Kernel methods for DNA barcoding

Pavel Kuksa, Vladimir Pavlovic

Department of Computer Science

Rutgers University

Outline

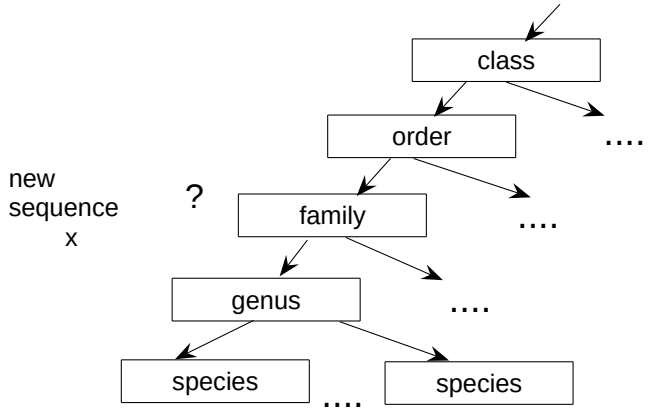
- Overview
- DNA barcoding
- Species recognition problem
- Kernel methods, string kernels
- Kernel-based species identification method
- New algorithms for string kernel computations
- Experimental results
- Conclusions

Problem definition

- Problem: species-level classification and identification of organisms based on sequence data
- We follow DNA-based approach to species identification: we examine nucleotide sequence of a given sample to determine its species identity
- Intuition behind genomic approaches to classification of species: exploit diversity among DNA sequences to identify organisms
- Motivation: transfer existing knowledge, reliable automatic and cost-effective bioidentification systems

Membership problem

- Given a new sequence, can we place it in its correct position within an existing hierarchy?



- Want to assign any unidentified organism to some taxonomic group: phylum, class, order, family, genus, species

DNA barcoding

- uses information from a standard gene region common across all taxa: in particular, mitochondrial genome of animals has been identified as a target for analysis
- DNA barcode is a short fragment of DNA obtained from the standard region and is used as a marker for identification and classification of the species
- Examples of markers: mitochondrial genes that encode ribosomal DNA, COI (cytochrome c oxidase I) gene, etc.

DNA barcoding

- Advantages of DNA barcodes: standardization, eases exchange of data, ability to continually **add in** more species and integrate data collected from different sources
- Problems that needs to be solved:
  - Accuracy of identification
  - Scalability of methods
  - Biological interpretability of the obtained models

Species recognition problem

Species identification problem definition:

- Given:
  - an unlabeled sample (specimen) X
  - a set SP of known species represented by their reference barcode sequences or models
- Want:
  - assign the given sample to one of the known species (or decide that this sample does not belong to any of the known categories).
- Our approach: we solve this global multiclass problem by dividing it into a collection of binary membership problems:
  - In a binary membership problem, the task is to decide whether an input sequence belongs to a particular class.
  - To solve each membership problem we build a binary classifi er.

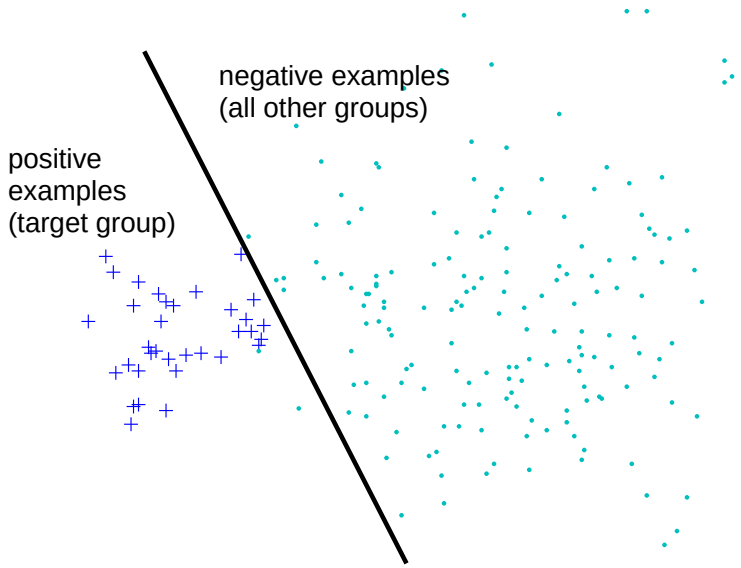
Discriminative approach

When we build binary classifiers for each of the **target taxonomic group** (one-vs-others), we adopt **discriminative approach** to the task of sequence classification

- sequences are seen as a set of **labeled** examples
  - **positive** if they are in the taxonomic group
  - **negative** otherwise

In **discriminative approach** one tries to **learn decision boundary** between positive and negative examples

Discriminative approach



How do we learn these classifiers?

