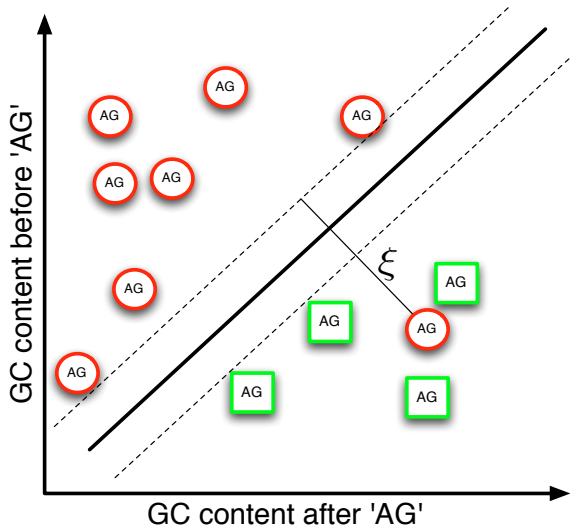
Lecture Thu 24.11.

KERNELS FOR SEQUENTIAL DATA

Soft-Margin SVM (Cortes & Vapnik, 1995)

The soft-margin SVM allows some of the training points to have smaller margin than $\gamma(x) = 1$, subject to a penalty:

$$\begin{aligned} \textit{Minimize} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ \textit{Subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geqslant 1 - \xi_i \\ & \text{for all } i = 1, \ldots, n. \\ & \xi_i \geq 0 \end{aligned}$$



- ξ_i is called the slack variable, when positive the margin $\gamma(x_i) < 1$
- The sum of slacks is to be minimized so the objective still favours hyperplanes that separates the classes well
- The coefficient $C > 0$ controls the balance between maximizing the margin and the amount of slack needed

Dual Soft-Margin SVM

A dual optimization problem (gives the same solution) to the soft-margin SVM is

$$\begin{aligned} \textit{Maximize} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \textit{Subject to} \quad & 0 \leq \alpha_i \leq C \\ & \text{for all } i = 1, \ldots, n. \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

- Note that the slack variables (ξ) and the weight vector ($\mathbf{w}$) have disappeared, the only variables to be optimized are the α_i's
- The data only appears through the kernel functions $k(x_i, x_j)$
- Derivation requires techniques of optimization theory (See e.g. Boyd S, Vandenberghe L, Convex Optimization. Cambridge University Press, 2004)

Common Kernels

$$
\begin{aligned}
\text{Polynomial} \quad & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) &&= (\langle \mathbf{x}, \hat{\mathbf{x}} \rangle + c)^d \\
\text{Sigmoid} \quad & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) &&= \tanh(\kappa \langle \mathbf{x}, \hat{\mathbf{x}} \rangle) + \theta) \\
\text{RBF} \quad & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) &&= \exp\left(-\|\mathbf{x} - \hat{\mathbf{x}}\|^2 / (2\,\sigma^2)\right) \\
\text{Convex combinations} \quad & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) &&= \beta_1 \mathrm{k}_1(\mathbf{x}, \hat{\mathbf{x}}) + \beta_2 \mathrm{k}_2(\mathbf{x}, \hat{\mathbf{x}}) \\
\text{Normalization} \quad & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) &&= \frac{\mathrm{k}'(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{\mathrm{k}'(\mathbf{x}, \mathbf{x}) \mathrm{k}'(\hat{\mathbf{x}}, \hat{\mathbf{x}})}}
\end{aligned}
$$

What if our data is not in the vector form already?

Kernels for non-vectorial data

Examples of data that is originally not in feature vector form:

- Sequences
- Graphs (e.g .molecular graphs)
- Images

How to compute kernels for them (efficiently)?

The String Kernel Recipe

General idea

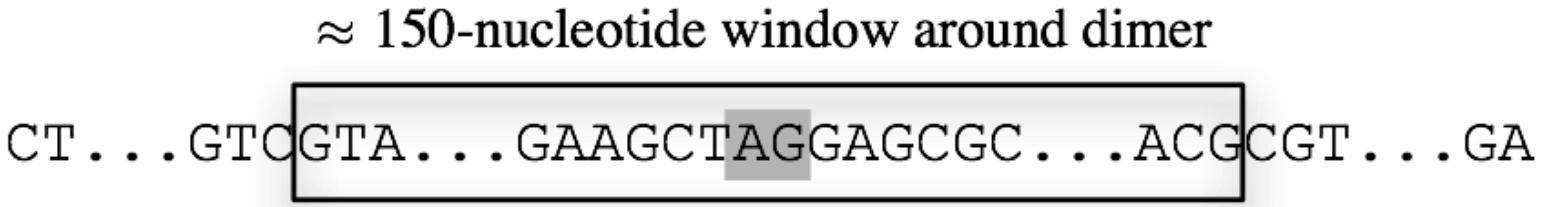
- Count substrings shared by two strings
- The greater the number of common substrings, the more two sequences are deemed similar

