Learning interpretable SVMs for biological sequence classification

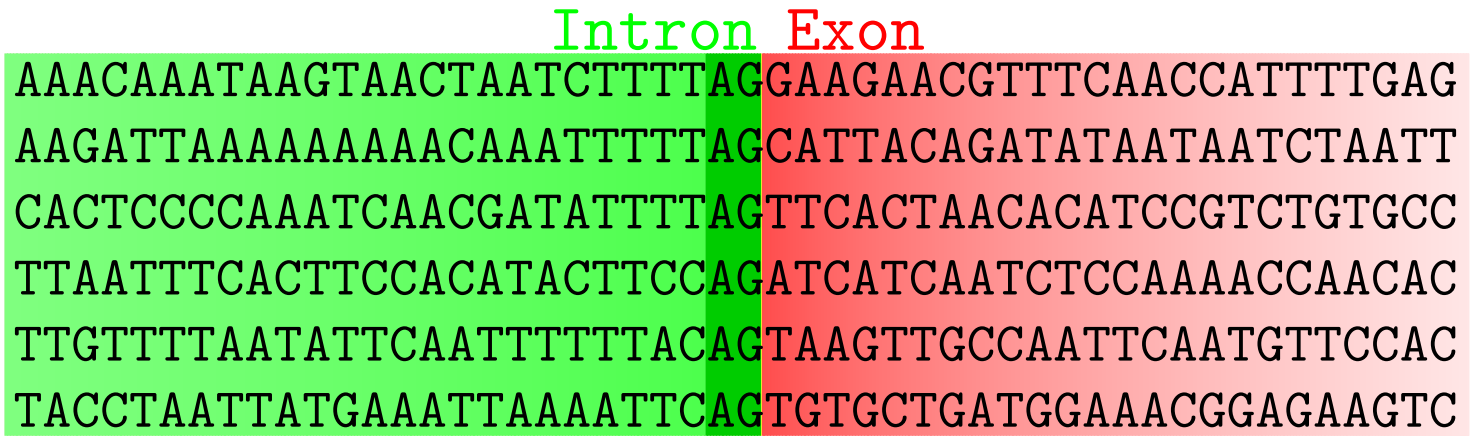
Sören Sonnenburg[†], Gunnar Rätsch[†♮], Christin Schäfer[†]

† Fraunhofer FIRST.IDA, Kekuléstr. 7, 12489 Berlin, Germany
♮ Friedrich Miescher Lab. of the Max Planck Society,
Spemannstr. 39, 72076 Tübingen, Germany
{Soeren.Sonnenburg, Christin.Schaefer}@first.fraunhofer.de,
Gunnar.Raetsch@tuebingen.mpg.de

Roadmap:

- The Motivating Application
- Multiple Kernel Learning (MKL)
- Derivation of the MKL Optimization Problem
- Algorithms
- Significance Analysis
- Results
- Outlook and Conclusion

Biology: Detection of Splice Sites



- aligned sequences of fixed length (AG always at position 61 & 62)
- Task: distinguish splice sites from fake - splice sites

$\Rightarrow$ 2-class classifcation problem

Approach: String Kernel + SVM

- use SVM classifier $$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i \mathrm{k}(\boldsymbol{x}, \boldsymbol{x}_i) + b\right)$$

- key ingredient is kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}') \Rightarrow$ gives means to compare 2 sequences

- Kernel here "Weighted Degree Kernel" β_k $(k = 1, \ldots, d)$:

