Sol – Stochastic Learning Toolkit

Technical Report

Sascha Fendrich
Department of Computational Linguistics
Heidelberg University
Heidelberg, Germany
fendrich@cl.uni-heidelberg.de

May 2012

Sol is an open source machine learning toolkit implemented in C++. It is based on stochastic gradient descent in a regularized risk minimization framework. At the current state it implements binary, multi-class, and multi-label classification; other classifiers are under development. Sol is designed with tasks from computational linguistics in mind and therefore works especially well with sparse data sets for which it scales up to billions of features. Compared to other implementations Sol reaches state of the art results.

1 Introduction

Sol is an open source[1] machine learning toolkit implemented in C++. It is based on stochastic gradient descent (SGD) in the large margin regularized risk minimization framework known from support vector machines (SVMs) [Vap98, SS01]. Besides binary classification Sol implements multi-class classification ([WW99, CS01, SS01]) which it solves as a structured prediction problem ([THJA04]), and multi-label classification ([EW01]) solved in a similar way.

This document is a summary of the theoretical background behind Sol from an implementation perspective. It documents the state of Sol at it's first public release and serves as a background reference for potential users as well as for discussing further development. The paper is organized as follows: Section 2 summarizes notation and basic concepts. Section 3 explains the theoretical background of the algorithms and discusses

[1] published at https://www.github.com/sfendrich/Sol

some implementation details. Section 4 presents some experiments that have been carried out to test functionality and to compare Sol to state of the art machine learning software. Finally, section 5 is concerned with open issues and future work.

2 Preliminaries

Let S be a set. With $\mathfrak{P}(S)$ we denote the power set, and with $|S|$ the cardinality of S. The *Kronecker delta* $\delta : S \times S \to \{0, 1\}$, and the *indicator function*[2] $I_A : S \to \{-1, +1\}$ of a subset $A \subseteq S$ are given by

$$\delta_{st} := \begin{cases} 1 & \text{if } s = t \\ 0 & \text{if } s \neq t \end{cases} \qquad I_A(s) := \begin{cases} +1 & \text{if } s \in A \\ -1 & \text{if } s \notin A. \end{cases}$$

For the real numbers we define the *signum function* $\operatorname{sgn} : \mathbb{R} \to \{+1, 0, -1\}$ and the *hinge function* $(\cdot)_+ : \mathbb{R} \to \mathbb{R}$, which cuts off all negative values setting them to zero, by

$$\operatorname{sgn}(x) = \begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases} \qquad (x)_+ = \max(0, x).$$

Given a set $\mathcal{X}$ of objects and a set $\mathcal{Y}$ of classes (or class labels), a *classifier* is a function $f : \mathcal{X} \to \mathcal{Y}$. A (supervised) *classification problem* is given by the task of learning an approximation h of an unknown classifier f from a given data sample $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$. h is also called a *hypothesis for* f. Depending on the nature of $\mathcal{Y}$ we distinguish several types of classification problems:

Binary classification: $|\mathcal{Y}| = 2$. In this paper we always assume $\mathcal{Y} = \{+1, -1\}$ which we also write as $\{+, -\}$ for short.

Multi-class classification: $\mathcal{Y}$ is finite. Here, we always assume $\mathcal{Y} = \{1, \ldots, k\}$ for a $k \in \mathbb{N}$. In contrast to the labels in multi-label classification, the classes are mutual exclusive here.

Multi-label classification: $\mathcal{Y} \subseteq \mathfrak{P}(\mathcal{L})$ for a finite set $\mathcal{L}$ of labels. In this paper we always assume $\mathcal{L} = \{1, \ldots, k\}$ for a $k \in \mathbb{N}$. In contrast to the classes in multi-class classification, several labels may be assigned to an object.

Regression: $\mathcal{Y} = \mathbb{R}$. In general, regression is not tractable. Therefore we will always make additional assumptions to f having some nice properties, e.g. continuity.

Ranking: $\mathcal{Y}$ is an ordered set. In the language of preference ranking [FH10], this is strictly speaking instance ranking. The terms bipartite ranking (in case $|\mathcal{Y}| = 2$) and multi-partite ranking are also used.

[2]Note that the indicator function is usually defined with a range of $\{0, 1\}$. But, as we take most decisions on the sign of some value, a range of $\{-1, +1\}$ is more convenient for our purpose.

Structured prediction: $\mathcal{Y}$ is a possibly infinite set of structured objects. Typically, for a given x there are many possible answers y from which we seek for the best.

The above problem types are not meant to be mutual exclusive, they rather describe the main perspective on a given classification problem. Most of them have at least some overlap, e.g. binary classification can be seen as a special case of all the others.

