STOCHASTIC SUBGRADIENT APPROACH FOR SOLVING LINEAR SUPPORT VECTOR MACHINES – AN OVERVIEW

Jan Rupnik
Department of Knowledge Technologies
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel: +386 1 4773934; fax: +386 1 4773315
e-mail: jan.rupnik@ijs.si

ABSTRACT

This paper is an overview of a recent approach for solving linear support vector machines (SVMs), the PEGASOS algorithm. The algorithm is based on a technique called the stochastic subgradient descent and employs it for solving the optimization problem posed by the soft margin SVM - a very popular classifier. We briefly introduce the SVM problem and one of the widely used solvers, SVM light, then describe the PEGASOS algorithm and present some experiments. We conclude that the algorithm efficiently discovers suboptimal solutions to large scale problems within a matter of seconds.

1 INTRODUCTION

Since the nineties Support Vector Machines (SVMs) have become one of the most popular supervised machine learning methods used for regression and classification problems [3]. Although SVMs can be used to find nonlinear classification or regression functions, this paper focuses on the case of linear classification SVMs. Training the algorithms for nonlinear SVMs scales super-linearly in the number of training examples and the algorithms can handle tens of thousands of data points. In recent years it has been shown that the linear SVMs on the other hand can be trained in linear time with respect to the number of training examples. These new approaches can deal with millions of training points. The purpose of this paper is to compare one of the most popular SVM implementations, SVM-light [2] and a recent solution, Pegasos [1] based on stochastic subgradient optimization.

2 SUPPORT VECTOR MACHINE

This chapter is composed of two subchapters. The first one will introduce the basic intuitions behind the support vector machines and some formal problem definitions and the second one will introduce the problem in the dual representation.

2.1 Intuitions and problem formulation

In a two class classification task we are presented with a training sample, *S*, of *m* labelled data points : $S = \{(x_i, y_i)\}_{i=1:m}$, where $x_i \in R^n$ are the training vectors and $y_i \in \{-1,1\}$ are their corresponding labels. The task is to find the linear functional $f\colon R^n \rightarrow R$, $f(x) = \langle w,x\rangle + b$ that satisfies a certain criterion. We denoted the inner product by $\langle .,. \rangle$, and we will also use the notation $w'x := \langle w,x\rangle$, where w' denotes vector *w* transposed. Vector *w* is commonly referred to as the normal of the classification hyperplane and *b* is referred to as bias. Let us first consider the case where data is linearly separable (there exists an *f* that perfectly classifies all the examples from the set *S*), a case also known as the hard margin SVM. SVM optimization criterion is based on finding the *f* that separates the data best in the sense of the highest minimal distance between the hyperplane and the data points. Figure 1 shows two possible hyperplanes (a blue and a black one). They both perfectly separate the data, but the margin (or the minimum distance) between data points and the black line is much higher than the margin of the blue line. The black hyperplane is more likely to perform better on new instances than the blue line. By using geometry we can show that the margin of a hyperplane *f* is proportional to $1/\langle w,w\rangle$. This can be formally stated as the following constrained optimization problem:

- Hard margin SVM

$$\text{Minimize: } w'w$$
$$\text{Subject to: } y_i (\langle w, x_i\rangle - b) > 1,\ i=1,\ldots,m$$

The constraints are equivalent to $f(x_i) = (\langle w, x_i\rangle - b) > 1$ if $y_i = 1$ and $f(x_i) = (\langle w, x_i\rangle - b) < -1$ if $y_i = -1$, for all $(x_i, y_i) \in S$, which are the conditions for correct classification of the training sample. All the constraints are linear functions of *w* and *b* and the objective is a quadratic function of *w*. Problems of this form are known as quadratic programs. Since data is usually noisy it is often the case that a separating hyperplane does not exist. In such cases we search for a hyperplane that misclassifies a few points but has a high margin with respect to the correctly classified points. This case is known as the soft margin SVM. The task is to find a hyperplane with a good trade-off between the training loss (high training loss usually leads to poor performance on new instances but small training loss can lead to overfitting) and the margin (large margins lead to good generalization ability, whereas small margins can lead to overfitting). The margin plays the role of regularizing the loss function and controls the complexity of the classification model. One part of the training task is to find a good trade-off parameter between the margin and the loss and this is usually accomplished by cross-validation. There are two equivalent formulations of the soft margin SVM optimization problem: regularized hinge loss formulation and slack variable formulation (softening the hard margin constraints). Here follows the latter formulation:

- Soft margin SVM – slack variables

$$\text{Minimize: } w'w + C\Sigma_i\xi_i$$
$$\text{Subject to: } y_i(<w, x_i> -b) \geq 1- \xi_i, \text{ for all } i=1,\dots,m$$

