FaLKM-lib v1.0: A LIBRARY FOR FAST LOCAL KERNEL MACHINES

Nicola Segata

April 2009

Technical Report # DISI-09-025

FaLKM-lib v1.0: a Library for Fast Local Kernel Machines

Nicola Segata[*]
segata@disi.unitn.it

April 24, 2009

Abstract

FaLKM-lib v1.0 is a library for fast local kernel machine implemented in C++. It contains a fast implementation of kernel k-nearest neighbors (kNN) using the Cover Tree data-structure called **FkNN**, the local support vector machine (LSVM) algorithm called **FkNNSVM**, a noise reduction technique based on a probabilistic version of LSVM called **FkNNSVM-nr**, a fast and scalable version of LSVM called **FaLK-SVM** (subdivided in the two modules: **FaLK-SVM-train** and **FaLK-SVM-predict**) and a fast and scalable noise reduction technique called **FaLKNR**. The library contains tools for model selection, local model selection and automatic tuning of kernel parameters. This document introduces the formulation of the algorithms in the software library; for a comprehensive discussion on the implemented techniques please refer to the papers cited in this document and for the use of the software refer to the README file included in the package. **FaLKM-lib v1.0** code is freely available for research and educational purposes at `http://disi.unitn.it/~segata/FaLKM-lib`.

1 Introduction

This document presents a software library called **FaLKM-lib v1.0** for fast and local kernel machines. **FaLKM-lib v1.0** combinis the machine learning techniques and the tools developed and discussed by Blanzieri and Melgani (2006, 2008); Segata and Blanzieri (2009a,b); Segata, Blanzieri, Delany and Cunningham (2008); Segata, Blanzieri and Cunningham (2009). The main common idea underlying the algorithms in the library is that very often (feature-space) locality plays a crucial role in machine learning classification problems. In particular locality can lead to the following advantages:

[*]Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy.

Improved accuracy. Local support vector machines (LSVM) which can be considered members of the class of local learning algorithms (LLA) introduced by Bottou and Vapnik (1992) are able to improve the generalization ability of support vector machines (SVM) which are considered the state-of-the-art classifiers (Cortes and Vapnik; 1995; Vapnik; 2000) and of kNN as studied by Segata and Blanzieri (2009a). Although implemented using specific data-structures for supporting neighborhood retrieval (we adopt the Cover Tree by Beygelzimer et al. (2006)) the computational overhead of LLA can be however consistent and for this reason LSVM are not indicated for large datasets. The algorithm in **FaLKM-lib v1.0** belonging to this class is **FkNNSVM** (and, considering the kNN as a LLA, also **FkNN**).

Improved speed and accuracy on large datasets. Large datasets require specific strategies to be efficiently tackled. The local approach to classification with kernels can be applied to large amount of data using some modifications of the LLA approach we presented in (Segata and Blanzieri; 2009b) permitting to obtain training and testing times dramatically lower than SVM ones. The algorithm for fast classification with local kernel machines presented here is **FaLK-SVM** (subdivided in the training module **FaLK-SVM-train** and the testing module **FaLK-SVM-predict**) and introduces novel improvements not discussed by Segata and Blanzieri (2009b) further decreasing the training and especially the testing time[1].

Improved ability of detecting noisy samples. Noise reduction consists in detecting and removing samples from a dataset based on their probability to be corrupted, mislabelled or noisy samples. Noise reduction can be used for data cleansing in bioinformatics and in the medical domain (Malossini et al.; 2006; Gamberger et al.; 2000; Lorena and Carvalho; 2004; Tang and Chen; 2008), for enhancing the generalisation ability of learning systems that are not noise tolerant like instance-based learning and case-based reasoning (Wilson and Martinez; 2000; Brighton and Mellish; 2002) and for enchancing case-based explanation (Leake; 1996; Cunningham et al.; 2003). In this context we developed two noise reduction algorithms based on a probabilistic variant of LSVM (Segata, Blanzieri, Delany and Cunningham; 2008) and on a variant of **FkNNSVM** (Segata, Blanzieri and Cunningham; 2009) available in **FaLKM-lib v1.0** under the names of **FkNNSVM-nr** and **FaLKNR** respectively.