Given a set of training instances $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, the general regularized risk minimization problem discussed in this paper is

$$\underset{w}{\text{minimize}} \quad \lambda\Omega(w) + \frac{1}{n}\sum_{i=1}^{n} l(w, x_i, y_i), \tag{1}$$

where the *parameter vector* $w \in \mathcal{W}$ is a real vector parametrizing the hypotheses $h_w \in \mathcal{H}$, $\Omega : \mathcal{W} \to \mathbb{R}_0^+$ is the *regularizer*, which is weighted by the non-negative, real-valued *regularization parameter* λ, and $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_0^+$ is a *loss function* quantifying the loss of quality when predicting $h_w(x_i)$ for the true label y_i. The combination of the regularizer and the empirical risk (i.e. the average loss on the given data) serves as an estimator of the true risk expected on the whole unknown data distribution.

Gradient descent (GD) is a class of iterative optimization techniques based on the gradient of the objective function. GD algorithms can be applied to unconstrained minimization problems if the objective function f is differentiable, and carried over to weaker conditions as long as a reasonable subgradient can be chosen. The basic idea is to iteratively make steps of reasonable size η in descent directions computed from the current gradient as shown for the objective function of problem (1) in algorithm 1.

Algorithm 1 GD

1: **for** $t = 1, \ldots, T$ **do**
2: choose η
3: $w = w - \eta(\lambda\nabla_w\Omega(w) + \sum_i^n \nabla_w l_i(w; x, y))$
4: **end for**

Stochastic gradient descent (SGD) approximates true GD by computing the gradient for a single instance (x_i, y_i) in each step (cf. algorithm 2). Although being a rough approximation, in each iteration SGD touches only one training instance, while GD has to iterate over the whole data sample $(x_1, y_1), \ldots, (x_n, y_n)$ to compute the sum in line 3 of algorithm 1. That is, after having touched n examples, SGD has made n updates, while GD has only made one (with approximately the same computational cost). This may lead to faster converging rates for SGD.

3 Algorithms

This section presents the concrete instances of algorithm 2 for the different problem types, by giving the gradients of the loss functions used in the update rule (line 4).

Sol is placed in the realm of linear discriminant functions $h_{w,b}(x) = \text{sgn}(\langle w, x\rangle + b)$, where the parameter vector (w, b) is formed of the *weight vector* $w \in \mathbb{R}^d$ and the *bias*

Algorithm 2 SGD

1: **for** $t = 1, \ldots, T$ **do**
2: choose $i \in \{1, \ldots, n\}$ at random
3: choose η
4: $w = w - \eta(\lambda \nabla_w \Omega(w) + \nabla_w l_i(w; x_i, y_i))$
5: **end for**

term $b \in \mathbb{R}$. That is, $\mathcal{X} = \mathbb{R}^d$ for some $d \in \mathbb{N}$, and the hypotheses are functions of the form $h_{w,b}$ or combinations of such functions.

Regarding the regularizer, Sol gives the choice between L_1- and L_2-regularization, i.e. between

$$\Omega_1(w) := ||w||_1 = \sum_{i=1}^{d} |w_i| \tag{2}$$

$$\Omega_2(w) := \frac{1}{2}||w||_2^2 = \sum_{i=1}^{d} w_i^2. \tag{3}$$

While Ω_2 is differentiable and therefore gives us a true gradient, we have to choose a reasonable subgradient for Ω_1, which is not differentiable in 0. We have:

$$\nabla_w \Omega_1(w) = (\text{sgn}(w_1), \ldots, \text{sgn}(w_d)) \tag{4}$$

$$\nabla_w \Omega_2(w) = w. \tag{5}$$

The loss function $l_i(b, w)$ is based on the hinge loss

$$l_i^+(w, b) := (1 - y_i(\langle w, x_i \rangle - b))_+, \tag{6}$$

but differs slightly depending on the type of the classification problem. Details are shown below. The hinge loss allows to transform the constrained SVM-optimization problem into an unconstrained optimization problem which can be solved by gradient descent techniques, due to convexity.

The general optimization problem that is solved by Sol is of the form

$$\text{minimize} \quad \lambda \Omega(w) + \frac{1}{n} \sum_{i=1}^{n} l_i(w, b), \tag{7}$$

the individual differences are shown in the following subsections.

