Assignment no. 4: Supplementary Notes

Spectrum Kernel

As discussed in class, the *spectrum kernel* is defined as

$$k(x,y) = \sum_{m \in \mathscr{M}} N(m,x) \cdot N(m,y), \tag{1}$$

where $\mathscr{M}$ is the set of all possible subsequences of length K and $N(m,x)$ denotes the number of occurrences/matches of sub-sequence m in string x. As a simple example, consider the two sequences $x = \texttt{AAATTTA}$ and $y = \texttt{TTTATT}$, obviously two sequences from the alphabet $\mathscr{A} = \{\texttt{A},\texttt{T}\}$. For $K = 2$, there are four possible subsequences of length $K = 2$, i.e. `AA`, `AT`, `TA`, and `TT`. The numbers of occurrences $N(m,x)$ are then given as follows:

$m \rightarrow$	`AA`	`AT`	`TA`	`TT`
$N(m,x)$	2	1	1	2
$N(m,y)$	0	1	1	3

Then the kernel value (1) is then given as the scalar product of the two rows of the above matrix:

$$k(x,y) = 2 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 + 2 \cdot 3 = 0 + 1 + 1 + 6 = 8$$

Implementation tricks: The table above is nothing else but computing an explicit feature representations $\Phi(x)$ / $\Phi(y)$ and computing the scalar product in the feature space. This is fine for such a simple example, but infeasible for large features spaces. As an example, if we use amino acid sequences (which corresponds to a 20-letter alphabet, i.e. letters `A`–`Z` except `B`, `J`, `O`, `U`, `X`, and `Z`) and $K = 5$, we arrive at a feature space of $20^5 = 3,200,000$ different patterns. At the same time, every sequence of length L can only contain up to $L - K + 1$ different features (possibly less, if patterns occur multiple times). One option would be to use a sparse matrix format, but this is too complicated to implement for such a simple exercise. Instead, you should rather employ one of the following strategies:

1. Create hashed lists of all patterns contained in each sequence and then, for a given pair of sequences, iterate over all patterns occurring in both sequences to compute the above scalar product.

2. The brute force approach would be to have a double loop for each pair of sequences that checks all pairs of subsequences of the two sequences. This is clearly less efficient.

Normalization: The spectrum kernel has the disadvantage that its range is influenced by the length of sequences, which is particularly unpleasant if the sequences in the data set are not of equal lengths. The standard way of dealing with this is *kernel normalization*. This is done by computing the kernel as described above and then to divide each value by the square root of the product of the two self-similarities. More specifically, the normalized kernel $k'(x,y)$ is given as

$$k'(x,y) = \frac{k(x,y)}{\sqrt{k(x,x)\cdot k(y,y)}},$$

where $k(x,y)$ is the original kernel before normalization.

One-hot Encoding of Sequences

One-hot encoding is a standard tool for converting categorical data with more than two classes to a vectorial representation. The idea is quite simple: if there are M categories, use M binary columns, where each column stands for one category. If a sample belongs to the i-th category, a 1 is placed in the i-th column, where the other columns remain 0.

Since every biological sequence is a sequence of symbols, one-hot encoding can be applied — at least as long all sequences have the same length. Let us consider a simple example of aligned DNA sequences (the alphabet is $\mathscr{A} = \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}$):

	pos. 1				pos. 2				pos. 3				pos. 4			
	A	C	G	T	A	C	G	T	A	C	G	T	A	C	G	T
AATT	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1
GGCG	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0
AGCT	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1

In Exercise 6, sequences are amino acid sequences (alphabet of 20 letters) of length 15, so the total number of binary features is $20 \cdot 15 = 300$.