The ξ_i variables are called slack variables and they allow the w and b variables to violate the hard margin constraints and by adding the sum $\Sigma_i\xi_i$ to the objective we penalize those violations. The parameter C controls the trade-off between the margin size and the amount of data that lies inside the margin or is even misclassified. This problem is also a quadratic programming problem:

- Soft margin SVM – regularized hinge loss

$$\text{Minimize: } w'w + C\Sigma(1 - y_i(w'x_i - b))_+$$

In the equation above $()_+$ represents the function $(x)_+ := max\{0,x\}$. Notice that this problem is an unconstrained optimization problem and that by contrast to the slack formulation it is not differentiable, since $()_+$ is not smooth.

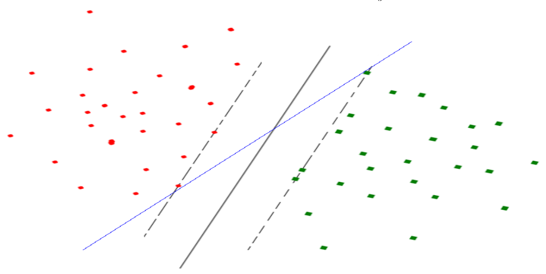

Figure 1: *Separating hyperplanes*

2.2 Optimization problem: dual

The formulations presented so far are searching for a w of the same dimension as training vectors x_i. We call such formulations primal formulations. By writing down the Lagrangian, analysing the Karush-Kuhn-Tucker (KKT) conditions [4] and some algebraic manipulation we can express the solution w as a m-dimensional (size of the training set) vector in terms of dual variables. This can be beneficial if the number of features is much higher than the number of training examples and these formulations can easily be adopted to handle nonlinear optimizations (this is known as the kernel trick, see [5]). We will omit the derivations and present the dual soft margin SVM optimization problem.

$$\text{Minimize: } \Sigma_i\alpha_i\alpha_j y_i y_j <x_i,x_j> - \Sigma_i\alpha_i$$
$$\text{Subjetc to: } 0 \leq \alpha_i \leq C,\ i= 1,\dots,m$$
$$\Sigma_i y_i\alpha_i = 0$$

We notice that this is again a quadratic problem with particularly simple linear constraints, box constraints. Vector w can be expressed as $w = \Sigma_i\alpha_i y_i x_i$. The solution is written as a linear combination of those training vectors whose corresponding α_i coefficients are non zero, and these vectors are called support vectors. One of the consequences of KKT theory is that the solution would remain the same even if we remove all but the support vectors from the training set S.

3 SOLVING THE OPTIMIZATION PROBLEM

This chapter will introduce two approaches to solving the SVM optimization problem. The first one is based on an active set method of the dual soft margin SVM and the other one is the main focus of this article – the stochastic subgradient descent optimization of the regularized hinge loss formulation of soft margin SVM.

3.1 Active set dual optimization: svm-light

One of the main problems with directly optimizing the dual soft margin SVM are the super-linear convergence rate and high memory requirement (quadratic in the number of training examples since the matrix of the quadratic objective function has m rows and m columns). We have mentioned that the solution of the problem is completely determined by the set of support vectors (or their corresponding α variables). Active methods try to identify that set by starting with a random set working set and then iteratively keep adding or removing variables from that set. In this way that they decompose the large problem into a series of smaller, tractable, quadratic problems, by optimizing only over the variables in the working set and fixing all the other variables. After that step the method tries to find a better working set. This can be posed as an optimization which can be efficiently solved. The solutions found by SVM-light are highly accurate.

3.2 Stochastic subgradient descent primal optimization: PEGASOS

Pegasos algorithm optimizes the primal view regularized hinge loss formulation instead of the quadratic program. It is based on a search method called stochastic subgradient descent. The method iteratively searches for the optimum of a function. It starts with a random starting point, finds the best search direction, computes the new point and repeats these steps until it converges. It uses subgradient descent, since the gradient of the hinge loss function does not exist and it uses the stochastic version, because computing the gradient of the optimization function can be expensive when the training set is large. We will first define the subgradient of a function and present the subgradient of the regularized hinge loss function.

