Multi-Threaded Support Vector Machines For Pattern Recognition

João Gonçalves[1], Noel Lopes[1,2], and Bernardete Ribeiro[1]

[1]CISUC, Department of Informatics Engineering, University of Coimbra, Portugal
[2]UDI/IPG - Research Unit, Polytechnic Institute of Guarda, Portugal
jcgonc@student.dei.uc.pt, noel@ipg.pt, bribeiro@dei.uc.pt

Abstract. Support Vector Machines (SVM) have become indispensable tools in the area of pattern recognition. They show powerful classification and regression performance in highly non-linear problems by mapping the input vectors nonlinearly into a high-dimensional feature space through a kernel function. However, the optimization task is numerically expensive since single-threaded implementations are hardly able to cope up with the complex learning task. In this paper, we present a multi-threaded implementation of the Sequential Minimal Optimization (SMO) which reduces the numerical complexity by parallelizing the KKT conditions update, the calculation of the hyperplane offset and the classification task. Our preliminary results both in benchmark datasets and real-world problems show competitive performance to the state-of-the-art tools while the execution running times are considerably faster.

Keywords: SVM, OpenMP, sequential minimal optimization (SMO)

1 Introduction

The increasing complexity and performance demands in pattern recognition applications require innovative and fast approaches to cope with the system non-linearities. In particular, the design of efficient and scalable systems depends on powerful tools to extract relevant (and meaningful) information. Additionally, the learning algorithms often require high-processing capabilities making current single-threaded algorithms unable to scale with the demanding processing power needed. Among the supervised learning algorithms, Support Vector Machines (SVMs) are the most widely used algorithm due to their generalization properties and regularization capability. SVMs are binary large margin classifiers which have found successful applications in many scientific fields such as bio-informatics [18], information management [1], finance and business [16]. The SVM aims to find the optimal decision hyperplane which is equivalent to reach the best trade-off between generalization and empirical errors. An important and crucial point in the SVM formulation is that it can provide a good generalization independent of the training set distribution by making use of the principle of structural risk minimization [15,8]. However, they usually require significant memory and computational burden for calculating the large Gram matrix [7].

To circumvent this limitation fast learning methods have successfully been proposed [9,10]. However, most implementations do no take advantage the multicore architecture of today CPU baseline computers. In this paper we focus on a multi-threaded parallel CPU standalone SVM version (MT-SVM) which builds up from the scratch an implementation of the Sequential Minimal Optimization (SMO) algorithm. Although previous approaches have been developed [5], our implementation includes a new kernel function, the Universal Kernel Function (UKF) [17] which leads to a broad spectrum of the generalization capabilities of the learning machine. Experiments performed on UCI datasets benchmarks [2] and real world problems such as MP3 Steganalysis [14] and the Peptidases detection [12] yield performance competitive results as compared to state-of-the-art LIBSVM tools while delivering better speedups on large datasets.

The paper is organized as follows: Section 2 describes the SVM training and classification tasks. Section 3 addresses the Sequential Minimal Optimization (SMO) algorithm. Section 4 describes the parallel implementation of both the training and classification tasks. We present our results in section 5. The conclusions as well as future work are addressed in section 6.

2 Support Vector Machines (SVM)

Given a set of n training points in a d dimensional feature space $\mathbf{x} \in \mathbb{R}^d$ each associated with a label $y_i \in \{-1, 1\}$ the binary soft-margin kernel SVM solves the linearly convex quadratic problem:

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \alpha_i \leq C, \;\; i = 1, 2, \ldots n \tag{2}$$

For each training point $\mathbf{x}_i$ there is an associated Lagrange multiplier α_i, bounded between 0 and the penalization constant C. The careful selection of this constant allows the SVM to balance the generalization and empirical error. The data points with $\alpha_i > 0$ are the Support Vectors (SVs) and define the decision boundary. Considering that n_{SV} is the number of SVs, after convergence the offset b is calculated as a weighted arithmetic average as follows:

$$b = \frac{1}{n_{SV}} \sum_{j=1}^{n_{SV}} \left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) \right) - y_j \tag{3}$$

