Stochastic Subgradient Approach for Solving Linear Support Vector Machines

Jan Rupnik

Jozef Stefan Institute

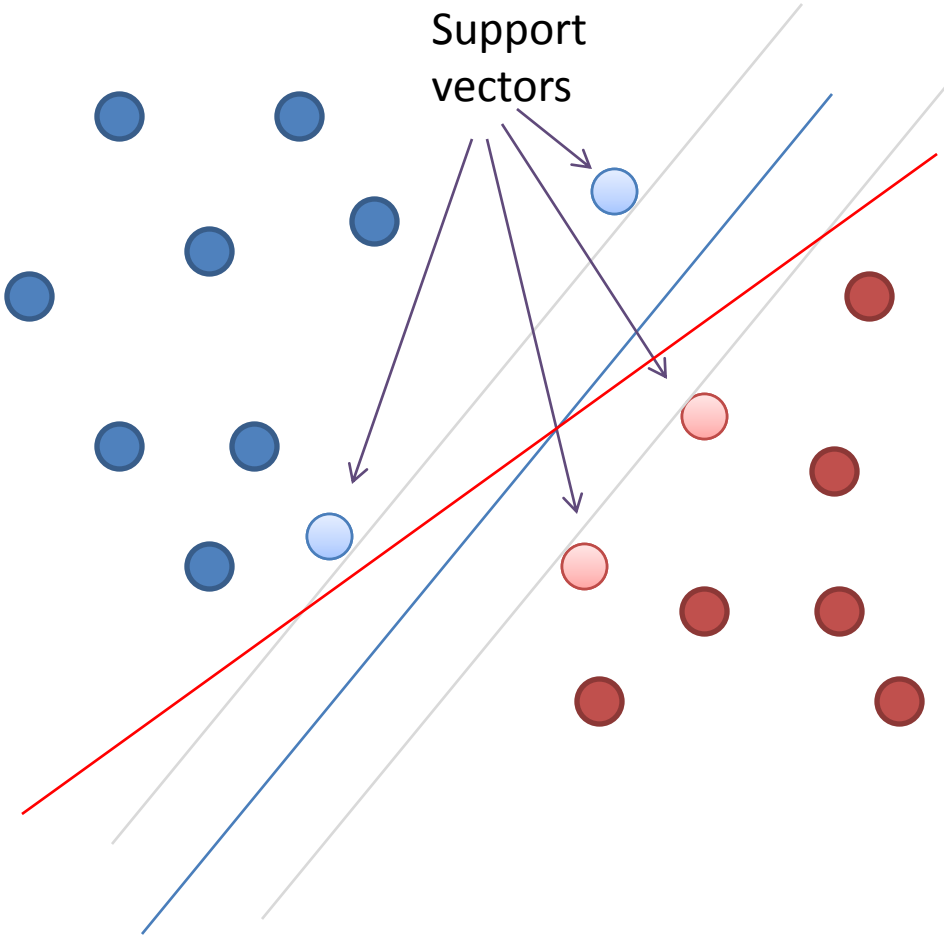
Outline

- Introduction
- Support Vector Machines
- Stochastic Subgradient Descent SVM - Pegasos
- Experiments

Introduction

- Support Vector Machines (SVMs) have become one of the most popular classification tools in the last decade
- Straightforward implementations could not handle large sets of training examples
- Recently methods for solving SVMs arose with linear computational complexity
- Pegasos: primal estimated subgradient approach for solving SVMs