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

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
.|.|.|||.|..||.|.|..|||.||...|....|...|||......|..
TACCTAATTATGAAATTAAAATTCAGTGTGCTGATGGAAACGGAGAAGTC
```

- L length of the $\boldsymbol{x}$'s, d maximal "match length" and $\boldsymbol{u}_{k,l}(\boldsymbol{x})$ subsequence of length k at position l of sequence $\boldsymbol{x}$

Success

Choosing a particular weighting

$$\beta_k = \frac{2(d-k+1)}{(d(d+1))}$$

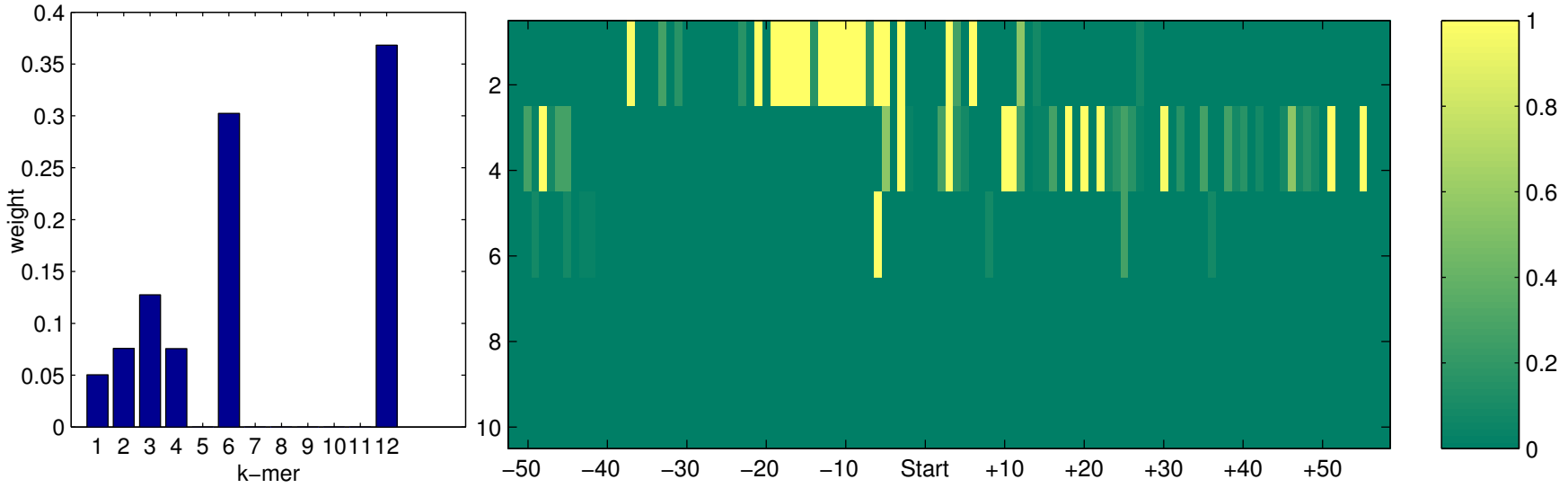
seems to solve the task: on 500,000 training examples test AUC $99,80\%$ (test error 0.78%)

Open questions

- Why does that weighting make sense ?
- Is there a better weighting ?
  - in terms of sense
  - classification performance
- Can we learn that weighting ?
- What does that have to do with optimization ?

GAIN

- automated model selection
- interpretability
  - which $k-$mers are of higher importance
  - even which $k-$mers where in the sequence are of higher importance



Reformulation: Multiple Kernel Learning

The Weighted Degree kernel is a linear combination of kernels !

$$\begin{aligned} \mathrm{k}(\boldsymbol{x}, \boldsymbol{x}') &= \sum_{k=1}^{d} \beta_k \sum_{l=1}^{L-k} \mathrm{I}(\boldsymbol{u}_{k,l}(\boldsymbol{x}) = \boldsymbol{u}_{k,l}(\boldsymbol{x}')) \\ &= \sum_{k=1}^{d} \beta_k \mathrm{k}_k(\boldsymbol{x}, \boldsymbol{x}') \end{aligned}$$

with

$$\mathrm{k}_k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{l=1}^{L-k} \mathrm{I}(\boldsymbol{u}_{k,l}(\boldsymbol{x}) = \boldsymbol{u}_{k,l}(\boldsymbol{x}')).$$

(This also holds for other String Kernels like the Spectrum Kernel)

$\Rightarrow$ need to solve the so called **Multiple Kernel Learning Problem**, i.e. determine $(\boldsymbol{\beta}, \boldsymbol{\alpha}, b)$ simultaneously.

FIRST

Motivation

- Kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}')$ used in standard SVM Classifier

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{\ell} y_i \alpha_i \mathrm{k}(\boldsymbol{x}, \boldsymbol{x}_i) + b\right)$$

- Now: linear combination of kernels (again a kernel)

