Fast Barcode-Based Species Identification Using String Kernels

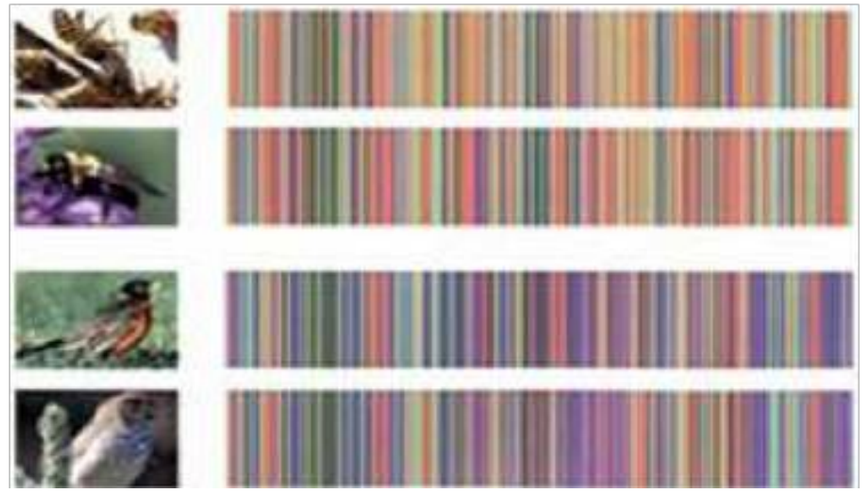
Pavel Kuksa
Vladimir Pavlovic

Department of Computer Science
Rutgers University

What Are We After?

- **Accurate** & **fast** identification and classification of species using barcodes
- Important biological problem, but also computationally challenging: Many species, large data sets, similar sequence content
- Sequence ID problem
- String kernels, robust classifiers, feature selection
- No phylogenetic trees

Barcoding



- Barcoding of Life: identification of organisms using ’barcodes’
- DNA barcode: short fragment (≈600bp) from standard mitochondrial gene region
- Assign an unidentified sample to the taxonomic group at a target level (class, family, species, etc.)using reference data with known class information
- Supported by many organizations: BOLI, CBOL, BOLD, etc.
- **Applications**: biodiversity monitoring and assessment, ecological studies, taxonomic research, etc.

Approach

- Kernel-based machine learning methods
  - some of the most accurate results in many sequence analysis and prediction tasks (e.g., remote homology)
  - often computationally expensive

- This work: general spectrum-like string kernel methods for DNA sequence analysis

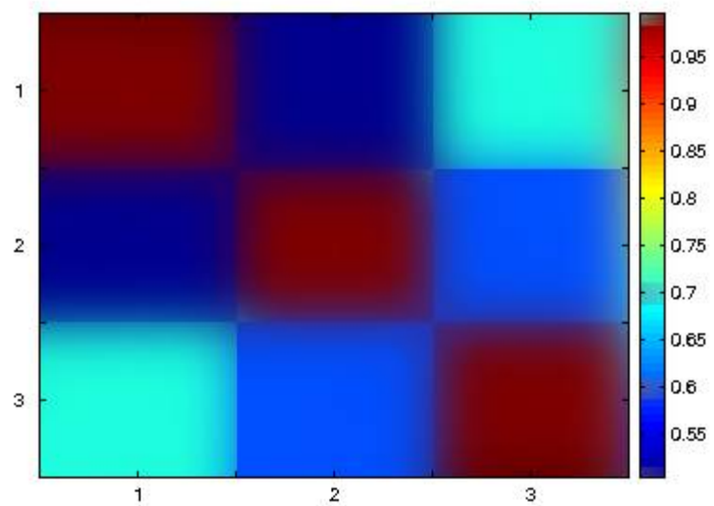
- **Fast** and **accurate** classification:
  - small space, fast to estimate, accurate prediction
  - “salient species features”

Kernel Methods for Sequence Analysis

Data

#	Sequence	Species
1	AATTTGAGCAGGACTAATTGGAACTTCTTTAAGATTACTTATTCGAACTGAATTAGGAACCCCAGGATCTTTAATTGGAGATGATCAAATT	A
2	NNNNTTTGAGCAGGACTAATTGGAACTTCTTTAAGATTACTTATTCGAACTGAATTAGGAACCCCAGGATCTTTAATTGGAGATGA	A
3	NNGGAATTTGAGCAGGACTAATTGGAACCTCCTTAAGATTACTTATTCGAACTGAATTAGGAACCCCAGGATCTTTAATTGGAGATGATCAAATTTATAATACAATTGT	B

Kernel



Predictor

$$\text{Predict}(\text{AATTGGAACCTCCTTAAGATTACTTATTCGAACTGAATTAGGAACCCCAGGATCTTTAATTGGAGATGATCAAATTTATAATACAATTGT}) = \begin{cases} \text{Is } \boldsymbol{A} \\ \text{Is } \boldsymbol{B} \\ \text{Not sure} \end{cases}$$

String spectrum features

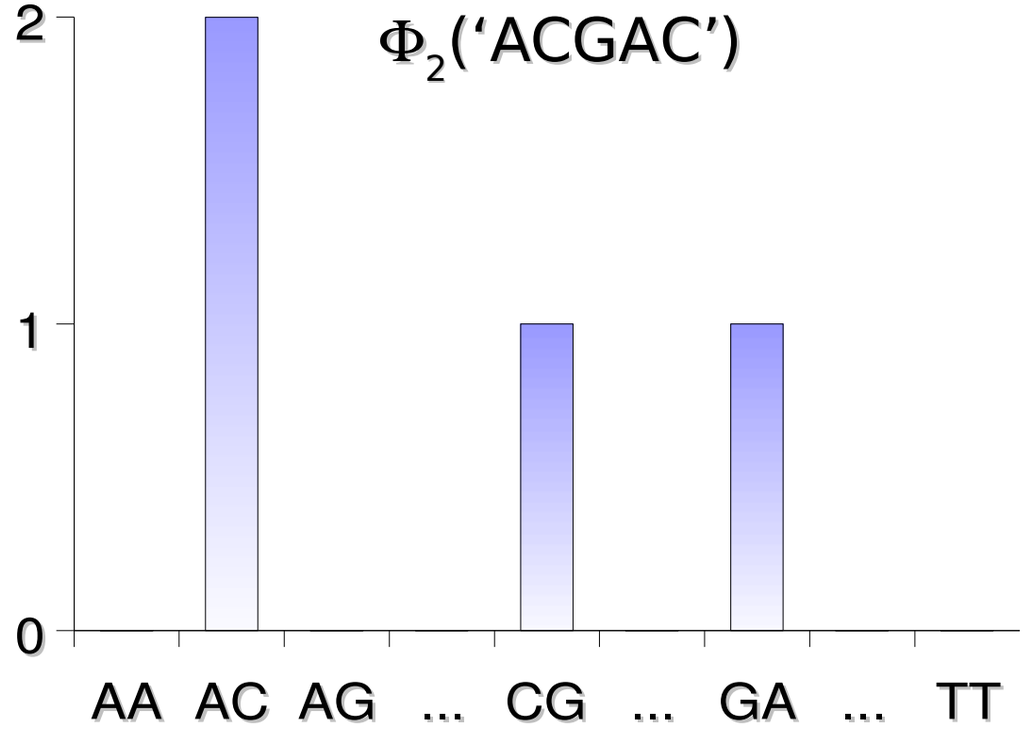
- General idea:
  - Compute **histograms (spectrum) of sequence feature frequencies** (e.g. short substrings or certain patterns)
  - compare the feature histograms
  - sequences with similar content have similar spectra

- Successfully used in protein remote homology detection, text clustering, etc.