Variations

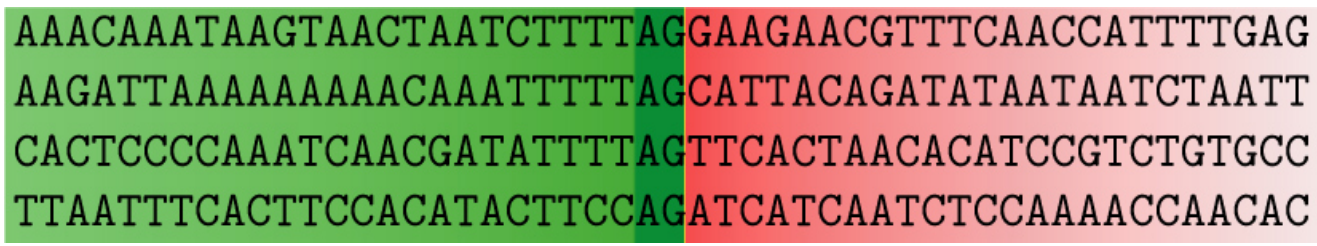
- Allow gaps
- Include wildcards
- Allow mismatches
- Include substitutions
- Motif kernels
- Assign weights to substrings

Recognizing Genomic Signals

Discriminate true signal positions from all other positions



- True sites: fixed window around a true site
- Decoy sites: all other consensus sites



Examples: Transcription start site finding, splice site prediction, alternative splicing prediction, trans-splicing, polyA signal detection, translation initiation site detection

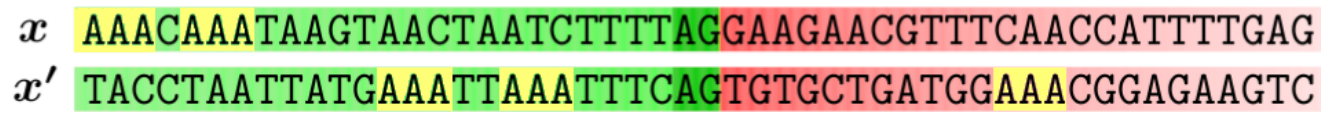
Types of Signal Detection Problems

Problem categorization

(based on positional variability of motifs)

Position-Independent

→ Motifs may occur anywhere,



for instance, tissue classification using promoter region

Problem categorization

(based on positional variability of motifs)

Position-Dependent

→ Motifs very stiff, almost always at same position,

for instance, splice site identification

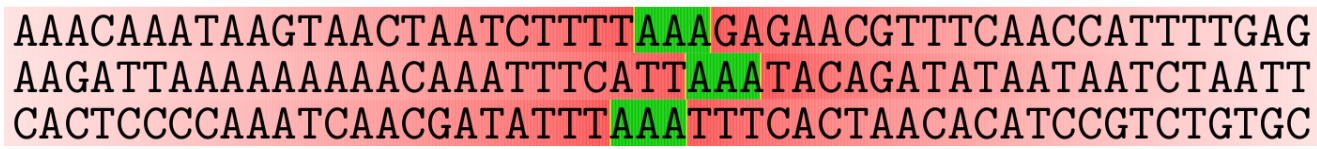
Types of Signal Detection Problems

Problem categorization

(based on positional variability of motifs)

Mixture of Position-Dependent/-Independent

$\rightarrow$ variable but still positional information



for instance, promoter identification

Spectrum Kernel

To make use of position-independent motifs:

- **Idea:** like the bag-of-words-kernel (cf. text classification) but for biological sequences (words are now strings of length k, called k-mers)
  - Count k-mers in sequence A and sequence B.
  - Spectrum Kernel is sum of product of counts (for same k-mer)

Example $k = 3$:

x AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG

x' TACCTAATTATGAAATTAAATTTCAGTGTGCTGATGGAAACGGAGAAGTC

3-mer	AAA	AAC	...	CCA	CCC	...	TTT
# in **x**	2	4	...	1	0	...	3
# in **x**′	3	1	...	0	0	...	1

$$k(\mathbf{x}, \mathbf{x}') = 2 \cdot 3 + 4 \cdot 1 + \ldots 1 \cdot 0 + 0 \cdot 0 \ldots 3 \cdot 1$$

Fast computation of spectrum kernels

Brute-force computation of k-mer spectrum would take $O(|\Sigma|^k)$ time, where Σ is the alphabet in use (DNA, Protein, ...)

- Fastest computation methods are based on suffix trees
- Principle: all suffixes of string s are stored in a suffix tree in $O(|s|)$ time, matches to k-mers in string t are read from the tree in $O(|t|)$
- Instead of suffix trees, suffix arrays can be used to save space

  [Teo and Vishwanathan, 2006]
- More about suffix data structure techniques: course 58093String Processing Algorithms