$$\mathrm{k}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{M} \beta_j \, \mathrm{k}_j(\boldsymbol{x}, \boldsymbol{x}'), \;\; \beta_j \geq 0$$

- **useful:** Polynomial kernels of different degree, kernels on different domain

- **but:** How to learn and constrain weights β_j ?

Constraining the weights

L_2-vs. L_1-Norm

- in max problem weights certain β_j would grow infinitely (min shrink to zero) $\Rightarrow$ **constraining β_j necessary**
- Dense $\|\boldsymbol{\beta}\|_2 = 1$ (Lin and Zhang 2004)
- vs. Sparse $\|\boldsymbol{\beta}\|_1 = 1$ (Bach, Lanckriet and Jordan 2004)
  - convex combination of kernels
  - sparse solution in terms of kernels
  - allows for interpretation of result

constraints on β_j:

$$\sum_{j=1}^{N} \beta_j = 1,\ \beta_j \geq 0$$

Standard SVM Optimization Problem

This we all know

SVM Primal formulation:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{N}\xi_n \\ \text{w.r.t.} \quad & \boldsymbol{w} \in \mathbb{R}^k, \boldsymbol{\xi} \in \mathbb{R}_+^N, b \in \mathbb{R} \\ \text{s.t.} \quad & y_i\left(\boldsymbol{w}^\top \Phi(\boldsymbol{x}_i) + b\right) \geq 1 - \xi_i, \forall i = 1, \ldots, N \end{aligned}$$

MKL Optimization Problem I

MKL Primal formulation:

$$\begin{aligned}
\min \quad & \frac{1}{2}\left(\sum_{j=1}^{M}\beta_j \left\|\boldsymbol{w}_j\right\|_2\right)^2 + C\sum_{i=1}^{N}\xi_n \\
\text{w.r.t.} \quad & \boldsymbol{w}=(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_M), \boldsymbol{w}_j\in\mathbb{R}^{k_j}, \boldsymbol{\xi}\in\mathbb{R}_+^N, \boldsymbol{\beta}\in\mathbb{R}_+^M, b\in\mathbb{R} \\
\text{s.t.} \quad & y_i\left(\sum_{j=1}^{M}\beta_j {\boldsymbol{w}_j}^\top \Phi_j(\boldsymbol{x}_i)+b\right)\geq 1-\xi_i, \forall i=1,\ldots,N \\
& \sum_{j=1}^{M}\beta_j = 1
\end{aligned}$$

Properties: equivalent to SVM for $M = 1$; solution sparse in "blocks"; each block j corresponds to one kernel

MKL Optimization Problem II

Dual Formulation (Bach, Lanckriet, Jordan 2004):

$$\begin{aligned}
\min \quad & \frac{1}{2}\gamma^2 - \sum_{i=1}^{N} \alpha_i \\
\text{w.r.t.} \quad & \gamma \in \mathbb{R}, \boldsymbol{\alpha} \in \mathbb{R}^N \\
