Fast String Kernels using Inexact Matching for Protein Sequences

C. Leslie, R. Kuang, 2004

presented by Laur Tooming
26 Feb 2008

The problem

- Given: a number of strings in an alphabet Σ.
- The strings must be classified into two groups.
- The classification is given on training data and must be learned from it.

ACTGGCATGTAGCTAGACT

TCGATCTACTACT

TCGATGTCTACGGACGA

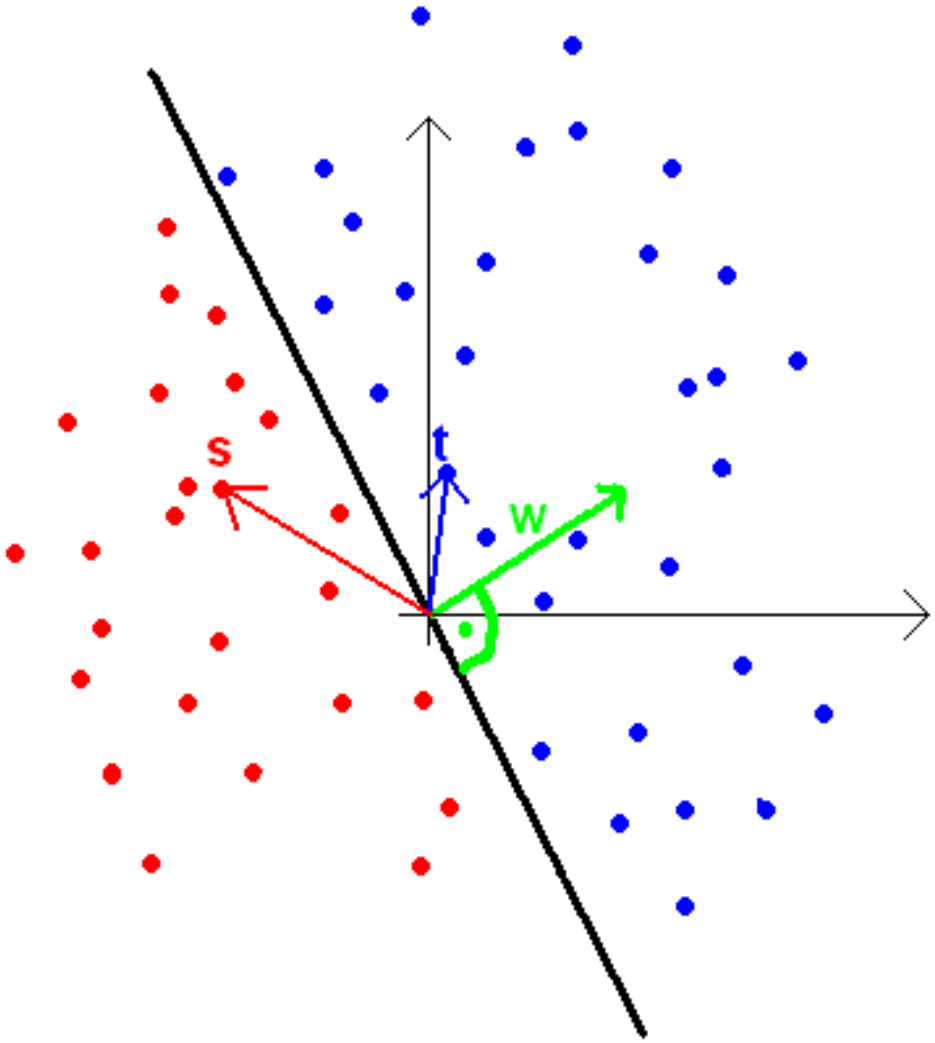
TAGCTGACTGTTACG

ACTGAGCTAGTCTATCGACGTGT

GACTGATCGTAGCTGA

SVMs

- Instead of strings, consider classifying points in n-dimensional space.
- We could try separating the two groups by a (n-1)-dimensional hyperplane.
- To test which side of the hyperplane a point is on, we need *dot products* with its normal vector.