3.1 Binary Classification

The goal of binary classification is to learn a function that is able to discriminate between two mutual exclusive classes $y \in \{+1, -1\}$. Our hypothesis space for binary classification

is parametrized by a single weight vector $w = (w_1, \ldots, w_d)$ and a single bias term b. The objective function for binary classification is that of the standard soft margin SVM:

$$\mathrm{SVM}^{\mathrm{bin}}(w, b) = \lambda\Omega(w) + \frac{1}{n}\sum_{i=1}^{n}(1 - y_i(\langle w, x_i\rangle + b))_+. \tag{8}$$

As stochastic subgradient of the loss we choose

$$\nabla_{w,b}\, l_i^{\mathrm{bin}}(w, b) = \begin{cases} -y_i(x_i, 1) & \text{if } y_i(\langle w, x_i\rangle + b) < 1 \\ 0 & \text{else.} \end{cases} \tag{9}$$

3.2 Multi-class Classification

Multi-class classification deals with the task of discriminating between a finite number of mutual exclusive classes $y \in \{1, \ldots, k\}$. This problem can be solved with structured prediction [THJA04]. There are other possibilities, like one versus rest and one versus one, which we don't discuss in this paper. The hypothesis space for multi-class classification is parametrized by $|\mathcal{Y}|$ weight vectors $w_y = (w_{y1}, \ldots, w_{yd})$ and bias terms b_y. The objective function for the structured prediction variant is:

$$\mathrm{SVM}_{\mathrm{sp}}^{\mathrm{mc}}(w, b) = \lambda\sum_{y\in\mathcal{Y}}\Omega(w_y) + \frac{1}{n}\sum_{i=1}^{n}(1 - \langle w_{y_i} - w_{y^*}, x_i\rangle - b_{y_i} + b_{y^*})_+, \tag{10}$$
$$\text{where } y^* = \operatorname*{argmax}_{y\neq y_i}\langle w_y, x_i\rangle$$

The chosen stochastic subgradient of the loss is:

$$\nabla l_i^{\text{mc-sp}}(w_y, b_y) = \begin{cases} -(\delta_{yy_i} - \delta_{yy^*})(x_i, 1) & \text{if } \langle w_{y_i} - w_{y^*}, x_i\rangle + b_{y_i} - b_{y^*} < 1 \\ 0 & \text{else} \end{cases} \tag{11}$$
$$\text{where } y^* = \operatorname*{argmax}_{y\neq y_i}\langle w_y, x_i\rangle$$

3.3 Multi-label Classification

The objective of multi-label classification is to assign non-exclusive labels $l \in \mathcal{L} = \{1, \ldots, k\}$ to objects, which is equivalent to assigning subsets $y \in \mathfrak{P}(\mathcal{L})$. This problem can be solved by learning a separate binary classifier for each label which can be combined to a single objective function. The Hypothesis space for this task is parametrized by $|\mathcal{L}|$ weight vectors $w_l = (w_{l1}, \ldots, w_{ld})$ and bias terms b_y. The objective function for multi-label classification is:

$$\mathrm{SVM}^{\mathrm{ml}}(w, b) = \lambda\sum_{l\in\mathcal{L}}\Omega(w_l) + \frac{1}{n}\sum_{i=1}^{n}\sum_{l\in\mathcal{L}}(1 - I_{y_i}(l)(\langle w_l, x_i\rangle + b_l))_+ \tag{12}$$

where we chose the following subgradient:

$$\nabla_{w_l,b_l} l_i^{\mathrm{ml}}(w_l, b_l) = \begin{cases} -I_{y_i}(l)(x_i, 1) & \text{if } I_{y_i}(l)(\langle w_l, x_i\rangle + b_l) < 1 \\ 0 & \text{else.} \end{cases} \tag{13}$$

3.4 Implementational Issues

Concerning the implementation, a key idea to the speed of Sol is the combination of stochastic updates with a special data representation presented by [SSSS07]:

The weight vector w is represented as a triple (v, α, ν) where v is a vector and α, ν are scalars with $w = \alpha v$ and $\nu = ||w||^2$. v is implemented as an array of floats (i.e. a dense vector) to have the fastest possible random access. This representation consumes 4 MB per million of features which easily scales to a billion features on a > 4 GB machine. ν is updated on each weight update such that we never need to iterate over w to calculate its L_2-norm. With this representation, scaling w is done by just updating α and ν. This makes L_2-regularization cheap.

For the data vectors a sparse representation was chosen which only holds features with a non-zero value. As all zero-values are irrelevant to the results of vector addition and inner product we only need to iterate over the (sparse) data vector to calculate the model score and the updates. The current implementation reads the whole data set into memory to have fast access. Of course this can be a problem on large non-sparse data sets. An extension for on-disk storage is planned, but currently not implemented.