Improved speed in performing model selection. Model selection is a crucial training step in the learning process to assure high classifica-

[1]Benchmark results can be found on the FaLKM-lib home page: `http://disi.unitn.it/~segata/FaLKM-lib`.

tion performances of kernel based methods. However it is often a very computationally heavy step (especially if the number of parameters to set is high) and the found parameters are the same for every sub-region of the datasets. With the local approach it is possible to tune locally the kernel and SVM parameters to better capture the local characterization of the problem and to use only a subset of the local model to estimate the parameters in order to unburden model selection. For these reasons in **FaLKM-lib v1.0**, in addition to the traditional grid search model selection, we implemented different strategies of local model selection in **FkNNSVM**, **FkNNSVM-nr**, **FaLK-SVM** and **FaLKNR**.

FaLKM-lib v1.0 is implemented in C++ and it is freely available for research and educational purposes (see the copyright file in the library package) at `http://disi.unitn.it/~segata/FaLKM-lib` and can be compiled both under Unix-based systems and Windows systems[2]. It uses LibSVM (Chang and Lin; 2001) version 2.88 for training and testing the local SVM models and our implementation of the Cover Tree data-structure by Beygelzimer, Kakade and Langford (2006) for nearest neighbor operations.

In the following we very briefly introduce the formulations of the implemented modules. For the usage of **FaLKM-lib v1.0** the user should refer to the README file in the software package.

2 The modules of **FaLKM-lib v1.0**

We introduce here the formulation and the theoretical complexity bounds of the modules, and the tools common to all of them. For a detailed discussion of the modules, please refer to the corresponding cited papers.

The mathematical formalism used to describe the methods assumes to have a classification problem with samples (x_i, y_i) with $i = 1, \dots, n$, $x_i \in \mathcal{H}$ where $\mathcal{H}$ is an Hilbert space ($\mathbb{R}^p$ as a special case), $y_i \in \{+1, -1\}$ and a mapping function $\Phi : \mathcal{H} \mapsto \mathcal{H}$ induced by a positive definite kernel function $K(\cdot, \cdot)$. Given a point x', it is possible to order the entire set of training samples X with respect to x' in the feature space. This corresponds to define the function $r_{x'} : \{1, \dots, n\} \to \{1, \dots, n\}$ as:

$$\begin{cases} r_{x'}(1) = \underset{i=1,\dots,n}{\operatorname{argmin}} \|\Phi(x_i) - \Phi(x')\|^2 & \\ r_{x'}(j) = \underset{i=1,\dots,n}{\operatorname{argmin}} \|\Phi(x_i) - \Phi(x')\|^2 & \begin{array}{c} i \neq r_{x'}(1), \dots, r_{x'}(j-1) \\ \text{for } j = 2, \dots, n \end{array} \end{cases} \qquad (1)$$

[2]Although we give the Windows binaries and Microsoft Visual C++ makefile, **FaLKM-lib v1.0** has been mainly developed and tested in Unix-based environments.

the computation can be expressed in terms of kernels as:

$$\begin{aligned} ||\Phi(x) - \Phi(x')||^2 &= \langle\Phi(x), \Phi(x)\rangle + \langle\Phi(x'), \Phi(x')\rangle - 2\langle\Phi(x), \Phi(x')\rangle = \\ &= K(x, x) + K(x', x') - 2K(x, x'). \end{aligned} \tag{2}$$

In the following k refers to the neighborhood size (i.e. the k parameter of kNN and of the other locality-based methods) and m to the number of testing samples.

2.1 FkNN

The decision rule of **FkNN** is simply the majority rule computed on the neighborhood of the testing point retrieved in the feature space:

$$\textsf{\textbf{FkNN}}(x) = \text{sign}\left(\sum_{i=1}^{k} y_{r_x(i)}\right).$$

Computational complexity: the bound on the complexity of the proposed implementation of **FkNN** which uses the Cover Tree data-structure (like all the other modules), is $\mathcal{O}(n \log n + m \cdot k \log n)$.