To improve performance on non-linearly separable classes, the above optimization task (see (1)) makes use of the kernel trick which allows the SVM to work on a higher dimension feature space by means of a dot-product between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. This result is calculated using the kernel projection $K(\mathbf{x}_i, \mathbf{x}_j)$. Therefore with a non-linear kernel the margin corresponds to a linear boundary in this

Algorithm 1 Sequential Minimal Optimization (SMO) algorithm

Require: $\mathbf{x}_i \in \chi$, $y_i \in \Omega$, $i \in \{1 \cdots n\}$
1: Initialize: α_i=0, f_i=$-y_i$, $\forall i \in \{1 \cdots n\}$
2: Initialize: $b_{high} = -1$, $b_{low} = 1$, $i_{high} = \min\{i : y_i = 1\}$, $i_{low} = \min\{i : y_i = -1\}$, $\forall i \in \{1 \cdots n\}$
3: Update: α_{ilow}, $\alpha_{i_{high}}$
4: **repeat**
5: Update optimality conditions f_i (see (9))
6: Compute: b_{high}, b_{low}, i_{high}, i_{low}
7: Update $\alpha_{i_{low}}$, $\alpha_{i_{high}}$, $\forall i \in \{1 \cdots n\}$
8: **until** $b_{low} \leq b_{high} + 2\tau$

new feature space. The standard kernel functions (linear, polynomial, Gaussian and sigmoid) have been considered as well as a recent kernel function, Universal Kernel Function (UKF) which has been proved to satisfy Mercer kernel [17] conditions. In the sequel is described as follows:

$$K(\mathbf{u}, \mathbf{v}) = L(\|\mathbf{u} - \mathbf{v}\|^2 + \sigma^2)^{-\alpha} \tag{4}$$

where L is a normalization constant, $\sigma > 0$ is the kernel width and $\alpha > 0$ controls the decreasing speed around zero. This kernel aims to gather points near to each other, in a higher dimension space, since they are strongly correlated. Hence, it can provide a small number of SVs and thus speeds up both the training and classification tasks. Additionally, it can yield better generalization [3]. Finally, the classification of a given sample $\mathbf{z}$ is done using a subset of the training set upholding the support vectors. The SVM classification task is:

$$y(\mathbf{z}) = \text{sign}\left(b + \sum_{i=1}^{n_{SV}} \alpha_i y_i K(\mathbf{x}_i, \mathbf{z})\right) \tag{5}$$

3 Sequential Minimal Optimization (SMO) Algorithm

The Sequential Minimal Optimization (SMO) algorithm was developed by Platt in 1998 [13]. At each step, only two Lagrange multipliers, α_i and α_j are required to be solved. Both multipliers must satisfy the constraints defined in (2) [13,5]. Algorithm 1 details the main steps of the soft-margin SMO algorithm using the kernel trick [5,4]. Initially the α_i are set to 0 as they satisfy the constraints defined in (2). At each step, after choosing i_{high} and i_{low}, the new values for the two Lagrange multipliers α_i^{new} are computed as follows:

$$\alpha_{i_{low}}^{\text{new}} = \alpha_{i_{low}} + y_{i_{low}} \frac{b_{high} - b_{low}}{\eta} \tag{6}$$

$$\alpha_{i_{high}}^{\text{new}} = \alpha_{i_{high}} + y_{i_{low}} y_{i_{high}} (\alpha_{i_{low}} - \alpha_{i_{low}}^{\text{new}}) \tag{7}$$

where η is defined as:

$$\eta = K(x_{i_{high}}, x_{i_{high}}) + K(x_{i_{low}}, x_{i_{low}}) - 2 \cdot K(x_{i_{high}}, x_{i_{low}}) \tag{8}$$

Naturally, $\alpha_{i_{low}}$ and $\alpha_{i_{high}}$ must satisfy (2). Thus, if $\alpha_{i_{low}}$ changes by δ then $\alpha_{i_{high}}$ changes by the same amount on the opposite direction $(-\delta)$. Next, the Karush-Kuhn-Tucker (KKT) conditions must be updated for each sample $\mathbf{x}_i$:

$$f_i = f_i^{old} + (\alpha_{i_{high}}^{new} - \alpha_{i_{high}}) y_{i_{high}} K(x_{i_{high}}, x_i) + (\alpha_{i_{low}}^{new} - \alpha_{i_{low}}) y_{i_{low}} K(x_{i_{low}}, x_i) \tag{9}$$

