1 Kernels for structures

A possible taxonomy of kernels for structures:

- Two papers have nothing to do with SVM or kernels: [8, 20].
- Two papers (the earliest ones) establish closure properties of kernels and *general mathematical design criteria* [5, 23]. Thereby, they are used to justify both, structure-based kernels and some kernels based on probabilistic models.
- *Structure based kernels:* i.e. kernels which directly refer to parts of the structure and compute distance measures comparing parts of the structures [2, 3, 4, 11, 12, 13, 14, 15, 19, 21, 22, 24, 25]. These can be further divided into:
  - kernels for *fixed structures* (e.g. sequences of fixed length) which enlarge standard kernels for fixed dimensional vectors e.g. incorporating local correlations. An important point is that two structures can in principle be compared via comparison of their respective elements (like in real vectors), the fact that the elements are contained in a structure (hence e.g. correlated to their neighbors) might propose specific interesting correlations. This includes:
    - [17, 25]: the *locality improved kernel* has been proposed for the comparison of sequences of the same length with elements in a finite alphabet. It measures the elementwise matching of the symbols in two strings and incorporates local correlations of nearby comparisons instead of global correlations.
    - [21]: a *P-kernel based on rare common substrings* for strings is proposed which compares strings of the same length with finite elements. The notion 'P-kernel' just refers to the fact that P-kernels over the product of a finite set X can be interpreted as joint probability distributions. The kernel extracts a number of (fixed positional) subsets of the indices, compares the respective entries of the two seqeunces at these positions and, in the case of matching, incorporates the probabilities for the strings and substrings. Thereby, rare substrings are emphasized. Efficient computation of this method is possbile for specific settings (independent variables, Markov chains and successive blocks via dynamic programming).
    - [19]: the *phylogenetic tree kernel* compares evolutionary profiles (i.e. labelings of the leaves of a phylogenetic tree, which correspond to the presence or absense of a gene in the respective species) for one fixed phylogenetic tree structure. This is basicaly a generalization of the previous approach to tree structures. Whereby some more work is necessary since only the labeling of the leaves, but not labeling of the entire tree is available. The basic idea is that each profile can in principle be extended to a labeling of the whole tree (respectively the probabilities for expansions can be computed, if some basic model about the probabilities attached to the edges and all their possible labeling is fixed. Here this is just some model which assumes fixed probabilities for vanishing of a gene/new invention of a gene.) Then the common occurences of rooted subtrees (i.e. subgraphs of the tree which contain the root and form tree structures) are summarized with appropriate factors. Thereby propbabilities have to be used because the labeling of interior nodes is unknown. Efficient computation is possible for Bayesian trees with dynamic programming.
  - Although the latter two approaches are designed for fixed structures, they are similar in spirit to a number of kernels which compare structures of different size comparing *occurences of substructures*. Essentially, a number of possible substructures (e.g. subsequences of length n) is fixed which might occure in a string at an arbitrary position. The vector which collect the information whether these substructures are present (somewhere in the structure) or not form the feature vectors. Thereby, the methods have all been proposed for strings of possibly different length over a finite alphabet. They differ with respect to the fact whether substrings of consecutive entries are compared or gaps may be included, whether a perfect match or partial match of substrings is considered, and how the kernels are computed efficiently (based on dynamic programming or a sparse feature representation).