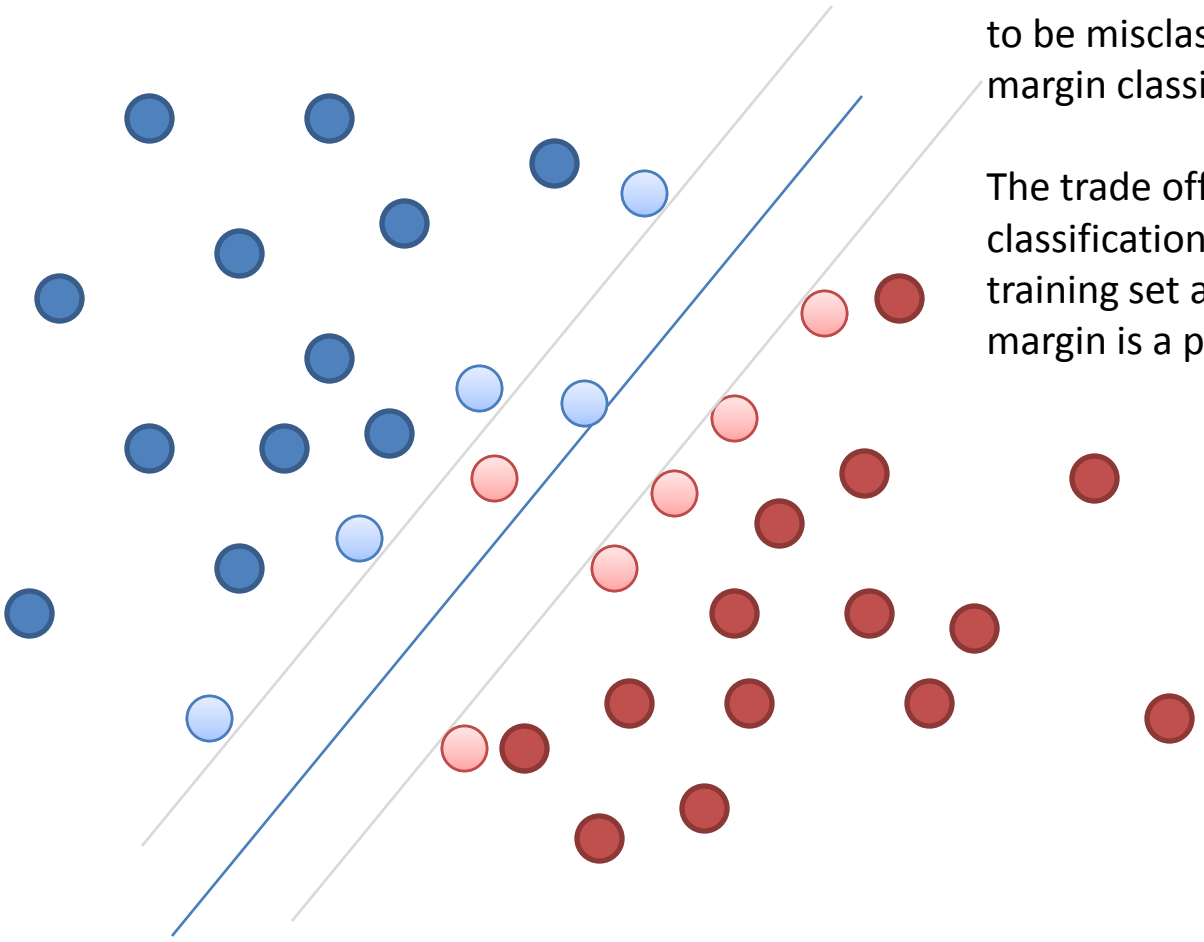
Hard Margin SVM



Many possible hyperplanes perfectly separate the two classes (e.g. red line).

Only one hyperplane with the maximum margin (blue line).

Soft Margin SVM



Allow a small number of examples to be misclassified to find a large margin classifier.

