Multivariate and generalized kernels for sequence analysis

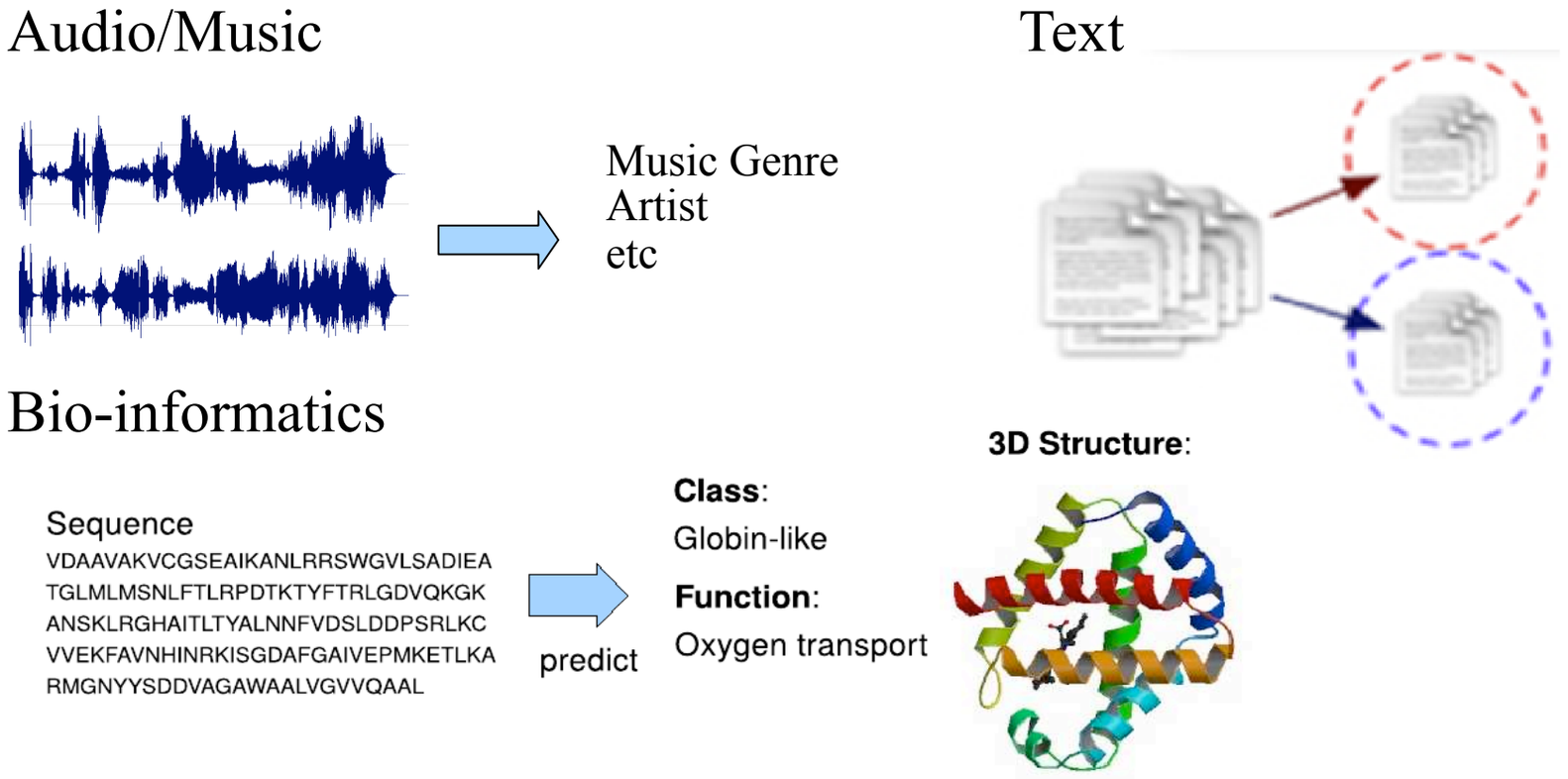
Pavel P. Kuksa
Machine Learning Department
NEC Laboratories America, Inc

Outline

- Part I: Sequence classification: broad overview, history, approaches
  - Kernel methods overview
- Part II: Standard string kernel methods
- Part III: Generalized similarity kernels
- Part IV: Multivariate string kernels
- Part V: Experiments: four fascinating and challenging problems

This talk

1. Classification of sequences
   - Inferring functions of proteins, genes, species labels from DNA, music genre, or text topic



   - Sequence = variable-length set of integers, reals, symbols, or vectors, indexed by position, time, or space, etc

Classification of sequences

- Motivation: problems of general interest
  - classification of text/documents
  - classification of speech/music
  - classification of time series/stock/etc
- Specific problems in bioinformatics, immunology, chemoinformatics etc
  - classification of proteins by function and structure
  - classification of small biomolecules (e.g., peptide classification: binding/non-binding)
  - classification of chemical compounds by biological activity

Sequence analysis problem difficulty

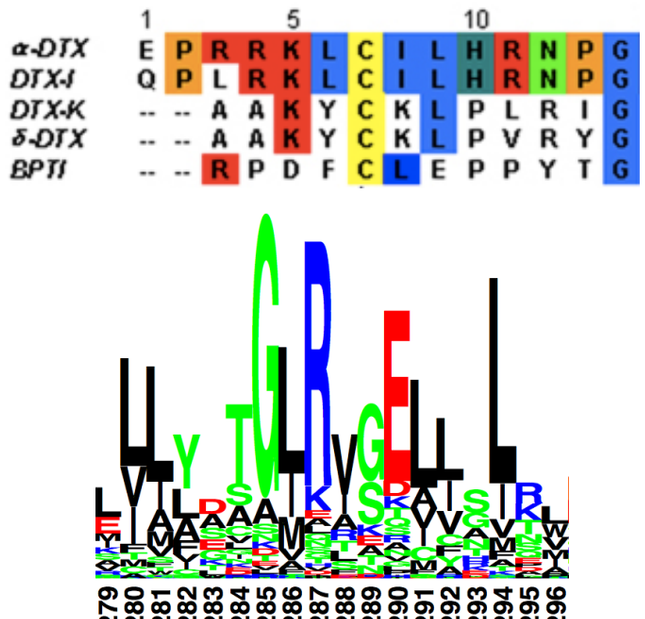
- Tasks in the space of sequence analysis are non-trivial:
  - Variable length: standard methods do not apply
  - No all-purpose/generic representation, similarity functions: representation problem
  - No straightforward way to do calculus on strings: clustering, classification, regression?
  - Size of data sets: methods need to scale to sets with millions of sequences

- Sequence analysis includes:
  - Comparison of sequences in order to find if sequences are related (homologous)
  - Identification of intrinsic sequence features (regulatory elements, exons, introns, etc)
  - Function, molecular structure, or feature prediction: classification, clustering, regression

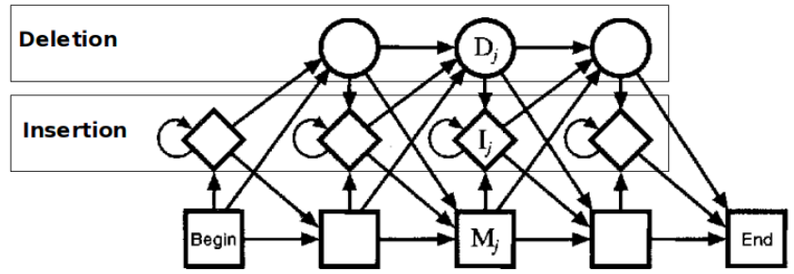
How to solve these tasks?

Approaches to analysis of sequences

- Sequence Alignment: Needleman-Wunsch (1970), Smith-Waterman (1981)
- Profile methods: profile comparison, PSSM (Gribskov, 1987), profiles constructed from multiple sequence alignment



- Hidden Markov Models: profile-HMM (Haussler, 1993)
- Kernel-based methods: clustering, classification, regression on *strings*!!



Problem Solving Framework

- Kernel Methods for Sequences
  - map the problem from original *sequence space X* into a *vector space (the feature space F)*
  - Infer class label of sequence X via similarity measures/kernels:

$$K(X,Y) = \langle F(X), F(Y) \rangle$$

dot-products of feature vectors $F(\cdot)$

$$\text{Predictor}(X) = \sum_{i \in SV} \alpha_i K(sv_i, X)$$

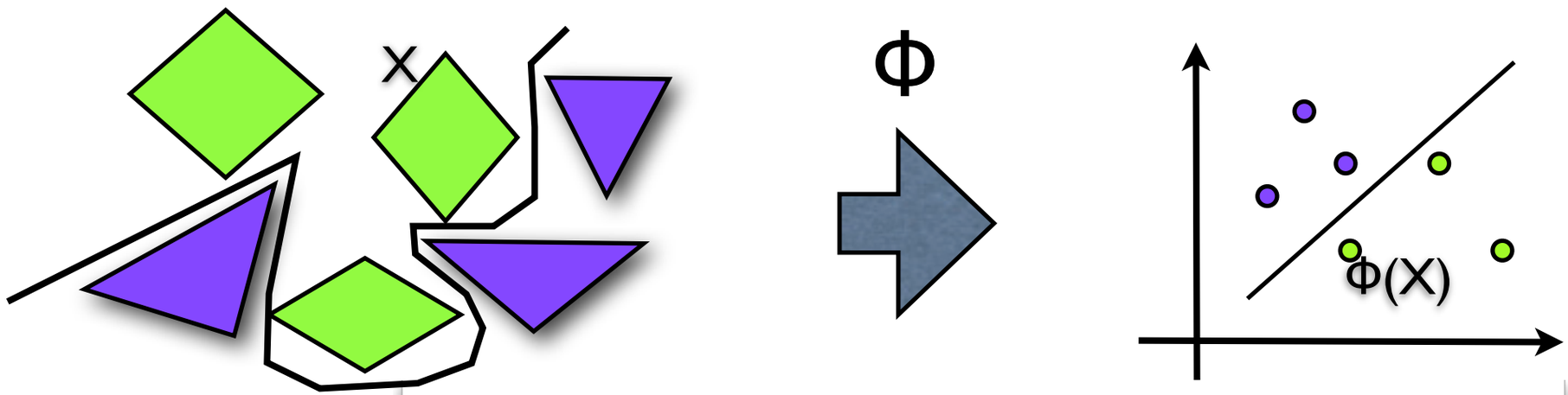
  - pairwise dot-product can be computed efficiently directly(!) from the original data using kernel function K

Basics on kernels

Main idea: map $X \rightarrow \Phi(X)$

Input space

Kernel space



Feature map:

$$\Phi(X) = [\Phi_1(X), \Phi_2(X), \ldots, \Phi_d(X)]$$

Φ-separation:

$$f(x) = w^T \phi(x) > 0, \quad x \in X^+$$
$$f(x) = w^T \phi(x) < 0, \quad x \in X^-$$

Representer theorem:

$$w = \sum \alpha_i \cdot \phi(x_i)$$
$$f(x) = \sum_i \alpha_i \phi(x)^T \phi(x_i) = \sum_i \alpha_i k(x, x_i)$$

kernel (dot-product)

Standard kernels

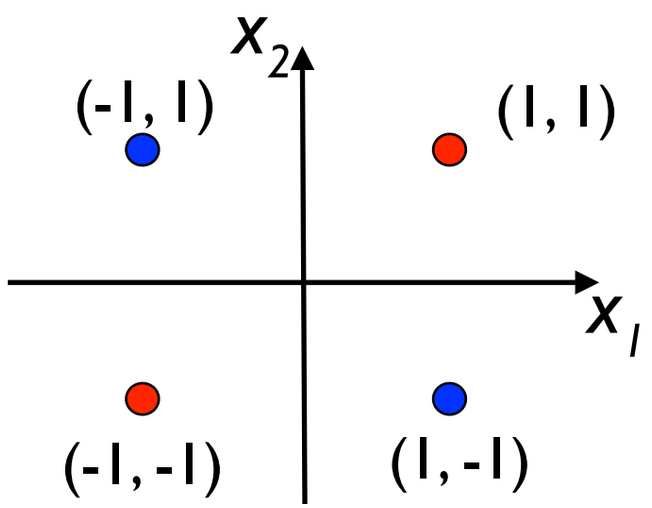
1. Polynomial: $K(x,y) = (x^T y+c)^d$
2. RBF: $K(x,y) = exp(-\frac{||x-y||^2}{\sigma^2})$