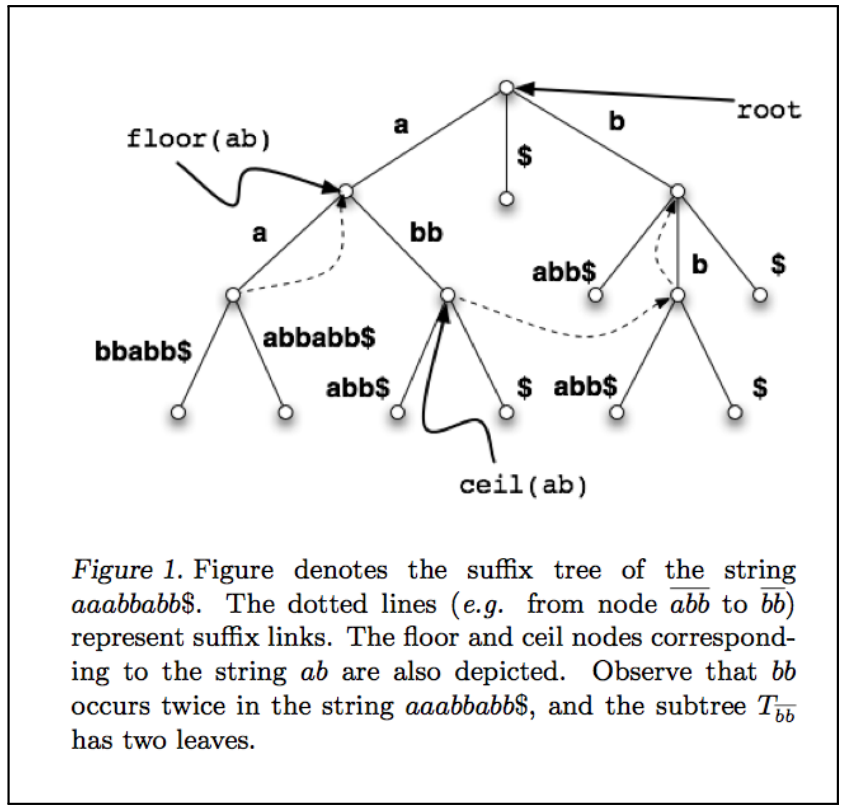


Figure 1. Figure denotes the suffix tree of the string *aaabbabb*\$. The dotted lines (*e.g.* from node $\overline{abb}$ to $\overline{bb}$) represent suffix links. The floor and ceil nodes corresponding to the string *ab* are also depicted. Observe that *bb* occurs twice in the string *aaabbabb*\$, and the subtree $T_{\overline{bb}}$ has two leaves.

Spectrum Kernel with Mismatches

General idea [Leslie et al., 2003]

- Do not enforce strictly exact matches
- Define mismatch neighborhood of ℓ-mer s with up to m mismatches $N_{\ell,m}(s)$: all length-ℓ sequences that differ from s by at most m mismatches
- Construct feature $\phi_\beta(s) = 1$ if $s \in N_{k,m}(\beta)$, and

$$\phi^{\text{Mismatch}}_{(l,m)}(s) = (\phi_\beta(s))_{\beta \in \Sigma^\ell}$$

- For sequence x of any length, the map is then extended as:

$$\phi^{\text{Mismatch}}_{(l,m)}(\mathbf{x}) = \sum_{\ell\text{-mers } s \text{ in } \mathbf{x}} (\phi^{\text{Mismatch}}_{(l,m)}(s))$$

- The mismatch kernel is the inner product in feature space defined by:

$$k^{\text{Mismatch}}_{(l,m)}(\mathbf{x}, \mathbf{x}') = \left\langle \Phi^{\text{Mismatch}}_{(l,m)}(\mathbf{x}), \Phi^{\text{Mismatch}}_{(l,m)}(\mathbf{x}') \right\rangle$$

Spectrum Kernel with weighted Gaps

General idea [Lodhi et al., 2002]

- Allow gaps in matches, down-weight matches with g gaps by λ^g, $0 < \lambda \leq 1$
- Let $\mathbf{i} = (i_1, \ldots, i_k)$ be a set of indices to string s, $gaps(\mathbf{i}) = i_k - i_1 + 1 - k$
- Feature $\phi_\beta(s) = \sum_{\mathbf{i}:s(\mathbf{i})=\beta} \lambda^{gaps(\mathbf{i})}$, sums up the weights of the matches
- Feature vector