Kernel methods

Kernel-based classification methods

- a general paradigm for the design of classifiers with well studied theoretical properties
- relies on the existence of positive-definite kernels, defined as inner products in feature spaces, that naturally induce **similarity metrics** among data points

Examples of kernels used: string kernels, Fisher kernels, etc.

Kernel functions

every data point in the input space (e.g. sequence) is mapped into high-dimensional space called **feature space** via some transformation $\Phi : x \rightarrow \phi(x)$
the inner product $K(x_i, x_j) =< \phi(x_i), \phi(x_j) >$ called **kernel**
Φ is called the feature map (mapping function).
In general, a kernel function is some function that corresponds to an inner product in some feature space.
Examples of typical kernel functions: RBF $exp(-\frac{(x_i - x_j)^2}{2\sigma})$, polynomial $(x_i x_j + 1)^d$, etc.
Other types of kernels: string kernels, Fisher kernels, etc.

String kernels

- String kernels compare sequences by the substrings they contain
- Examples of string kernels: exact **spectrum kernel**, **mismatch kernel**, and profile kernel
- all the string kernels have general form of

$$k(x, y) = \sum_i \phi_i(x)\phi_i(y)$$

where $\phi_i(x), i = 1...|\Phi(x)|$ are components of the feature vector $\Phi(x)$ that defines mapping of the input string x to the feature space

The k-spectrum of a sequence

The k-**spectrum of a sequence** is the set of all k-length (contiguous, $k \geq 1$) subsequences that it contains. We refer to such a k-length subsequence as a k-**mer**
Dimension of k-mer feature space is $|\Sigma|^k$, where $|\Sigma|$ is the size of alphabet:

- $|\Sigma|$ = 4 for the DNA alphabet
- $|\Sigma|$ = 20 for the alphabet of amino acids
- Example: 2-spectrum of sequence ACGT is {AC, CG, GT}

The feature map (mapping function) of spectrum kernel

k-spectrum mapping function Φ_k for sequence x:
$\Phi_k(x) : x \rightarrow [(\phi_a(x))]_{\forall a \in \Sigma^k}$
where $\phi_a(x)$ = **number of times** a occurs in x
Example: sequence ACCGT for $k = 3$ maps to $4^3 = 64$-dimensional vector:
Φ_3(ACCGT) = [AAA/0, AAC/0, ... , ACC/1, ..., CCG/1, ..., CGT/1, ..., TTT/0]
Note: sequence x is mapped to a vector with each element indicating number of occurrences of some k-long subsequence in x. Note: exact match only, no mismatches are allowed, hence projection is based on explicit features only

k-spectrum kernel

k-spectrum kernel $K_k(x, y)$ for two sequences x and y is obtained by taking the **inner product in feature space**:
$K_k(x, y) =< \Phi_k(x), \Phi_k(y) >$
This kernel counts the occurrences of k-length subsequences for each of the sequence in consideration
This kernel gives a simple notion of sequence similarity:
two sequences will have a large k-spectrum kernel value if they have many of the k-mers in common

(k,m)-mismatch kernel

Additional parameter m of mismatch kernel specifies upper bound on number of mismatches that is allowed in the counting of occurrences

Mapping function is defined first for k-mer a as:

$\Phi_{(k,m)}(a) = (\phi_b(a))_{b \in \Sigma^k}$

$\phi_b(a) = 1$ if b is within m mismatches from a, and 0 otherwise.