- Ex: for polynomial *d*=2, *dim*(x)=2

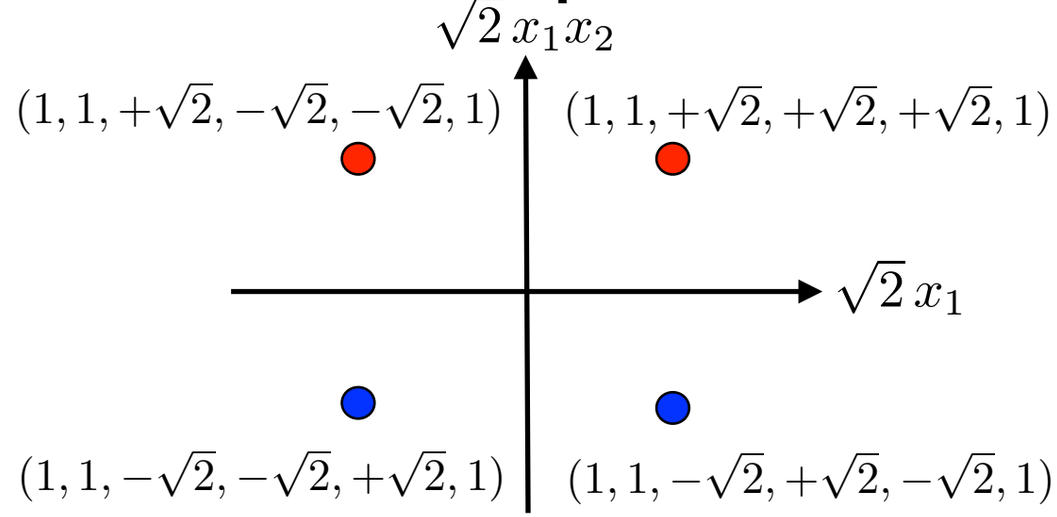
$$\Phi(x=(x_1,x_2)) = [x_1^2,\ x_2^2,\ \sqrt{2}x_1x_2,\ \sqrt{2c}x_1,\ \sqrt{2c}x_2,\ c]$$

- XOR problem

original space



kernel space



Linearly **Non-separable**!

Linearly **separable**!! (with x_1x_2=0)

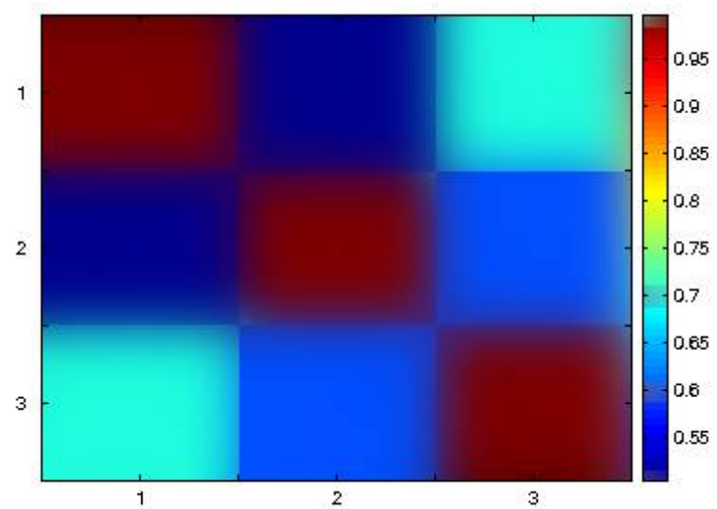
Kernel framework

Data D = $\{(X_i,C_i)\}_{i=1,...,N}$

#	Peptide X	Class C
1	NATEIRASV....	B
2	MVLSAFDER...	NB
	.....	
N	SALILRGSV....	B

$\mathbf{K}(X_i,X_j)$

Kernel matrix *K*



Predictor (SVM)

Predict($_{\text{YLEENPSAG}}$) =

B
NB
SB

$$f(X)=\sum_{i \in SV} \alpha_i K(sv_i,X)$$

estimated during training!

Advantages: can capture *complex relations* in data, can combine heterogeneous information (sequence, structure, etc)

Kernel methods

- Benefits:
  - Efficiency: kernel K is often more efficient to compute than Φ and dot product
    - e.g. product in high-dimenstional kernel feature space can be computed in input space (RBF, polynomial, etc)
  - Non-linear separation
  - Can operate on many types of data: sequences, text, images, graphs, vectors

Part II:

Standard string kernels

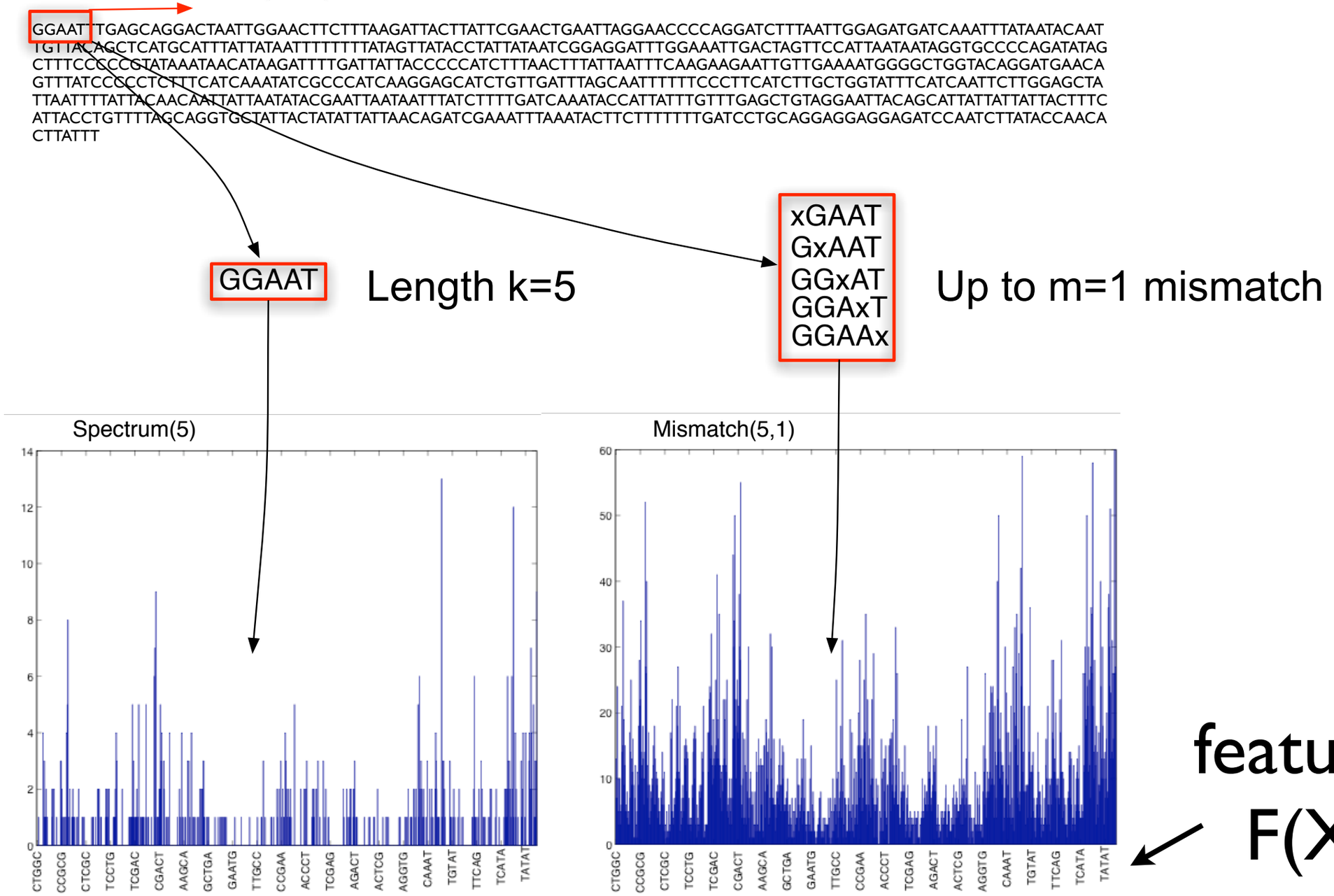
Sequence kernels

- Definition: kernels defined on pairs of strings
- Motivation: computational biology (protein, DNA), text analysis and classification, time series etc
- Idea: two sequences/strings are related if they share many similar substrings or subsequences

Inference using String Kernels (1)

1. Represent sequence X using fixed-dimensional feature vector F(X)

X



Similar X and Y ~ Similar F(X) and F(Y)

K(X,Y) = <F(X), F(Y)>

Inference using String Kernels (2)

2. Compute similarity K(X,Y) using feature vectors F(X),F(Y)

Sequence X

GTATATGATCAGGACTAGTTGGTACAGCTCTTAGACTATTAATT
CGAGCTGAACTAGGCCAACCAGGAGCTCTTCTGGGGGATGA
TCAATTATATAATGTAATTGTAACCGCCCACGCTTTTGTTATAAT
TTTCTTTTTAGTAATACCTATAATGATTGGAGGATTTGGAAACT
GATTAGTTCCTTTAATACTAGGAGCCCCAGACATGGCATTCCC
ACGTCTAAATAATATAAGTTTCTGACTTCTTCCGCCTGCCCTT
CTACTACTCCTTTCTTCAGCAGCAGTCGAAAGTGGAGTTGGA
ACTGGATGAACCGTCTACCCTCCTTTAGCCGGAAACCTTGCT
CACGCATTGACAGACCGAAACTTTAACACCGCTTTCTTTGAT
CCAGCAGGAGGTGGAGA

?

Sequence Y

ATGGTTAAATTCTCCTCAGAACCACATTTGGCTCAGGTAGTC
GCAGAAGACCTTCTTTCTCCTAGCGTGGTGGATGTGGGTGA
CTTCACAATATCAATCAACGAGGGTCTCCCCTCTGGGGTGCC
CTGCACCTCCCAATGGAACTCCATCGCCCACTGGCTTCTCA
CTCTCTGTGCGCTCTCTGAAGTTACAAATCTGTCCCCTGACA
TCATACAGGCTAATTCCCTCTTCTCCTTCTATGG

F(X)

?

F(Y)

dot-product

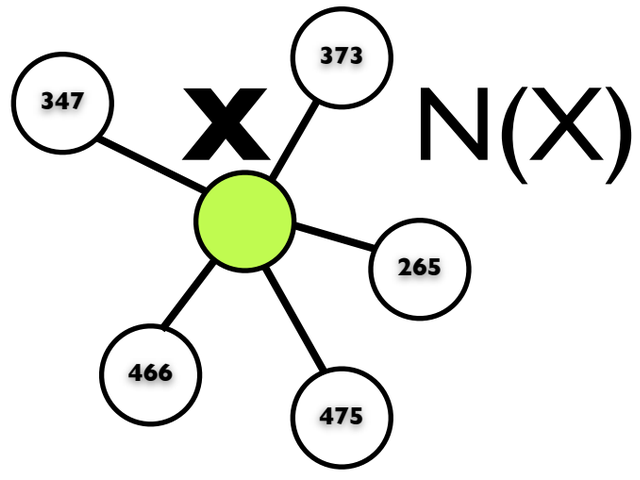
Similarity score: **K(X,Y) = < F(X), F(Y) >**

String kernel

Fixed-length vector of features derived from sequence

Using external information to help prediction

- Computing F(X) can use other information (besides X itself) to improve predictive performance
  - general sequence databases (UNIPROT, PDB, etc)
  - specific catalogs (e.g., plnTFDB)
  - gene ontology, etc
- Eg. $F(X) = F(X|U) = F(N_U(X))$
  - U is sequence database, and $N_U(X)$ is a set of sequences from U that are similar to X
  - this can *significantly* improve predictive accuracy



String Kernels: Overview

- Original string kernels

  Pairwise-alignment algorithms (Needleman-Wunsch)
  Not Mercer kernels [Vert et al.'04]

  Pair HMMs [Watkins'99], convolution kernels [Haussler'99], gappy n-gram kernels [Lodhi et al.'02], rational kernels [Cortes et al.'02]

  O(n²) complexity in sequence length ***n*** / pair

- *Spectrum-like kernels*

  Spectrum kernels [Leslie'02], *mismatch kernels* [Leslie'04], substring kernels [Vishwanathan & Smola'02]

  O(n) complexity in sequence length ***n*** / pair

Accuracy & Algorithmic complexity still insufficient for large-scale accurate sequence comparison / annotation