    - [4, 11, 12, 22]: the *spectrum kernel* considers substrings of consecutive values of length k. Thereby, mismatches are allowed and punished according to expert knowledge (in this case given by the BLOSUM matrices from computational biology – note that this incorporates probabilistic information about the data). Efficient computation is possible via suffix trees, which represent the possible features in a sparse way. The paper [4] allows mismatches at specific (previously defined) positions of the strings. The paper [22] just describes the eficient computation of the above kernel (and similar kernels for tree structres) via suffix trees.
    - [13, 14, 15]: here *string kernels* which measure the coocurrence of substrings which might be not contiguous are defined. Thereby, [15] also incorporates the notion of partial matches (since the kernel works on syllables as simplest units instead of letters). Efficient computation is always done based on dynamic programming. Interestingly, [15] finds a stochastic interpretation such that closenes to Fisher-kernels is also established.
    - [2, 24]: these articles use the same ideas, detecting of comon substructures, but for *tree models*. Thereby they use also dynamic programming for efficient computation. [2] can be interpreted as a mixture between fixed structure and variable structure approaches: the respective parents of the trees are directly compared, but matching children might occur somewhere (due to parse trees where the top has a fixed meaning which can directly be compared).
  - kernels arising from *structural transduction*. A fairly general aproach is [3] which proposes a distance measure which occurs from the products of weights if structural transduction (along all possible paths) from one structure to the other one is done. Thereby, symmetrie is to be ensured, hence the transducter is composed with its inverse. Alternatively formulated, the sum of the weights for all possible linking structures is computed. Efficient computation is again possible with dynamic programing. This kernel contains as a special case kernels based on substrings. However, the general formulation is different, focussing on structure transformation (edit distance etc.) than on (static) common substructures.

- *Kernels based on probabilistic models:* these methods start from a probabilistic model of the data and either derive a kernel classifier directly from probabilistic decision models (e.g. via linearization) or use parameters of the probabilistic model instead of the original data to compare the datae. Training usually incorporates information about the data and might be computationally expensive if e.g. statistical models are to be fitted to compute the kernel. Nevertheless, they combine benefits of two approaches, precise and efficient clustering via e.g. SVM and maximum information extraction via statistical modelling. These models include [1, 6, 7, 9, 10, 16, 18]. Note that some of the above kernels (e.g. string kernels) can be interpreted as kernels derived from probabilistic models, and some kernels involve statistical elements (e.g. weighting schemes, transition probabilities).

  - kernels derived as possibly improved *implementation of a statistical decision model* start with a statistical formulation and transform the problem until it has been reduced to a classification problem including kernels (possibly with statistical elements). This includes:
    - [1]: this paper does not propose a kernel for structures but uses standard kernels (polynomial or Gaussian). However, the SVM is used in an unusual way, breaking down a probabilistic problem to a standard SVM-problem. The paper proposes *generalized linear discriminants kernels* for speaker identification. It starts based on the assumption that a generalized linear discriminant (e.g. an SVM with some simple e.g. polynomial kernel) for the probability P of an output, given a speaker is given. Using Bayes rule and some computation and Taylor approximation this allows to efficiently compute the probability of a speaker, given data. An SVM (with standard kernel) is know simply trained to give the above probability P, whereby the kernel is not adapted. But the training method is adapted because the expectation over a number of observations is to be learned in practice. This approach does not require complex fitting of probabilstic models.
    - [17]: the *tangent vector of log-odds* also starts from the formulation of the classification problem (classification into two classes) as an optimum decision given probabilistic mod-

els for both classes. Applying Bayes law and Taylor approximation allows to formulate this problem as a linear classification problem where data are previously transformed via a kernel. This kernel is very similar to the Fisher kernel. It refers to the derivative of generative models for both classes. In the paper, these are obtained as hidden Markov models for sequences trained previously on the data. In principle, every type of data can be tackled in this way if generative models for the two classes and its derivative are available.