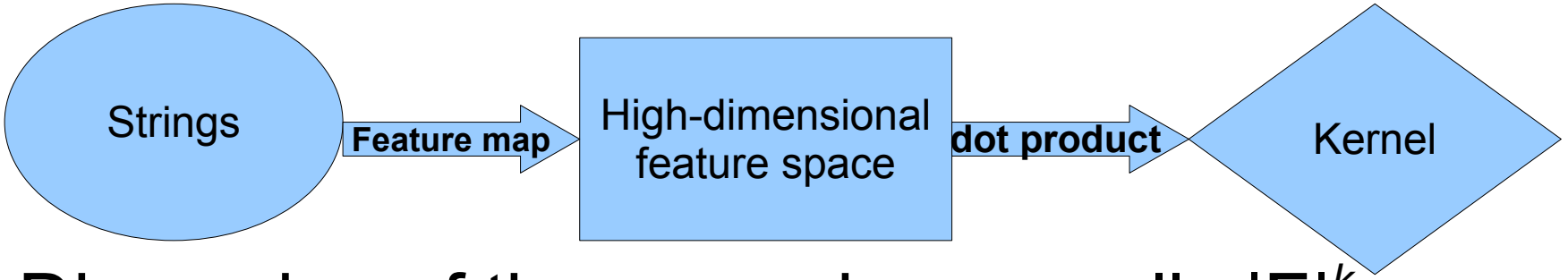


$\langle s, w \rangle < 0$

$\langle t, w \rangle > 0$

Kernels

- To use SVMs on strings, we must map the strings into a vector space.
- „Similar“ strings should become close points.



- Dimension of the space is generally $|\Sigma|^k$.
- Kernel $\langle \Phi(s), \Phi(t) \rangle$ should be directly computable from the strings s and t.