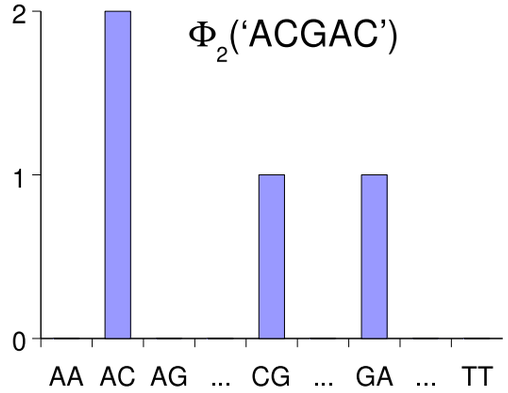
Standard kernels on strings

- Spectrum kernels (Leslie, 2002)
- Mismatch kernels (Leslie, 2004)
- Profile kernels (Kuang, 2005)
- Spatial sample kernels (Kuksa, 2010)

Spectrum Kernels

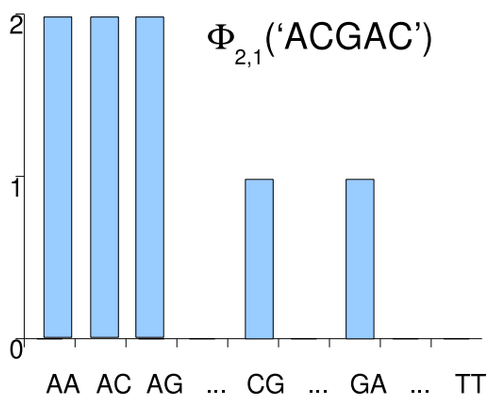
- Measure similarity between sequences based on co-occurrence of fixed-length substrings (*k*-mers)
- Feature map for exact spectrum kernel (no mismatches)

$$\Phi(X) = \left(\sum_{\substack{\alpha \in X \\ |\alpha| = k}} I(\alpha, \gamma) \right)_{\gamma \in \Sigma^k}$$



- Feature map for mismatch kernel (*m* mismatches)

$$\Phi(X) = \left(\sum_{\substack{\alpha \in X \\ |\alpha| = k}} I_m(\alpha, \gamma) \right)_{\gamma \in \Sigma^k}$$



- more dense representation (up to $k^m|\sum|^m$ more dense)

Profile kernel

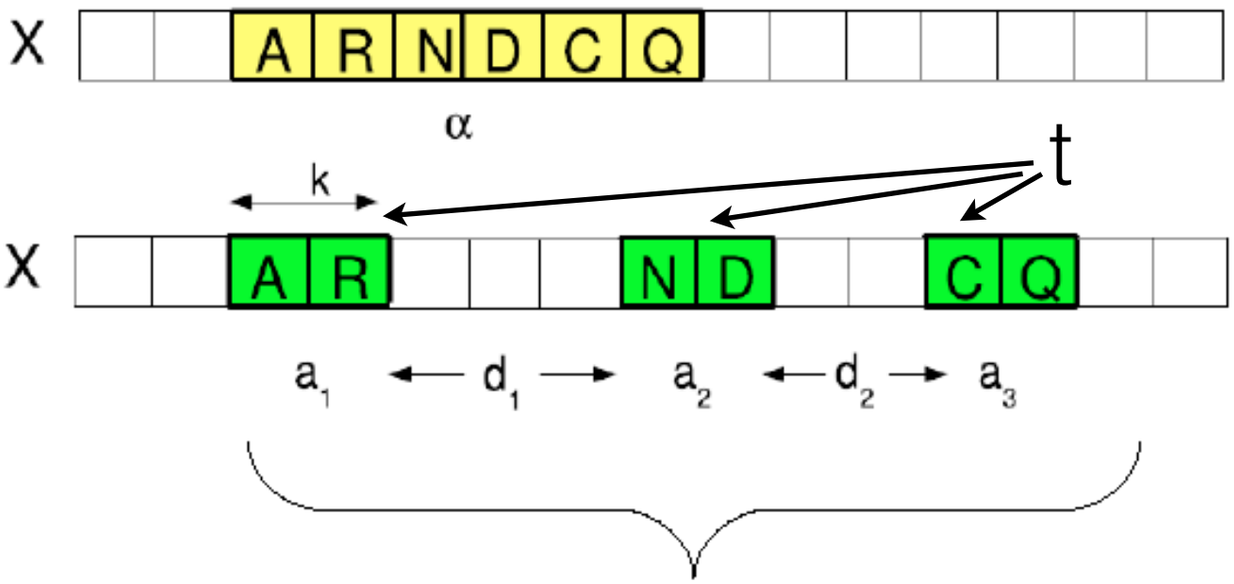
- *Probabilistic* string representation
- Replaces string representation $\mathrm{nx1}$ with a $\mathrm{nx}|\sum|$ profile
- Each character in a sequence is replaced with a probability distribution over alphabet characters
- Measures sequence similarity base on co-occurrence (in a probabilistic sense!) of k-mers

$$\Phi(X) = \left(\sum_{M_\sigma \in X} I(\gamma \in M_\sigma) \right)_{\gamma \in \Sigma^k}$$

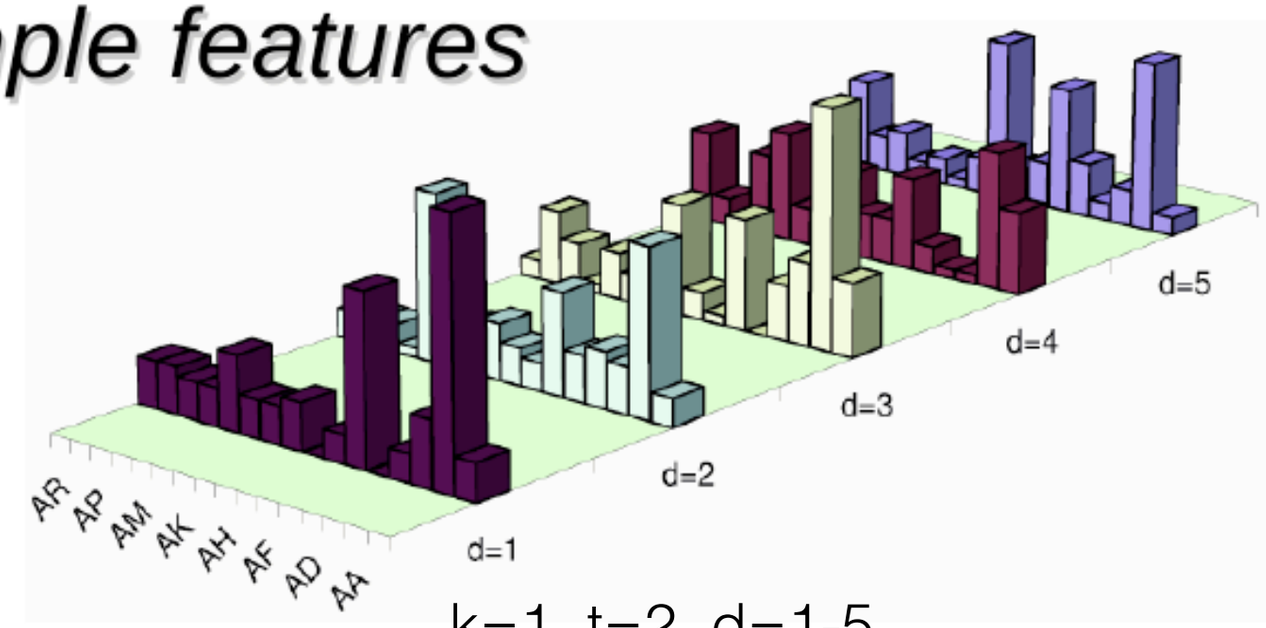
$M_\sigma = \{a : P(a) > \sigma\}$(mutation neighborhood)

Sparse Spatial Sample Embedding

Spatial arrangements of simple features



SSS feature



k=1, t=2, d=1-5

SSS embedding

$$S = a_1 \overset{d_1}{\leftrightarrow} a_2, \overset{d_2}{\leftrightarrow}, \cdots, \overset{d_{t-1}}{\longleftrightarrow} a_t\,, |a_i| = k, 0 \leq d_i < d$$

$$K(X, Y|k, t, d) = \sum_{\substack{(a_1, d_1, a_2, \ldots, d_{t-1}, a_t), \\ a_i \in \Sigma^k, 0 \leq d_i \leq d-1}} C(a_1, d_1, \ldots, d_{t-1}, a_t|X) C(a_1, d_1, \ldots, d_{t-1}, a_t|Y)$$

- Models *multi-scale spatial/temporal* dependencies
- Matching, comparison, classification **more efficient** O(cn)

- Differences in sequence modeling

S = HKYNQLIM

S′= HKINQIIM

mismatch (5,1)

XKYNQ	XYNQL
HXYNQ	KXNQL
HKXNQ	KYXQL
HKYXQ	KYNXL
HKYNX	YKNQX
XNQLI	XQLIM
YXQLI	NXLIM
YNXLI	**NQXIM**
YNQXI	NQLXM
YNQLX	NQLIX

XKINQ	XINQI
HXINQ	KXNQI
HKXNQ	KIXQI
HKIXQ	KINXI
HKINQ	KINQX
XNQII	XQIIM
IXQII	NXIIM
INXII	**NQXIM**
INQXI	NQIXM
INQIX	NQIIX

double-(1,5)

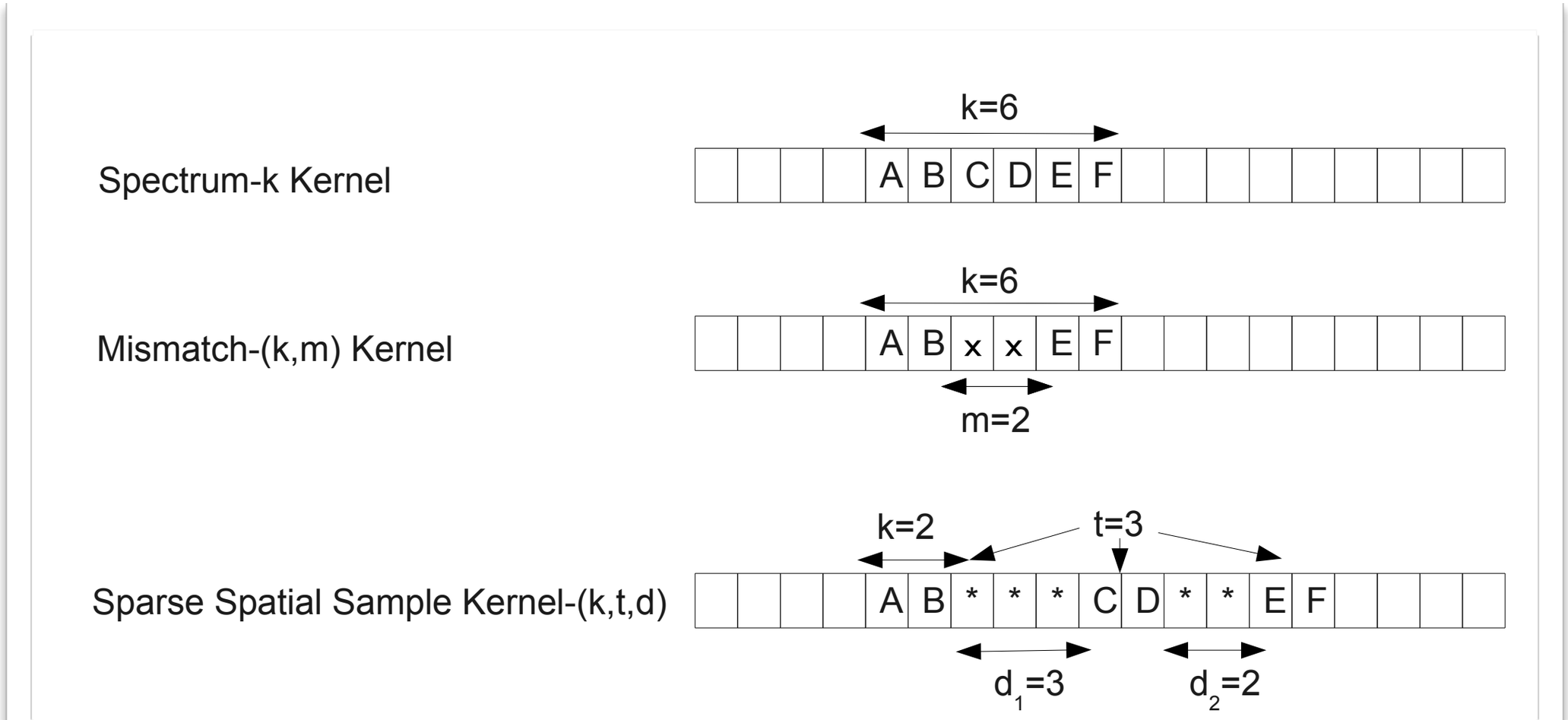
HK	H_Y	**H__N**	**H___Q**	H____L
KY	**K_N**	**K__Q**	K___L	**K____I**
YN	Y_Q	Y__L	Y___I	Y____M
NQ	N_L	**N__I**	**N___M**	
QL	**Q_I**	**Q__M**		
LI	L_M			
IM				

