CMPE59H - Bioinformatics
Project Report
Multiple Kernel Learning for Extracting Protein-protein Interactions

Göker Erdoğan

14.01.2012

1 Introduction

Proteins are the workforces of organisms taking crucial roles in vast amounts of biological processes. Knowing that two proteins interact carry highly important information both theoretically and practically. Understanding inner workings of organisms, predicting functions of proteins and designing drugs are some of the problems that benefit greatly from such knowledge [8, 9, 10]. Since carrying out experiments to confirm an interaction is costly and labor intensive, given the amount of possible interactions, other methods for extracting protein-protein interactions (PPI) are valuable. One source of information for finding interacting proteins is biomedical literature where a vast amount of possible interactions are documented by researchers. However, again this procedure for mining PPIs from literature requires a lot of human labor and there are many methods developed for extraction of PPIs by automated software. One approach to PPI extraction relies on machine learning and natural language processing methods to learn models discriminating between positive and negative interactions based on linguistic features of sentences. These features span a wide spectrum from shallow features such as frequency of words, part of sentence (POS) tags to syntax trees and dependency graphs of sentences. These features obtained from text are used for training learning algorithms to separate false interactions from possible interactions. A high amount of research is carried out on extracting features that convey the information necessary for discrimination as much as possible. Literature is filled with kernel functions; which measure the similarity between two feature vectors, for extraction of PPIs. Tikk et al. [12] present a comprehensive benchmark of such kernels in literature to assess performance of these from multiple perspectives. After the dazzling success of support vector machines [4], machine learning researchers built a wide, deep body of knowledge in kernel based learning methods. A fruitful approach in this regard has been multiple kernel learning [1] where a combination of kernels are utilized for learning better models. We believe that problem of PPI extraction; given the high number of different kernels, may benefit from such an approach. In this project, we apply multiple kernel learning algorithms on 3 kernels that use different linguistic features mentioned in [12] and analyze the results from accuracy and kernels' importance perspective.

2 Method

In this section, we will first introduce the kernels we will be employing for PPI extraction and then present a summary of multiple kernel learning theory before detailing the specific multiple kernel methods we will be using.

2.1 Kernels

A kernel function $k(x_i, x_j) : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ measures the similarity between two vectors; x_i and x_j mapping from possibly high dimensional input space of vectors to real values [1]. Kernel functions have great signifi-

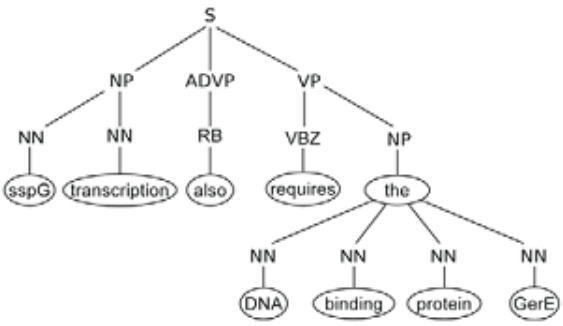

Figure 1: Syntax tree of sentence *SsgG transcription also requires the DNA binding protein GerE.* [12]

cance due to their role in support vector machines where they can transform a non-linear problem to linear space. Tikk et al. [12] present a multitude of kernels based on various linguistic features for PPI extraction. We choose 3 of these kernels in our work. We have picked one kernel from each of the linguistic feature types; shallow, syntax tree and dependency graph and tried to make sure that each one is a top performer among the kernels that are based on the same linguistic features.

2.1.1 Shallow Linguistic Kernel

This kernel is the simplest and least computationally expensive one which is based only on shallow parsing information [6]. Actually, similarity value calculated by the kernel is sum of two kernels; global context and local context kernel. Global context kernel uses bag of words representation to calculate the number of common words between two PPIs from different places in the sentence. 3 subsets of the sentence are considered; from the two proteins in the interaction, all the words before the first one appearing in the sentence and words between the proteins constitute the first subset. Secondly, only words in between two proteins are considered. Lastly, words between the proteins and after the second protein make up the last subset. Number of common words between these 3 subsets in two sentences are called the global context kernel. Local context kernel uses POS tag, lemma of the words to the left and right of proteins in interaction and applied scalar product to obtain kernel value. Finally, values for global and local context kernels are summed to form shallow linguistic kernel.