The indices of the next Lagrange multipliers i_{low} and i_{high} are chosen from two corresponding sets:

$$I_{low} = \{i : 0 < \alpha_i < C\} \cup \{i : y_i > 0, \alpha_i = C\} \cup \{i : y_i < 0, \alpha_i = 0\} \tag{10}$$

$$I_{high} = \{i : 0 < \alpha_i < C\} \cup \{i : y_i > 0, \alpha_i = 0\} \cup \{i : y_i < 0, \alpha_i = C\} \tag{11}$$

The optimality coefficients b_{low} and b_{high} are calculated as:

$$b_{low} = \max\{f_i : i \in I_{low}\} \tag{12}$$

$$b_{high} = \min\{f_i : i \in I_{high}\} \tag{13}$$

For simplicity, to choose i_{low} and i_{high} we use the first order heuristic [11]. For the next iteration, these indices are calculated as:

$$i_{low} = \arg\max\{f_i : i \in I_{low}\} \tag{14}$$

$$i_{high} = \arg\min\{f_i : i \in I_{high}\} \tag{15}$$

The algorithm is executed until the following inequality holds:

$$b_{low} \leq b_{high} + 2\tau \Leftrightarrow b_{low} - b_{high} \leq 2\tau \tag{16}$$

where $\tau : 0 < \tau < 1$ is the tolerance of the solution optimality and in fact the stopping criteria. After converging, the parameter b can be calculated using (3).

4 Parallel SMO implementation

Our implementation was developed in C++ using OpenMP. This API allows to write multi-threaded shared memory (also named Uniform Memory Access (UMA)) applications in either C/C++ or Fortran. Programs written using this approach are in the Flynn's taxonomy classified as Single Process Multiple Data (SPMD) because the same program is executed by different threads, each processing a different subset of the data.

Our approach consists of identifying the SMO steps that are simultaneously responsible for large portions of the overall computation and that could be safely parallelized. One of such steps, as noted by Cao et al. [4], is the KKT conditions update (using f_i and the kernel Gram matrix). Since each f_i can be computed independently, this step can fully take advantage of CPU multi-core architectures. Another phase which can be accelerated is the computation of the next b_{low}, b_{high}, α_{low} and α_{high}. Since this is done by performing a first order heuristic search, it can be executed in parallel using reduction operations. Each thread

works on a subset of both I_{high} and I_{low} while the master thread waits for the results and then applies the reduction operators. The offset b is also computed in parallel for each SV. Thus, the only sequential steps are the Lagrange multipliers (α_{ilow} and $\alpha_{i_{high}}$) update and the convergence verification.

The above parallel tasks are divided into equal parts, each one assigned to a corresponding thread. In theory, if the original single-threaded SMO training process takes T_s time, using a processor with P cores, the multi-threaded SMO would execute in $T_p = \frac{T_s}{P}$ and would offer a speedup of $P\times$. However, this theoretical speedup is rarely achieved in practice because part of the code is not parallelized. Even though the algorithm can be fully parallel, the sequential sections always exist (Amdahl's law) due to: (i) synchronization, where some threads must wait for the completion of others, (ii) memory bandwidth, which is shared by all CPU cores, and (iii) mutual exclusion areas, among other reasons.

5 Experimental Setup, Results and Discussion

In order to evaluate our MT-SVM implementation w.r.t. the performance and speedup we compared the results obtained for several benchmarks with the corresponding results of the state-of-the-art LIBSVM (version 3.11) [6]. Both tools use the SMO algorithm. For fairness we set LIBSVM cache to one Megabyte since currently our implementation does not make use of a kernel cache. For our implementation we set the number of threads to 4. The system used for testing has an Intel Quad Core i5-750 processor with the clock set to 3.33 GHz. Moreover, the machine used had 12 GB RAM.