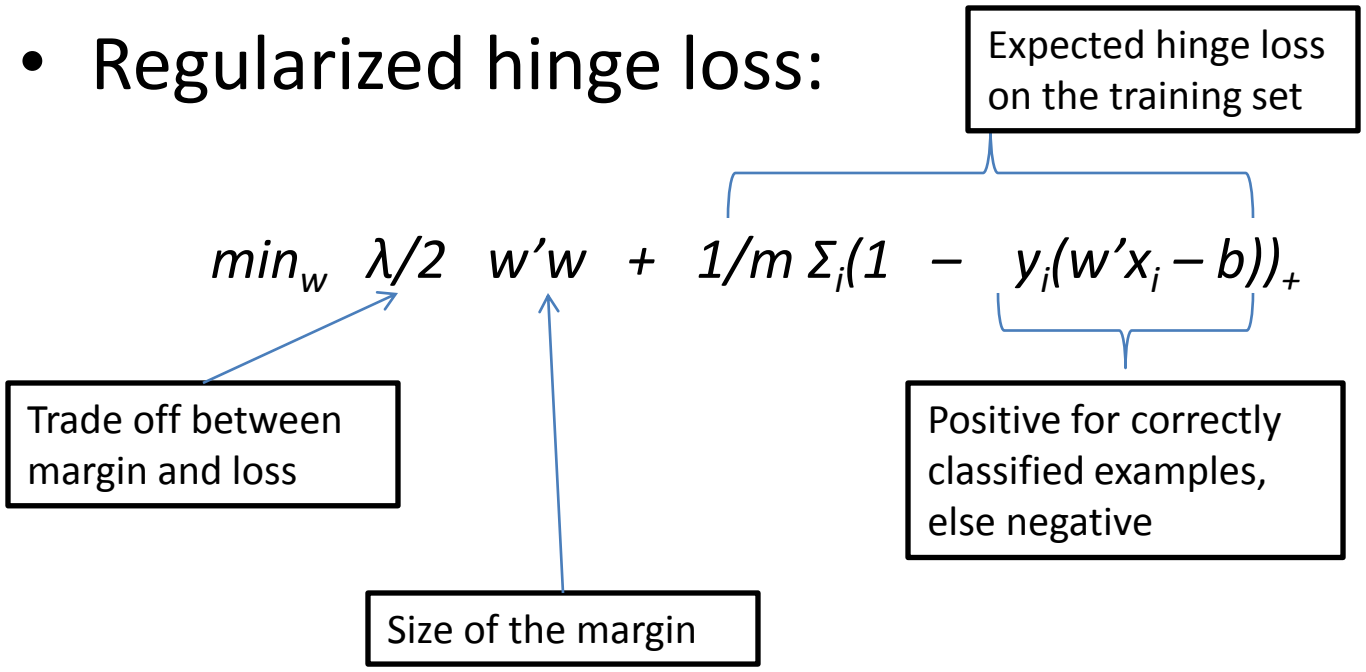
The trade off between the classification accuracy on the training set and the size of the margin is a parameter for the SVM

Problem setting

- Let $S = \{(x_1,y_1),\ldots, (x_m, y_m)\}$ be the set of input-output pairs, where $x_i \in R^N$ and $y_i \in \{-1,1\}$.
- Find the hyperplane with the normal vector $w \in R^N$ and offset $b \in R$ that has good classification accuracy on the training set S and has a large margin.
- Classify a new example x as $sign(w'x - b)$

Optimization problem

- Regularized hinge loss:



$(1 - z)_+ := \max\{0, 1\text{-}z\}$ (hinge loss)

First summand is a quadratic function, the sum is a piecewise linear function. The whole objective: piecewise quadratic.

Perceptron

- We ignore the offset parameter *b* from now on (*b = 0*)
- Regularized Hinge Loss (SVM):

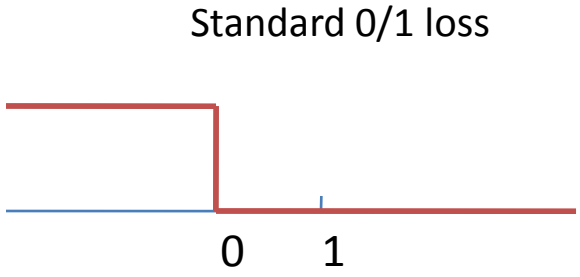
  $min_w \ \lambda/2 \ w'w \ + \ 1/m \ \Sigma_i(1 \ - \ y_i(w'x_i))_+$

- Perceptron

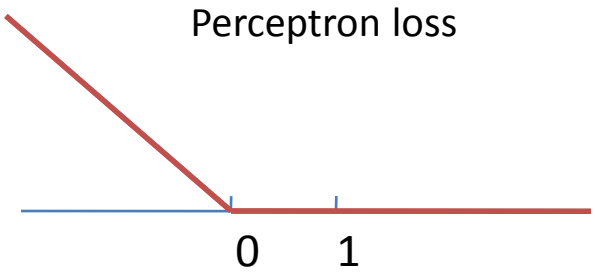
  $min_w \ 1/m \ \Sigma_i(-y_i(w'x_i))_+$

Loss functions

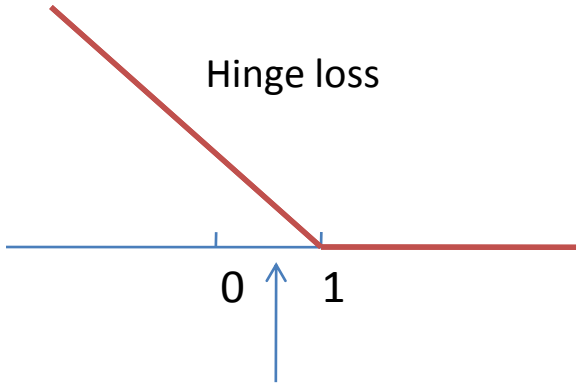
Penalizes all incorrectly classified examples with the same amount