$$\phi^{\mathrm{Gap}}(s) = (\phi_\beta(s))_{\beta \in \Sigma^\ell}$$

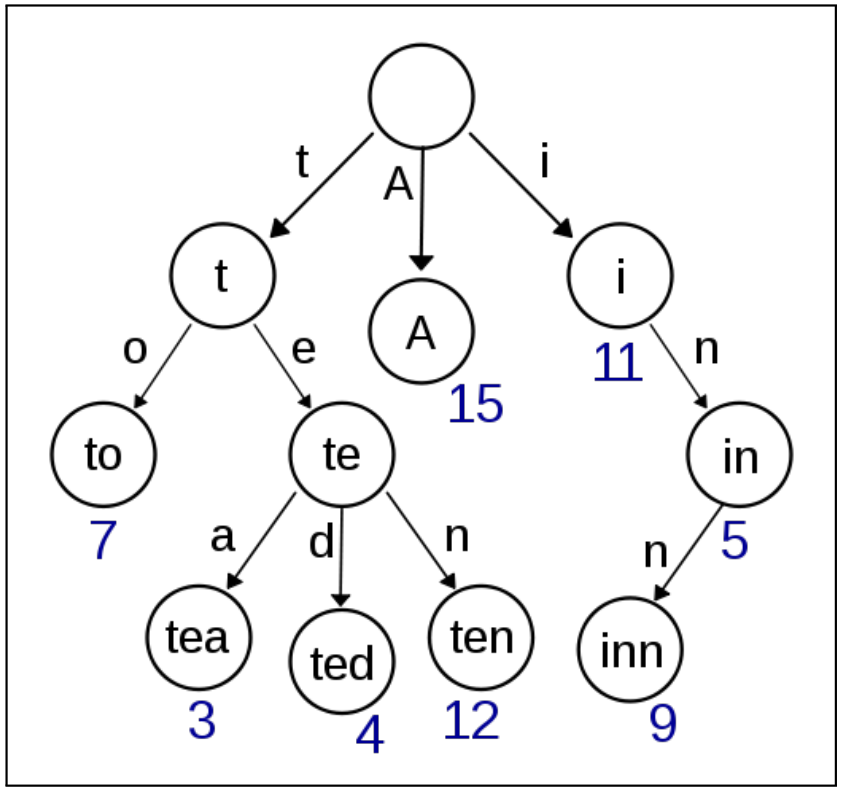
Fast Algorithms for Gap-Weighted Spectrum Kernels

- The dynamic programming algorithm by [Lodhi et al., 2002] computes the gap-weighted spectrum kernels in time $O(k|s||t|)$ where k is the number of non-gap characters in the subsequences
  - Idea intuitively: tabulate the sum of match weights for all prefixes of the two sequences for $l = 1, \ldots, k-1$ non-gap characters. This requires filling $k-1$ times a $O(|s||t|)$dynamic programming table
- We will take a look at an algorithm based on trie-data structure that works in time proportional to the number of matching subsequences [Rousu and Shawe-Taylor, 2005]
  - Works better than the dynamic programming method when compared sequences are long and total number of gaps is limited

Trie-based computation of gap-weighted spectrum kernels

[Rousu and Shawe-Taylor, 2005]

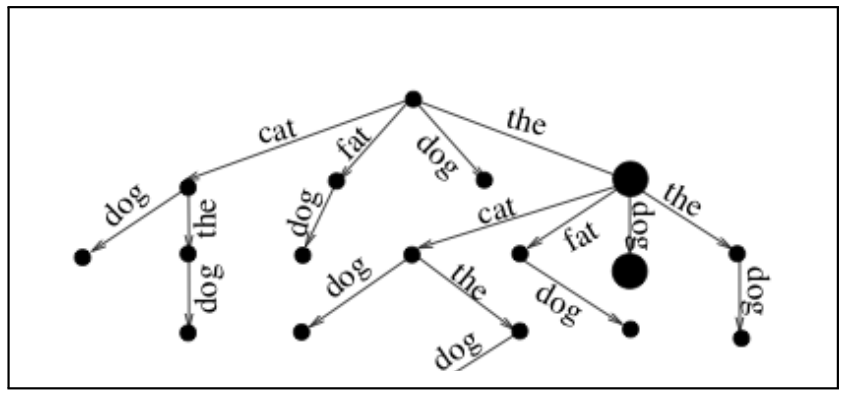
- Trie: tree shaped data structure for a (set of) string(s).
- Root corresponds to empty string,
- Internal nodes correspond to subsequences .
- Children of a node corresponds of extensions of the substring with one character
- Nodes of the trie may contain links to occurences, or counts, or both



Trie-based computation of gap-weighted spectrum kernels

[Rousu and Shawe-Taylor, 2005]

- One can also build the tree for words or motifs: lets us compare chains of words/motifs (we call these phrases)
- Key idea for kernel $k(s, t)$: build the tree so that it only contains the phrases contained in both sequences s and t

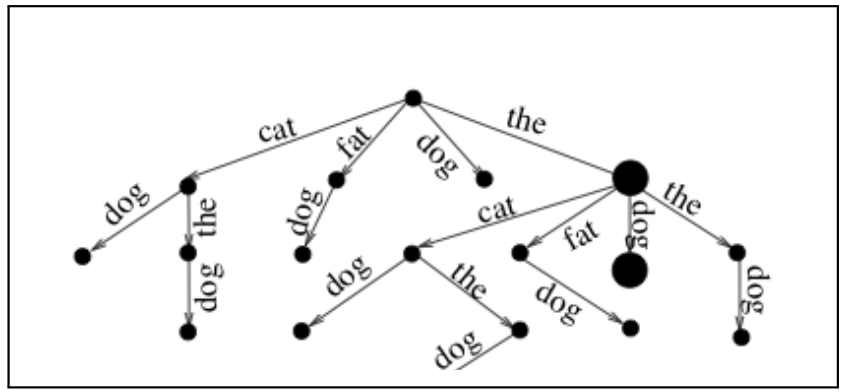


Trie built for the pair $s =$ 'The cat was chased by the fat dog' and $t =$ 'The fat cat bit the dog'

Trie-based computation of gap-weighted spectrum kernels

[Rousu and Shawe-Taylor, 2005]