HK	H_I	**H__N**	**H___Q**	H____I
KI	**K_N**	**K__Q**	K___I	**K____I**
IN	I_Q	I__I	I___I	I____M
NQ	N_I	**N__I**	**N___M**	
QI	**Q_I**	**Q__M**		
II	I_M			
IM				

- mismatch: very few feature retained
- spatial: still significant overlap in features

State-of-the-art kernels

- Features employed by state-of-the-art kernels



k - length of the feature/sample
m - # mismatches
t - number of samples
d - max. distance between samples

* - gap
x - wildcard

Traditional String Kernels

- *Feature space*: indexed by all possible *k*-mers: sparse embedding (spectrum, SSSK), dense embedding (mismatch, profile)

- *Hamming distance*-based matching

$$K(X,Y|k,m) = \sum_{\gamma \in \Sigma^k} \Phi_{k,m}(\gamma|X)\Phi_{k,m}(\gamma|Y)$$

$$= \sum_{\alpha \in X} \sum_{\beta \in Y} \sum_{\gamma \in \Sigma^k} I_m(\alpha,\gamma) I_m(\beta,\gamma)$$

$$I(\alpha,\beta) = \begin{cases} 1, h(\alpha,\beta) \leq m \\ 0, otherwise \end{cases}$$

- Drawback: Hamming distance may not reflect true underlying similarity/dissimilarity:
  - ex: different pairs of AA induce different levels of similarity
  - matching of word k-grams should reflect semantic similarity, not only simple character-level differences

=> need more "precise" similarity measures!

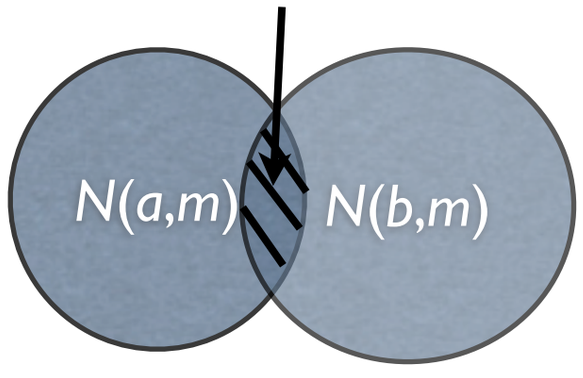
Part III:
Generalized Similarity Kernels

General similarity kernels

- Traditional string kernels: *k*-mer similarity is measured as the number of identical substrings in the mismatch neighborhoods (i.e. intersection size)

$$\sum_{\gamma \in \Sigma^k} I_m(\alpha, \gamma) I_m(\beta, \gamma)$$

intersection size



How to define more "precise" and meaningful similarity?

General Similarity kernels

- Idea: replace intersection size-based similarity with metric similarity function

$$\sum_{\gamma \in \Sigma^k} I_m(\alpha, \gamma) I_m(\beta, \gamma) \rightarrow \mathcal{S}(\alpha, \beta)$$

- General similarity kernel

$$K(X, Y | k, \mathcal{S}) = \sum_{\alpha \in X} \sum_{\beta \in Y} \mathcal{S}(\alpha, \beta)$$

  - Complexity in general becomes quadratic!
  - This is not a proper kernel!

=> How to incorporate general metric S and preserve linear time complexity?

General Similarity kernels

- Idea: replace quadratic time computation with linear time Hamming computation!
- Two approaches:

1. *Cluster kernel:* cluster *k*-mers according to S, match *k*-mers from same clusters
2. *Similarity-preserving kernel*: use *symbolic* approximation of S

1. Cluster similarity kernel

Feature set: $\mathcal{F}$ = set of *k*-mers

Cluster indicator function (according to S):

$$L : \mathcal{F} \rightarrow \{C_1, \ldots, C_n\}$$

- defines partitioning of the feature set into n disjoint subsets

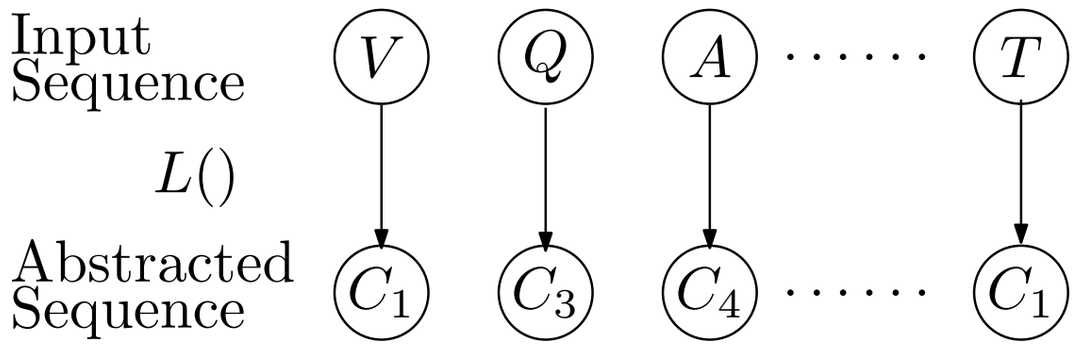
Cluster similarity kernel:

$$K_L(X,Y) = \sum_{\gamma \in \mathcal{F}} \sum_{\alpha \in X} I_{\mathcal{S}}(\alpha,\gamma) \sum_{\beta \in Y} I_{\mathcal{S}}(\beta,\gamma)$$
$$= \sum_{\alpha \in X} \sum_{\beta \in Y} \sum_{\gamma \in \mathcal{F}} I_{\mathcal{S}}(\alpha,\gamma) I_{\mathcal{S}}(\beta,\gamma)$$

$$I_{\mathcal{S}}(\alpha,\beta) = \begin{cases} 1, \text{if } L(\alpha) = L(\beta) \\ 0, \text{otherwise} \end{cases}$$

1. Cluster similarity kernel

- Constructs abstracted sequence using similarity metric S



- Benefits:
  - can incorporate non-symbolic/non-Hamming interrelationships between symbolic *k*-mers
  - still linear time computation

- Drawback: does not necessarily quantify similarity level, i.e., how similar are the two *k*-mers?

2. Similarity-preserving kernel

1. Learn symbolic embedding *E* for which Hamming distance accurately approximates *S*

$$h(E(a),E(b)) \sim S(a,b)$$

2. Use *Hamming-type computation* to efficiently evaluate kernel $K(X, Y|\mathcal{S})$ in linear time

$$K(X, Y|k, \mathcal{S}) = \sum_{\alpha \in X} \sum_{\beta \in Y} \mathcal{S}(\alpha, \beta)$$

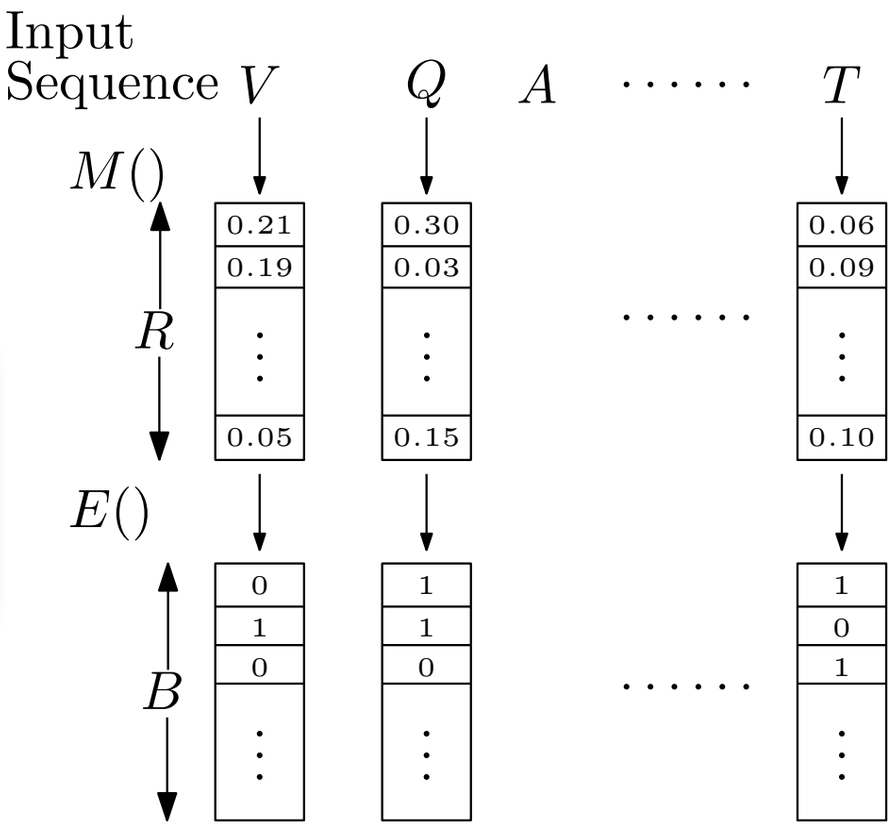
2.1: symbolic embedding

Similarity hashing:

$$\min \sum_{\alpha,\beta} S(\alpha,\beta) h(E(\alpha), E(\beta))^2$$

s.t. $E(\alpha) \in \{-1, 1\}^B$

Idea: minimize average Hamming distance between similar data points

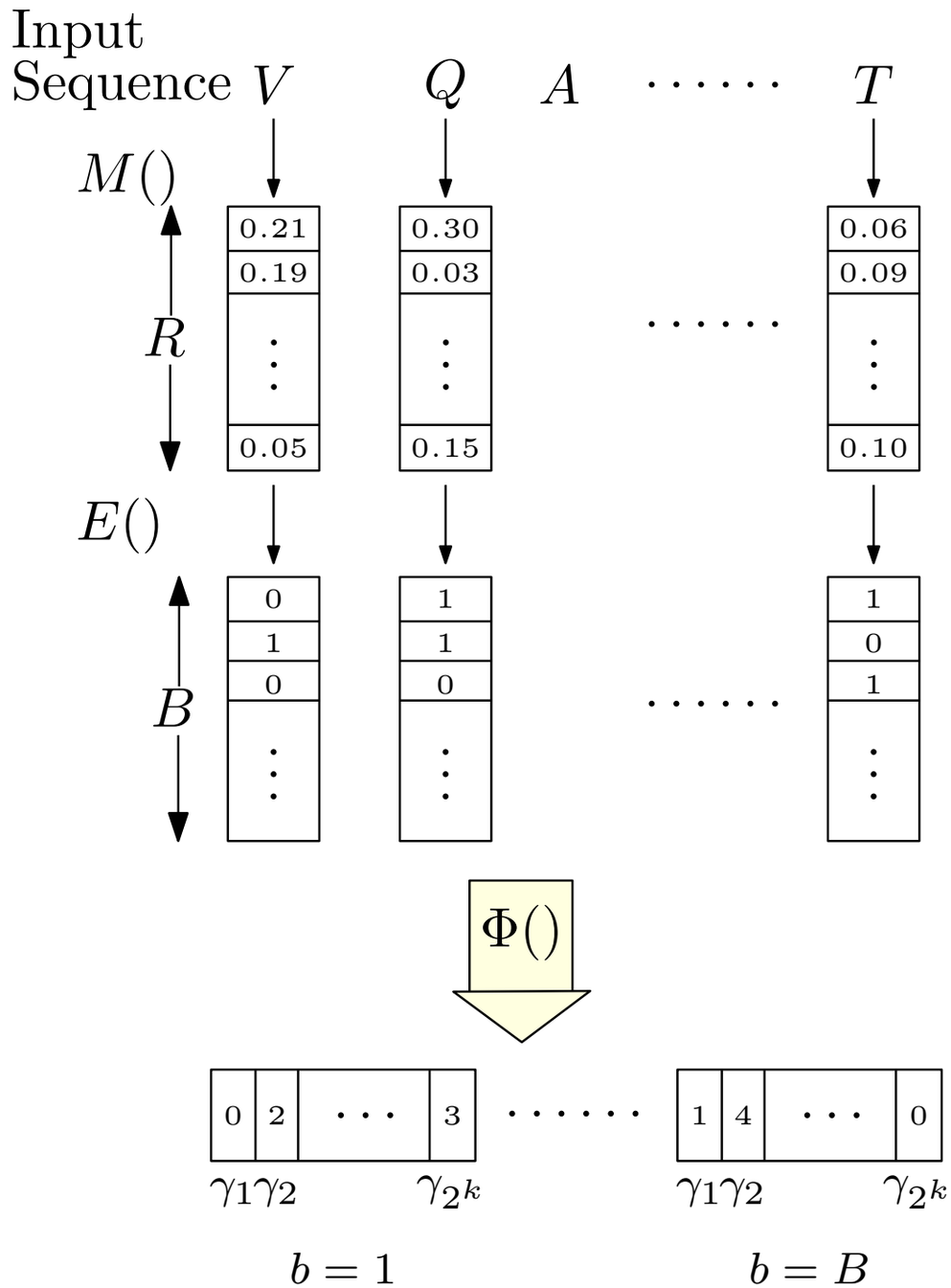


$$E(X) = E(x_1), ..., E(x_n)$$

$$E(x_i) = e_1^i e_2^i \dots e_B^i \qquad e_j^i \in \{0, 1\}$$

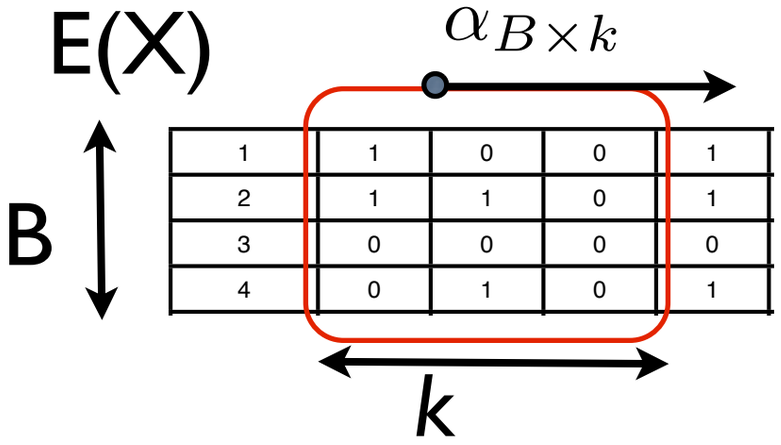
$$h(E(\alpha), E(\beta)) \propto \mathcal{S}(\alpha, \beta)$$

2.2: similarity-preserving feature embedding



- separate 2^k feature map for each dimension $b=1...B$

Similarity-preserving kernel



$$K_k(E(X), E(Y)) = \langle \Phi_k(E(X)), \Phi_k(E(Y)) \rangle \tag{5.9}$$

$$= \sum_{b=1}^{B} \sum_{\gamma \in \{0,1\}^k} \left(\sum_{\alpha \in E(X)} I(\alpha[b], \gamma) \sum_{\beta \in E(Y)} I(\beta[b], \gamma) \right)$$

$$= \sum_{b=1}^{B} \sum_{\alpha \in E(X)} \sum_{\beta \in E(Y)} I(\alpha[b], \beta[b]) \tag{5.10}$$

$$= \sum_{\alpha \in E(X)} \sum_{\beta \in E(Y)} \underbrace{\sum_{b=1}^{B} I(\alpha[b], \beta[b])}_{s_{\alpha\beta}} \tag{5.11}$$

number similar rows in a and b

Kernel value approximates general similarity function

$$K(X, Y|k, \mathcal{S}) = \sum_{\alpha \in X} \sum_{\beta \in Y} \mathcal{S}(\alpha, \beta)$$

$$\max\{h_{\alpha\beta} - (k-1)B, 0\} \leq s_{\alpha\beta} \leq \frac{h_{\alpha\beta}}{k}$$

$$h_{\alpha\beta} = h(E(\alpha), E(\beta)) \propto \mathcal{S}(\alpha, \beta)$$

=> K(E(X),E(Y)) ~ K(X,Y|S)

Similarity-preserving kernels

- Benefits:
  - Incorporate metric similarity function S into matching
    - Enhances representation by capturing interrelationships between sequence elements and features according to S
      - =>Refine matching of otherwise symbolically different sequence elements/features
    - Computationally efficient: linear time!

Generalized similarity kernels summary

- Incorporate general (*non-Hamming*) metric similarity function *S*
  - *by matching k-mers that are similar according to S (cluster kernel)*
  - *by retaining (approximate) actual similarity values (similarity-preserving kernel)*
- Computationally efficient: *linear time*
- *Empirically, improved predictive performance on many important problems*

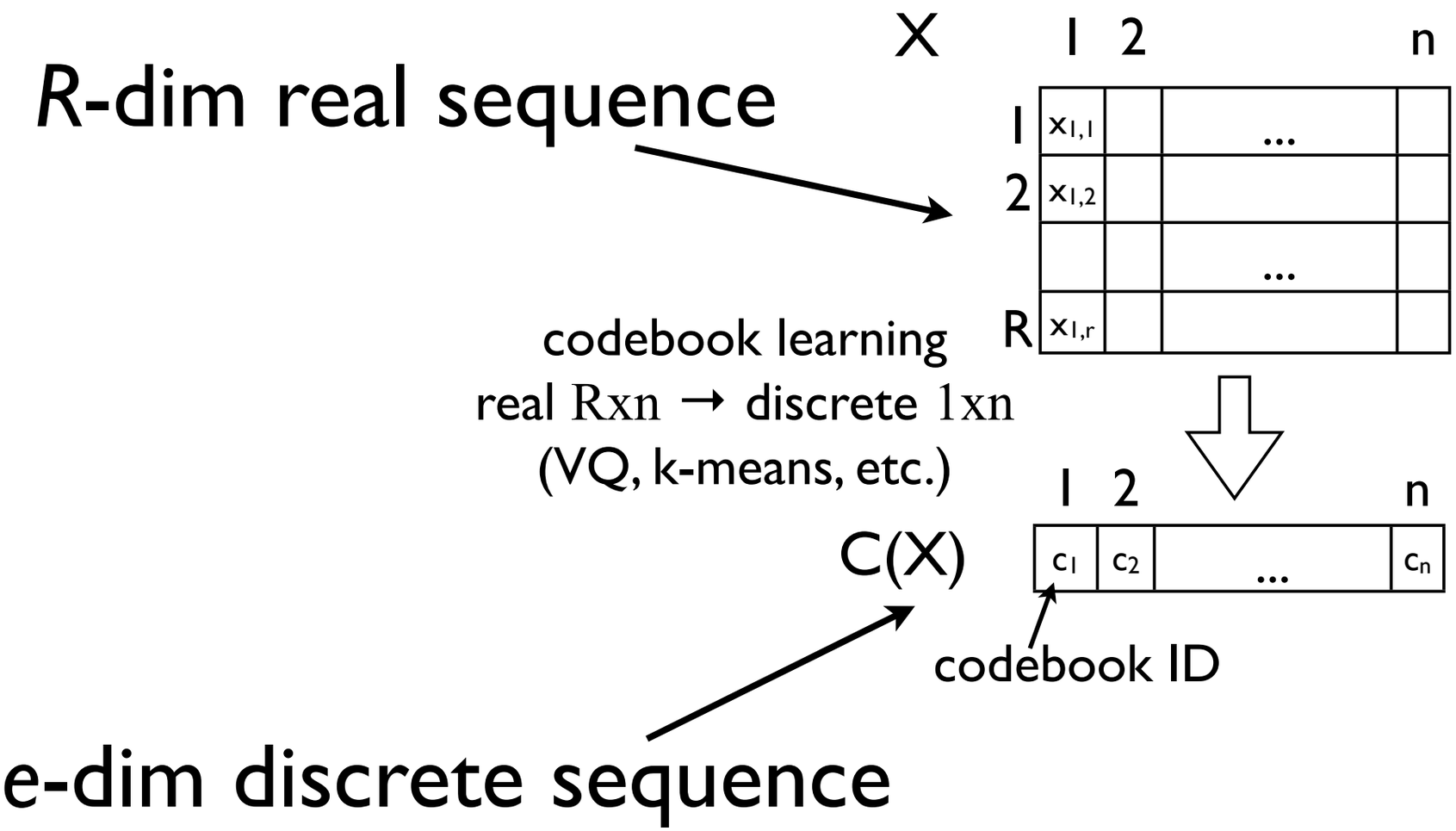
Part IV:
Multivariate sequence kernels

Multivariate sequence analysis

- Original data is often in the form of *multivariate* sequences, i.e. sequences of identically-sized feature vectors
  - Ex1: audio sequence = sequence of 13-dim MFCC vectors
  - Ex2: protein sequence = sequence of AA descriptors (eg., profile)
  - Ex3: image = 2D sequence of feature vectors for image patches in a grid
- Standard kernels: defined for one-dimensional strings! => how to solve multivariate sequence problem?

I: Codebook learning approach

- Use standard string kernel K(C(X),C(Y)) defined on pairs of codeword sequences:



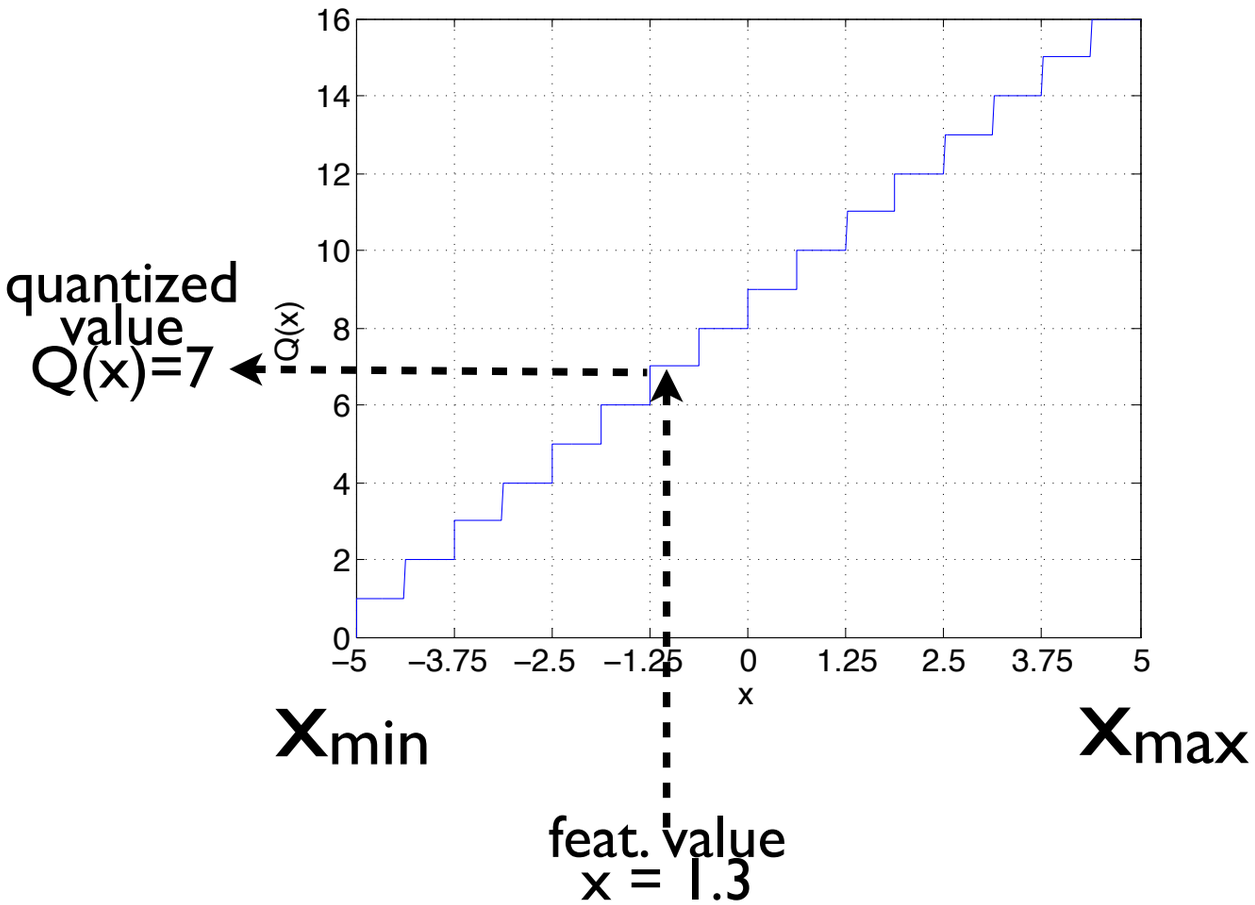
one-dim discrete sequence

Codebook learning approach

- Drawbacks:
  - not all dimensions are equally important
  - codeword derived from high-dim noisy samples
  - distance computation introduces bias for certain dimensions
  - does not retain features-time correlation information