Penalizes incorrectly classified examples x proportionally to the size of |w'x|

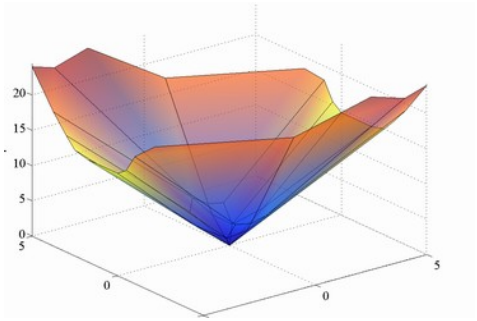


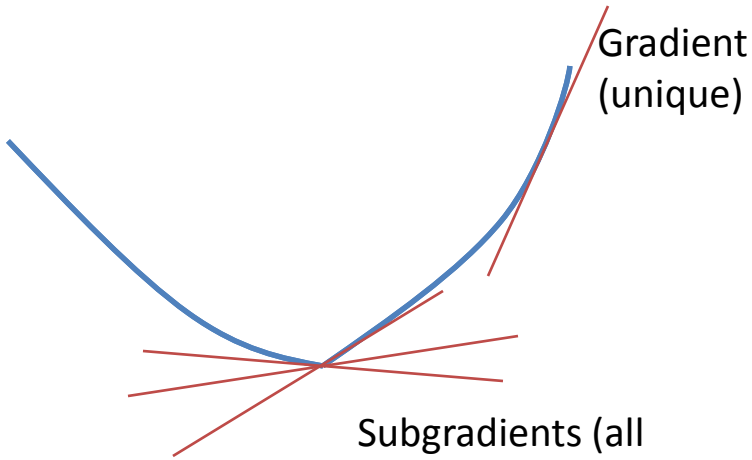
Penalizes incorrectly classified examples and correctly classified examples that lie within the margin



Stochastic Subgradient Descent

- Gradient descent optimization in perceptron (smooth objective)
- Subgradient descent in pegasos (non differentiable objective)





Stochastic Subgradient

- Subgradient in perceptron: $1/m\ \Sigma -y_i x_i$ for all misclassified examples
- Subgradient in SVM: $\lambda w + 1/m\ \Sigma_i (1 - y_i x_i)$ for all misclassified examples
- For every point *w* the subgradient is a function of the training sample S. We can estimate it from a smaller random subset of S of size *k, A,* (stochastic part) and speed up computations.
- Stochastic subgradient in SVM:
  $\lambda w + 1/k\ \Sigma_{(x,y)\ \epsilon\ A\ \&\&\ \text{misclassified}} (1 - yx)$

Pegasos – the algorithm

Input: S, λ, T, k

Initialize: Choose w_1 randomly so that $\|w_1\| \leq \frac{1}{\sqrt{\lambda}}$

For $t = 1, 2, \ldots, T$

Choose A_t, a random subset of S of size k

Set $B_t = \{(x, y) \in A : y w_t' x < 1\}$

Set $\mu_t = \frac{1}{\lambda t}$

Set $w_{t+\frac{1}{2}} = (1 - \mu_t \lambda) w_t + \frac{\mu_t}{k} \sum_{(x,y) \in B_t} yx$

Set $w_{t+1} = \min\left\{1, \frac{1}{\left(\sqrt{\lambda} \left\| w_{t+\frac{1}{2}} \right\|\right)}\right\} w_{t+\frac{1}{2}}$

Output w_{T+1}

Subsample

Learning rate

Subgradient step

Subgradient is zero on other training points

Projection into a ball (rescaling)

What's new in pegasos?