\text{s.t.} \quad & 0 \leq \boldsymbol{\alpha} \leq C, \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& \underbrace{\sum_{r=1}^{N}\sum_{s=1}^{N} \alpha_r \alpha_s y_r y_s K_j(\boldsymbol{x}_r, \boldsymbol{x}_s)}_{=:S_j(\boldsymbol{\alpha})} - \gamma^2 \leq 0, \ \forall j = 1, \ldots, M
\end{aligned}$$

"partial Lagrangian:"

$$L := \frac{1}{2}\gamma^2 - \sum_{i=1}^{N} \alpha_i + \sum_{j=1}^{M} \beta_j (S_j(\boldsymbol{\alpha}) - \gamma^2)$$

MKL Optimization Problem II

Reformulation as Semi-Infinite Linear Program:

$$\max_{\boldsymbol{\beta}} \min_{\boldsymbol{\alpha}} \sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right)$$

$$\text{s.t. } 0 \leq \boldsymbol{\alpha} \leq C, \sum_{i=1}^{N} \alpha_i y_i = 0, \sum_{j=1}^{M} \beta_j = 1$$

$$\begin{array}{ll} \max & \theta \\ \text{w.r.t.} & \theta \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}_+^M \text{ with } \sum_{j=1}^{M} \beta_j = 1 \\ \text{s.t.} & \sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right) \geq \theta \\ & \text{for all } \boldsymbol{\alpha} \text{ with } 0 \leq \boldsymbol{\alpha} \leq C \text{ and } \sum_{i=1}^{N} y_i \alpha_i = 0 \end{array}$$

$\Rightarrow$ **Linear, but infinitely many constraints**

The Semi-Infinite Linear Program I

$$
\begin{aligned}
\max \quad & \theta \\
\text{w.r.t.} \quad & \theta \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}_+^M \text{ with } \sum_{j=1}^{M} \beta_j = 1 \\
\text{s.t.} \quad & \sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right) \geq \theta \\
& \text{for all } \boldsymbol{\alpha} \text{ with } 0 \leq \boldsymbol{\alpha} \leq C \text{ and } \sum_{i=1}^{N} y_i \alpha_i = 0
\end{aligned}
$$

Properties:

- optimize a convex combination
- infinitely many constraints
- quite easy to identify violated constraints

The Semi-Infinite Linear Program II

$$
\begin{aligned}
\max \quad & \theta \\
\text{w.r.t.} \quad & \theta \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}_+^M \text{ with } \sum_{j=1}^{M} \beta_j = 1 \\
\text{s.t.} \quad & \sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right) \geq \theta \\