- We build the tree by scanning the two strings from each position
- Store the indices of the matches to the nodes of the trie (one set for matches of subsequence u in s, another set for t)
- When extending a match, scan forward from each location, adding gaps



g	A_s('the dog',g)
0	
1	{8}
2	
3	
4	
5	
6	{8}

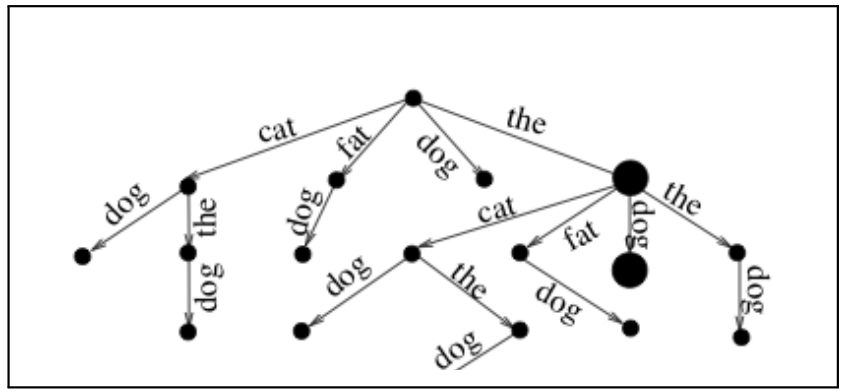
g	A_t('the dog',g)
0	{6}
1	
2	
3	
4	{6}
5	
6	

Trie built for the pair s = 'The cat was chased by the fat dog' and t = 'The fat cat bit the dog'

Trie-based computation of gap-weighted spectrum kernels

[Rousu and Shawe-Taylor, 2005]

- While a match at certain index can be extended, it remains *alive*
- When at the required dept of the trie, compute the required gap weighting
- Multiply the counts stored in the match sets with the gap weights, and the sum over all pairs of matches in the two strings



g	A_s('the dog',g)
0	
1	{8}
2	
3	
4	
5	
6	{8}

g	A_t('the dog',g)
0	{6}
1	
2	
3	
4	{6}
5	
6	

Trie built for the pair s = 'The cat was chased by the fat dog' and t = 'The fat cat bit the dog'

Efficiency of trie-based computation [Rousu and Shawe-Taylor, 2005]

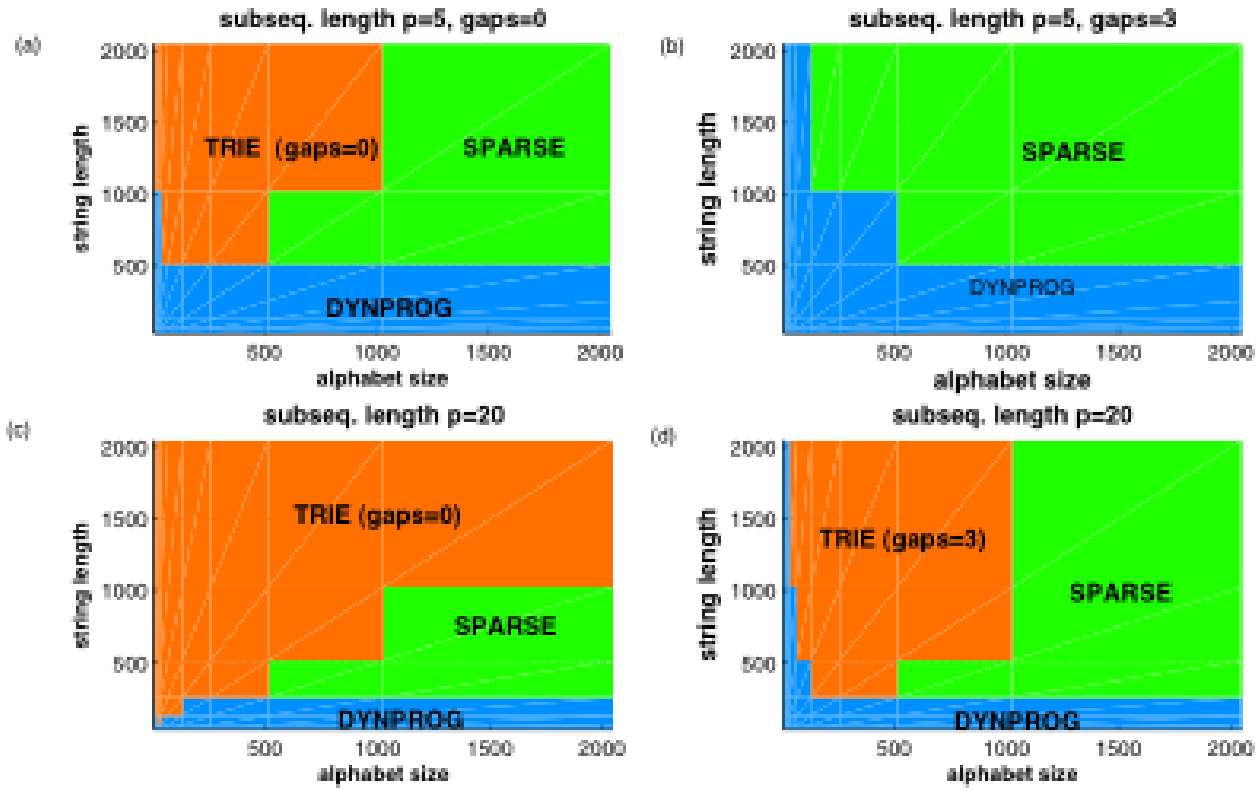


Figure 9: The fastest algorithm as a function of string length and alphabet size, with different subsequence lengths (p) and upper bounds for the number of gaps in the TRIE algorithm.