- Sub-gradient descent technique 50 years old
- Soft Margin SVM 14 years old
- Typically the gradient descent methods suffer from slow convergence
- Authors of Pegasos proved that aggressive decrease in learning rate μ_t still leads to convergence.
  - Previous works: μ_t = 1/(λ√t))
  - pegasos: μ_t = 1/ (λt)
- Proved that the solution always lies in a ball of radius 1/√λ

SVM Light

- A popular SVM solver with superlinear computational complexity
- Solves a large quadratic program
- Solution can be expressed in terms a small subset of training vectors, called support vectors
- Active set method to find the support vectors
- Solve a series of smaller quadratic problems
- **Highly accurate** solutions
- Algorithm and implementation by Thorsten Joachims

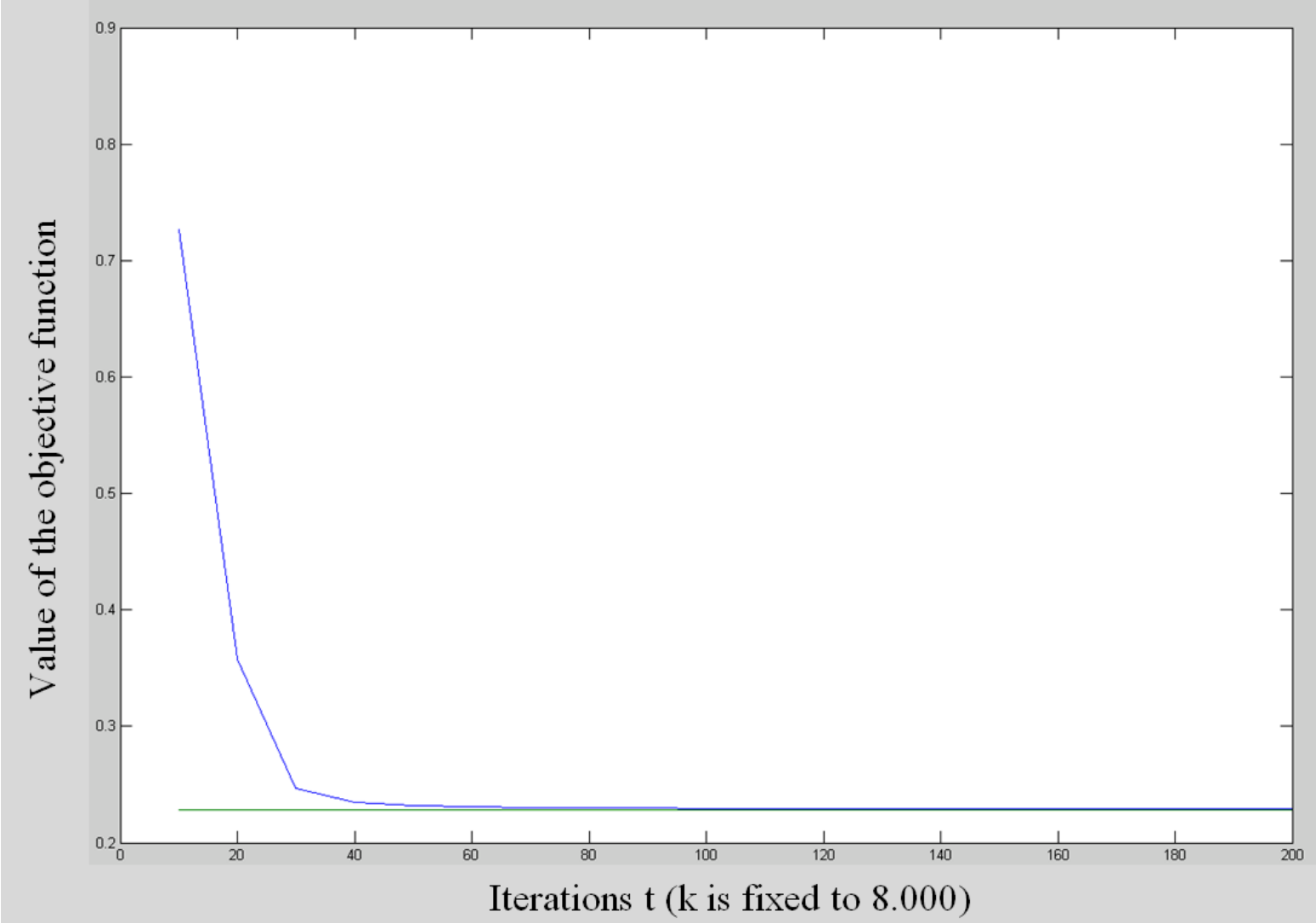
T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999

Experiments

- Data
  - Reuters RCV2
  - Roughly 800.000 news articles
  - Already preprocessed to bag of word vectors, publicly available
  - Number of features roughly 50.000
  - Sparse vectors
  - category **CCAT**, which consists of 381.327 news