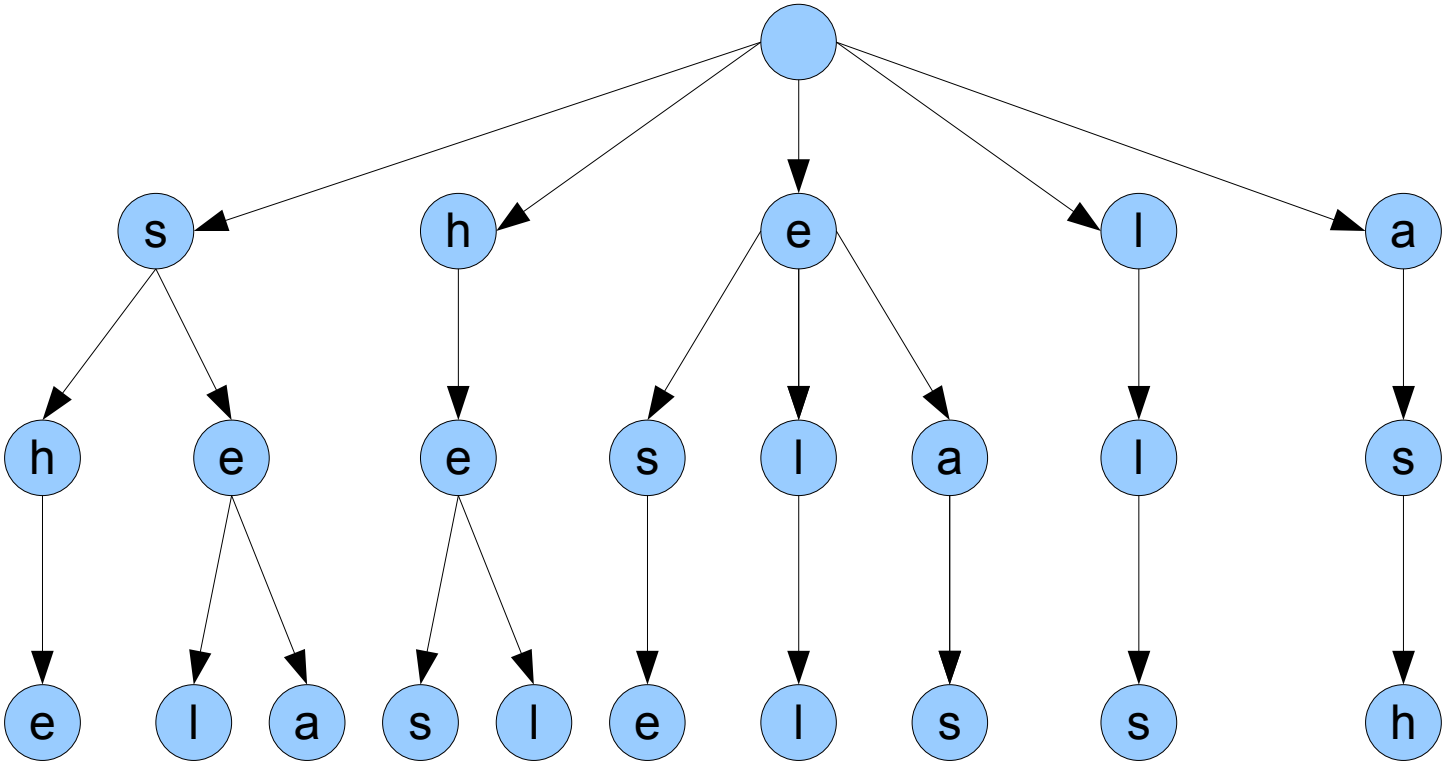
k-spectrum kernel

- |Σ|^*k*-dimensional feature space, indexed by all *k*-length strings (*k*-mers)
- A coordinate of Φ(s) shows how many times the given *k*-mer occurs in s.
- Example: s = „abracadabra“, *k* = 4, Σ = {a,b,c,d,r}
- Φ(s) = (0 , 0 , ... , 2 , ... , 1 , ... , 0)

  aaaa aaab abra acad rrrr
- The kernel: K(„abracadabra“,“acadarabracbrac“) =
  = 2*1 (abra) + 1*2 (brac) + 1*1 (acad) + 1*1 (cada) = 6.
- A kernel can also be considered a similarity measure between strings.

k-spectrum kernel computation

- s = „shesells“, t = „seashells“, $k = 3$
- Time complexity $O(k(|s| + |t|))$
- Trie
- Similar algorithms work on other fast string kernels



(*k,m*)-mismatch kernel

- The spectrum kernel counted only exact matching substrings. The following examples allow inexact matching.
- We consider continuous *k*-mers of a string and allow up to *m* mismatching characters.
- For a *k*-mer b, the b-coordinate of Φ(s) is the count of *k*-mers in s differing from b in at most *k* positions.
- s = „abracadabra“, *k*=4, *m*=2: „raca“-coordinate is 2; „aaaa“-coordinate is ....
- $O(k^{m+1}|\Sigma|^m(|s| + |t|))$ (i.e. depends on alphabet size)
- There are quite a lot of non-zero coordinates in the feature vectors.

(k,m,λ)-wildcard kernel

- In each k-substring, it is allowed to replace up to m characters with wildcards („abcd“ -> „*b*d“)
- Feature space is indexed by k-mers with up to m wildcards.
- b-coordinate of $\Phi(s)$ is the number of k-mers in s matching b, multiplied by $\lambda^{\text{number of wildcards in b}}$. λ: wildcard penalty, between 0 and 1.
- $O(k^{m+1}(|s|+|t|))$ - no dependence on alphabet size.
- Example: k=4, m=2, s=“string“, t=“seeing“, λ=0.8
- $K(s,t) = 0.8^2 \cdot 0.8^2$ (s**i) + $0.8^2 \cdot 0.8^2$ (**in) + $0.8 \cdot 0.8$ (*ing) + $0.8^2 \cdot 0.8^2$ (**ng) + $0.8^2 \cdot 0.8^2$ (*i*g)+ $0.8^2 \cdot 0.8^2$ (*in*)