Currently our implementation is designed exclusively for binary tasks, thus we have specifically chosen binary class datasets for the experimental setup. With the exception of the Two-Spiral, the MP3 Steganalysis [14] and the Peptidases detection [12], the remainder datasets where obtained from the UCI Machine Learning repository [2]. The Two-Spiral dataset consists of learning to discriminate between data distributed on two distinct spirals that coil around each other in the x-y plane. This dataset was used in order to assess the Universal Kernel Function (UKF) kernel efficiency. The MP3 Steganalysis dataset was extracted from a real problem using the four methods described in Qiao et al. [14]. The dataset is composed of two classes: the first corresponds to normal MP3 audio files (cover) and the second are the same MP3 files with hidden information (stego). The Peptidases detection problem is described in Lopes et al. [12]. Peptidases are a class of enzymes that catalyze chemical reactions, allowing the decomposition of protein substances into smaller molecules. The task consists of discriminating between peptidases and non-peptidases. Table 1 lists the main characteristics of the datasets as well as the best RBF kernel parameters determined by grid search. The optimality gap τ was set to 0.01. The datasets were normalized before being processed using a standard score procedure. We ran the experiments 10 times for each dataset with 5-fold cross validation. Table 2 shows the speedups obtained by MT-SVM as compared to LIBSVM, both for training and classification tasks. For the smaller datasets (Breast Cancer, Haber-

Table 1. Datasets and RBF kernel parameters used in the experiments.

Dataset	#Samples	#Features	C	γ
Adult	32561	14	1.0	0.100
Breast Cancer	569	30	3.0	0.050
German	1000	59	1.0	0.050
Haberman	306	3	1.0	1.000
Heart	270	20	0.1	0.050
Ionosphere	351	34	1.0	0.500
Sonar	208	30	3.0	0.050
Tic-tac-toe	958	9	1.0	0.001
Two-Spiral	2097152	2	3.0	0.250
MP3 Steganalysis	1994	742	0.1	0.250
Peptidases	20778	24	0.56	11.30

Table 2. Training and classification times (in seconds) and speedups obtained for the MT-SVM implementations as compared to LIBSVM.

	MT-SVM	LIBSVM	MT-SVM	LIBSVM	Training	Classification
Dataset	Training		Classification		MT-SVM Speedup	
adult	17.733±0.063	32.492±0.071	1.033±0.717	2.240±0.586	**1.83**×	**2.17**×
Breast Cancer	0.028±0.001	0.004±0.003	0.008±0.001	0.008±0.003	0.14×	**1.10**×
German	0.088±0.002	0.126±0.001	0.009±0.005	0.024±0.004	**1.42**×	**2.61**×
Haberman	0.028±0.001	0.004±0.000	0.006±0.002	0.001±0.004	0.15×	0.14×
Heart	0.012±0.002	0.005±0.001	0.007±0.002	0.002±0.003	0.44×	0.23×
Ionosphere	0.029±0.002	0.009±0.001	0.007±0.001	0.003±0.003	0.32×	0.41×
Sonar	0.021±0.001	0.007±0.001	0.007±0.004	0.003±0.003	0.32×	0.38×
Tic-tac-toe	0.147±0.001	0.079±0.001	0.007±0.008	0.006±0.010	0.53×	0.78×
Two-Spiral	21.259±0.116	146.723±0.664	3.018±13.569	11.720±2.025	**6.90**×	**3.88**×
MP3 Steganalysis	0.344±0.003	2.367±0.016	0.019±0.053	0.573±0.017	**6.87**×	**29.53**×
Peptidases	3.973±0.033	12.079±0.008	0.418±0.124	1.690±0.204	**3.04**×	**4.04**×

man, Heart, Ionosphere, Sonar and Tic-tac-toe) the speedup is actually negative (< 1). In this case, the amount of data simply does not justify the overhead of launching additional threads and synchronizing their execution. However, for bigger datasets, with a sufficient large number of samples and/or features (Adult, German, Two-Spiral, MP3 Steganalysis and Peptidases detection), the MT-SVM implementation can boost significantly both the training and the classification tasks. This takes particular relevance for the training task as it can considerably reduce the time required to perform a grid search (a fundamental process to obtain good generalization models). As illustrated in Table 3, MT-SVM yields competitive performance as compared to LIBSVM. In terms of classification (accuracy and F-Score), the Wilcoxon signed ranked test found no statistical evidence of any of the implementations performing worse than the other. However, the null hypothesis that MT-SVM generates a model with a number of SVs greater or equal than the number generated by LIBSVM is rejected at 0.05 significance level. Table 4 presents the UKF results. The additional number of parameters greatly increases the complexity of performing a grid search. Having said that, it is possible that the results could be improved by narrowing the search. Nevertheless, the results show the usefulness of the UKF kernel. UKF yield better or equal F-Score results than the RBF kernel, in almost a half of the

Table 3. MT-SVM and LIBSVM classification performance and number of SVs.