Example: toy

- 2-spectrum of ’ACGAC’:

  $4^2 = 16$ vector

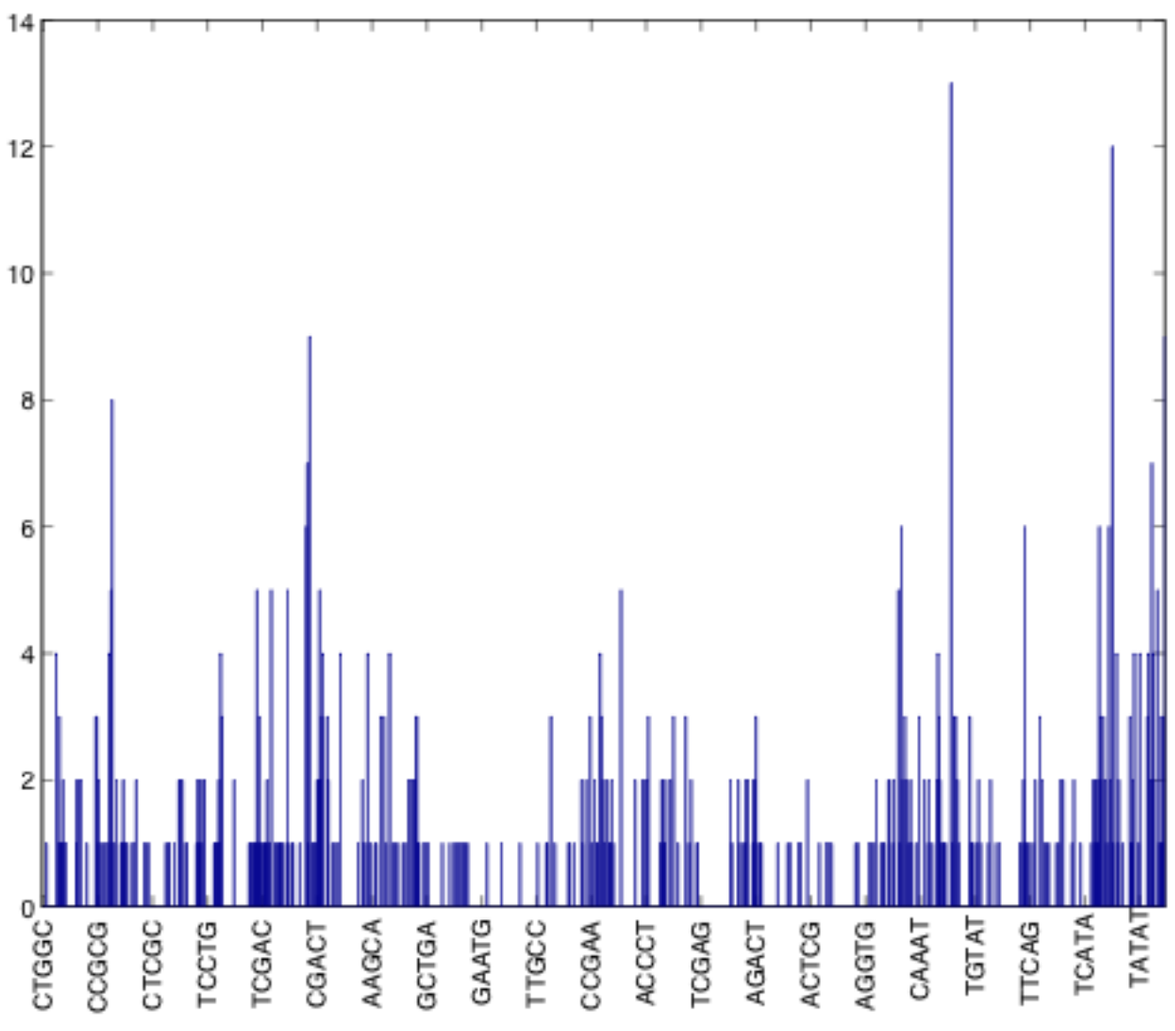


Example: Astraptes

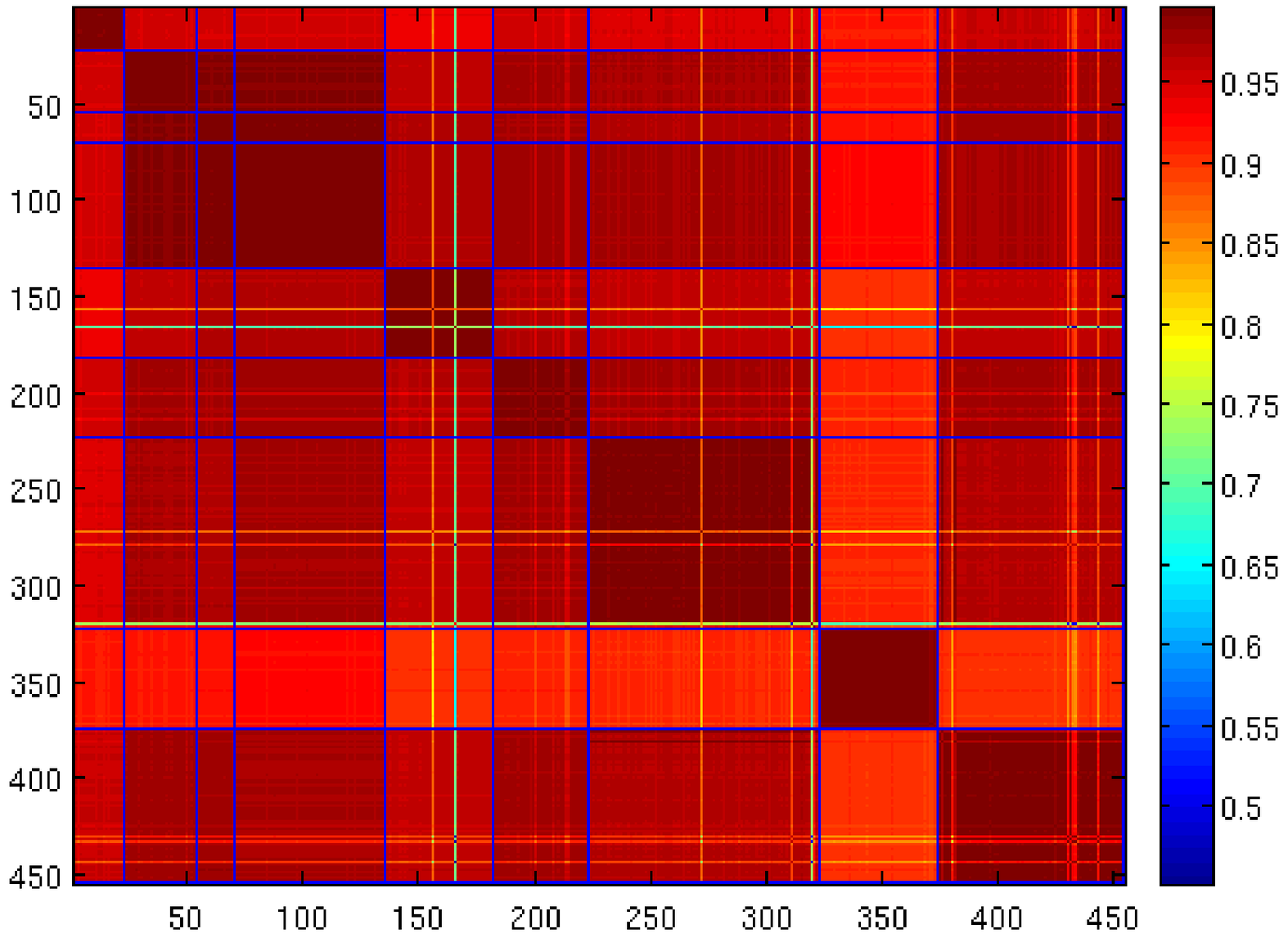
$X=$

GGAATTTGAGCAGGACTAATTGGAACTTCTTTAAGATTACTTATTCGAACTGAATTAGGAACCCCAGGATCTTTAATTGGAGATGATCAAATTTATAATAC
AATTGTTACAGCTCATGCATTTATTATAATTTTTTTTATAGTTATACCTATTATAATCGGAGGATTTGGAAATTGACTAGTTCCATTAATAATAGGTGCCCCA
GATATAGCTTTCCCCCGTATAAATAACATAAGATTTTGATTATTACCCCCATCTTTAACTTTATTAATTTCAAGAAGAATTGTTGAAAATGGGGCTGGTACA
GGATGAACAGTTTATCCCCCTCTTTCATCAAATATCGCCCATCAAGGAGCATCTGTTGATTTAGCAATTTTTTCCCTTCATCTTGCTGGTATTTCATCAATT
CTTGGAGCTATTAATTTTATTACAACAATTATTAATATACGAATTAATAATTTATCTTTTGATCAAATACCATTATTTGTTTGAGCTGTAGGAATTACAGCAT
TATTATTATTACTTTCATTACCTGTTTTAGCAGGTGCTATTACTATATTATTAACAGATCGAAATTTAAATACTTCTTTTTTTGATCCTGCAGGAGGAGGAGA
TCCAATCTTATACCAACACTTATTT

$\Phi_5(x)$



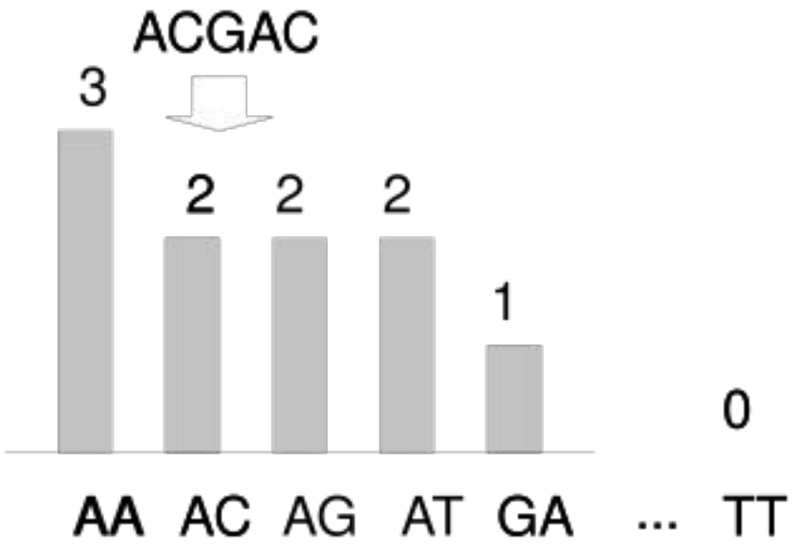
Example: Spectrum kernel (Astraptes)