2.1.2 Subtree Kernel

Based on syntax tree representation of sentences, subtree kernel counts the number of common subtrees in two sentences to calculate a similarity measure [13]. A subtree is considered to be a node with all its descendants in the tree and two subtrees are considered identical if the node labels and order of children are identical for all nodes.

2.1.3 k-band Shortest Path Spectrum Kernel (k-BSPS)

The most complex of the kernels we will be experimenting with uses walks on shortest path between nodes denoting the proteins in the dependency graph [12]. All walks of length in $[q_{min}, q_{max}]$ lying on the shortest path between proteins are compared with the walks of same length from other sentence to calculate the similarity score. Nodes that are within distance k from the shortest path are also included when finding all walks of a certain a length. When two walks are being compared matches and mismatches are weighted according to the type of node; dependency type (D), entity (E) and other surface tokens (L). Besides, it is possible to specify some mismatches as intolerant meaning that such a mismatch sets the similarity score between two walks to 0. Kernel value for two PPIs are calculated with the following formula where L, E, D sets if that type of mismatch is tolerated (0 being tolerated, -1 being in-tolerated), l, e, d are the weights matches and mismatches of L, E, D types and p^q is the set of walks of length q from k-band shortest path

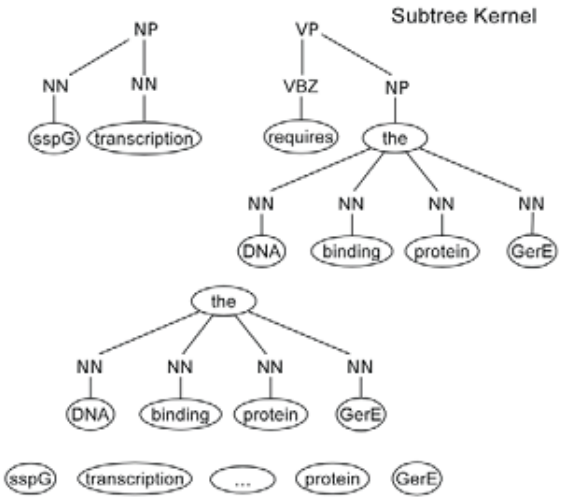

Figure 2: Subtrees of syntax tree in figure 1 [12]

of pair p.

$$S(p_i, p_j) = \sum_{q=q_{min}}^{q=q_{max}} max_{i \in p_i^q, j \in p_j^q}(score_{L,E,D,l,e,d}(i,j)) \tag{1}$$

2.2 Multiple Kernel Learning

Support vector machine (SVM) [4] is a highly researched and widely popular classification algorithm that learns the linear discriminant with maximum margin between two classes. Given a sample of N training instances $\{(x^t, y^t)\}_{t=1}^N$ where x^t are D-dimensional input vectors and $y^t \in \{-1, +1\}$ are class labels, SVM learns the linear boundary of the following form where w is the vector of coefficients and b is the bias term.

$$f(x) = wx + b$$

Maximizing the distance of training samples from the boundary can be achieved by the following quadratic optimization problem [1].

$$\begin{aligned} \text{minimize } & \frac{1}{2}||w||_2^2 + C\sum_{t=1}^{N}\xi^t \\ \text{w.r.t } & w \in \mathbb{R}^D, \xi \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to } & y^t(wx^t + b) \geq 1 - \xi^t \ \ \forall t \end{aligned}$$

In the above problem, ξ denotes slack variables and C is a parameter of the method that adjusts between model simplicity and classification error. Writing the dual formulation of this problem and solving it results in the following discriminant function where decision boundary is defined as a weighted sum of training samples.

$$f(x) = \sum_{t=1}^{N} \alpha^t y^t x^t x + b$$

In the above function α^t is the weight of training sample x^t. After training only some of the α^t values are non-zero and these training samples are called support vectors. In addition, discriminant actually does not directly use feature representations of input samples since it only needs $x^t x$ scalar product which

measures the similarity between two samples. It is possible to replace this product with any kernel function $k(x^t, x^p) : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ which enables us to learn non-linear boundaries by mapping input samples to a new space with a non-linear kernel function. Moreover, quadratic optimization problems can be solved optimally meaning that we do not need to resort to locally optimal learning algorithms when solving SVMs.