Regarding L_1-regularization, we face the problem, that it cannot be reduced to a simple scaling of w. Here, applying the regularization update essentially means adding or subtracting the regularization parameter λ to/from each feature weight w_i (for $i = 1, \ldots, d$) depending on the sign of w_i. As this process penalizes large feature weights by driving w_i towards 0, we also speak of applying the *regularization penalty*. We have to iterate over the whole weight vector which is expensive if we have a large number of features. To reduce the computational cost, it is possible to apply the regularization penalty only every n-th update. Note that this should be combined with a different value for λ. Furthermore, as long as $|w_i| < \lambda$, applying the regularization penalty at each update would make w_i oscillate between two values which is not desirable. To prevent this behavior we use the simplest form of *truncated gradient descent* ([LLZ09]), and truncate w_i to 0 if $|w_i|$ gets smaller than λ.

3.5 Meta-Parameters

The number of updates T can be chosen as a fixed number in advance. Same for the greatest possible feature id, which can also be estimated from the greatest feature id in the training set. The learning rate η (cf. algorithm 2) can be chosen as a fixed number, or as a decreasing number as in [SSSS07]. Additionally, the ball-projection described in the same paper is optionally available for binary classification. The regularization parameter λ can be set to a fixed number, if set to zero the regularizer component is switched off totally. As described earlier, it is possible to choose between L_1- and L_2-regularization, and a regularization interval can be set to apply regularization every n updates, only.

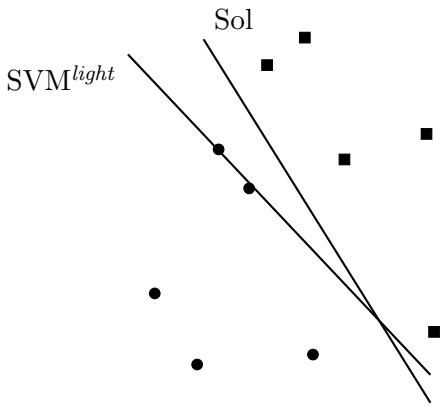

Figure 1: Separating hyperplanes on binary testing data set.

4 Experiments

For testing and evaluation purposes Sol has been applied to the following data sets:

Testing: A very small artificial data set.

Sentiment: A multi-domain sentiment classification data set.

CLTE: SemEval-2012 task on cross lingual textual entailment.

MLcomp: Various data sets on the MLcomp platform.

Regarding the testing, sentiment, and CLTE data sets Sol was compared to the widely used state of the art machine learning tool SVM^{light} [3] [Joa02, Joa99], reaching comparable accuracy results.

4.1 Testing Data Set

The testing data set (testing) is a very small, artificial data set for testing and debugging purposes. It was manually generated and comes in a binary and a ternary version. Due to its small size the learning process and the results can easily be checked and backtracked manually. Figure 1 shows a comparison between Sol and SVM^{light}. Figure 2 shows the model hyperplanes and the effective class boundaries, which are given by the angle bisectors between the hyperplanes.

4.2 Sentiment Classification

[BDP07] have collected a multi-domain sentiment classification data set[4], which was taken to compare Sol's performance with SVM^{light}. The data set consists of the four

[3] `http://svmlight.joachims.org/`
[4] `http://www.cs.jhu.edu/~mdredze/datasets/sentiment/`

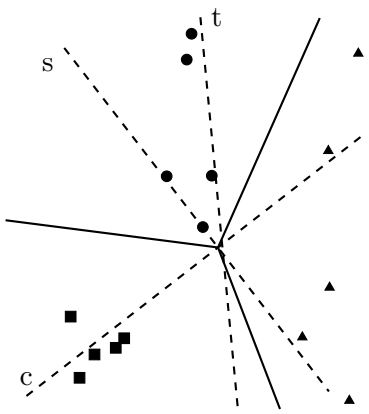

Figure 2: Model hyperplanes (dashed) and class boundaries (continuous) on ternary testing data set. Positive halfspaces are marked with c, s, and t for circles, squares, and triangles (Sol structured prediction multi-class).

training set	1	2	4	5	total
books	1106	1147	1088	1059	4400
dvd	1050	1001	1014	1029	4094
electronics	1166	1180	1194	1180	4720
kitchen	1039	1053	1005	1022	4119
test set	1	2	4	5	total
books	280	253	293	275	1101
dvd	271	243	259	251	1024
electronics	302	286	289	304	1181
kitchen	261	260	257	252	1030

Table 1: Sentiment classification: number of instances by stars.

domains books, dvd, electronics, and kitchen. The data vectors are built from unigram and bigram counts of product reviews and are rated with one or two stars for negative sentiment, and four or five stars for positive sentiment. The data set is nearly balanced, the sizes of the four domains are given in table 1.