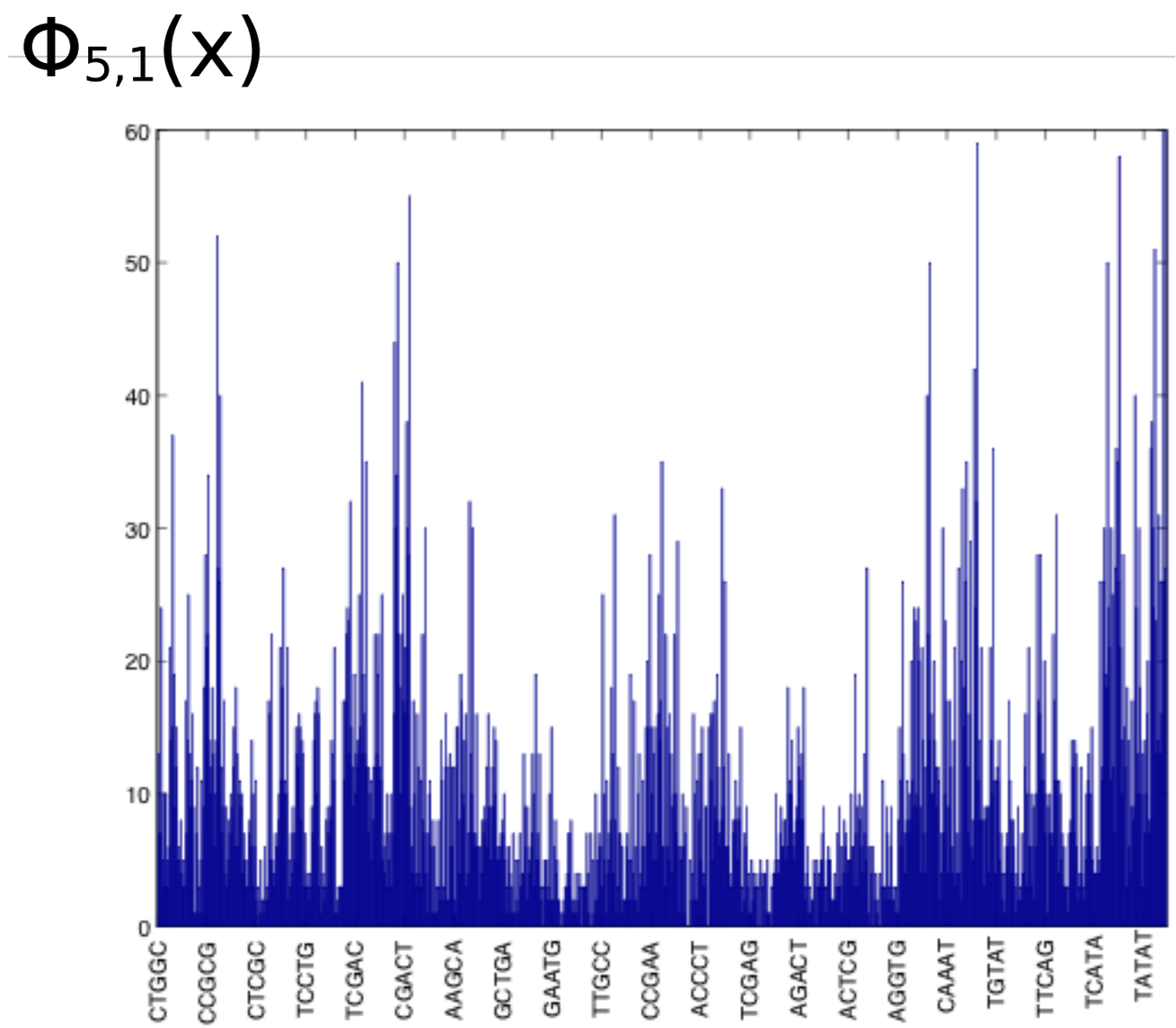


12 species

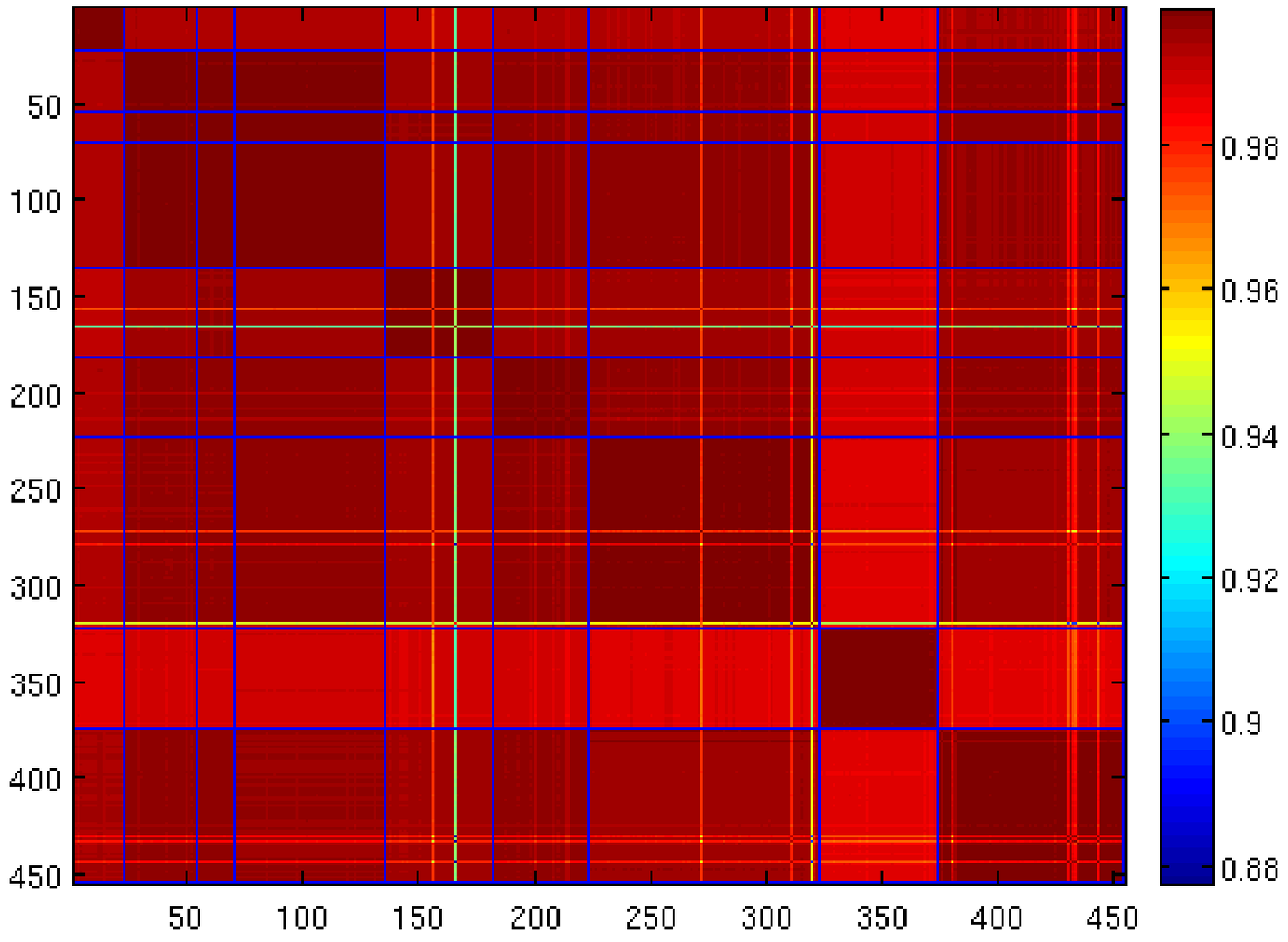
Mismatch Features

- ***k***-long substrings with up to ***m*** mutations





Example: Mismatch Kernel (Astraptes)



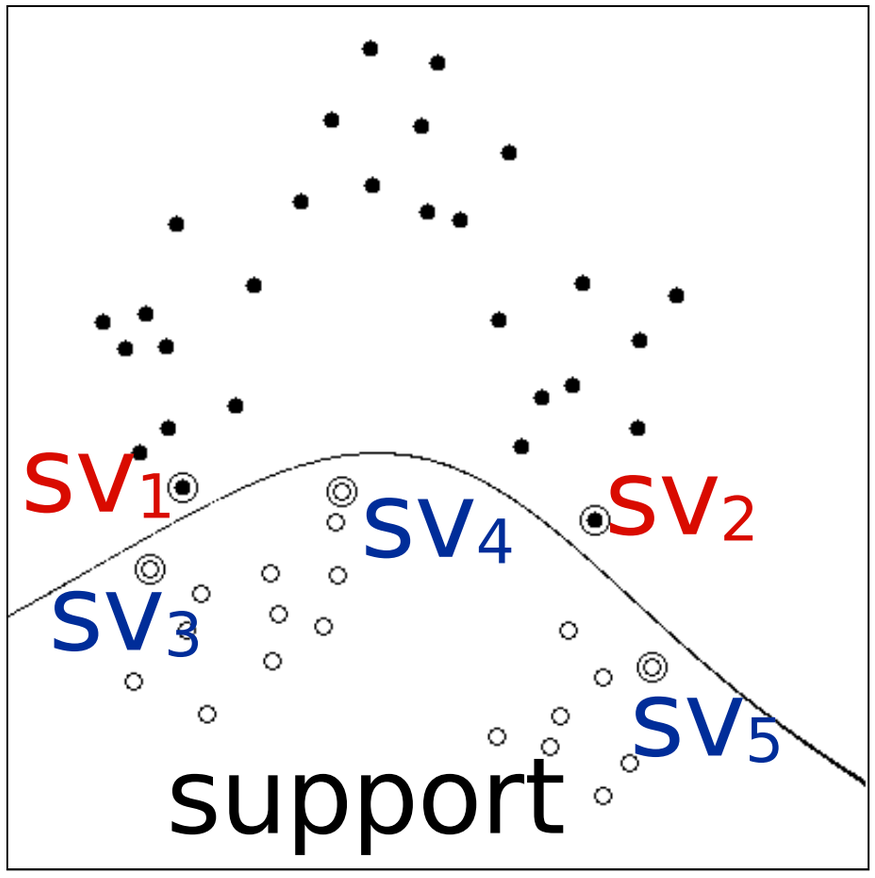