In recent years, multiple kernel learning (MKL) methods, where instead of using a single kernel for training a combination of kernels are used, drew much attention. By combining multiple kernels we may aim to find the kernels which work best for our problem by letting the MKL method to do the choosing. Besides, different kernels may depend upon different feature representations of input data thus providing more information to algorithm and making it possible to learn a better boundary. Now, instead of using only values of a single kernel $k(x^t, x^p)$, we define a new kernel by using a function f_{comb} that combines information from multiple kernels k_m into a single one.

$$k(x^t, x^p) = f_{comb}(\{k_m(x^t, x^p)\}_{m=1}^P)$$

There are many variants of multiple kernel learning algorithms in literature each varying in a way from others. When viewed from a broader perspective these variants fall into specific categories from different perspectives. When we focus on the combination functions employed by different MKL methods, we can say that roughly 3 categories exist. In fixed combination, single kernels are combined in a predefined manner and an SVM is trained on the obtained kernel. Some methods use heuristics based on various information to find the relative weights of each kernel in combination while methods in third category combine kernels by taking their linear, conic or convex sums [1]. From training perspective, we can speak of two classes of methods where two-step methods calculate weights and train SVM to find support vectors as separate steps while coupled methods learn the parameters of combination function and support vectors simultaneously. For the purpose of combining kernels for PPI extraction, we will review 3 MKL methods one from each combination category.

2.2.1 Rule Based Multiple Kernel Learning (RBMKL)

In rule based multiple kernel learning (RBMKL), Cristianini et al [5] propose to take the mean or product of single kernels to use it as our combined kernel. Then, an SVM is trained on the obtained kernel. We use mean combination rule in our experiments.

$$k(x^t, x^p) = \frac{1}{P} \sum_{m=1}^{P} k_m(x^t, x^p)$$

2.2.2 Alignment Based Multiple Kernel Learning (ABMKL)

He et al. [7] present a solution to finding the kernel weights in the combination function by minimizing the distance between the combined kernel and ideal kernel matrix with the following optimization problem. Ideal kernel for a binary classification task is given by yy^T.

$$\begin{aligned} \text{minimize } & \|K_w - yy^T\|_F^2 \\ \text{w.r.t } & w \in \mathbb{R}_+^P \\ \text{subject to } & \sum_{m=1}^{P} w_m = 1 \end{aligned}$$

ABMKL learns a combined kernel K_w which is a convex combination of individual kernels. During training, firstly optimal kernel weights are found via above problem and then an SVM is trained with the combined kernel.

	Positive pairs	Negative pairs
AIMed	1000	4834
LLL	164	166

Table 1: Data sets used for evaluation

2.2.3 Multiple Kernel Learning by Bach et al. [2] (BMKL)

BMKL method by Bach et al. [2] learns the support vectors and kernel weights simultaneously with the following optimization problem

$$\begin{aligned} \text{minimize } & \frac{1}{2}(\sum_{m=1}^{P} \|w_m\|_2)^2 + C\sum_{t=1}^{N}\xi_i \\ \text{w.r.t } & w \in \mathbb{R}^{S_m}, \xi \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{subject to } & y^t(\sum_{m=1}^{P} w_m x^t + b) \geq 1 - \xi_i \ \forall i \end{aligned}$$

where S_m is the dimensionality of the feature space of kernel K_m. BMKL method finds a linear combination of input kernels.

3 Experiments

For evaluating multiple kernel methods on PPI extraction task, we have used two widely used data sets; AIMed and LLL [3, 11]. We have obtained kernel gram matrices for these data sets by modifying the source code provided in the benchmark study [12]. From the kernels we have experimented with, SL does not use any parameters. For ST and k-BSPS optimum parameter values reported are used (for SL; $c = 1$,$j = 2$,$\lambda = 0.4$,$\mu = 0.4$ and for k-BSPS; $L = 0, E = 0, D = 0, l = 1, e = 6, d = 3, q_{min} = 0, q_{max} = 2, k = 0, j = 2$). Multiple kernel learning methods detailed in the previous section are run with 10 fold CV on the splits provided by Tikk et al. For cost parameter of kernel methods, we have executed a parameter search on values $\{0.01, 0.1, 1, 10, 100\}$. In the results, we observe area under the ROC curve and F_1 measures as well as kernel weights found by the methods.