Multivariate direct feature quantization

- Idea: direct quantization of each dimension



$$\delta = (f_{max} - f_{min})/B$$

$$Q(f) = \lfloor (f - f_{min})/\delta \rfloor$$

- Example: MVDFQ representation

Original multivariate time series X

	t=1	t=2	t=3	t=4	t=5
dim 1	0.43	1.43	3.79	2.53	3.29
dim 2	-0.34	0.91	2.97	1.68	2.12
dim 3	-0.41	0.40	2.22	1.15	1.74

Discrete multivariate representation (DFQ(X), B=64 bins)

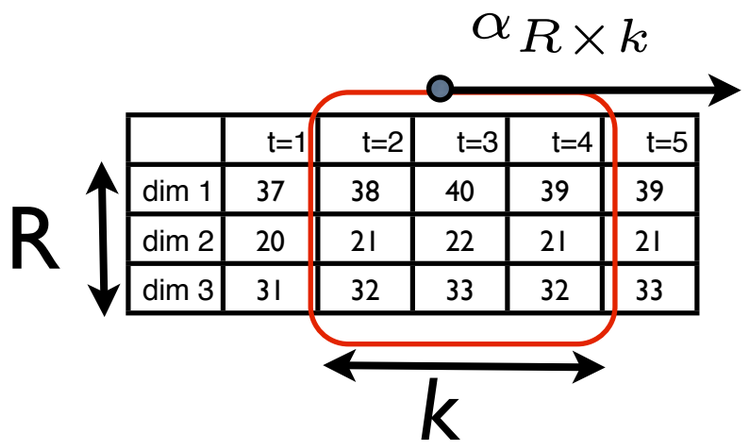
	t=1	t=2	t=3	t=4	t=5
dim 1	37	38	40	39	39
dim 2	20	21	22	21	21
dim 3	31	32	33	32	33

Univariate representation (VQ(X), codebook size=2048)

	t=1	t=2	t=3	t=4	t=5
dim 1	309	173	484	1148	1252

MVDFQ kernel:

$$K_{MVDFQ}(DFQ(X), DFQ(Y)) = \sum_{\alpha_{R\times k} \in DFQ(X)} \sum_{\beta_{R\times k} \in DFQ(Y)} \mathcal{K}(\alpha_{R\times k}, \beta_{R\times k})$$



Submatrix kernel:

$$\mathcal{K}(\alpha_{R\times k}, \beta_{R\times k}) = \sum_{r=1}^{R} I_{1\times k}(\alpha^r_{R\times k}, \beta^r_{R\times k})$$

similar matrices => many similar rows => high kernel value

Complexity: linear *O*(*Rkn*)

Manifold embedding

- Multinomial manifold (L1 embedding)

$$\hat{\Phi}(X) = \left(\frac{\phi_1(X)}{\sum_i \phi_i(X)}, \ldots, \frac{\phi_d(X)}{\sum_i \phi_i(X)} \right)$$

- Bhattacharyya affinity

$$K_{manifold}(X,Y) = < \sqrt{\hat{\Phi}(X)}, \sqrt{\hat{\Phi}(Y)} > = \sum_i \sqrt{\hat{\phi}_i(X)} \sqrt{\hat{\phi}_i(Y)}$$

- MVDFQ-Manifold kernel

$$K_{MVDFQM}(X,Y) = \sum_{r=1}^{R} \sum_{\gamma} \sqrt{\phi^r_\gamma(X)} \sqrt{\phi^r_\gamma(Y)}$$

$$\phi^r_\gamma(X) = \sum_{\alpha^r_{R\times k} \in DFQ(X)} I(\alpha^r_{R\times k}, \gamma)$$

Part V:

Experimental evaluation

Experiments

- Content-based automatic music classification
  - music genre recognition
  - artist identification
- MHC-peptide binding prediction
- Protein remote homology prediction
- Bio-medical text classification (BioCreative II)

Music classification

- Datasets:

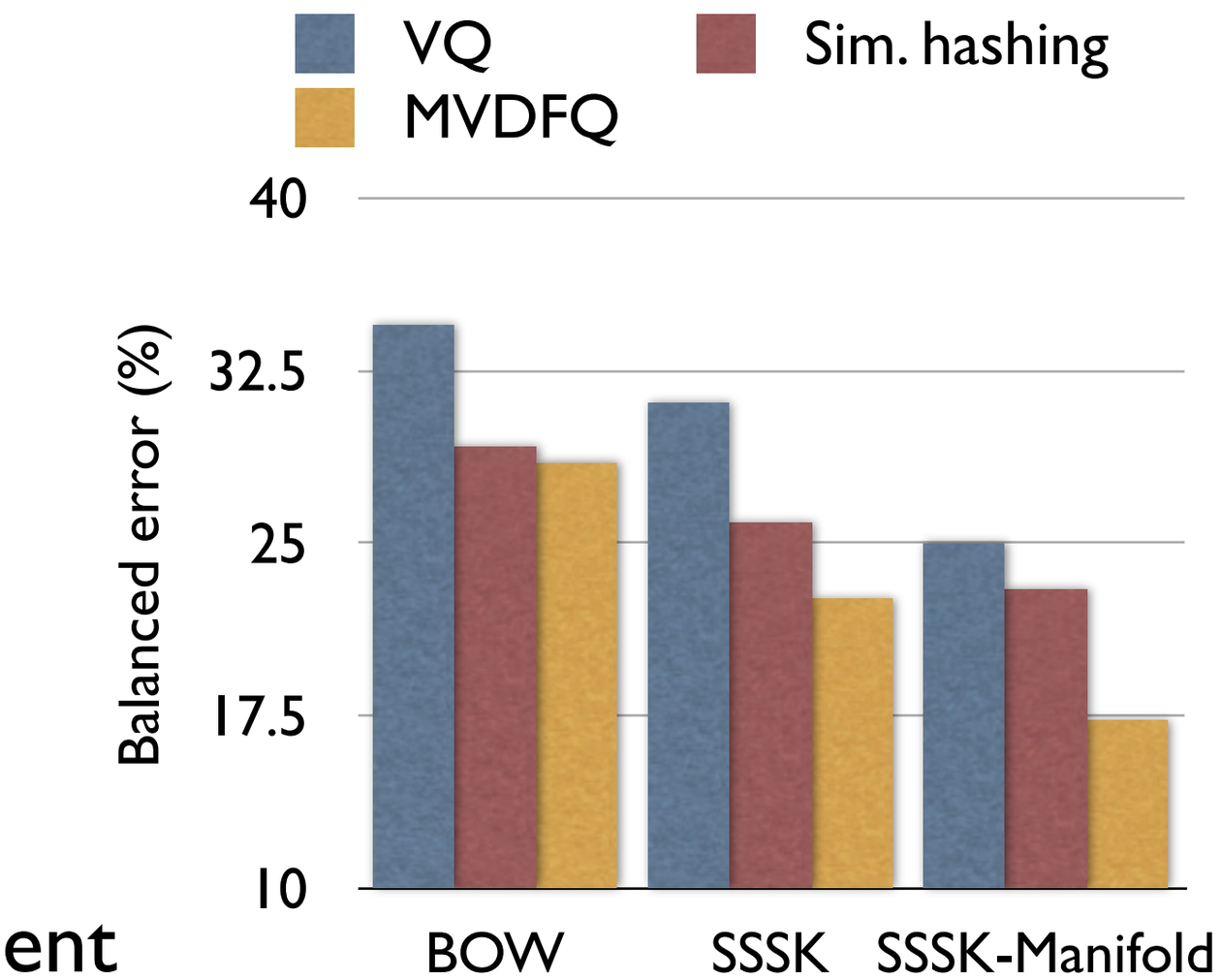
	#classes	#seqs	features	ave. seq length
GTZAN	10 genres	1000 songs	13-dim MFCC	6980
ISMIR	6 genres	1458	13-dim MFCC	10151
Artist ID	20 artists	1413	13-dim MFCC	19708

- Settings:

VQ	$\|\Sigma\|$ = 2048 codewords
Sim. hashing	$\|E\|$=32,64 bits
MVDFQ	B = 32 bins

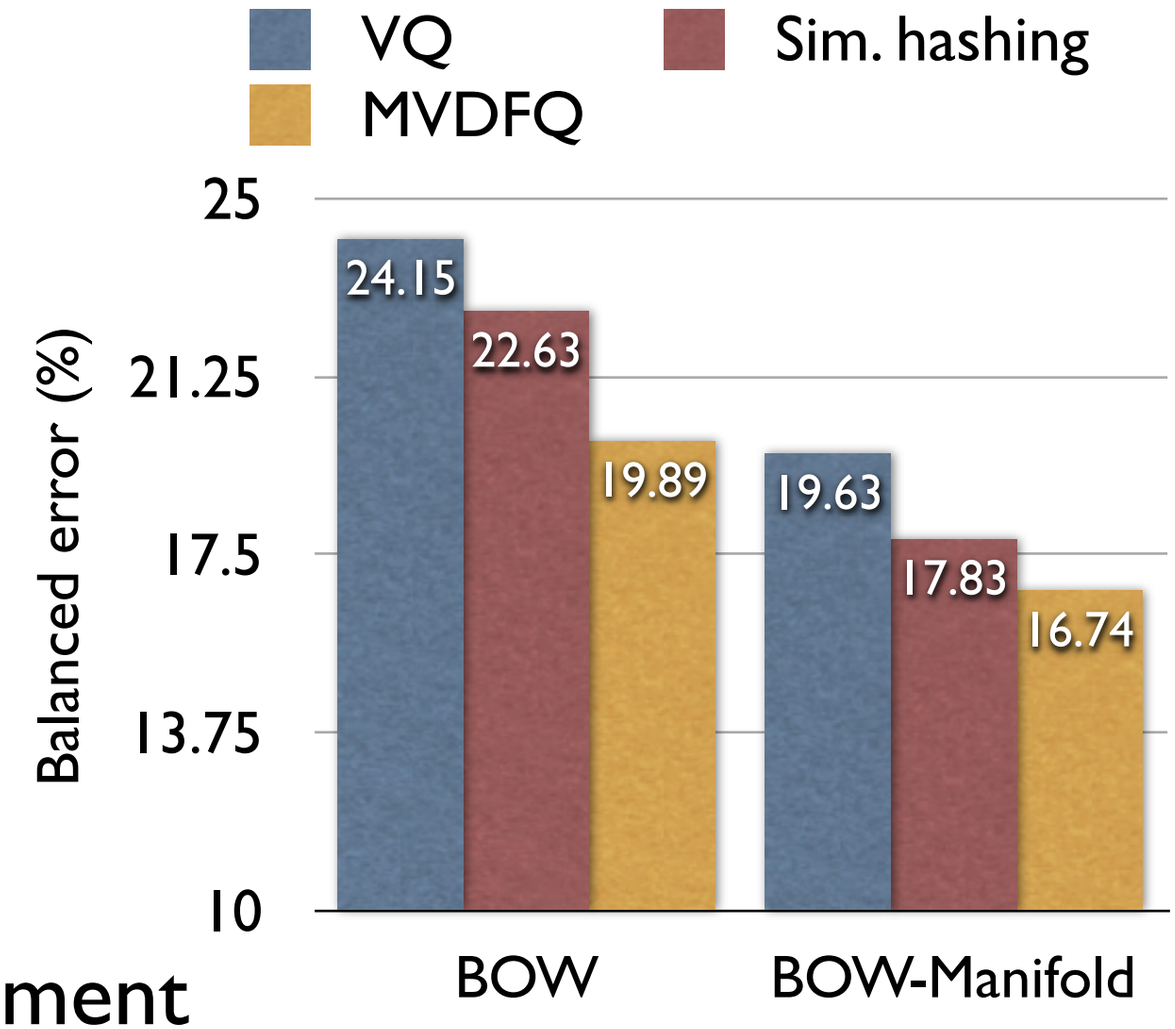
Music genre recognition

- GTZAN:



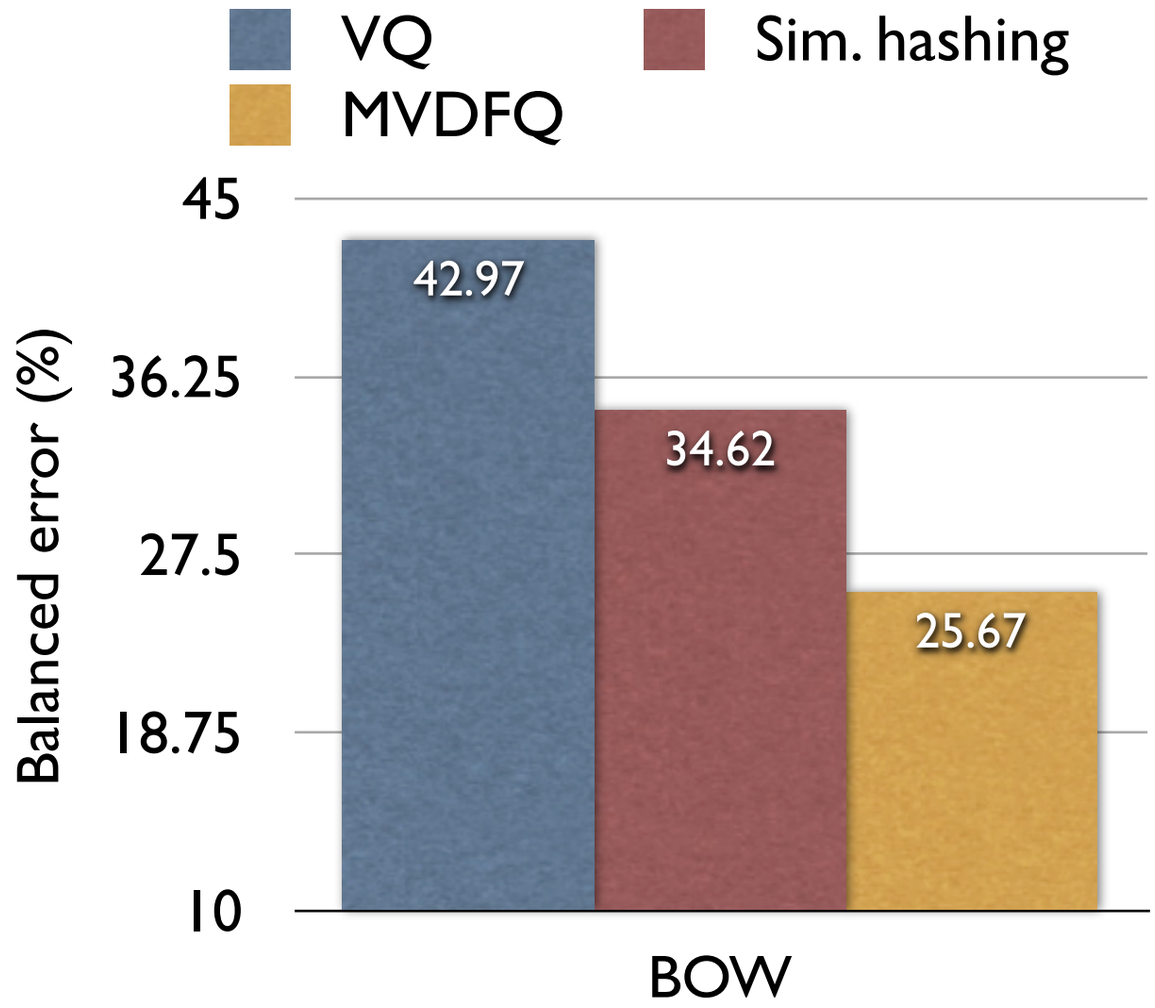
Improvement (MVDFQ vs VQ): 17% 26% 30%

ISMIR contest



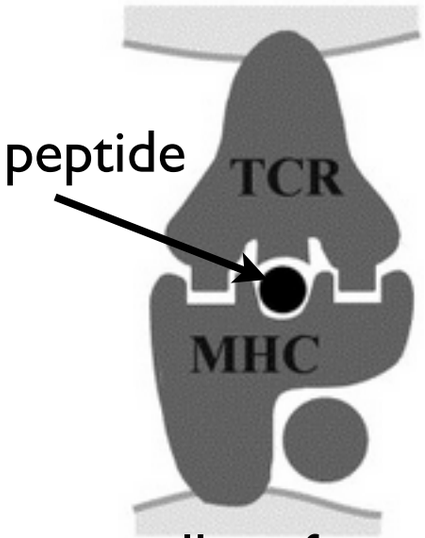
Improvement (MVDFQ vs VQ): 17% 14%

Artist ID


VQ
Sim. hashing
MVDFQ
45
36.25
27.5
18.75
10
Balanced error (%)
42.97
34.62
25.67
BOW

- Task 2: MHC-peptide binding prediction

- MHC-Peptide binding prediction task



Position	Peptide	binding strength
1	MVTEFIFLG	3.47
2	VTEFIFLGL	6.50
3	TEFIFLGLS	3.185
..	.....	....
..	.....	....

MVTEFIFLGLSD.....

antigen
(target protein)

Peptide binding prediction

Test ROC scores

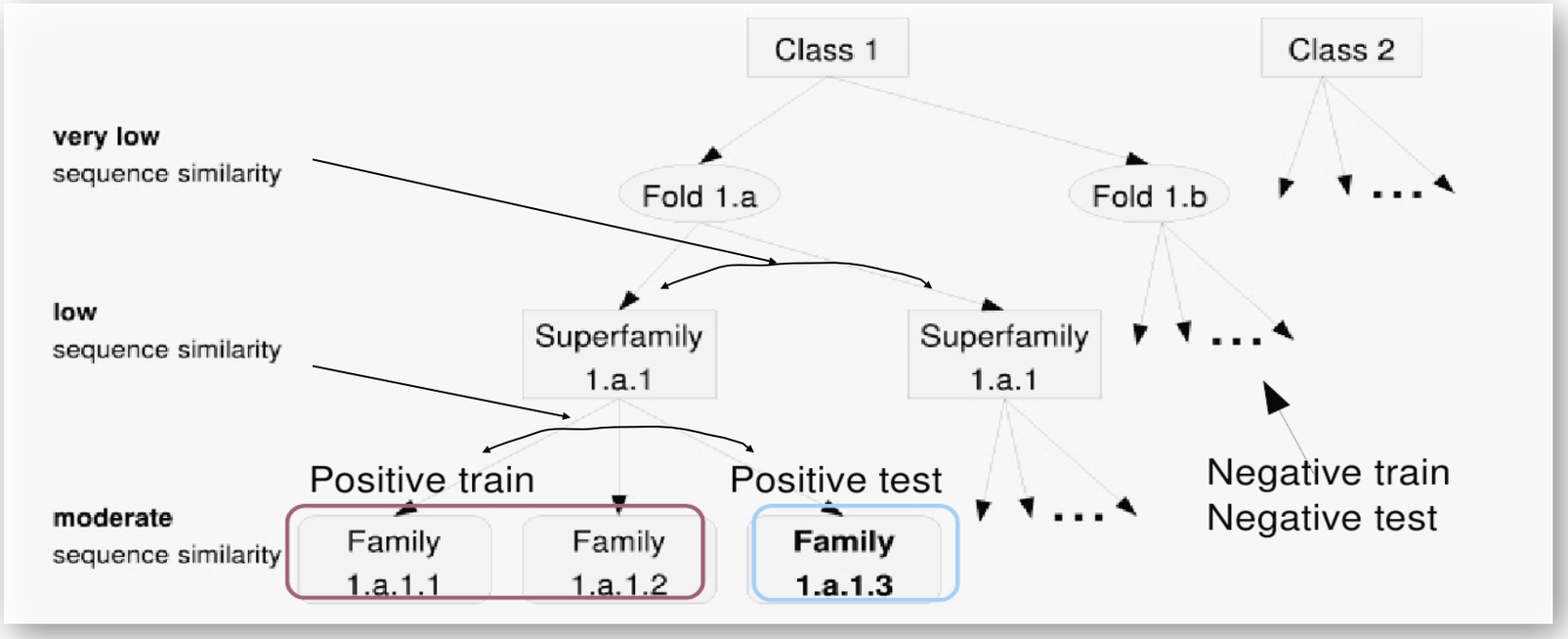
Allele	QHMM	NetMHC	MVSK
A0201	62.1	76.9	**80.17**
A0206	67.03	**83.93**	77.67
A2402	66.51	76.35	**85.24**

- QHMM = committee of HMMs
- NetMHC = highly optimized neural network predictor

- Task 3: Protein remote homology prediction

Protein remote homology prediction

Structural classification of proteins (SCOP)



Ex: prediction task

Target superfamily: 1.a.1		
	positive	negative
train	1.a.1.1 1.a.1.2	other sf/ folds
test	1.a.1.3	other sf/ folds

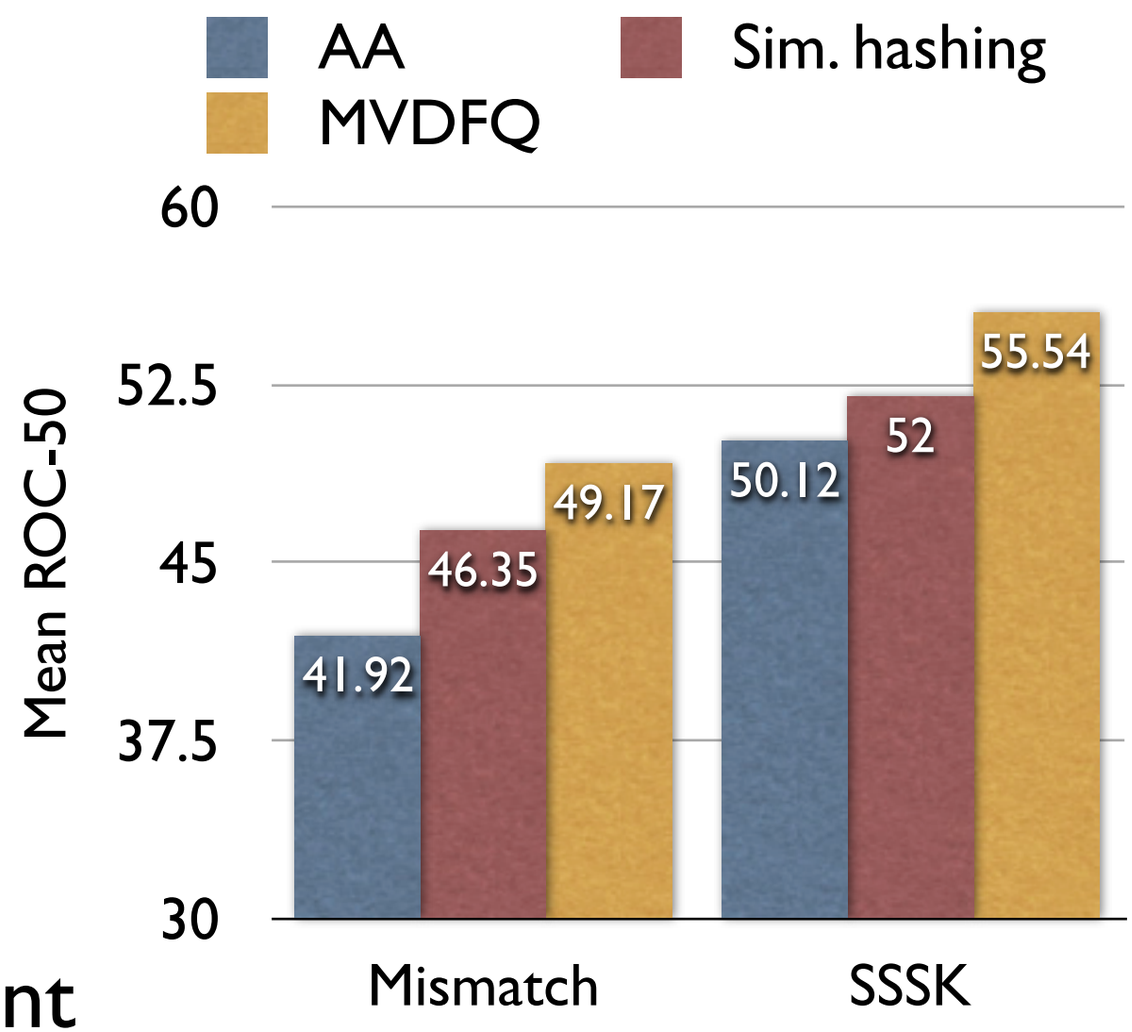
- Task: infer membership of a protein in the target superfamily
  - performance is averaged over 54 tasks