String Kernel Representation

- **Benefit 1:**
  No need for alignment!

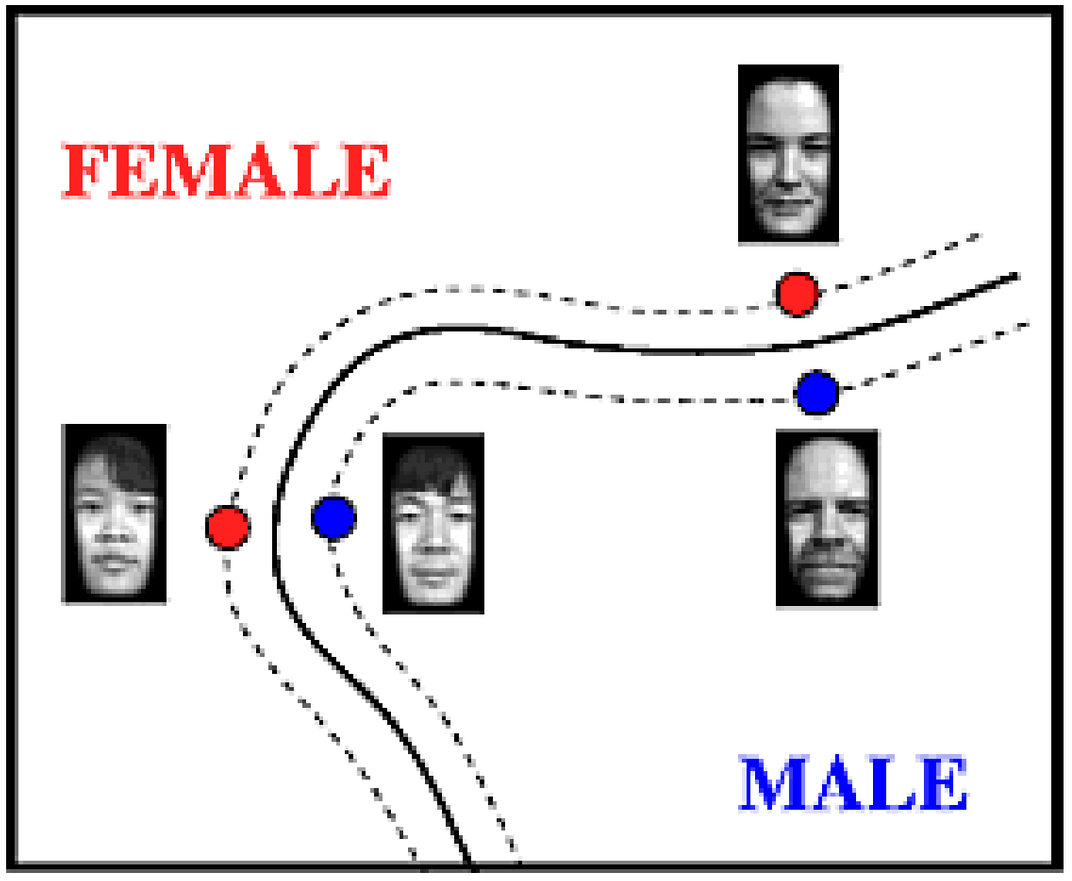
- **Benefit 2:**
  Can select specific substrings that are important for identification of particular species, making classification more robust.