Example: (k,m)-mismatch neighborhood of a = ACG

Neighborhood(ACG) = {AC**A**, AC**C**, AC**T**, ..., **T**CG}

(k,m)-mismatch kernel

Mapping function for the sequence x is then defined as a sum over mappings of all k-mers in x:
$\Phi(x) = \sum_{\text{k-mers a in x}}(\Phi_{(k,m)}(a))$
(k,m)-mismatch kernel $K_{k,m}(x,y)$ is again an inner product of two vectors in the feature space:
$K_{k,m}(x,y) =< \Phi_{k,m}(x), \Phi_{k,m}(y) >$
Note: projection involves now not only explicit features, but also implicit features from the mismatch neighborhood
Note: now we take into into account possibility of substitutions that naturally occur during evolution

Support Vector Machine (SVM) Classifier

Linear classifier defined in feature space:
$f(x) =< w, \Phi(x) > +b = \sum \alpha_i K(x, x_i) + b$
where w is a linear combination of **support vectors**, a subset of the training vectors: $w = \sum \alpha_i x_i$
We want to define a mapping function from space of DNA sequences to vector space.
Our goals at this step:

- computational efficiency of training and classification
- find near perfect kernel function s.t. sequences from the same taxonomic group after mapping form distinct group(s) in the feature space, whereas different taxonomic groups at the same time are well apart from each other

Kernel-based species identification method

We use the kernel-based formalism to design of classifiers for binary species identification.
Given a training set of species barcodes SP and their corresponding species labels S, a sequence kernel is used in a SVM setting to learn a species classifier for new sequences:

$$species(x = s) = \begin{cases} \text{yes}, & \sum_{i \in M} \alpha_{i,s} k_s(x, x_i) > 0 \\ \text{no}, & \text{otherwise} \end{cases}$$

where $\alpha_{i,s}$ and $M \subseteq SP$ are estimated in the learning process using standard SVM methods.
A **critical point** in this formalism, when applied to the domain of sequences, is the **complexity of kernel computations**

Basic idea:

- cluster combined k-spectrum feature set S = $\bigcup_{i=1}^{N} Spectrum(x_i)$ and naturally find groups of features that are (k, m)-neighbors. Each cluster in the resulting partition of the combined spectrum will correspond to some particular k-mer.
- size of the resulting clusters (subclusters that correspond to different input strings) gives desired counts of number of times features occur in the input strings

- *Divide step*: set S of k-spectrum features composed of all k-mers extracted from N input sequences is partitioned into subsets $S_1, \ldots, S_{|\Sigma|}$ using character-based clustering
- *Conquer step*: The same procedure (Divide step) is applied to each of the obtained subsets recursively. At depth k of the recursion tree kernel matrix is updated at each node according to the contribution of the node feature f: $K(upd_f, upd_f) = K(upd_f, upd_f) + c_f^T c_f$ where $upd_f = \{i : f \in x_i\}$ and $c_f = [num_f(x_i)]_{i \in upd_f}$ is a vector of feature counts.

Complexity analysis

Complexity of kernel matrix computations:

- Input parameters:
  - number of sequences N
  - typical sequence length n
  - size of the alphabet $|\Sigma|$
- Kernel parameters:
  - k for spectrum kernel
  - (k, m) for mismatch kernel

Time and space complexity of kernel computations

Let c_K be the complexity of kernel computations for two sequences x and y
Two main approaches to compute kernel matrices:

- compute each element $k(x_i, x_j)$ separately: complexity of this method is then $N^2 \cdot c_K$
- compute the entire matrix through series of its updates:
  - for each unique feature (k-mer) one update is performed
  - number of updates u = number of unique features (k-mers) in the combined spectrum of all N sequences
  - complexity of updating is then $u \cdot N^2$ since each update affects the entire matrix in the worst case
  - set of unique features can be found in linear time $O(nN)$

Time and space complexity of kernel computations

Spectrum kernel matrix computation:
Brute force method: $O((kn + n^2)N^2)$ time and $O(|\Sigma^k| + nN)$ space
Suffix trees: $O(knN^2 + nN^2)$ and $O(kn + nN)$ space
Bucket sort: $\Theta(knN + uN^2)$ time and $\Theta(nN)$ space
Divide-and-conquer: $\Theta(knN + uN^2)$ and $\Theta(knN)$ space
Bottom line: No need in suffix trees! Benefits: simple and efficient implementation, no large time constants and memory requirements usually associated with suffix tree construction algorithms (e.g.., Ukkonen algorithm), scalable, online algorithms with minimal memory requirements

Time and space complexity of kernel computations

- Mismatch kernel matrix using **incremental** mismatch kernel algorithm:

  $T(n, N, k, m) = u \cdot \sum_{l=1}^{k} \sum_{i=0}^{\min(m,l)} \binom{l}{i} (\Sigma - 1)^i + u \cdot N^2$

- Complexity of mismatch kernel computation (for small $m \ll k$) can be expressed as