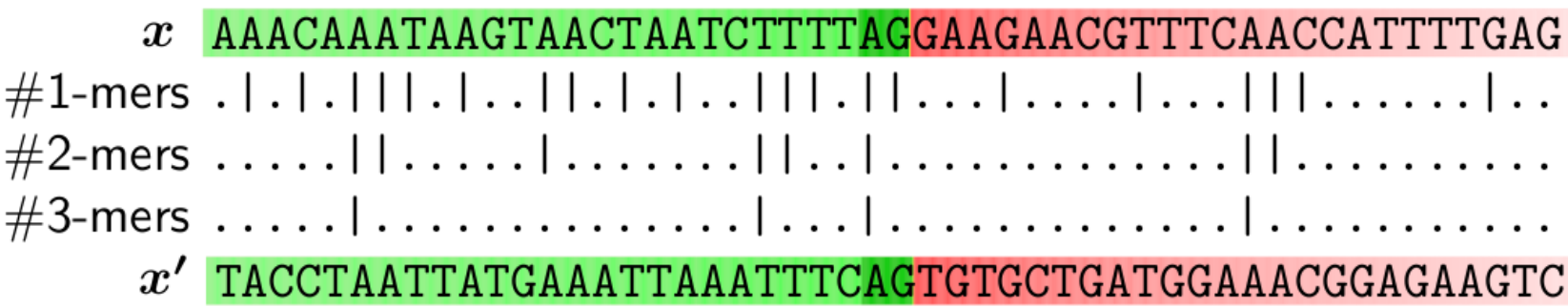
Weighted Degree Kernel = Spectrum kernels for each position

To make use of position-dependent motifs:

$$k(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{d} \beta_k \sum_{l=1}^{L-k} \mathbf{I}(\mathbf{u}_{k,l}(\mathbf{x}) = \mathbf{u}_{k,l}(\mathbf{x}'))$$

- $L :=$ length of the sequence $\mathbf{x}$
- $d :=$ maximal "match length" taken into account
- $\mathbf{u}_{k,l}(\mathbf{x}) :=$ subsequence of length k at position l of sequence $\mathbf{x}$

Example degree $d = 3$:



$$k(\mathbf{x}, \mathbf{x}') = \beta_1 \cdot 21 + \beta_2 \cdot 8 + \beta_3 \cdot 4$$

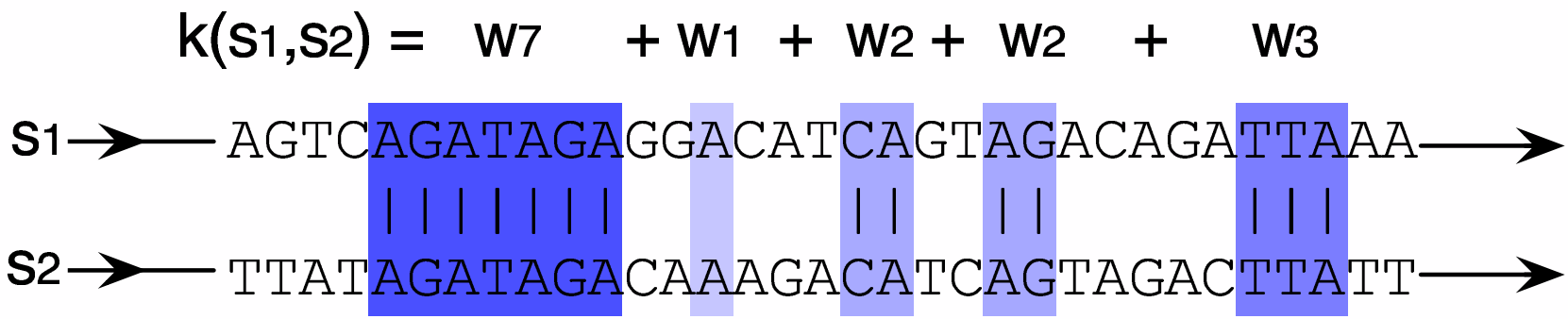
Difference to Spectrum kernel:

- Mixture of Spectrum kernels (up to degree d)
- Each position is considered independently

Weighted Degree Kernel

- As weighting we use $\beta_k = 2\frac{d-k+1}{d(d+1)}$:
  - Longer matches are weighted less, but they imply many shorter matches
- Computational effort is $O(L \cdot d)$

Speed-up Idea: Reduce effort to $O(L)$ by finding matching "blocks" (computational effort $\mathcal{O}(L)$)