Three systems have been trained – SVM^{light} regression, SVM^{light} binary, and Sol binary – on each domain individually as well as on all domains at once. The systems were evaluated separately on each domain. Table 2 shows that the systems perform comparably.

4.3 Cross-lingual Textual Entailment

The SemEval 2012 Cross-lingual Textual Entailment task [NMM$^+$12] was to recognize the presence of semantic entailment between two sentences E and F from different languages. The question here is if the meaning of F follows from the meaning of E

training	SVM^{light} regression				SVM^{light} binary				Sol binary			
↓ test →	b	d	e	k	b	d	e	k	b	d	e	k
books	.813	.759	.720	.741	.814	.760	.717	.737	**.830**	**.796**	**.763**	**.775**
dvd	**.787**	.801	.768	.773	.777	.801	.766	.780	**.787**	**.828**	**.774**	**.783**
electronics	**.738**	.744	.848	**.868**	.727	.741	.839	.855	.724	**.766**	**.851**	.844
kitchen	**.723**	**.766**	.848	**.888**	.717	.745	.842	.876	.717	.752	**.853**	.881
all	.832	.846	**.885**	**.895**	.833	.828	.875	.893	**.845**	**.860**	.880	.886

Table 2: Sentiment classification: accuracy of different systems with best results in **bold**.

and vice versa, which we write as $E \to F$ (E entails F) and $F \to E$, respectively. This leads to the four possible entailment relations *bidirectional*, *forward*, *backward*, and *no entailment*. For each of four language pairs (english-spanish, english-italian, english-french, and english-german) a training set with 500 sentence pairs was provided. The participating systems were evaluated on additional 500 sentence pairs. We applied SVM^{light}in two and Sol in three different variants to features extracted from this data:

- Multi-class classification with the four entailment directions as classes. (Sol and SVM^{light})
- Multi-label classification with forward and backward entailment as labels, which are interpreted as bidirectional if both labels are predicted and as no entailment if no label is predicted. (Sol)
- multi-label classification as above, but each label was predicted by a separately trained binary classifier. (Sol and SVM^{light})

The features as well as the official results are reported in an upcoming paper [WF12]. The results of Sol and SVM^{light}are comparable.

4.4 MLcomp

MLcomp[5] is an online platform for the comparison of machine learning tools and data sets. Any user can upload programs and data sets, and can run any available program on any available data set. We uploaded Sol for binary classification. Although promising compared to other linear classifiers, we don't report results here, because they are under constant change. This is due to data sets being added and deleted, new runs being submitted etc. Especially, each time a program is updated, all runs of the program are deleted. The current results of Sol are found at `http://mlcomp.org/programs/1017`.

5 Future Work

There are several open issues with the current implementation which are documented as `// TODO: ...` comments in the source code. Of special interest for users is that the

[5] `http://mlcomp.org/`

multi-label learner is currently limited to a small number of labels. The input format was not intended to provide multiple labels. As a fast hack, a set of labels is encoded bitwise as an 32-bit integer, each bit indicating the presence (1) or absence (0) of the corresponding label. Therefore, multi-label classification is currently limited to 32 labels, yielding to up to 2^{32} possible label combinations. There are multi-label data sets with hundreds or thousands of labels, so this is in fact an issue.

Furthermore, it is planned to extend Sol with different classifiers and features, as pairwise ranking, regression, online learning, multi-task learning, on-disk storage for large data sets, implicit parameter tuning, different feature selection techniques, sparse representations for very large models, and more.

References

[BDP07] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, 2007.

[CS01] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[EW01] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *In Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press, 2001.

[FH10] Johannes Fürnkranz and Eyke Hüllermeier, editors. *Preference Learning*. Springer, 1. edition, October 2010.

[Joa99] Thorsten Joachims. Making large-scale svm learning practical. In Bernhard Schölkopf, C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[Joa02] Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines*. PhD thesis, Department of Computer Science, Cornell University, May 2002.

[LLZ09] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

[NMM+12] Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. Semeval-2012 Task 8: Cross-lingual Textual Entailment for Content Synchronization. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012)*, 2012.

[SS01] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001.

[SSSS07] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 807–814, New York, NY, USA, 2007. ACM.

[THJA04] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21 st International Conference on Machine Learning*, 2004.

[Vap98] Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley Interscience, 1998.

[WF12] Katharina Wäschle and Sascha Fendrich. Hdu: Cross-lingual textual entailment with smt features. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 467–471, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics.

[WW99] Jason Weston and Chris Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, 1999.