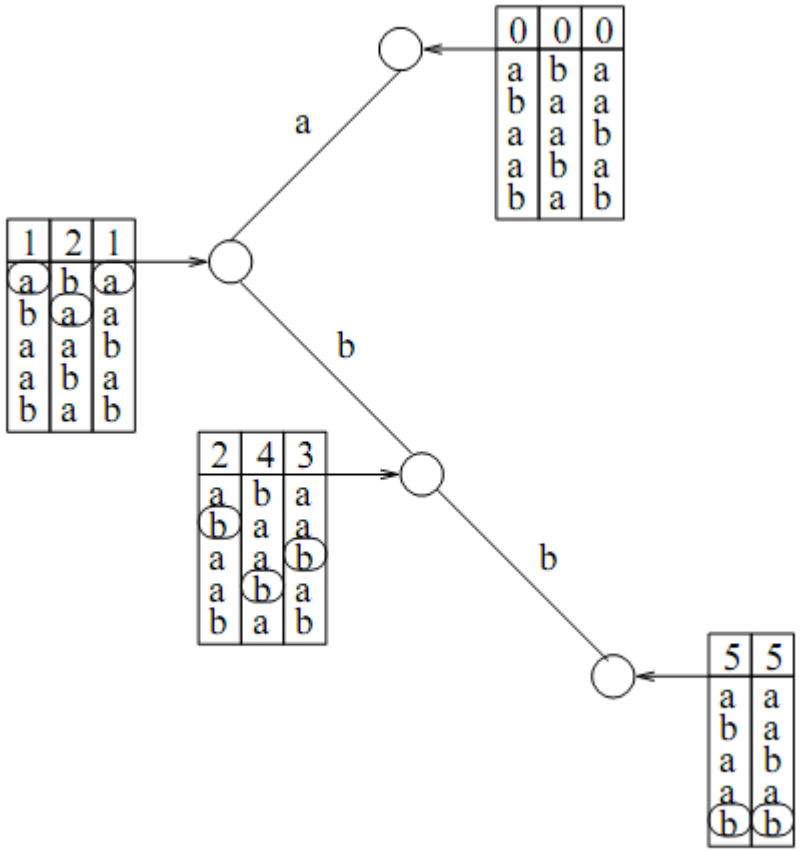
Substitution kernel

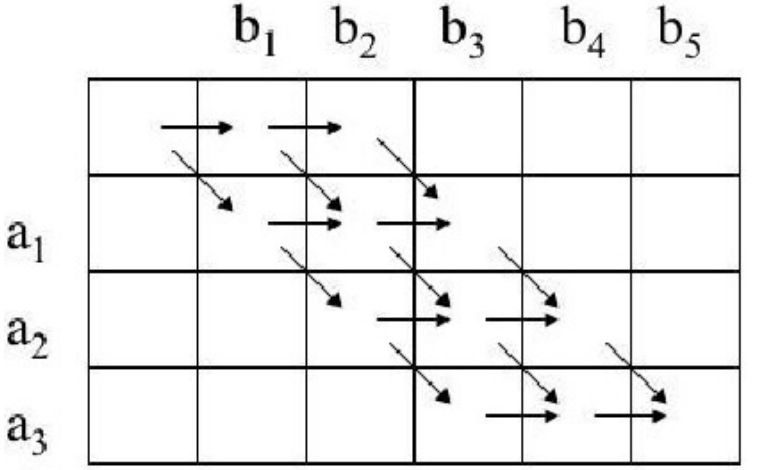
- Probabilistic model: for each pair of characters *a* and *b*, P(*b*|*a*) is the probability of *a* mutating into *b.* (Presumably, P(*a*|*a*) is large.)
- Parameters: *k* (*k*-mer size) and *σ* (probability treshold).
- The mutation neighbourhood of *k*-mer ($a_1 a_2 ... a_k$) consists of *k*-mers ($b_1 b_2 ... b_k$) such that
  $-[\log P(b_1|a_1) + ... + \log P(b_k|a_k)] < \sigma.$
- b-coordinate of Φ(s) is the number of *k*-substrings of s that are in the mutation neighbourhood of b.
- $O(kN(|s| + |t|))$, where *N* is max neighbourhood size.

(g,k)-restricted gappy kernel

- In the feature vector, we count k-subsequences whose beginning and end are no more than g apart.
- s = „abracadabra“, k=3, g=6: k-mer „aaa“ occurs in 1st, 3rd, 4th, 6th g-mer, so it gets coordinate 4.
- Optional parameter λ <= 1 can be used to penalise gaps. For example, „aaa“ in each position has 2 gaps, so the coordinate would be $4\lambda^2$.
- Restriction by g is important, as it reduces time complexity from $O(|s||t|)$ to $O[g^{g-k}(|s|+|t|)]$ (unweighted) or $O[k(g-k)g^{g-k}(|s|+|t|)]$ (weighted).