GC-Content-based Splice Site Recognition

Recall the previous results:

Kernel	auROC
Linear	88.2%
Polynomial $d = 3$	91.4%
Polynomial $d = 7$	90.4%
Gaussian $\sigma = 100$	87.9%
Gaussian $\sigma = 1$	88.6%
Gaussian $\sigma = 0.01$	77.3%

SVM accuracy of acceptor site recognition using polynomial and Gaussian kernels with different degrees d and widths σ. Accuracy is measured using the area under the ROC curve (auROC) and is computed using five-fold cross-validation

Sequence-based Splice Site Recognition

Kernel	auROC
Spectrum $\ell = 1$	94.0%
Spectrum $\ell = 3$	96.4%
Spectrum $\ell = 5$	94.5%
Mixed spectrum $\ell = 1$	94.0%
Mixed spectrum $\ell = 3$	96.9%
Mixed spectrum $\ell = 5$	97.2%
WD $\ell = 1$	98.2%
WD $\ell = 3$	98.7%
WD $\ell = 5$	98.9%

The area under the ROC curve (auROC) of SVMs with the spectrum, mixed spectrum, and weighted degree kernels for the acceptor splice site recognition task for different substring lengths ℓ.

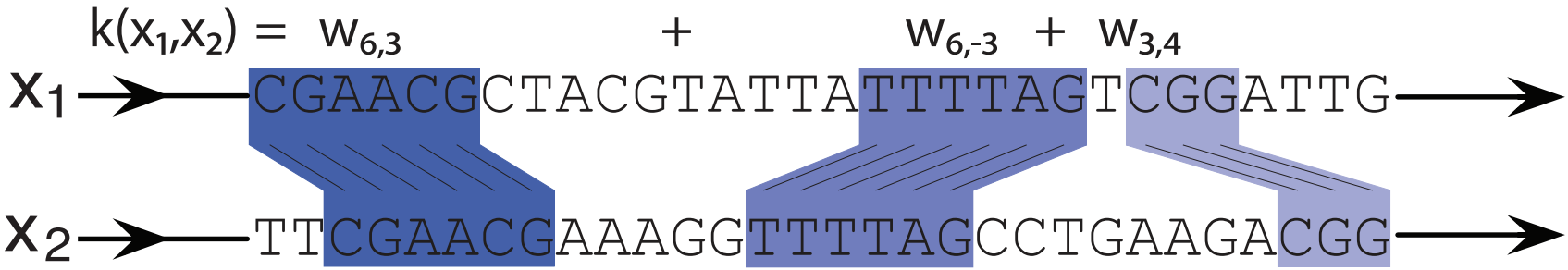
Weighted Degree Kernel with *Shifts*

To make use of partially position-dependent motifs:

- If sequence is slightly mutated (e.g. indels), WD kernel fails
- Extension: Allow some positional variance (shifts $S(l)$)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{K} \beta_k \sum_{l=1}^{L-k+1} \gamma_l \sum_{\substack{s=0 \\ s+l \leq L}}^{S(l)} \delta_s \, \mu_{k,l,s,\mathbf{x}_i,\mathbf{x}_j},$$

$$\mu_{k,l,s,\mathbf{x}_i,\mathbf{x}_j} = \mathbf{I}(\mathbf{u}_{k,l+s}(\mathbf{x}_i) = \mathbf{u}_{k,l}(\mathbf{x}_j)) + \mathbf{I}(\mathbf{u}_{k,l}(\mathbf{x}_i) = \mathbf{u}_{k,l+s}(\mathbf{x}_j)),$$



[Rätsch et al., 2005]

Oligo Kernel

Oligo kernel

$$k(\mathbf{x}, \mathbf{x}') = \sqrt{\pi}\sigma \sum_{\mathbf{u}\in\Sigma^k} \sum_{p\in S_{\mathbf{u}}^{\mathbf{x}}} \sum_{q\in S_{\mathbf{u}}^{\mathbf{x}'}} e^{-\frac{1}{4\sigma^2}(p-q)^2},$$