- Possible problem:
  - choice of length & mismatch parameters
  - loss of longer spatial dependencies
  - potentially expensive to compute

Predictors: Support Vector Machines



support vectors



Ming-Hsuan Yang

$$\text{Predictor}(u) = \sum \alpha_i K(sv_i, u)$$

Spectrum Kernel Computations

- *N* training specimens of average length *n*:
  - Single kernel computations: compute kernel K(x, y) for two sequences
  - Kernel matrix computations: compute NxN matrix for N sequences
  - Kernel predictor computations: compute Nx1 kernel vector for sequence x and N support sequences
  - Focus on kernel matrices and vectors (SVM training, evaluation/testing, clustering, etc.)

Exact spectrum kernel computations

- Suffix tree-based algorithms:
  - for each pair (x, y) build a suffix tree for sequence features, and compute the kernel

    $K_k(x, y)$ at the tree leaves
  - $O(N^2kn)$ for N sequences of length n each
- Explicit mapping (EMap) algorithms
  - explicitly map sequences to feature vectors of size $|\Sigma^k|$, kernel matrix is then computed as K = M.M’ , where $M_{N\times|\Sigma|^k}$ is a feature matrix
  - $O(Nkn + |\Sigma^k|N^2)$
- Sorting-based algorithms:
  - Represent k-mers as numbers and use favorite sorting algorithm

Counting-sort spectrum kernel

- Algorithm:
  - Extract k-length substrings from input sequences and store them in a list L, O(knN)
  - Sort the list L using k passes of counting sort, O(knN).
  - Scan the sorted list updating the kernel matrix on each change in the feature value
- Complexity: $O(Nnk + min(u, n) \cdot N^2)$, u - number of unique k-mers in the input
- Update step:
  - $K(upd_f , upd_f) = K(upd_f , upd_f) + c_f \, c_f^T$
  - $upd_f = \{i : f \in x_i \}$ = input sequences containing f
  - c_f = vector of feature counts

Mismatch kernel using sorting

- Sorting-based computations:
  - Extract unique features using sorting
  - Expand a set of unique features to include their neighbors
  - Sort the resulting set
  - Scan the sorted list and update kernel matrix on each change in the feature value
- Complexity: $O(Nnk + uvk + uN^2)$

Divide-and-Conquer Mismatch Kernel

- Basic idea:
  - Infer count for a k-mer f using counts of its neighbors
  - Cluster the combined feature set $S = \sum^{N}_{i=1} Spectrum(x_i)$ to find sets of neighboring features
  - The size of resulting clusters/subclusters gives desired counts of feature occurrences

- For DNA, since u << nN , improve performance by using a unique features instead of the original redundant set.

Divide-and-Conquer Method

- **Divide step**: the combined feature set S is partitioned into subsets S_1 , . . . , $S_{|\Sigma|}$ using character-based clustering
- **Conquer step**: The same procedure (Divide step)is applied to each of the obtained subsets. After k divisions, kernel matrix is updated according to the contribution of the corresponding k-mer *f*.
- Complexity:

$$
\begin{aligned}
T(n,N,k,m) &= u \cdot \sum_{l=1}^{k} \sum_{i=0}^{\min(m,l)} \binom{l}{i} (\Sigma - 1)^i + u \cdot N^2 \\
&= O(uvk + u'N^2)
\end{aligned}
$$

Sparse kernel

- Can we fur ther reduce computation costs?
  - Preselect features (e.g. using filtering) and evaluate kernel for a set F of selected features
  - $K = M_F^T M_F$, M_F is $|F| \times N$ matrix of feature counts

- Reduces complexity of computations:
  - Spectrum kernel:

    $O(Nnk + FN^2)$ *vs* $O(Nnk + uN^2)$

  - Mismatch kernel:

    $O(Nnk + Fvk + FN^2)$ *vs* $O(Nnk + uvk + uN^2)$

Complexity comparison: Spectrum

- Previously known bounds: O(knN² + nN²)
- New bounds: O(knN + uN²) [WABI'07]

Algorithm	Time	Space
EMap	$O(Nnk + \|\Sigma\|^k N^2)$	$O(\|\Sigma\|^k N^2)$
Suffix tree 1 (N^2 trees)	$O(N^2 nk + nN^2)$	$O(Nn)$
Suffix tree 2 (single tree)	$O(Nnk + \min(u,n)N^2)$	$O(Nnk)$
Counting sort	$O(Nnk + \min(u,n)N^2)$	$O(Nnk)$

- Advantages of counting sort-based computations:
  - more time efficient
  - smaller memory requirements in practice than suffix trees
  - easier to implement

Complexity comparison: Mismatch

- Previously known bounds: $O(nk^{m+1}|\Sigma|^m N^2)$
- New bounds: $O(uk^{m+1}|\Sigma|^m + uN^2)$ [WABI'07]

	Mismatch kernel matrix	Mismatch kernel matrix with feature selection
EMap	$Nnvk + \|\Sigma\|^k N^2$	$Nnvk + \|F\|N^2$
EMap+Sort	$Nnk + uvk + uvN + \|\Sigma\|^k N^2$	$Nkn + uvk + uvN + \|F\|N^2$
DC	$Nnvk + u'N^2$	$Nnvk + \|F\|N^2$
DC+Sort	$Nkn + uvk + u'N^2$	$Nkn + uvk + \|F\|N^2$
Sorting	$Nkn + uvk + u'N^2$	$Nkn + \|F\|vk + \|F\|N^2$

EMap=explicit map, EMap+Sort=EMap with presorting,
DC=divide and conquer

v=neighborhood size,
u=number of different k-mers in the input,
u´=number of different k-mers including neighbors