2.2 FkNNSVM

The local SVM decision function, as defined in (Blanzieri and Melgani; 2006, 2008), for a point x and a training set X, is:

$$\textsf{\textbf{FkNNSVM}}(x; X) = \text{sign}\left(\sum_{i=1}^{k} \alpha_{r_x(i)} y_{r_x(i)} K(x_{r_x(i)}, x) + b\right)$$

where $r_t(i)$ is the ordering function defined by Eq. 1 and $\alpha_{r_t(i)}$ and b come from the training of an SVM on the k-nearest neighbors of x in the feature space.

Computational complexity: the bound on the complexity of the proposed implementation of **FkNNSVM** is $\mathcal{O}(n \log n + m \cdot k \log n + m \cdot k^3)$.

2.3 FkNNSVM-nr

The probabilistic variant of **FkNNSVM** on which **FkNNSVM-nr** (Segata, Blanzieri, Delany and Cunningham; 2008) is based, can be obtained following a refinement of the Platt (1999) method developed by Lin et al. (2007):

$$\widehat{p}^{\,\textsf{\textbf{FkNNSVM}}}(y = +1|x; X) = \frac{1}{1 + \exp(A \cdot \textsf{\textbf{FkNNSVM}}(x; X) + B)}.$$

Choosing a threshold $\gamma \in \mathbb{R}$ with $0.0 \leq \gamma \leq 1.0$ the decision of removing a sample (corresponding to the negative output) is based on the probability that the predicted label of a training point x_i is the true label accordingly the SVM built on its neighborhood (not considering x_i itself):

$$\textsf{\textbf{FkNNSVM-nr}}(x_i; X) = \text{sign}\left(\widehat{p}^{\,\textsf{\textbf{FkNNSVM}}}(y = y_i | x_i; X \setminus \{x_i\}) - \gamma\right)$$

Computational complexity: the bound on the complexity of the proposed implementation of **FkNNSVM-nr** is $\mathcal{O}(n \cdot k \log n + n \cdot k^3)$.

2.4 FaLK-SVM

FaLK-SVM precomputes a set of local SVMs in the training set and assigns to each model all the points lying in the central neighborhood of the k points on which it is trained. The prediction is performed applying to the query point the model corresponding to its nearest neighbor in the training set. The formal definition, based on Segata and Blanzieri (2009b) starts from a generalization of **FkNNSVM** for applying an SVM to a point x possibly different from the point t for which the neighborhood is retrieved:

$$\textsf{\textbf{FkNNSVM}}_t(x) = \text{sign}\left(\sum_{i=1}^{k} \alpha_{r_t(i)} y_{r_t(i)} K(x_{r_t(i)}, x) + b\right).$$

The decision function of **FaLK-SVM** is:

$$\textsf{\textbf{FaLK-SVM}}(x) = \textsf{\textbf{FkNNSVM}}_c(x) \quad \text{with } c = cnt(x_{r_x(1)})$$

where *cnt* is the function that retrieve the local SVM model defined as

$$cnt(x_i) = x_j \in \mathcal{C}$$
$$\text{with } j = \min\big(z \in \{1, \ldots, n\} \big| x_z \in \mathcal{C} \text{ and } \nexists c \in \mathcal{C} \text{ s.t. } r_{x_i}(c) < r_{x_i}(x_z)\big)$$

and $\mathcal{C}$ is the set of the centers of the models selected as follows:

$$c_i = x_j \in X$$
$$\text{with } j = \min\big(z \in \{1, \ldots, n\} \big| x_z \in X_{CT} \setminus X_{c_i}\big)$$
$$\text{where } X_{c_i} = \bigcup_{l<i} \left\{ x_{r_{c_l}(h)} \mid h = 1, \ldots, k' \right\}.$$

where $1 \leq k' \leq k$ and X_{CT} is the training set reordered using the separation invariant of Cover Tree which assures to take as centers points that are as distant as possible thus further reducing the number of local SVM that need to be trained. **FaLK-SVM** is divided in two submodules: the training module **FaLK-SVM-train** which takes the training set and builds the model and the **FaLK-SVM-predict** which uses the model to perform the predictions. We also implemented a faster variant of **FaLK-SVM-predict** which perform the nearest neighbor operations to retrieve the trained local model not with respect to all the training points but with the points in $\mathcal{C}$ only.