Dataset	MT-SVM	LIBSVM	MT-SVM	LIBSVM	MT-SVM	LIBSVM
	Accuracy (%)		F-Score (%)		#SVs	
Adult	84.65±0.39	**84.72±0.38**	90.13±0.28	**90.38±0.24**	**9781.8±56.4**	9788.4±48.9
Breast Cancer	97.48±2.26	**97.76±1.49**	96.59±1.90	**96.96±2.02**	**113.3±05.2**	114.4±04.9
German	**73.61±1.65**	73.03±1.32	83.44±1.08	**83.46±0.82**	**713.7±05.5**	718.7±05.1
Haberman	71.85±4.42	**72.92±3.50**	82.14±3.13	**83.49±2.31**	**149.1±04.8**	151.2±04.4
Heart	**83.18±4.64**	82.37±4.96	**85.44±4.09**	85.30±4.00	**175.7±03.1**	177.2±03.1
Ionosphere	**89.66±3.38**	89.06±3.58	**91.16±3.13**	90.72±3.29	**215.1±03.1**	217.7±03.2
Sonar	**85.77±4.90**	84.65±4.78	**87.30±4.39**	86.83±4.00	**151.1±02.6**	153.7±02.4
Tic-tac-toe	97.70±1.22	**97.72±1.26**	98.28±0.90	**98.29±0.93**	**548.4±10.1**	551.8±10.9
Two-Spiral	**100.00±0.0**	**100.00±0.0**	**100.00±0.0**	**100.00±0.0**	**939.9±58.2**	1053.1±67.5
MP3 Steganalysis	**97.05±0.87**	96.92±1.00	**97.06±0.88**	96.94±0.99	**346.2±07.1**	348.1±07.2
Peptidases	**96.25±0.24**	96.04±0.23	**97.85±0.14**	97.75±0.13	6849.4±23.7	**3829.6±17.3**

Table 4. UKF kernel results with MT-SVM.

Dataset	Time (seconds)		Classification		
	Training	Classification	Accuracy (%)	F-Score (%)	#SVs
Adult	15.518±0.160	2.210±0.029	83.36±0.37	89.47±0.26	12543.6±56.6
Breast Cancer	0.026±0.010	0.001±0.000	**98.11±1.00**	**97.39±1.46**	**63.1±03.0**
German	0.139±0.018	0.005±0.001	71.79±3.15	83.04±2.08	795.6±13.9
Haberman	0.027±0.008	0.001±0.000	72.45±4.91	82.85±3.58	231.1±04.0
Heart	0.014±0.001	0.001±0.000	82.93±3.59	84.85±3.41	181.5±05.5
Ionosphere	0.014±0.002	0.001±0.000	**94.28±3.04**	**95.61±2.30**	**101.4±04.0**
Sonar	0.014±0.003	0.000±0.000	85.10±4.81	86.19±4.69	**129.8±03.8**
Tic-tac-toe	0.208±0.021	0.002±0.001	98.17±0.94	**98.59±0.76**	**475.6±11.0**
Two-Spiral	13.210±0.004	2.940±0.010	**100.00±0.00**	**100.00±0.00**	**324.0±05.0**
MP3 Steganalysis	0.915±0.045	0.052±0.007	93.02±0.78	93.12±0.83	1019.4±08.5
Peptidases	7.803±0.827	0.638±0.056	96.73±0.25	**98.14±0.14**	5097.6±57.5

datasets. Using the Wilcoxon signed ranked test we found no statistical evidence of the UKF kernel performing worse than the RBF kernel and vice-versa. It is interesting to note that, with the exception of the Sonar dataset, UKF yields better F-Score results in the datasets that present a smaller number of SVs than the corresponding number for the RBF kernel. This seems to indicate that UKF presents better classification performance when it is able to gather points near to each other, in a higher dimension space, as intended.