Quick convergence to suboptimal solutions

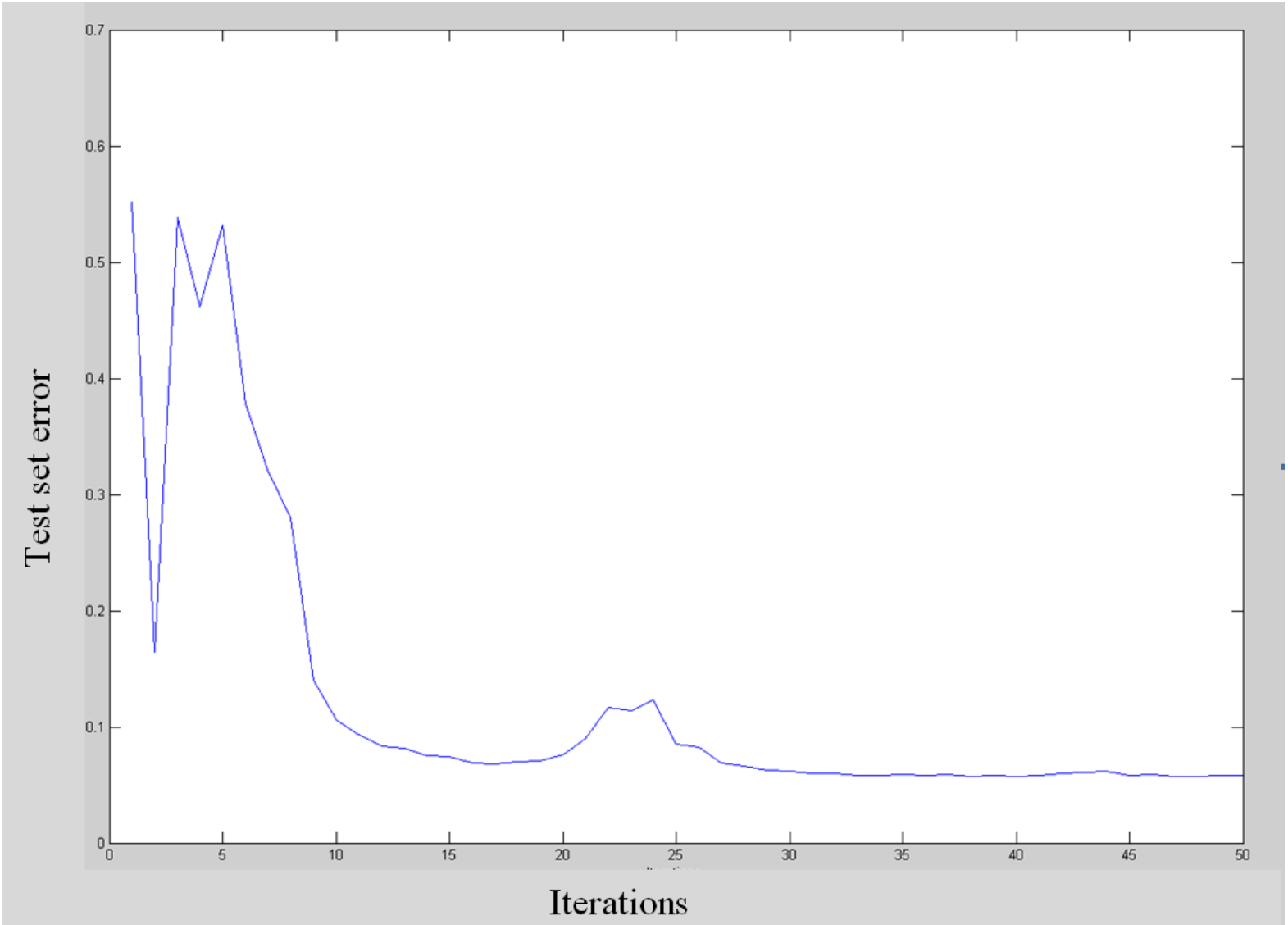
- 200 iterations took **9.2 CPU seconds**, the objective value was 0.3% higher than the optimal solution
- 560 iterations to get a 0.1% accurate solution
- SVM Light takes roughly **4 hours of CPU time**