Protein remote homology prediction

Data	Supervised setting	Semi-supervised setting
SCOP 1.59	7359 seqs	+1M unlab seqs (NRDB)

- MVDFQ, Sim. Hashing: map each amino acid into 20-dim descriptor using BLOSUM (BLOcks SUbstitution Matrix)
  - Sim. hashing: each protein is encoded as 8-dim sequence, |E|=8 bits
  - MVDFQ: each protein is encoded as 20-dim sequence with B=16 bins

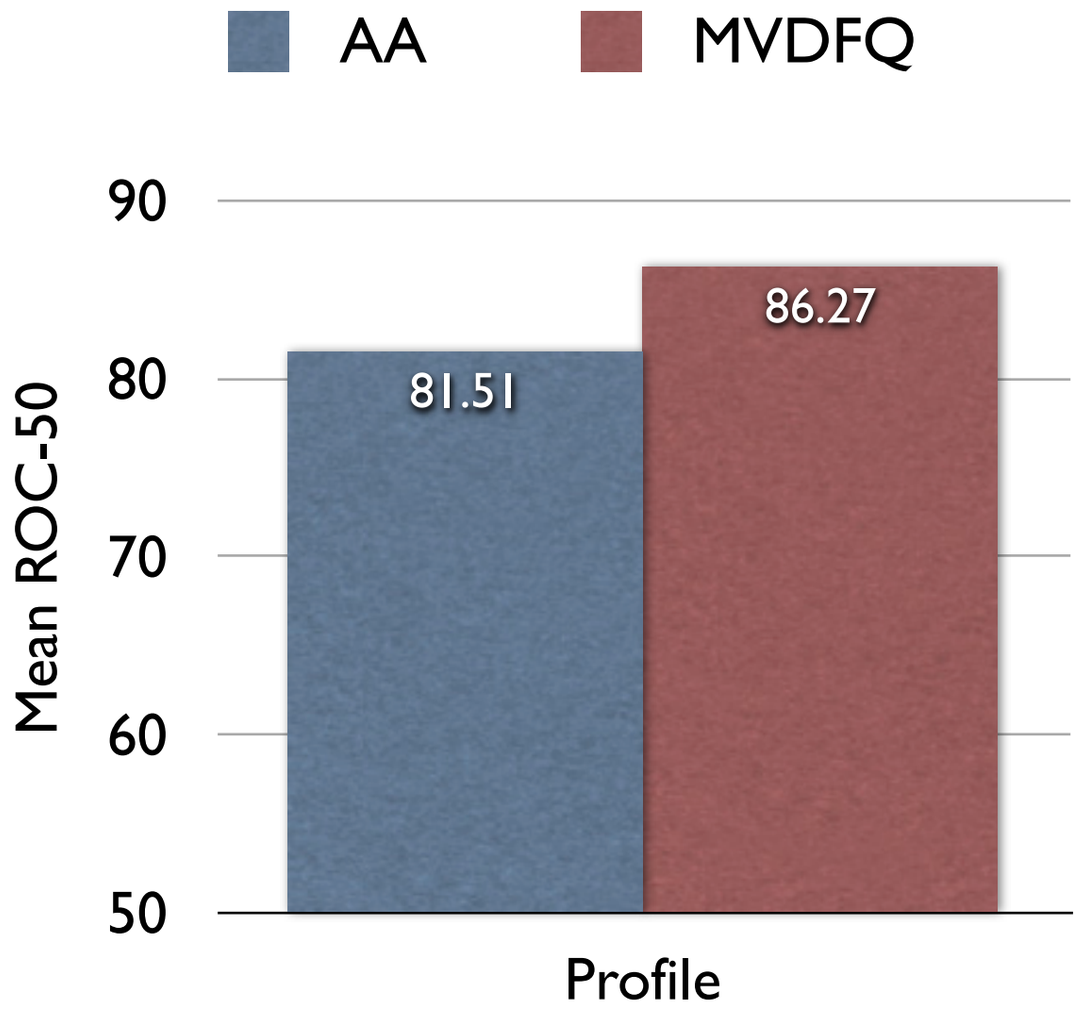
Protein Remote homology



Improvement (MVDFQ vs AA): 17% 11%

AA = string kernel defined on amino acid sequences

Protein remote homology



BioCreative II

- IAS task: classify article abstracts into protein-protein interaction (PPI)-related or not
  - Labeled data: 5495 abstracts for training / 677 for testing
  - Unlabeled data: 60M sentences from PubMed (95-today) (~1.3G words)
    - use unlabeled data to construct word descriptors (ECML, 2010)

BioCreative II contest

- IAS test performance

method	F1	Accuracy
BioCreative II (best)	78.00	75.33
BioCreative II (rank 2)	77.95	77.10
String kernel (best)	77.17	74.30
MVSK	**80.11**	**79.03**

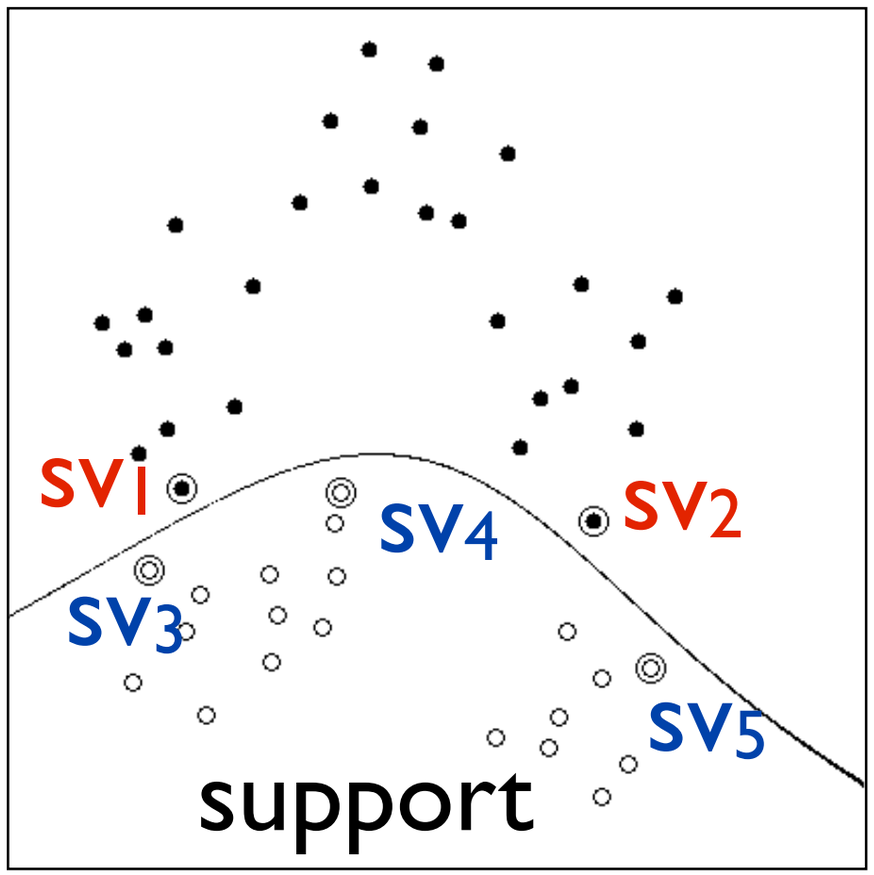
Conclusions

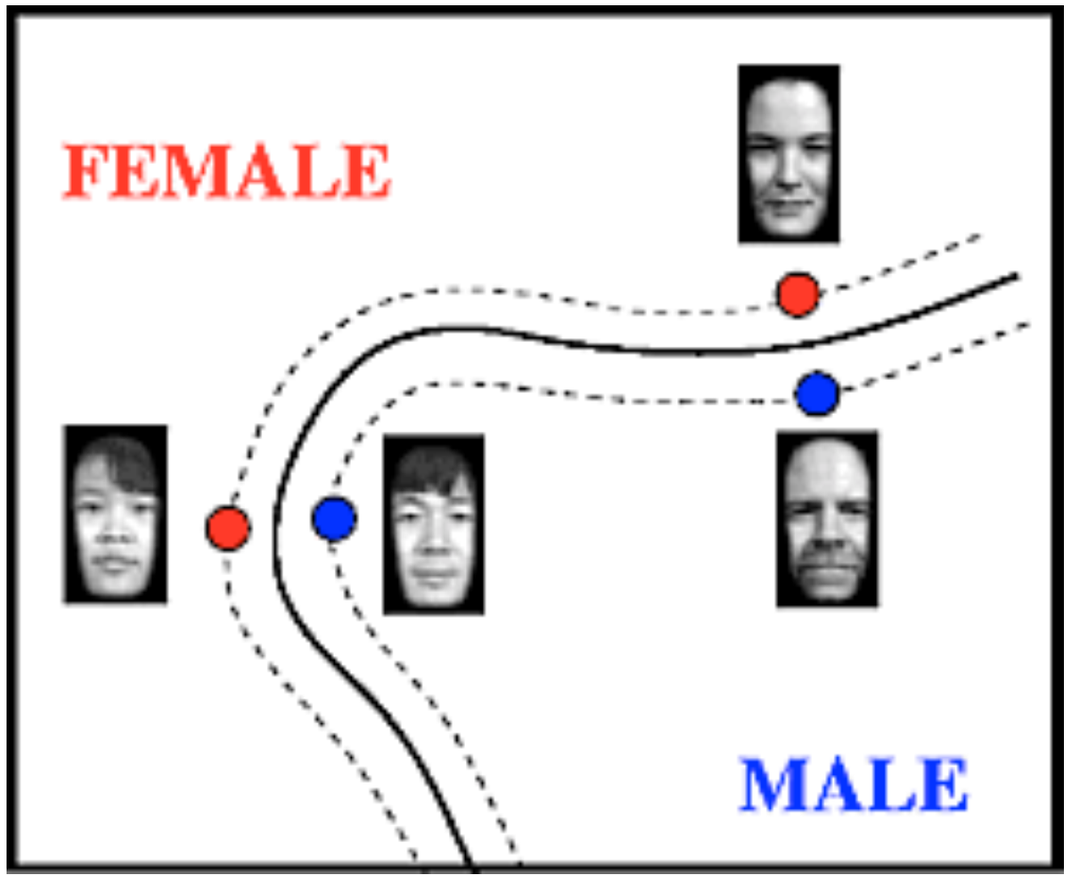
- Generalized similarity kernels: using general (non-Hamming) metric similarity functions helps classification
- Multivariate DFQ: more effective for classification compared to standard codebook learning
- Generalize across many domains: discrete (AA sequences, text), real (music)
- Computationally efficient (linear time)

References

1. Pavel P. Kuksa, Pai-Hsi Huang, Vladimir Pavlovic. *Scalable Algorithms for String Kernels with Inexact Matching.* NIPS, 2008
2. Pavel P. Kuksa and Vladimir Pavlovic. *Efficient evaluation of large sequence kernels*. In *KDD*, 2012
3. Pavel P. Kuksa. *2D similarity kernels for biological sequence classification*. In *BIOKDD*, 2012.
4. Pavel P. Kuksa, Imdadullah Khan, and Vladimir Pavlovic. *Generalized Similarity Kernels for Efficient Sequence Classification*. In *SDM*, 2012.
5. Pavel P. Kuksa, Yanjun Qi, Bing Bai, Ronan Collobert, Jason Weston, Vladimir Pavlovic, and Xia Ning. *Semi-Supervised Abstraction-Augmented String Kernel for Multi-Level Bio-Relation Extraction*. In *ECML*, 2010
6. Christina Leslie and Rui Kuang. *Fast string kernels using inexact matching for protein sequences*. JMLR, 2004
7. Pavel P. Kuksa and Vladimir Pavlovic. *Spatial Representation for Efficient Sequence Classification*. ICPR, 2010
8. Christina S. Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. *Mismatch string kernels for svm protein classification*. NIPS , 2002

Predictors: Support Vector Machines





Ming-Hsuan Yang

$$\text{Predictor}(X) = \sum_{i \in SV} \alpha_i K(sv_i, X)$$

Advantages: can capture *complex relations* in data and scale to *large problems*

- Two subproblems
  - one related to *representation*, i.e. devising a proper set of descriptors
  - one related to *statistical learning*, i.e. finding smooth, stable function that maps descriptors into output (e.g., class label)