4 Results

In table 4, we provide F_1 measure values for each kernel trained independently and all three kernels combined with three different multiple kernel learning methods. Firstly, F_1 values we reach for single kernels are lower than the reported figures in [12] which may be attributed to lack of parameter fine tuning of kernels and methods. Also, since our purpose is to observe the contribution of multiple kernel learning methods rather than reaching higher accuracies, we did not strive to match the reported performances. When we analyze results for single kernels, we see that SL being simplest kernel performs best or comparably in two data sets. Subtree kernel (ST) exhibits the lowest performance while most complex kernel k-BSPS performs best or second in two data sets. These results also confirm the finding by Tikk et al. where SL kernel is one of the performers.

From the three multiple kernel learning methods, RBMKL and ABMKL consistently performs lower than the highest accuracies for single kernels. This result implies that combining kernels in a fixed manner or with an heuristic may not be able to exploit the information diversity among different kernels. However, BMKL method always matches or reaches a higher performance than single kernels showing us that combining multiple kernels for PPI extraction is indeed well justified. The fact that BMKL's performance on LLL is similar with best single kernel performance may result from low sample size of data set and single kernels already being able to reach highest possible performance. When we observe the weights of kernels in the combination for BMKL and ABMKL method in table 4, SL always gets a much higher weight while k-BSPS

	SVM with SL	SVM with ST	SVM with k-BSPS	BMKL	RBMKL	ABMKL
LLL	0.744	0.706	**0.793**	**0.793**	0.681	0.722
AIMed	0.518	0.272	0.439	**0.545**	0.317	0.357

Table 2: F_1 Measure

	BMKL			ABMKL		
	SL	ST	k-BSPS	SL	ST	k-BSPS
LLL weights	3.4	0.3	0.1	0.57	0.4	0.03
AIMed weights	2.8	0.1	0.01	0.66	0.3	0.07

Table 3: Kernel weights

nearly vanishes in all combinations. It is rational for SL to get higher weight but it is somewhat unexpected for k-BSPS kernel being the only kernel based on dependency graph to have such low weight. It may be speculated that SL and ST kernels together provide the information represented by dependency graphs but this claim requires more experimentation and formal arguments to support it. In table 4, we provide the results for area under the ROC curves (AUC) for each method and kernel. We omit its discussion since it is similar to above analysis and AUC is not a reliable measure for PPI extraction problems. This work may be extended largely by applying the multiple kernel methods on other data sets and including more MKL methods in evaluations.

	SVM-sl	SVM-st	SVM-kbsps	BMKL	RBMKL	ABMKL
LLL	0.825	0.701	**0.837**	**0.837**	0.804	0.772
AIMed	0.830	0.659	0.739	**0.833**	0.596	0.655

Table 4: Area under the ROC curve

References

[1] Ethem Alpaydin and Mehmet Gonen. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12, 2011.

[2] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 6–, New York, NY, USA, 2004. ACM.

[3] Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun Kumar Ramani, and Yuk Wah Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine (special issue on Summarization and Information Extraction from Medical Documents)*, (2):139–155, 2005.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. 10.1007/BF00994018.

[5] Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

[6] Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, Trento, Italy, April 2006.

[7] Junfeng He, Shih-Fu Chang, and Lexing Xie. Fast kernel learning for spatial pyramid matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –7, june 2008.

[8] Robert Hoffmann, Martin Krallinger, Eduardo Andres, Javier Tamames, Christian Blaschke, and Alfonso Valencia. Text Mining for Metabolic Pathways, Signaling Cascades, and Protein Networks. *Sci. STKE*, 2005(283):pe21+, May 2005.

[9] Trey Ideker and Roded Sharan. Protein networks in disease. *Genome Research*, 18(4):644–652, April 2008.

[10] Samira Jaeger, Sylvain Gaudan, Ulf Leser, and Dietrich Rebholz-Schuhmann. Integrating protein-protein interactions and text mining for protein function prediction. *BMC Bioinformatics*, 9(Suppl 8):S2, 2008.

[11] C. Ndellec. Learning language in logic - genic interaction extraction challenge. In *Proceedings of the Learning Language in Logic 2005 Workshop at the International Conference on Machine Learning*, 2005.

[12] Domonkos Tikk, Philippe Thomas, Peter Palaga, Jörg Hakenberg, and Ulf Leser. A comprehensive benchmark of kernel methods to extract proteinprotein interactions from literature. *PLoS Comput Biol*, 6(7):e1000837, 07 2010.

[13] S. V. N. Vishwanathan and Alex Smola. Fast Kernels for String and Tree Matching. *Advances in Neural Information Processing Systems*, 15, 2003.