Test set error

- Optimizing the objective value to a high precission is often not necessary

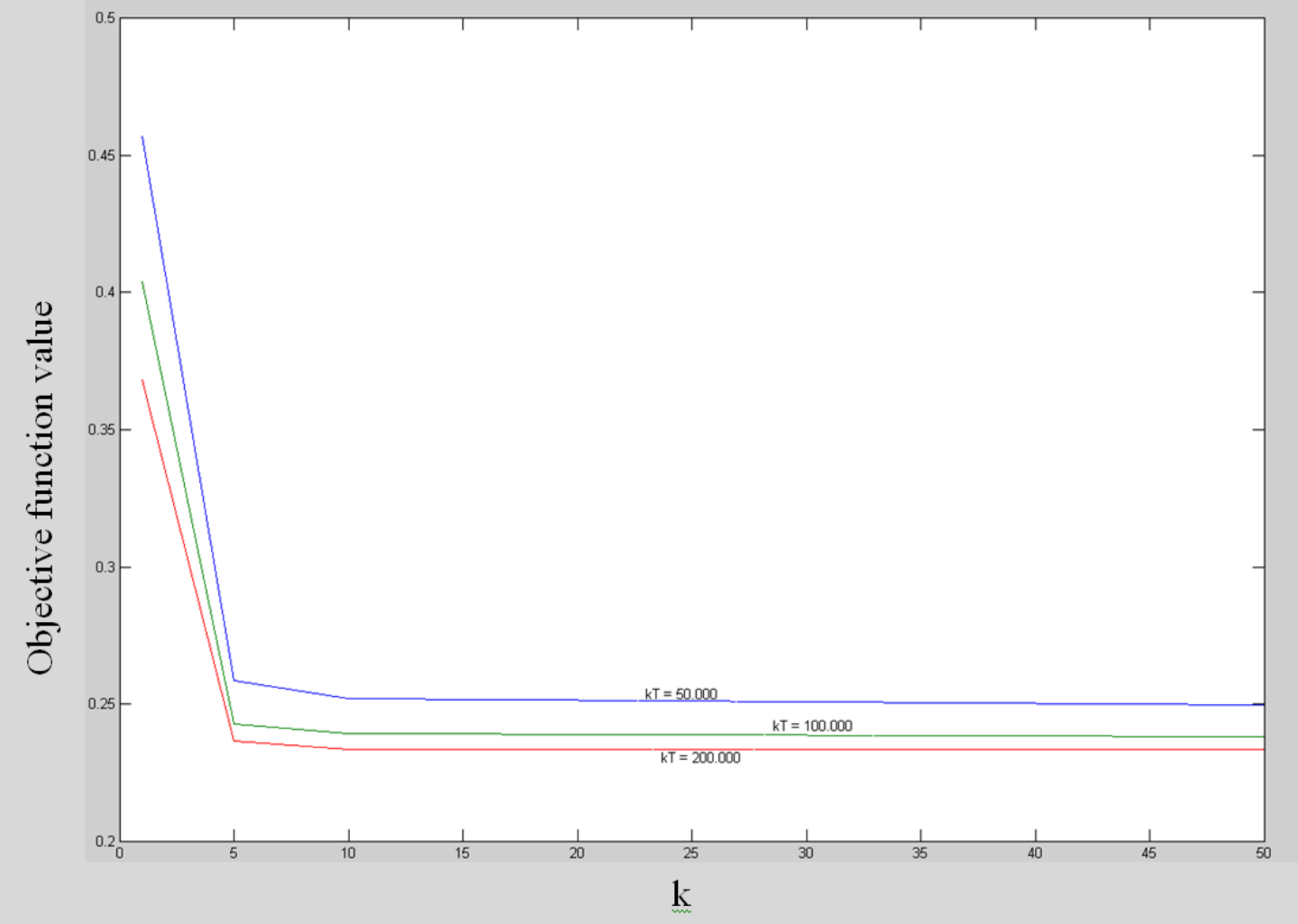
- The lowest error on the test set is achieved much earlier



Parameters k and T

- The product of kT determines how close to the optimal value we get

- If kT is fixed the k does not play a significant role



Conclusions and final notes

- Pegasos – one of the most efficient suboptimal SVM solvers
- Suboptimal solutions often generalize to new examples well
- Can take advantage of sparsity
- Linear solver
- Nonlinear extensions have been proposed, but they suffer from slower convergence

Thank you!