  $O(uk^{m+1}|\Sigma|^m + uN^2)$

Bounds on string kernels algorithms

- Previously known bounds:
  - mismatch kernel matrix computation: $O(nk^{m+1}|\Sigma|^m N^2)$
  - spectrum kernel matrix computation: $O(knN^2 + nN^2)$
- New bounds:

	Time	Space
spectrum kernel matrix	$O(knN + uN^2)$	$O(nN)$
mismatch kernel matrix	$O(uk^{m+1}\|\Sigma\|^m + uN^2)$	$O(nN)$
spectrum kernel	$O(kn + u)$	$O(n)$
mismatch kernel	$O(uk^{m+1}\|\Sigma\|^m + u)$	$O(n)$

Feature selection

Can we do better (i.e. reduce computational cost further)? Can we select small number of features while preserving accuracy of identification?

Two main approaches to feature selection:

- Feature fi ltering methods:
  - use train data statistics to score features, then select "top" features
- Wrapper methods:
  - train classifi er, eliminate features with small weights
  - repeat previous two steps until target number of features remained

Note: wrapper methods is not suitable when features do not have individual weights

Feature selection criteria

Objective: select most informative features, minimizing their number while preserving prediction accuracy

Compute score for each feature (k-mer) a as:

score(a) = $\frac{\text{number of occurrences in class}}{\text{number of occurrences in all classes}}$

and select top n features (k-mers)

Note: suitable for imbalanced data sets when number of positive and negative examples differs significantly

Note: similar to mutual information when prior is uniform

Feature selection criteria

Intuition: when we select features, we want to know how much of uncertainty regarding sequence membership is removed after seeing some particular feature: will it help us to consider this feature? if yes, then what is the amount of information we gain from looking at this sequence feature?

Mathematically, mutual information

$MI(\text{Feature, Class}) = H(\text{Class}) - H(\text{Class|Feature})$

$MI(Feature, Class) = 0$ when observing a feature does not give us any new information (feature and class are independent, $H(\text{Class|Feature}) = H(\text{Class})$)

String kernels with feature selection

Bounds for string kernel algorithms with feature selection:

- brute force algorithms: $O(|F|(knN + N^2))$
- our framework:
  - spectrum kernel with feature selection: $O(knN + |F|N^2)$ time
  - mismatch kernel with feature selection: $O(|F|(k \cdot u + N^2) + knN)$ time

Note: in case of DNA sequences, for typical values of k, n and N, maximum number of features $|\Sigma|^k \ll nN$, i.e. $u \ll nN$, which gives substantial performance improvement.

Experimental results

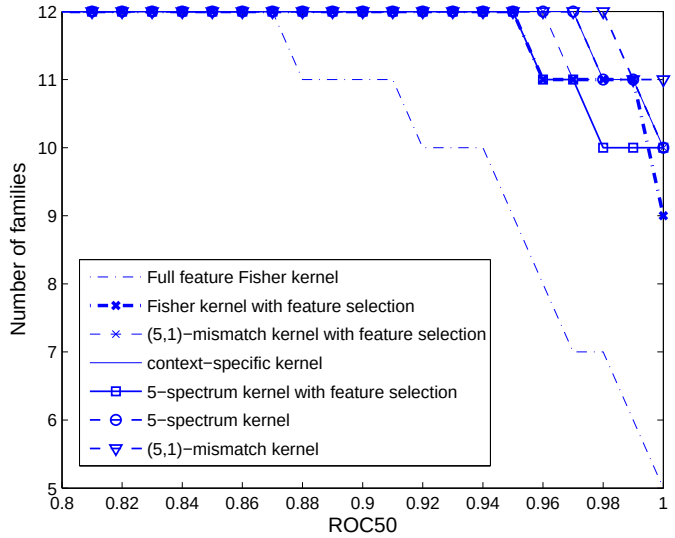
Three data sets:

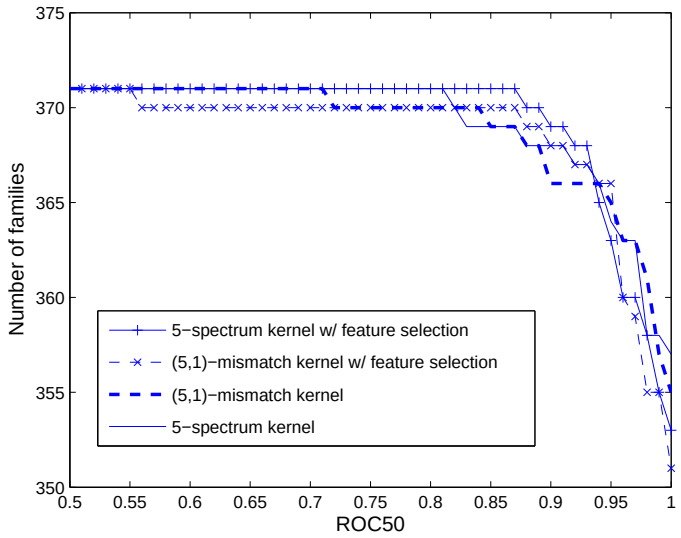
- Fish sequences (7 species, 56 barcodes)
- Astraptes dataset (12 species, 466 barcodes)
- Hesperiidae dataset (371 species, 2135 barcodes)

To measure performance of our methods we use cross-validation (CV) error rates and ROC (Receiver Operating Characteristic) scores

Experimental results: classification performance

Classification performance (accuracy of identification):





Feature selection improves performance!
We observed more than 90% reduction in the number of sequence features

Comparison of running times

Mismatch kernel computation (k=5, m=1):

Barcode dataset	Time/algorithm 1	Time/algorithm 2
Dataset 1 Astraptes N=466, n=600	21 secs	47 minutes
Dataset 2 Hesperiidae N=2135, n=600)	305 secs	> 10 hrs

Algorithm 1 = new algorithm, algorithm 2 = old mismatch algorithm

Note: machine configuration: 2.8Ghz CPU, 1GB RAM

Note: order of magnitude improvement

Related work

- Work on exact match string kernels [Leslie 02a, Vishwanathan 02]
- Work on kernels with inexact string matching [Leslie 02b, Kuang 04]
- Work of [P.D.N. 03] on biological identification through DNA barcodes

Main results

- Showed efficiency of mismatch/spectrum kernels as a sequence similarity measures
- Presented a method for efficient computation of kernel matrices $K_{N\times N}$: a divide-and-conquer algorithm for the (k,m)-mismatch kernel that improves previously known bound on running time and is more memory efficient and easy to implement
- Using counting statistics of mismatch kernel, we have extracted small subsets of important k-mers in order to find most discriminative regions/sequence features in the sequence taxonomic group

Conclusions

- The use of kernel-based approaches for DNA barcoding resulted in a highly accurate and scalable species identification method
- Mismatch/spectrum kernels proved to be very efficient as sequence similarity measure
- have demonstrated that the use of feature selection applied to the high dimensional space of string sequence features can often result in dramatic reduction in the number of features

References

[Jaakkola 00] Tommi Jaakkola, Mark Diekhans & David Haussler. *A Discriminative Framework for Detecting Remote Protein Homologies.* Journal of Computational Biology, vol. 7, no. 1-2, pages 95–114, 2000.

[Kuang 04] Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund & Christina Leslie. *Profile-Based String Kernels for Remote Homology Detection and Motif Extraction*. In CSB '04: Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference (CSB'04), pages 152–160, Washington, DC, USA, 2004. IEEE Computer Society.

[Leslie 02a] Christina S. Leslie, Eleazar Eskin & William Stafford Noble. *The Spectrum Kernel: A String Kernel for SVM Protein Classification.* In Pacifi c Symposium on Biocomputing, pages 566–575, 2002.

[Leslie 02b] Christina S. Leslie, Eleazar Eskin, Jason Weston & William Stafford Noble. *Mismatch String Kernels for SVM Protein Classification.* In NIPS, pages 1417–1424, 2002.

[P.D.N. 03] Hebert P.D.N., A. Cywinska, S.L. Ball & J.R. deWaard. *Biological identifications through DNA barcodes*. In Proceedings of the Royal Society of London, pages 313–322, 2003.

Thanks for listening

Kernels with position information

- **Fisher kernel** [Jaakkola 00]
  - each taxonomic group is modeled by profile HMM
  - each sequence is mapped to the space of fixed dimensionality using profile

Feature selection proved to be very effective in case of Fisher kernels: subsets of features induced after feature elimination step indicate that there is a **small set of positions** that can be used as a **signature** of each class that places it apart from all other classes. Unlike Fisher kernels, string kernels inherently contain no positional information. One way to induce position information is via **context-specific kernels**.