6 Conclusions and Future Work

As the amount of data produced grows at an unprecedented rate fast machine learning algorithms that are able to extract relevant information from large repositories have become extremely important. To partly answer to this challenge in this paper we proposed a multi-threaded parallel MT-SVM which parallelizes the SMO algorithm. Our implementation uses the power available on multi-core Central Processing Units (CPUs) and efficiently learns (and classifies) within several domains, exposing good properties in scaling data. Speedups up to 7×

on training and up to 30× on classification tasks were achieved. Additionally, the UKF kernel which has good generalization properties in the high-dimensional feature space has been included, although more parameters are needed to fine tune the results. In future work we will account for vectorization (SSE or AVX) as well as support for kernel caching which may drastically decrease the amount of computation.

References

1. Ando, R., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. J. of Machine Learning Research 6, 1817–1853 (2005)
2. Asuncion, A., Newman, D.J.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml/index.html`
3. Ayat, N., Cheriet, M., Suen, C.: KMOD - a two-parameter SVM kernel for pattern recognition. In: Proc. of ICPR. pp. 30331–30334 (2002)
4. Cao, L., Keerthi, S., Ong, C.J., Zhang, J., U. Periyathamby, X.J.F., Lee, H.: Parallel sequential minimal optimization for the training of support vector machines. IEEE Transactions on neural networks 17(4), 1039–1049 (2006)
5. Catanzaro, B., Sundaram, N., Keutzer, K.: Fast support vector machine training and classification on graphics processors. In: Proc. of the 25th Int Conf on Machine learning. pp. 104–111. ICML '08, ACM, NY, USA (2008)
6. Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines. Dep. of Computer Science National Taiwan University, Taipei, Taiwan (2011)
7. Chen, B., Zhao, S., Zhu, P., Príncipe, J.C.: Quantized kernel least mean square algorithm. IEEE Trans. Neural Netw. Learning Syst. 23(1), 22–32 (2012)
8. Cortes, C., Vapnik, V.: Support-vector networks. In: Machine Learning. pp. 273–297 (1995)
9. De Brabanter, K., De Brabanter, J., Suykens, J., De Moor, B.: Optimized fixed-size kernel models for large data sets. Comput. Stat. Data Anal. 54(6), 1484–1504 (2010)
10. Hoegaerts, L., Suykens, J.A.K., Vandewalle, J., De Moor, B.: Subset based least squares subspace regression in RKHS. Neurocomput. 63, 293–323 (2005)
11. Keerthi, S., Shevade, S., Bhattacharyya, C., Murthy, K.: Improvements to Platt's SMO algorithm for SVM classifier design. Neural Comput. 13(3), 637–649 (2001)
12. Lopes, N., Correia, D., Pereira, C., Ribeiro, B., Dourado, A.: An incremental hypersphere learning framework for protein membership prediction. In: Int. Conf. on Hybrid Artificial Intelligence Systems. pp. 429–439. LNCS 7208 (2012)
13. Platt, J.: Sequential minimal optimization: A fast algorithm for training support vector machines, vol. 208, pp. 1–21. MIT Press (1998)
14. Qiao, M., Sung, A.H., Liu, Q.: Feature mining and intelligent computing for MP3 steganalysis. In: Int Joint Conf on Bioinformatics, Systems Biology and Intelligent Computing. pp. 627–630. IEEE Computer Society (2009)
15. Vapnik, V.: The nature of statistical learning theory. Springer-Verlag (1995)
16. Wu, W.W.: Beyond business failure prediction. Expert Systems with Applications 37, 2371–2376 (2010)
17. Zhang, R., Wang, W.: Facilitating the applications of support vector machine by using a new kernel. Expert Systems with Applications 38, 14225–14230 (2011)
18. Zien, A., Rätsch, G., Mika, S., B. Schölkopf, C.L., Smola, A., Lengauer, T., Mueller, K.R.: Engineering support vector machine kernel that recognize translation initiation sites. Bioinformatics 16(9), 799–807 (2000)