& \text{for all } \boldsymbol{\alpha} \text{ with } 0 \leq \boldsymbol{\alpha} \leq C \text{ and } \sum_{i=1}^{N} y_i \alpha_i = 0
\end{aligned}
$$

Solving the SILP:

- Column Generation
  - fast, but no convergence rate
- Use Boosting like techniques: Arc-GV or AdaBoost*
  - known convergence rate $\mathcal{O}(\log(M)/\varepsilon^2)$
- SMO like algorithm
  - consider suboptimal SVM solutions: empirically 3-5 times faster

Solving the SILP: Column Generation I

$$
\begin{aligned}
\max \quad & \theta \\
\text{w.r.t.} \quad & \theta \in \mathbb{R}, \boldsymbol{\beta} \in \mathbb{R}_+^M \text{ with } \sum_{j=1}^{M} \beta_j = 1 \\
\text{s.t.} \quad & \sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right) \geq \theta \\
& \text{for all } \boldsymbol{\alpha} \text{ with } 0 \leq \boldsymbol{\alpha} \leq C \text{ and } \sum_{i=1}^{N} y_i \alpha_i = 0
\end{aligned}
$$

- solved by taking set of most violated constraints into account
- most violated constraints given by SVM solution for fixed $\boldsymbol{\beta}$

$$
\sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right) = \frac{1}{2} \sum_{r=1}^{N} \sum_{s=1}^{N} \alpha_r \alpha_s y_r y_s \sum_{j=1}^{M} \beta_j k_j(\boldsymbol{x}_r, \boldsymbol{x}_s) - \sum_{i=1}^{N} \alpha_i,
$$

- iteratively find most violated constraints, solve linear program with current constraints, ..., till convergence to the global optimum

Solving the SILP: Column Generation II

$D^0 = 1$, $\theta^1 = 0$, $\beta_k^1 = \frac{1}{M}$ for $k = 1, \ldots, M$
for $t = 1, 2, \ldots$ **do**
  obtain SVM's $\boldsymbol{\alpha}^k$ with kernel

$$\mathrm{k}^t(\boldsymbol{x}_i, \boldsymbol{x}_j) := \sum_{k=1}^{M} \beta_k^t \, \mathrm{k}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

  for $k = 1, \ldots, M$ **do**
    $D_k^t = \frac{1}{2} \sum_{r,s} \alpha_r^t \alpha_s^t y_r y_s \, \mathrm{k}_k(\boldsymbol{x}_r, \boldsymbol{x}_s) - \sum_r \alpha_r^t$
  end for
  $D^t = \sum_{k=1}^{M} \beta_k^t D_k^t$
  $(\boldsymbol{\beta}^{t+1}, \theta^{t+1}) = \operatorname{argmax} \ \theta$
    w.r.t. $\boldsymbol{\beta} \in \mathbb{R}_+^M, \theta \in \mathbb{R}$ with $\sum_k \beta_k = 1$
    s.t. $\sum_{k=1}^{M} \beta_k D_k^r \geq \theta$ for $r = 1, \ldots, t$
