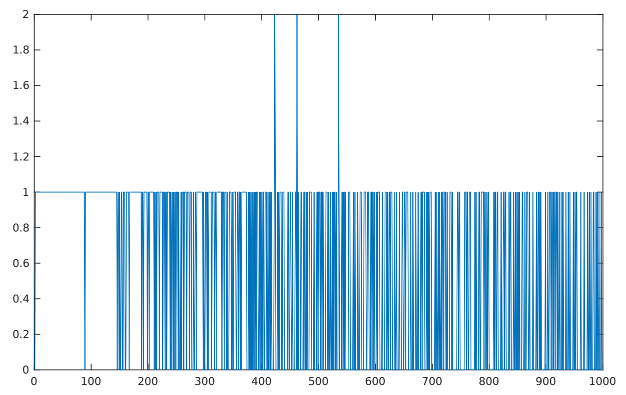


Figure 5.1: Number of iterations needed before the first update method converges after each point addition. Zero means that monitoring decided that no update was needed, while one means that the node where the violator arrived solved the local problem, broadcasted the updated global vector and the algorithm converged. Number of sites is 10.

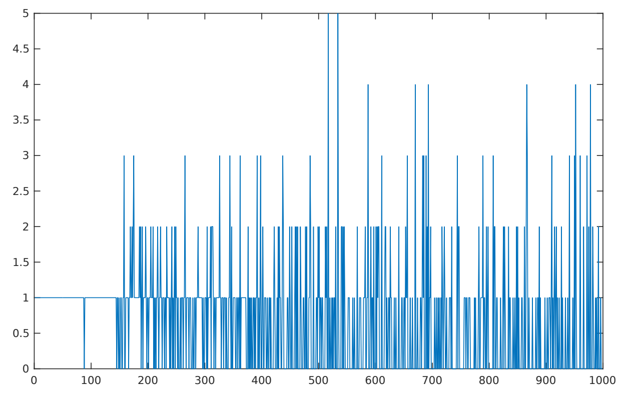


Figure 5.2: Number of vector broadcasts (without counting broadcasting the coefficients) needed before the first update method converges after each addition. Zero means that monitoring decided that no update was needed. Number of sites is 10.

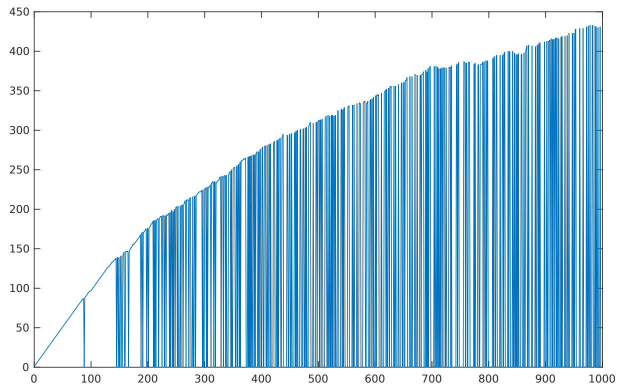


Figure 5.3: Number of support vector after each update. Zero means that monitoring decided that no update was needed. Number of sites is 10.

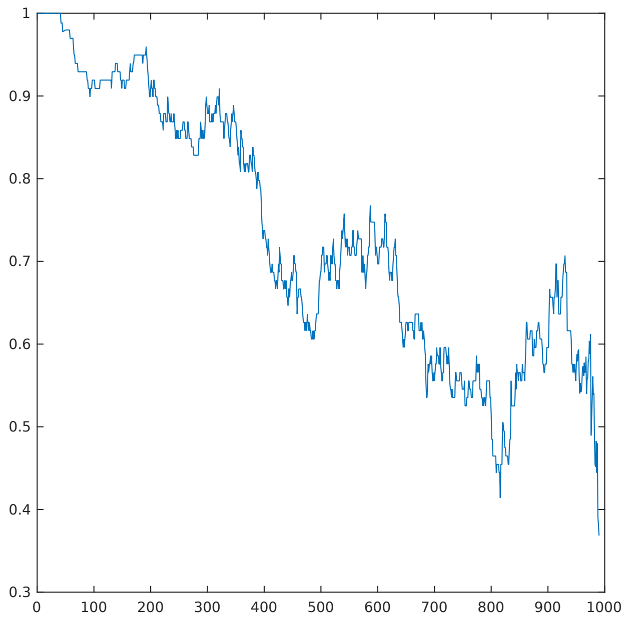


Figure 5.4: Moving window average of length 100 over the number of transmitted vectors after each update. Zero means that monitoring decided that no update was needed. Number of sites is 10.

Chapter 6

Conclusion

Concluding, we proposed communication efficient methods for tracking the minimum of any differentiable convex function f over the convex hull of a fully dynamic distributed set of points. Our methods for the exact minimization case have been extended to the approximate case, relaxing the constraints that trigger the updates.

We also proposed a novel geometric method with only one communication stage for updating our model, when an update is needed.

Finally, we showed that our methods can be applied to train L_2 Support Vector Machines, even in the case of an explicit feature space, making our methods applicable for training distributed online kernelized L_2 SVMs.

Chapter 7

Bibliography

[1] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[2] V. Franc and V. Hlaváč. An iterative algorithm learning the maximal margin classifier. *Pattern Recognition*, 36(9):1985–1996, 2003.

[3] M. Garofalakis, D. Keren, and V. Samoladas. Sketch-based geometric monitoring of distributed stream queries. *Proceedings of the VLDB Endowment*, 6(10):937–948, 2013.

[4] B. Kozinec. Recurrent algorithm separating convex hulls of two sets. *Learning algorithms in pattern recognition*, pages 43–50, 1973.

[5] A. Lazerson, I. Sharfman, D. Keren, A. Schuster, M. Garofalakis, and V. Samoladas. Monitoring distributed streams using convex decompositions. *Proc. VLDB Endow.*, 8(5):545–556, Jan. 2015.

[6] G. Loosli and S. Canu. Comments on the“core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 8(Feb):291–301, 2007.

[7] M. E. Mavroforakis, S. Theodoridis, et al. A geometric approach to support vector machine (svm) classification. *IEEE transactions on neural networks*, 17(3):671–682, 2006.

[8] J. Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[9] T. Poggio and G. Cauwenberghs. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, 13:409, 2001.

[10] P. Rai, H. Daumé III, and S. Venkatasubramanian. Streamed learning: one-pass svms. *arXiv preprint arXiv:0908.0572*, 2009.

[11] M. Schlesinger, V. Kalmykov, and A. Suchorukov. Sravnitelnyj analiz algoritmov sinteza linejnogo reshajushchego pravila dlja proverki slozhnych gipotez (comparative analysis of algorithms synthesising linear decision rule for analysis of complex hypotheses). *Automatika*, 1(1):3–9, 1981.

[12] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Transactions on Database Systems (TODS)*, 32(4):23, 2007.

[13] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. In *Journal of Machine Learning Research*, pages 363–392, 2005.

[14] K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui, and E. Y. Chang. Parallelizing support vector machines on distributed computers. In *Advances in Neural Information Processing Systems*, pages 257–264, 2008.