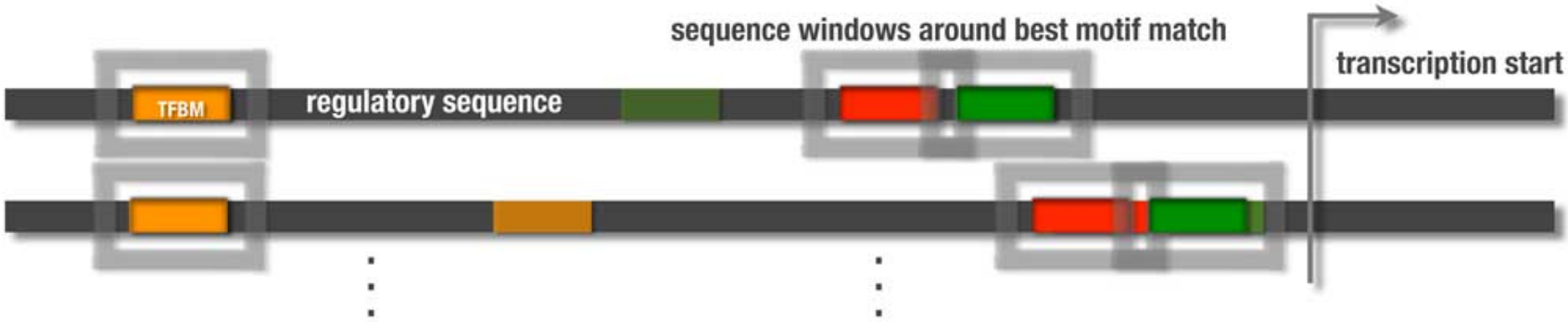
where

- $0 \leq \sigma$ is a smoothing parameter
- $\mathbf{u}$ is a k-mer and
- $S_{\mathbf{u}}^{\mathbf{x}}$ is the set of positions within sequence $\mathbf{x}$ at which $\mathbf{u}$ occurs as a substring

Similar to WD kernel with shifts.

[Meinicke et al., 2004]

Regulatory Modules Kernel [Schultheiss et al., 2008]



- Search for overrepresented motifs $m_1, \ldots, m_M$ (colored bars)
- Find best match of motif m_i in example x_j; extract windows $s_{i,j}$ at position $p_{i,j}$ around matches (boxed)
- Use a string kernel, e.g. k_{WDS}, on all extracted sequence windows, and define a combined kernel for the sequences:

$$k_{seq}(\mathbf{x}_j, \mathbf{x}_k) = \sum_{i=1}^{M} k_{WDS}(s_{i,j}, s_{i,k})$$

- Use a second kernel k_{pos}, e.g. based on RBF kernel, on vector of pairwise distances between the motif matches:

$$\mathbf{f}_j = (p_{1,j} - p_{2,j}, p_{1,j} - p_{3,j}, \ldots, p_{M-1,j} - p_{M,j})$$

- Regulatory Modules kernel: $k_{RM}(\mathbf{x}, \mathbf{x}') := k_{seq}(\mathbf{x}, \mathbf{x}') + k_{pos}(\mathbf{x}, \mathbf{x}')$

Local Alignment Kernel

In order to compute the score of an alignment, one needs:

- substitution matrix $S \in \mathbb{R}^{\Sigma \times \Sigma}$
- gap penalty $g : \mathbb{N} \to \mathbb{R}$

An alignment π is then scored as follows:

```
CGGSLIAMM----WFGV
|...|||||....||||
C---LIVMMNRLMWFGV
```

$$s_{S,g}(\pi) = S(C,C) + S(L,L) + S(I,I) + S(A,V) + 2S(M,M)$$
$$+S(W,W) + S(F,F) + S(G,G) + S(V,V) - g(3) - g(4)$$

Smith-Waterman score (not positive definite)

$$SW_{S,g}(\mathbf{x},\mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x},\mathbf{y})} s_{S,g}(\pi)$$

Local Alignment Kernel

Local Alignment kernel [Vert et al., 2004]

$$K^{\beta}(\mathbf{x}, \mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{x},\mathbf{y})} \exp(\beta s_{S,g}(\pi))$$

- This kernel is positive semi-definite for certain values of $\beta > 0$
- Dynamic programming algorithm exist to compute the kernel

References I

C. Leslie, E. Eskin, J. Weston, and W.S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4), 2003.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

P. Meinicke, M. Tech, B. Morgenstern, and R. Merkl. Oligo kernels for data mining on biological sequences: a case study on prokaryotic translation initiation sites. *BMC Bioinformatics*, 5(169), 2004.

G. Rätsch and S. Sonnenburg. Accurate splice site detection for *Caenorhabditis elegans*. In *Kernel Methods in Computational Biology*. MIT Press, 2004.

G. Rätsch, S. Sonnenburg, and B. Schölkopf. RASE: recognition of alternatively spliced exons in *C. elegans*. *Bioinformatics*, 21(Suppl. 1):i369–i377, June 2005.

J. Rousu and J. Shawe-Taylor. Efficient computation of gapped substring kernels on large alphabets. *Journal of Machine Learning Research*, 6(2):1323, 2005.

References II

S.J. Schultheiss, W. Busch, J.U. Lohmann, O. Kohlbacher, and G. Rätsch. Kirmes: Kernel-based identification of regulatory modules in euchromatic sequences. In *German Conference on Bioinformatics*, Lecture notes in Informatics, pages 158–167, Heidelberg, 2008. GI, Springer Verlag. URL `http://www.fml.tuebingen.mpg.de/raetsch/projects/kirmes`.

S. Sonnenburg, G. Rätsch, and K. Rieck. Large-scale learning with string kernels. In *Large-Scale Kernel Machines*. MIT Press, 2007.

C.H. Teo and SVN Vishwanathan. Fast and space efficient string kernels using suffix arrays. In *Proceedings of the 23rd international conference on Machine learning*, pages 929–936. ACM, 2006.

J.-P. Vert, H. Saigo, and T. Akutsu. Local alignment kernels for biological sequences. In *Kernel Methods in Computational Biology*. MIT Press, 2004.