- A very popular model which is derived from general mathematical considerations and which tackles basically all types of data is the *Fisher kernel* as introduced in [6, 7]. The idea is to substitute the comparison of structures by the comparison of relevant parameters of a generative model fitted to the data. In the case of the Fisher kernel, a manifold of parameterized generative models is considered. A model fitted to the data corresponds to one point in this manifold. The tangent space at this point (derivatives of the model with respect to the parameters at given data points) can be identified with the data points, hence structures can be compared comparing the derivatives, i.e. the vectors which indicate how much the parameters influence the respective given structures to be compared. Thereby, the natural metric is given by the metric in the manifold which is related to the Fisher information matrix (but this is often dropped in applications). This formulation has the advantage that it is independent of the paramaterization of the models. Since it is a fairly general approach, one can embedd some string kernels into it assuming appropriate probabilstic models.
  - [6, 7]: these papers invent the Fisher kernel. It show some nice properties like the fact that no information relevant for differentiation of structures is lost if one goes from the generative models to the Fisher kernel if the label is contained as latent variable. The kernel is applied including hidden Markov models as generative model.
  - [16]: just another application and tests of the sensitivity with respect to training parameters of SVM, the kernel is thereby based on a single Gaussian resp. a mixture of Gaussians.
  - [17]: an application for DNA sequences, again based on hidden Markov models.
  - [18]: the *marginalized kernel* just describes the general principle that, given a kernel for the data and latent variables, one can get a kernel for the data taking expectation. Thereby, the kernel started with is not further defined. Starting with the Fisher kernel for the joint variables, one gets the Fisher kernel for the data taking expectation as shown in the paper.
- *Diffusion kernel* as introduced in [9, 10] can be seen as a generalization of the Fisher kernel which does not only take local information of the manifold into account, but also global information. One can show that a kernel exists which fulfills heat equations (differential equations in the Fisher kernel) and hence incorporates infromation spread along the manifold. This kernel can be approximated using a Taylor expansion. ...I did not understand this approach ... ;-)

Possible lines along which the kernels can be distinguished are

- which type of data are dealt with (sequences/trees/graphs)?
- fixed or variable structures?
- what is the complexity of training?
- is the kernel adaptive for the given data?
- robustness with respect to noise?

References

[1] W.M.Campbell. Generalized linear discriminant sequence kernels for speaker recognition.

[2] M.Collins and N.Duffy. Convolution kernels for natural language.

[3] C.Cortes, P.Haffner, and M.Mohri. Rational kernels.

[4] R.Fink, S.Merchant, S.Mohiuddin, and S.Sidiroglou-Douskos. Wildcarrd spectrum kernels for SVM protein classification.

[5] D.Haussler. Convolution kernels on discrete structures.

[6] T.Jaakkola, M.Diekhans, and D.Haussler. Using the Fisher kernel method to detect remote proteing homologies.

[7] T.Jaakkola and D.Haussler. Exploiting generative models in discriminative classifiers.

[8] I.Jonyer, A.J.Cook, and L.B.Holder. Graph-based hierarchical conceptual clustering.

[9] R.I.Kondor and J.Lafferty. Diffusion kernels on graphs and other discrete input spaces.

[10] J.Lafferty and G.Lebanon. Information diffusion kernels.

[11] C.Leslie, E.Eskin, A.Cohen, and W.S.Noble. Mismatch string kernels for SVM protein classification.

[12] C.Leslie, E.Eskin, and W.S.Noble. The spectrum kernel: a string kernel for SVM protein classification.

[13] H.Lodhi, C.Saunders, J.Shawe-Taylor, N.Cristinanini, and C.Watkins. Text classification using string kernels.

[14] C.Saunders, J.Shawe-Taylor, and A.Vinokourov. String kernels, Fischer kernels and finite state automata.

[15] C.Saunders, H.Tschach, and J.Shawe-Taylor. Syllabes and other string kernels extensions.

[16] N.Smith and M.Niranjan. Data-dependent kernels in SVM classification of speech patterns.

[17] S.Sonnenburg, G.Rätsch, A.Jagota, and K.-R.Müller. New methods for splice site recognition.

[18] K.Tsuda, T.Kin, and K.Asai. Marginalized kernels for biological sequences.

[19] J.-P.Vert. A tree kernel to analyze phylogenetic profiles.

[20] J.-P.Vert. Text categorization using adaptive context trees.

[21] J.-P.Vert. Support vector machine prediction of signal peptide cleavage site using a new class of kernels for strings.

[22] S.V.N.Vishwanathan and A.J.Smola. Fast kernels for string and tree matching.

[23] C.Watkins. Dynamic alignment kernels.

[24] D.Zelenko, C.Aone, and A.Richardella. Kernel methods for relational extraction.

[25] A.Zien, G.Rätsch, S.Mika, B.Schölkopf, T.Lengauer, and K.-R.Müller. Engineering support vector machine kernels that recognize translation initiation sites.