Computational complexity: the bounds on the complexity of the proposed implementations of **FaLK-SVM-train** and **FaLK-SVM-predict** are $\mathcal{O}(n \log n + |\mathcal{C}| \cdot k \log n + |\mathcal{C}| \cdot k^3)$ and $\mathcal{O}(m \cdot k \log n)$ respectively which reduce to $\mathcal{O}(n \log n)$ and $\mathcal{O}(m \log n)$ for reasonably low k values. The variant of **FaLK-SVM-predict** retrieving the local models using the centers only, takes $\mathcal{O}(m \cdot k \log |\mathcal{C}|)$.

2.5 FaLKNR

FaLKNR is a version of **FkNNSVM-nr** without the probabilistic approach which is applicable for large and very large datasets thanks to the integration of the approach of **FaLK-SVM** as detailed by Segata, Blanzieri and Cunningham (2009). The removal of a sample $x_i \in X$ (corresponding to the negative output) is based on the following formulation:

$$\textbf{FaLKNR}(x_i) = 1 - |\textbf{FkNNSVM}_c(x_i) - y_i|$$

Computational complexity: the bound on the complexity of the proposed implementation of **FaLKNR** is $\mathcal{O}(n \log n + |\mathcal{C}| \cdot \log n \cdot k + |\mathcal{C}| \cdot k^3 + k \cdot n)$ which is $\mathcal{O}(n \log n)$ for reasonably low k values.

2.6 Tools common to all modules

The following tools are common to all (or some of) the **FaLKM-lib v1.0** modules.

Multi-class classification. The generalization from binary class classification to multi-class classification is straightforward for the kNN-based techniques, whereas for the local SVM models we use LibSVM (Chang and Lin; 2001) which handles multi-class classification with the one-versus-all approach.

Model selection. The traditional model selection is implemented with cross validation and can be applied to the SVM regularization parameter C, to the kernel parameters and to the neighborhood size k.

Local model selection. The local model selection is implemented with various options; basically a local cross validation step is performed on a subset (or all) of the local models in the training set. The local κ-fold cross validation consists of the following steps:

1. the k' most internal samples, called S, are separated from the remaining external points, called S^E;
2. S is randomly splitted in κ disjoint internal validation sets S_i with $0 < i < \kappa$;

3. for each fold i a model is trained on $S^E \cup (S \setminus S_i)$ and evaluated on the correspondent S_i set, taking the mean of the accuracies on the κ folds. A particular strategy for locally tune the width parameter of the Gaussian RBF kernel based on the histogram of the distances of the local neighborhood is also available.

Classification measures. The classification modules and the (local) cross validation gives as output the classification accuracy and (if a binary class dataset is used) the precision, recall and F-measure.

3 Benchmarks

Benchmarks of computational performances, classification accuracy, and noise reduction ability of preliminary versions of **FaLKM-lib v1.0** modules can be found in the correspondent papers. The empirical comparisons are made with kNN, SVM and approximated SVM algorithms like (Tsang et al.; 2005; Bordes et al.; 2005) for the classification modules and with tradition case-based competence enhancing techniques for the noise reduction modules. The performances of the last version of **FaLKM-lib v1.0** are however slightly better and the current benchmark results are available on **FaLKM-lib v1.0** home page at `http://disi.unitn.it/~segata/FaLKM-lib`.

4 Acknowledgements

The author of **FaLKM-lib v1.0** would like to thank Enrico Blanzieri for the fruitful discussions related to local approaches for learning with kernels and his crucial support in the implementation work, and Pádraig Cunningham for his suggestions about the implemented noise reduction algorithms.