Experimental results

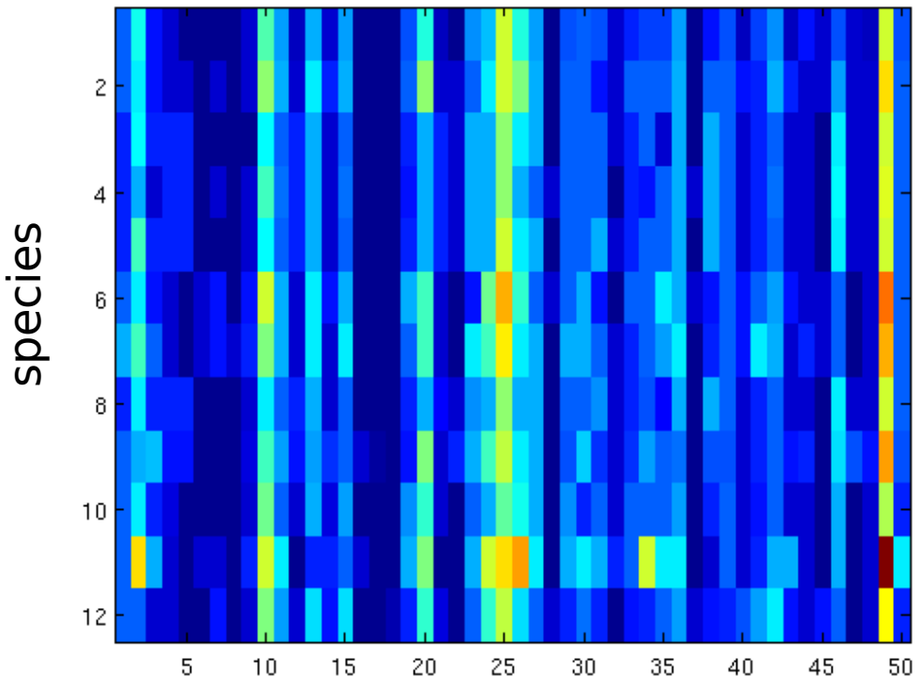
- Barcode datasets

dataset	# of classes	N	n
Astraptes	12	466	600
Hesperiidae	355	2135	613
Fish larvae	7	56	1113

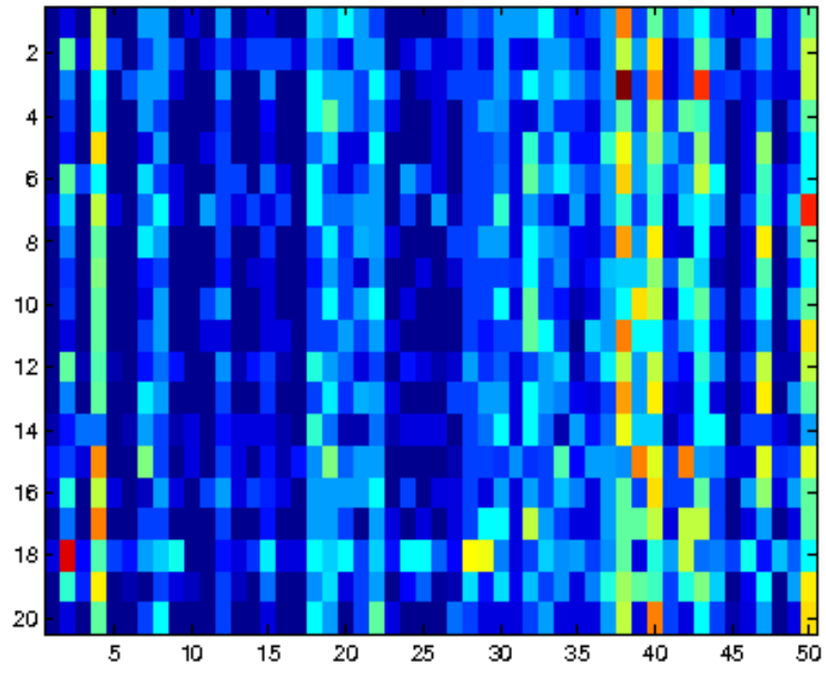
- Classification: multiclass & binary (CV/ROC/ROC50)
- Kernels: Fisher, Spectrum, Mismatch
- Algorithms: SVM, ridge regression, 1-NN
- Running time analysis:
  - training (matrix) and testing (vector)
  - different kernel parameters: k, m
  - different feature selection levels

Spectra (per species)