3.2.1 Stochastic subgradient

Vector v is a subgradient of function f at a point x_0, if:

$$f(x) - f(x_0) \geq v'(x - x_0)$$

for every x in some open neighbourhood of x_0.
The authors of Pegasus optimize the following function (slightly different trade-off constant and ignoring the bias coefficient)

$$f(w) = \lambda/2\ w'w + 1/m\ \Sigma_i\ (1 - y_i\ w'x_i)_+$$

Subgradient of each of the summands in the above sum is equal to 0 if $y_i w'x_i > 1$ and equal to $-y_i x_i$ otherwise (non zero loss case). Subgradient of the full expression is thus equal to:

$$\partial f = \lambda\ w - 1/m\ \Sigma_{i+}\ y_i x_{i,}$$

where Σ_{i+} denotes the sum over the indices with nonzero loss. Computing a stochastic subgradient is very similar, the only difference is that we create a subsample of the training points, A, and compute the subgradient of an approximated function:

$$f_A(w) = \lambda/2\ w'w + 1/k\ \Sigma_{i \in A}\ (1 - y_i\ w'x_i)_+,$$

where k is the size of the subsample A.

3.2.2 The algorithm

The algorithm has an additional step besides the subgradient descent step in each iteration and that is projection onto a ball with diameter $1/\sqrt{\lambda}$. It can be proven that the optimal solution always lies in that ball and if the current iterate moves out of that ball, projecting it brings it only closer to the optimal solution. The step size, η, is initialized as $1/\lambda$ and keeps decreasing with the number of iterations. In the t-th iteration we set it to $\eta_t = 1/(\lambda t)$. We denote the total number of iterations as T, and the size of the subsample in each step as k, which we chose manually .

```
Algorithm :
INPUT:   S, $\lambda$, T, k
INITIALIZE:   Choose $w_1$ randomly so that $||w_1|| \leq 1/\sqrt{\lambda}$
FOR   t = 1, 2, ..., T
        Choose $A_t$, a random subset of $S$ of size $k$
        Set $A' = \{(x,y) \in A_t : yw_t', x < 1\}$
        Set $\eta_t = 1/(\lambda t)$
        Set $w_{t+1/2} = (1 - \eta_t\lambda)w_t + \eta_t/k \sum_{(x,y) \in A'} yx$
        Set $w_{t+1} = \min\{1, 1/(\sqrt{\lambda}||w_{t+1/2}||)\}\ w_{t+1/2}$
OUTPUT:   $w_{T+1}$
```

3.2.3 Remark

One of the reasons why such an old technique has not been successfully applied to this problem until a few years ago is that the researchers used slower, less aggressive, learning rates. The Pegasos fast learning rate and the fact that it provably converges are the key to the success of the algorithm.

4 EXPERIMENTS

We will first describe the data set and proceed with analyzing several properties of the Pegasos algorithm.

4.1 Data

The experiments were conducted on the Reuters RCV2 corpus [6], which consists of 804.414 news documents. The documents are represented as 47.236 dimensional sparse vectors (bag of words document representation), with the sparsity 0,16%. Each document in the collection is assigned to a category from a hierarchy of categories. The four major categories are: CCAT, GCAT, ECAT, MCAT. We focused on testing the algorithms on the category CCAT, which consists of 381.327 positive documents (the rest are negative), and is very balanced.

4.2 Robustness to random initializations

We first investigated the robustness of Pegasos solutions to different choices of initial random starting vectors w. Figure 2 depicts several curves corresponding to different starting points. Each curve represents the value of the objective function as the iterations increase. One can notice that the behaviour of the Pegasos algorithm is more or less independent of the choice of initial solution.

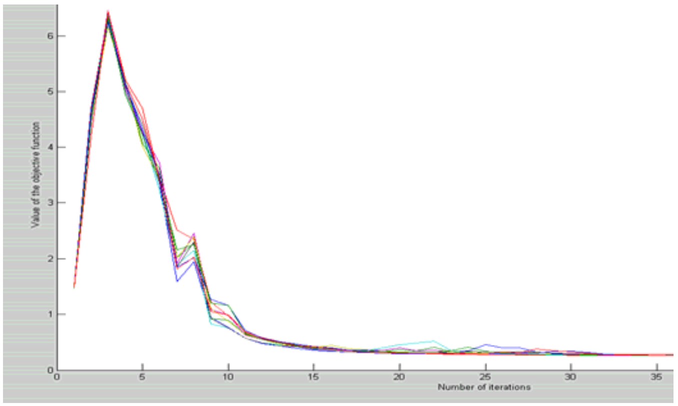

Figure 2: *Several runs with different initial vectors*

4.3 Convergence of pegasos

We evaluated the convergence speed of the Pegasos algorithm. Optimum objective value was computed by SVM-light, which took roughly four hours of CPU time. Computing the 200 iterations of the Pegasos algorithm took 9.2 seconds of CPU time and the the objective value was 0.3% close to the optimum. Pegasos needed 560 iterations to get within 0.1% error of the true optimum. This experiment demonstrates the rapid convergence of Pegasos towards approximate solutions. The k parameter was set to

8.000 and the λ parameter was set to 0.0001, as this value was observed to be optimal for the Reuters corpus and the category CCAT.