References

Beygelzimer, A., Kakade, S. and Langford, J. (2006). Cover Trees for Nearest Neighbor, *Internationa Conference on Machine Learning (ICML-06)*, ACM Press New York, NY, USA, pp. 97–104.

Blanzieri, E. and Melgani, F. (2006). An adaptive SVM nearest neighbor classifier for remotely sensed imagery, *IGARSS 2006*, pp. 3931–3934.

Blanzieri, E. and Melgani, F. (2008). Nearest neighbor classification of remote sensing images with the maximal margin principle, *IEEE Transactions on Geoscience and Remote Sensing* **46**(6).

Bordes, A., Ertekin, S., Weston, J. and Bottou, L. (2005). Fast kernel classifiers with online and active learning, *Journal of Machine Learning Research* **6**: 1579–1619.

Bottou, L. and Vapnik, V. (1992). Local learning algorithms, *Neural Computation* **4**(6).

Brighton, H. and Mellish, C. (2002). Advances in instance selection for instance-based learning algorithms, *Data Mining and Knowledge Discovery* **6**(2): 153–172.

Chang, C. C. and Lin, C. J. (2001). *LIBSVM: a library for support vector machines. Software available at* `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

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

Cunningham, P., Doyle, D. and Loughrey, J. (2003). An evaluation of the usefulness of case-based explanation, *In Proceedings of the Fifth International Conference on Case-Based Reasoning*, Springer, pp. 122–130.

Gamberger, A., Lavrac, N. and Dzeroski, S. (2000). Noise detection and elimination in data preprocessing: experiments in medical domains, *Applied Artificial Intelligence* pp. 205–223.

Leake, D. B. (1996). CBR in context: The present and future, *in* D. B. Leake (ed.), *Case Based Reasoning: Experiences, Lessons, and Future Directions*, MIT Press, pp. 3–30.

Lin, H. T., Lin, C. J. and Weng, R. (2007). A note on platt's probabilistic outputs for support vector machines, *Machine Learning* **68**(3): 267–276.

Lorena, A. C. and Carvalho, A. C. P. L. F. d. (2004). Evaluation of noise reduction techniques in the splice junction recognition problem, *Genetics and Molecular Biology* **27**: 665 – 672.

Malossini, A., Blanzieri, E. and Ng, R. T. (2006). Detecting potential labeling errors in microarrays by data perturbation, *Bioinformatics* **22**(17): 2114–2121.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, *Advances in Large Margin Classifiers*, pp. 61–74.

Segata, N. and Blanzieri, E. (2009a). Empirical assessment of classification accuracy of Local SVM, *The 18th Annual Belgian-Dutch Conference on Machine Learning (Benelearn 2009).* Accepted for Pubblication.

Segata, N. and Blanzieri, E. (2009b). Fast local support vector machines for large datasets, *International Conference on Machine Learning and Data Mining (MLDM 2009).* Accepted for Pubblication.

Segata, N., Blanzieri, E. and Cunningham, P. (2009). A scalable noise reduction technique for large case-based systems, *International Conference on Case-Based Reasoning (ICCBR 09)*. Accepted for Pubblication.

Segata, N., Blanzieri, E., Delany, S. and Cunningham, P. (2008). Noise reduction for instance-based learning with a local maximal margin approach, *Technical Report DISI-08-056*, DISI, University of Trento, Italy.

Tang, S. and Chen, S.-P. (2008). Data cleansing based on mathematic morphology, *2nd International Conference on Bioinformatics and Biomedical Engineering (ICBBE 2008).*, pp. 755–758.

Tsang, I. W., Kwok, J. T. and Cheung, P. M. (2005). Core Vector Machines: Fast SVM Training on Very Large Data Sets, *The Journal of Machine Learning Research* **6**: 363–392.

Vapnik, V. (2000). *The Nature of Statistical Learning Theory*, Springer.

Wilson, D. R. and Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms, *Machine Learning* **38**(3): 257–286.