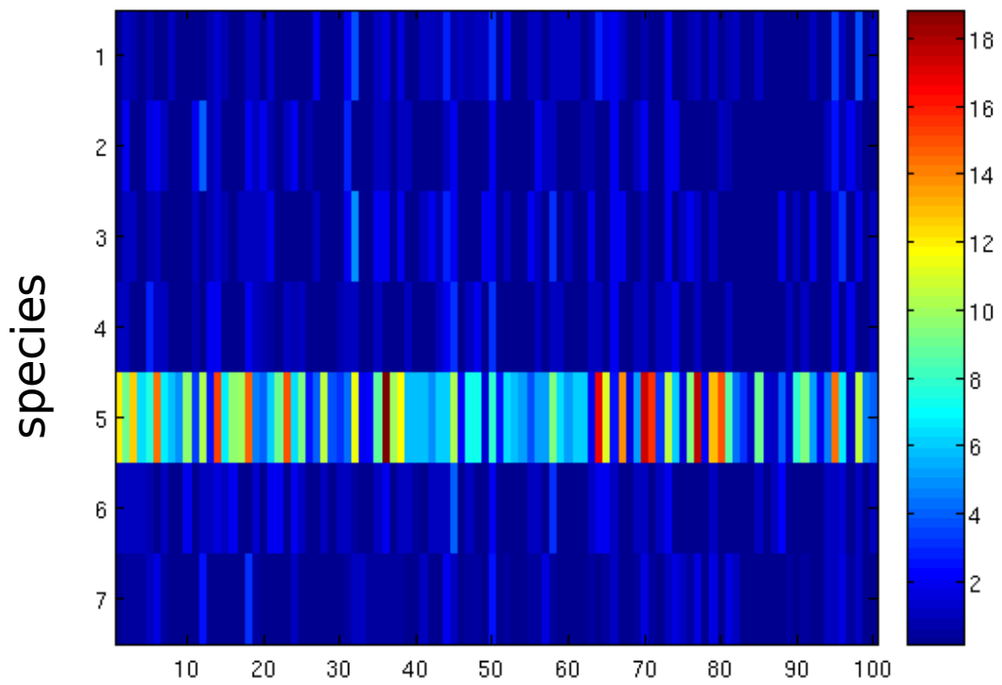
Astraptes



Hesperiidae



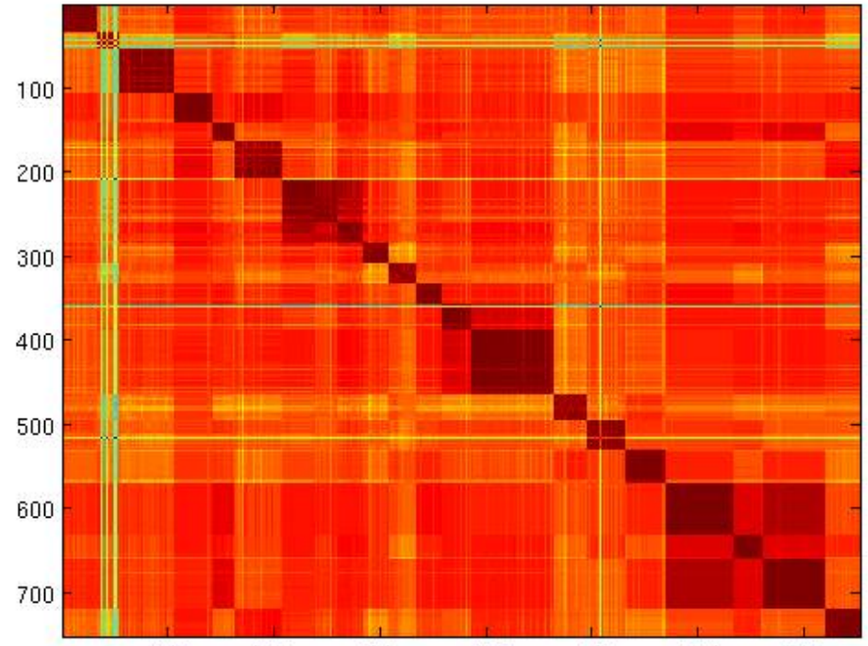
Fish larvae

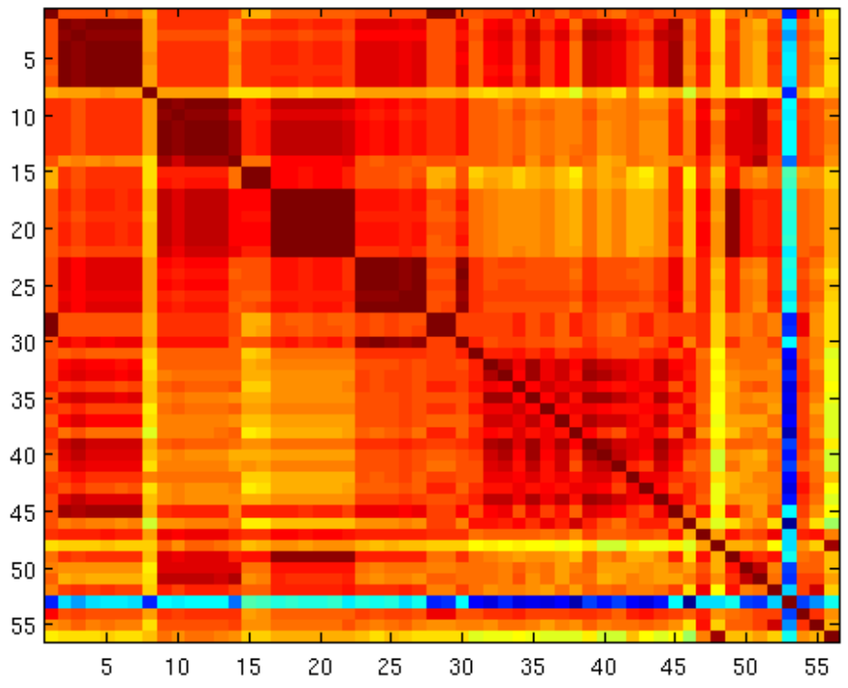


*first n mismatch features shown

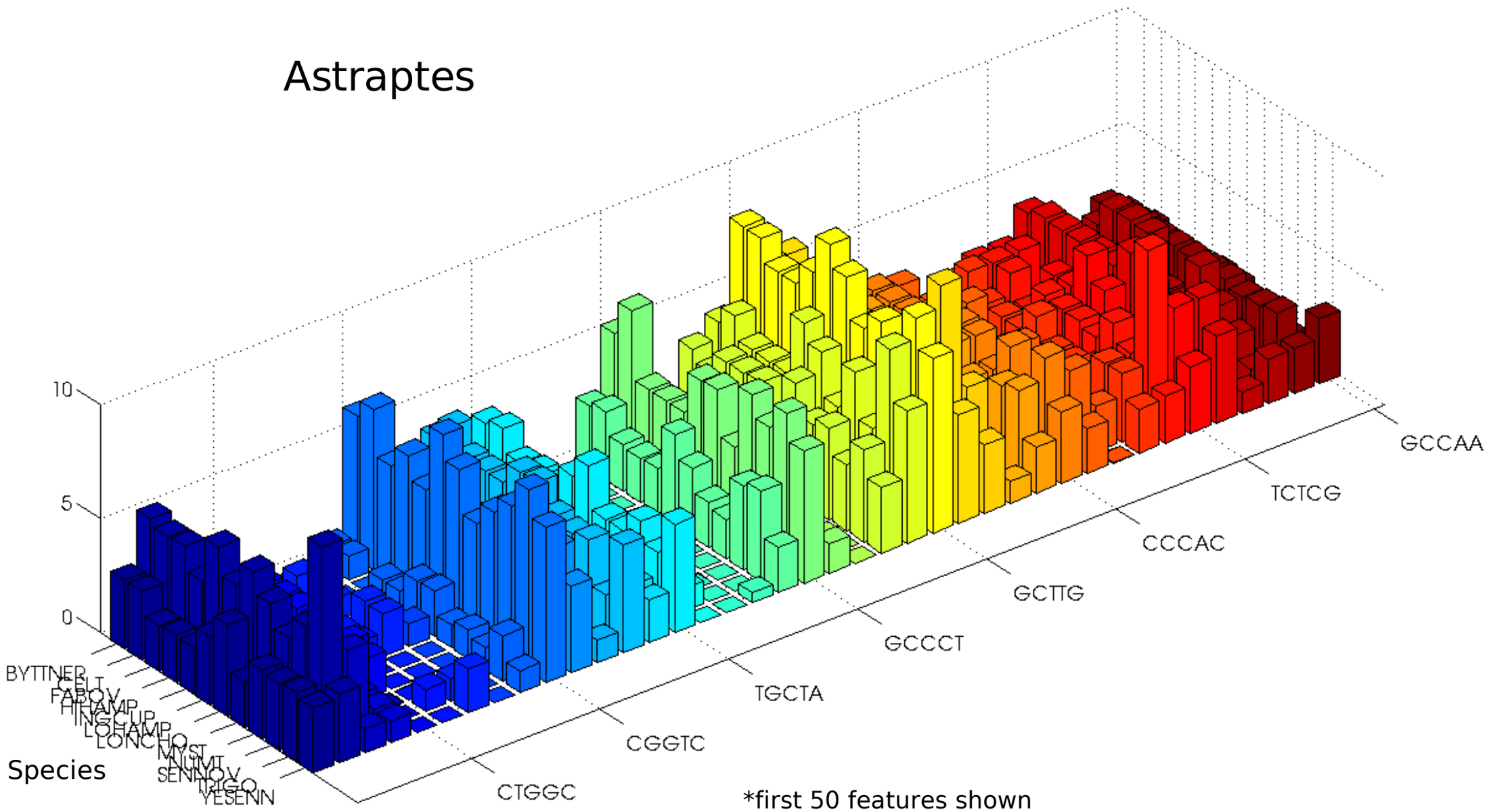
Kernels


50
100
150
200
250
300
350
400
450
0.98
0.96
0.94
0.92
0.9
0.88
50
100
150
200
250
300
350
400
450


100
200
300
400
500
600
700
100
200
300
400
500
600
700


5
10
15
20
25
30
35
40
45
50
55
5
10
15
20
25
30
35
40
45
50
55

Astrapes: spectra of different species



*first 50 features shown
*12 species

Classification performance: multiclass

%Error

method	Astraptes	Hesperiidae	Fishes
SVM-MK	1.28±1.79	3.61±0.87	4.17 ± 9.00
SVM-SMK(100)	0.85±1.1	2.1±0.32	5.00±10.54
SVM-SK	1.9±2.54	3.72±0.90	5.00±11.25
SVM-SSK(100)	1.07±1.5	2.3±0.53	7.26 ± 12.53
FK	4.15±1.82	-	12.45±9.03
PSI-BLAST	9.63±6.58	1.6±0.42	16.15±7.65

- 10-fold cross-validation
- MK=mismatch kernel, SK=spectrum kernel, SMK, SSK=with feature selection, FK=Fisher kernel
- **10% features, classifiers improve/retain performance**
- **Improved** performance compared to previous studies in [Matz & Nielsen, 2005] and [Nielsen & Matz, 2006](**rates of 9-20%**)