4.4 Testing the classification accuracy

The Reuters data set was split into the first 700.000 documents for training and the rest 104.414 documents for testing (the original order was preserved). We investigated how the classification error on the test set decreases with the number of iterations of the Pegasos algorithm. Figure 3 depicts the error on the test set with the number of iterations of the algorithm. We can see that the algorithm achieves 5.8% classification error within 50 iterations. SVM-light achieved the error of 5.6%.

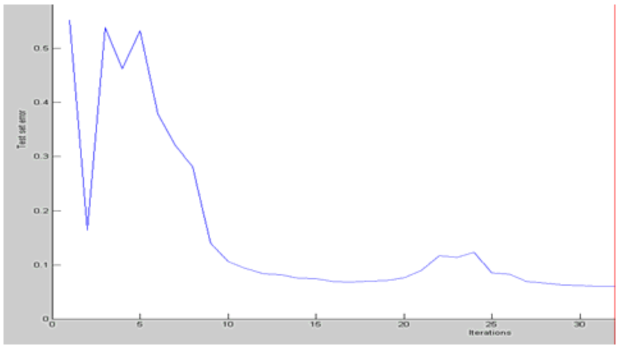

Figure 3: *The decrease of test error with the number of iterations*

4.5 Parameters k and T

One of the experiments involved examining the influence of different values of *k* and *T* parameters on the objective value. Figure 4 depicts the relationship between *k* and *T* when their product is fixed. The first thing to notice is that curves with larger *kT* are always dominated by curves with smaller *kT* (convergence). All three settings for different values of *kT* yielded similar curves, and we can notice that seting *k* too small can slow down convergence rate. This result is unexpected, since the authors of the Pegasos algorithm experimentally showed that the value of k is not important as long as the value of *kT* is fixed, although they left deeper analysis for future work. Note that they did not use the same data set for their experiments. They recommended setting the value of *k* to 1, although our experiments imply setting *k* to a higher value. One possible reason for slower convergence with low values of k, for example 1, is that the algorithm converges to solutions with low number (5 to 10 percent) of misclassified training examples very rapidly, even though the margin is still suboptimal. This means that when we sample a training point in each of the following iterations it is very likely to be correctly classified, so the value of *w* would not change in that iteration. The value of *t* on the other hand would still increase and consequently the step size will decrease too quickly. One possible way to prevent that is not to increase *t* in those cases. This can improve convergence rate although the convergence remains slower.

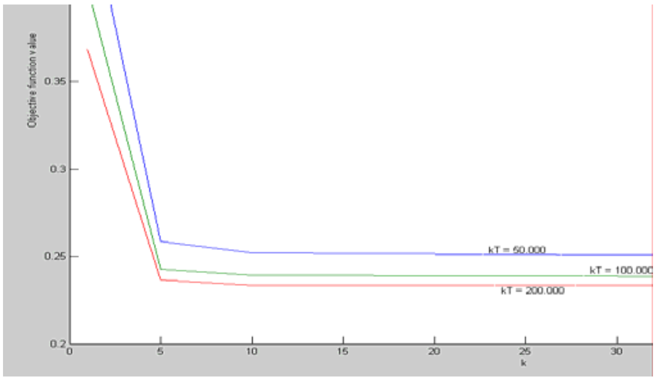

Figure 4: *Parameters k and T. The horizontal scale represents different values of k as the product kT is fixed.*

5 CONCLUSIONS

We have presented an examination of Pegasos - an efficient algorithm for solving the linear SVM optimization problem. The approach is based on stochastic subgradient descent and has strong convergence guarantees. The approach is easy to implement and converges extremely fast to a suboptimal solution. We have also demonstrated that high precision optimization of the objective function on the training set can be unnecessary, since optimal classification error on a test set can be achieved much sooner. SVM-light algorithm, a high precission SVM solver, was chosen as a baseline for comparison.

6 ACKNOWLEDGMENTS

This work was supported by the Slovenian Research Agency and the IST Programme of the EC under SMART (IST-033917) and PASCAL2 (IST-NoE-216886).

References

[1] S. Shalev-Shwartz, Y. Singer, N. Srebro, *Pegasos: Primal Estimated sub-GrAdient SOlver for SVM*, pp. 807-814., 2007

[2] T. Joachims, *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999

[3] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995

[4] H. W. Kuhn, A. W. Tucker, *Nonlinear programming*. Proceedings of 2nd Berkeley Symposium: 481-492, Berkeley: University of California Press, 1951

[5] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[6] D. Lewis, Y. Yang, T. Rose, F. Li, *Rcv1: A new benchmark collection for text categorization research*. Journal of Machine Learning Research (JMLR), 5:361–397, 2004.