Gappy kernel computation





(A)

	a	b	a	b	b
	1	1	1			
a		1	λ	λ^2+1		
b			1	λ	$2\lambda^2+1$	
b				0	λ	$2\lambda^2+\lambda+1$

(B)

Transducer representation

- A *transducer* consists of:
  - a set of *states*, among them some *input states* and some *output states*
  - *transitions* from one state to another, each of which may *accept* one input symbol, may *emit* one output symbol and has a *weight*.
  - The weight of a path is the product of transition weights.
- Define mapping into feature space indexed by all output strings. For a string s, consider all paths in the given transducer that accept s. For a possible output string
- That way, every (regular) transducer defines a kernel, and many of previously seen string kernels can be represented by a transducer.

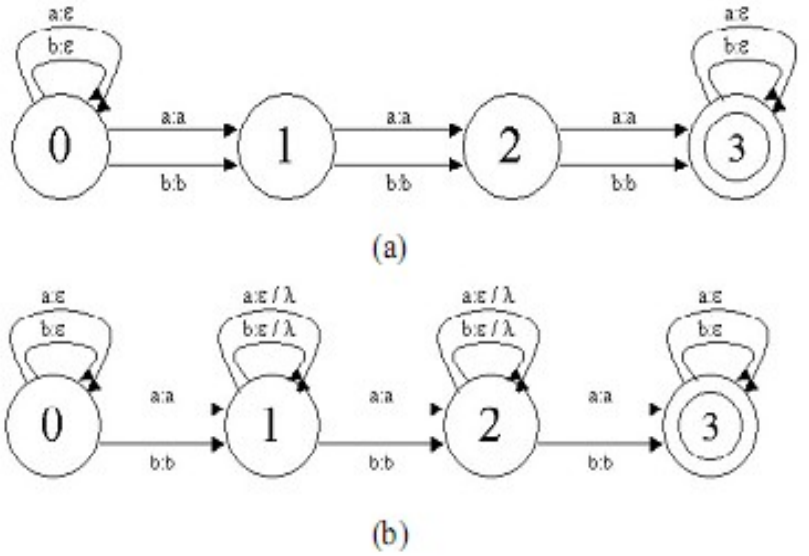


Figure 3: **The k-gram and gappy k-gram transducers.** The diagrams show the 3-gram transducer (a) and the gappy 3-gram transducer (b) for a two-letter alphabet. Here, the edge label $a : \varepsilon : \lambda$, for example, means "accept symbol a, output the empty symbol ε, multiply the weight by λ".

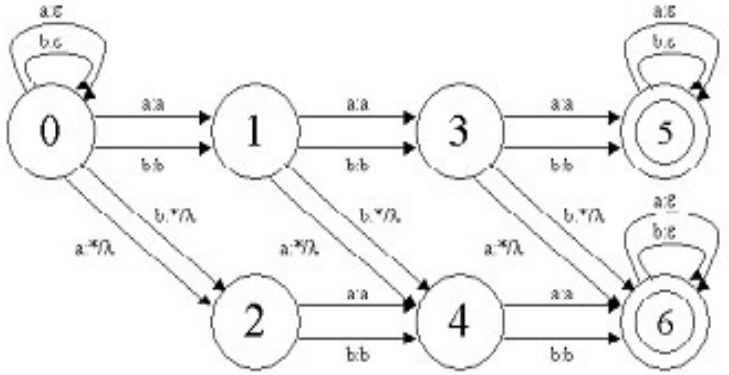


Figure 4: **The (k, m)-wildcard transducer.** The diagram shows the $(3, 1)$-wildcard transducer for a two-letter alphabet.

Evaluation

- SCOP (Structural Classification of Proteins) data
- Using SVM classifyers to learn families of proteins
- ROC (receiver operating characteristic) score calculated for each family with each kernel
- Kernel alignment scores
- Testing for best parameters for each kernel
- Comparable to older quadratic-time kernels (Fisher kernel), although not better
- Combining many kernels may be a good idea

References

- Leslie, Esin, Noble. The spectrum kernel: a string kernel for SVM protein clasification. 2002
- Leslie, Eskin, Cohen, Weston, Noble. Mismatch string kernels for discriminative protein classification. 2004
- Leslie, Kuang. Fast string kernels using inexact matching for protein sequences. 2004