Classification performance: binary

- 10-fold cross-validation error rates

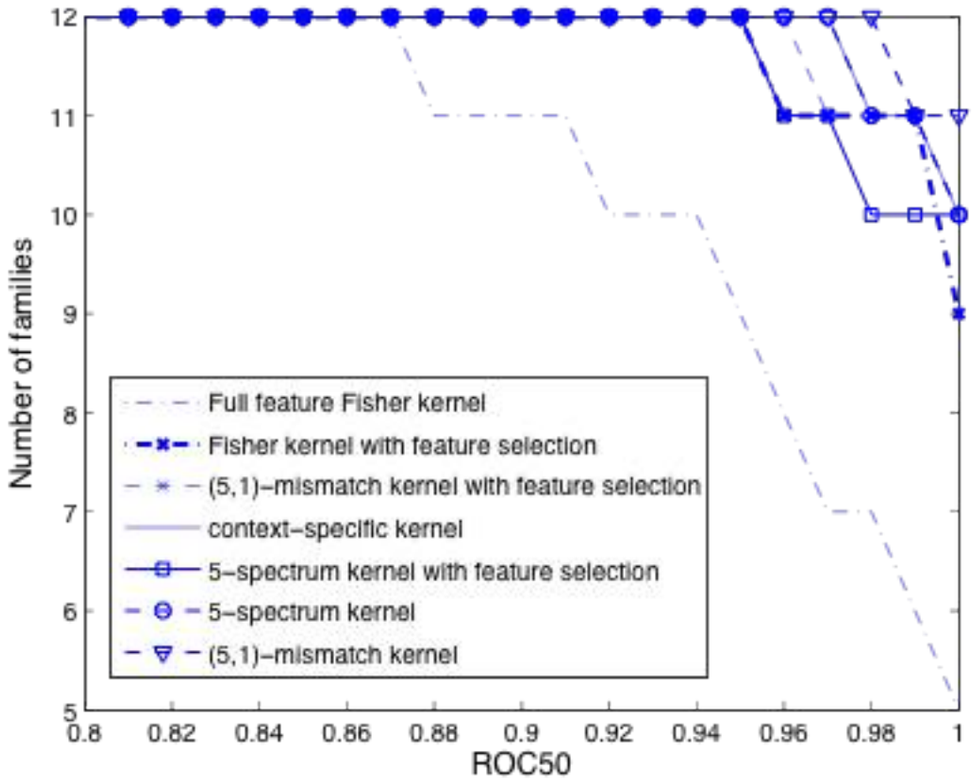
method	Astraptes	Hesperiidae	Fishes
SVM-MK	0.23±0.36	0.2±1.27	1.95±2.48
SVM-SMK(100)	0.18±0.61	0.05±0.1	2.48±2.34
SVM-SK	0.18±0.35	0.04±0.07	1.71±1.73
SVM-SSK(100)	0.82 ± 1.77	0.07±0.14	1.71±1.73

- Average ROC/ROC50 scores

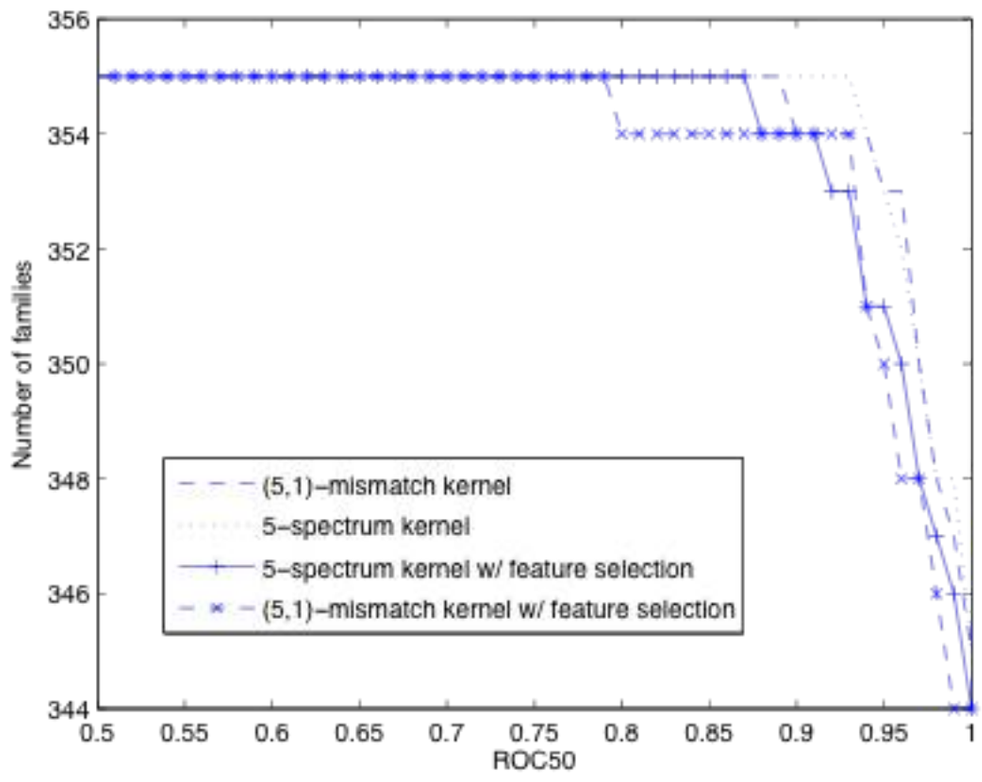
	Astraptes		Hesperiidae		Fish	
Method	ROC	ROC50	ROC	ROC50	ROC	ROC50
SVM-MK	0.986	0.986	0.942	0.857	0.943	0.943
SVM-SMK	0.986	0.986	0.954	0.897	0.943	0.943
SVM-SK	0.966	0.966	0.948	0.894	0.943	0.943
SVM-SSK	0.993	0.993	0.943	0.88	0.943	0.943

Classification performance: ROC

Astraptes



Hesperiidae



- Feature selection improves performance
- 90% reduction in the number of features

Running time: Mismatch

- Running time, seconds

data set	K	EMap	EMap+Sort	D&C	SK package
Astraptes	5	202.11	3.14	16.92	2820
Hesperiidae	5	938.79	14.73	240.36	75219

- Significant time improvement compared to the state-of-art spectrum kernel implementation
- EMap requires much larger storage than D&C
- Pre-sorting significantly improves computing time for EMap

Running time: Mismatch + Feature Selection

	Astraptes data			Hesperiidae data		
% feat	EMap	EMap+ Sort	D&C	EMap	EMap+ Sort	D&C
90	212.23	4.03	12.91	961.94	17.75	163.80
70	185.62	3.88	10.41	982.72	17.20	122.34
50	193.74	2.31	8.54	989.4	18.56	89.25
30	192.54	2.51	6.31	977.07	14.85	52.32
10	184.75	3.60	3.74	966.68	10.57	23.92

- D&C scales almost linearly with the number of features

Running time: Prediction

k	m	Astraptes data			Hesperiidae data		
		D&C	DC+ Sort	EMap+ Sort	D&C	D&C+ Sort	EMap+ Sort
5	1	9.25	3.25	3.06	53.36	14.44	13.70
7	1	18.01	9.15	5.84	78.39	16.35	48.62
9	1	39.88	42.70	-	200.62	99.37	-
5	2	33.28	3.9	5.30	197.34	26.23	20.03
7	2	80.65	10.14	12.50	537.61	49.31	75.63
9	2	192.04	70.88	-	1166.6	173.9	-

- D&C outperforms EMap in many cases while requiring only linear space

Size of Support Vector Sets

method	Astraptes	Hesperiidae	Fishes
#samples/sp	3-100	2-21	1-77
SVM-MK	11	6	5
SVM-SMK(100)	19	7	4
SVM-SK	8	7	6
SVM-SSK(100)	7	5	3

Summary of results

- Spectrum kernels for **accurate and fast** DNA barcode-based species identification
- Efficient computation of spectrum kernel matrices and vectors
- **Salient sequence features** can successfully discriminate species
- **Signatures** (support vectors + short snippets) for many taxonomic groups

Future work

- Position-aware string kernels: taking feature interactions into account
- Efficient feature selection methods
- Semi-supervised setting: many unlabeled & few labeled specimens

References

- [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 Pacific 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] Heber t 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.

- [Vishwanathan 02] S. V. N. Vishwanathan & Alexander J. Smola. Fast Kernels for String and Tree Matching. In NIPS, pages 569–576, 2002.