  if $|1 - \frac{\theta^{t+1}}{D^t}| \leq \epsilon$ **then break**
end for

Solving the SILP: Boosting I

Boosting

Primal

$$\begin{aligned} \max_{\boldsymbol{\alpha},\rho} \quad & \rho \\ \text{subject to} \quad & y_n \sum_{j=1}^{J} \alpha_j h_j(\boldsymbol{x}_n) \geq \rho \\ & \alpha_j \geq 0 \\ & \textstyle\sum_j \alpha_j = 1 \end{aligned}$$

Dual

$$\begin{aligned} \min_{\mathbf{d},\delta} \quad & \delta \\ \text{subject to} \quad & \sum_{n=1}^{N} d_n y_n h_j(\boldsymbol{x}_n) \leq \delta \\ & d_n \geq 0 \\ & \textstyle\sum_n d_n = 1 \end{aligned}$$

- max w.r.t. $-\delta$; d_n corresponds to β_n; h_j corresponds to expression using a certain $\boldsymbol{\alpha}$
- Arc-GV and AdaBoost*: ϱ and δ are optimized

The number of hypotheses can be very large or infinite!

Solving the SILP: Boosting I

An Arc-GV like Algorithm

$D^0 = 1$, $\rho^1 = \tau_k^1 = 0$, $\beta_k^1 = \frac{1}{M}$ for $k = 1, \ldots, M$

for $t = 1, 2, \ldots$ **do**

obtain SVM's $\boldsymbol{\alpha}^k$ with kernel

$$\mathrm{k}^t(\boldsymbol{x}_i, \boldsymbol{x}_j) := \sum_{k=1}^{M} \beta_k^t \, \mathrm{k}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

for $k = 1, \ldots, M$ **do**

$D_k^t = \frac{1}{2} \sum_{r,s} \alpha_r^t \alpha_s^t y_r y_s \, \mathrm{k}_k(\boldsymbol{x}_r, \boldsymbol{x}_s) - \sum_r \alpha_r^t$

end for

$D^t = \sum_{k=1}^{M} \beta_k^t D_k^t$

$\gamma_t = \mathrm{argmin}_{\gamma \in [0,1]} \sum_{k=1}^{M} \beta_k^t \exp\left\{\gamma(D_k - \rho^t)\right\}$

for $k = 1, \ldots, M$ **do**

$\tau_k^{t+1} = \tau_k^t + \gamma_t D_k^t$

$\beta_k^{t+1} = \beta_k^t \exp(\gamma_t D_k^t) / \left(\sum_{k'} \beta_{k'}^t \exp(\gamma_t D_{k'}^t)\right)$

end for

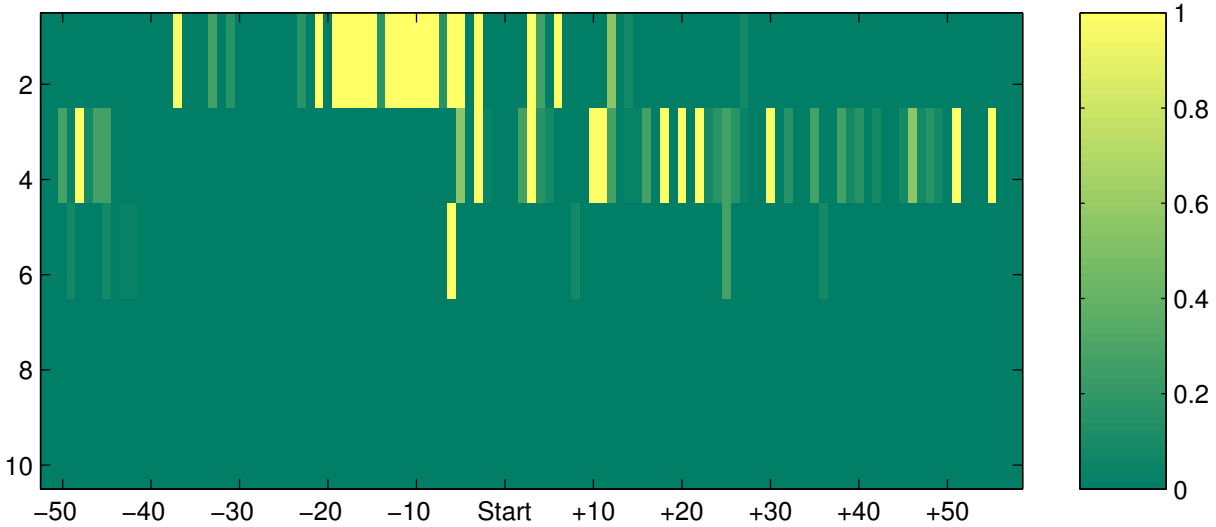
$\rho^{t+1} = \max_k \tau_k^{t+1} / \sum_{r=1}^{t} \gamma_r$

if $|1 - \frac{\rho^{t+1}}{D^t}| \leq \epsilon$ **then break**

end for

Stability of the solution ?

We obtain a weighting

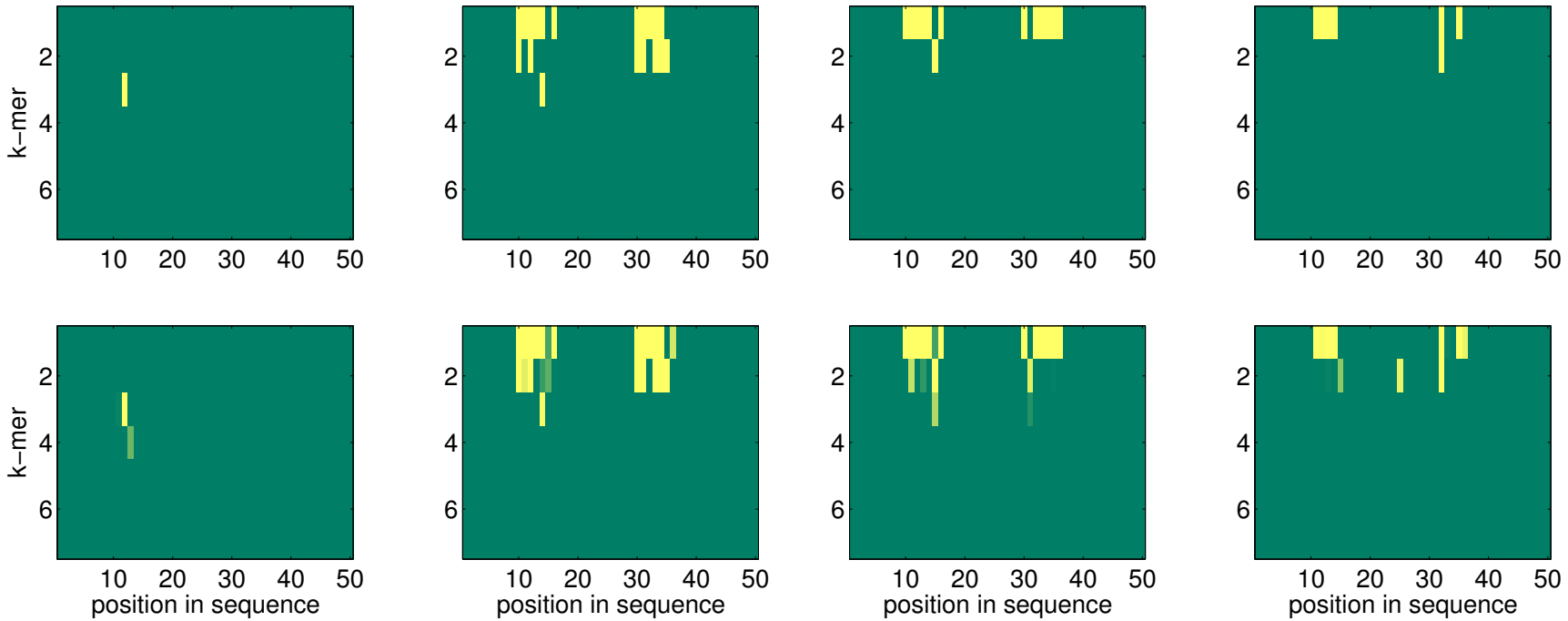


Questions:

- Can we use that weighting to interpret the SVM solution ?
  - Stability of the weighting β ?
  - Which weights are significant ?

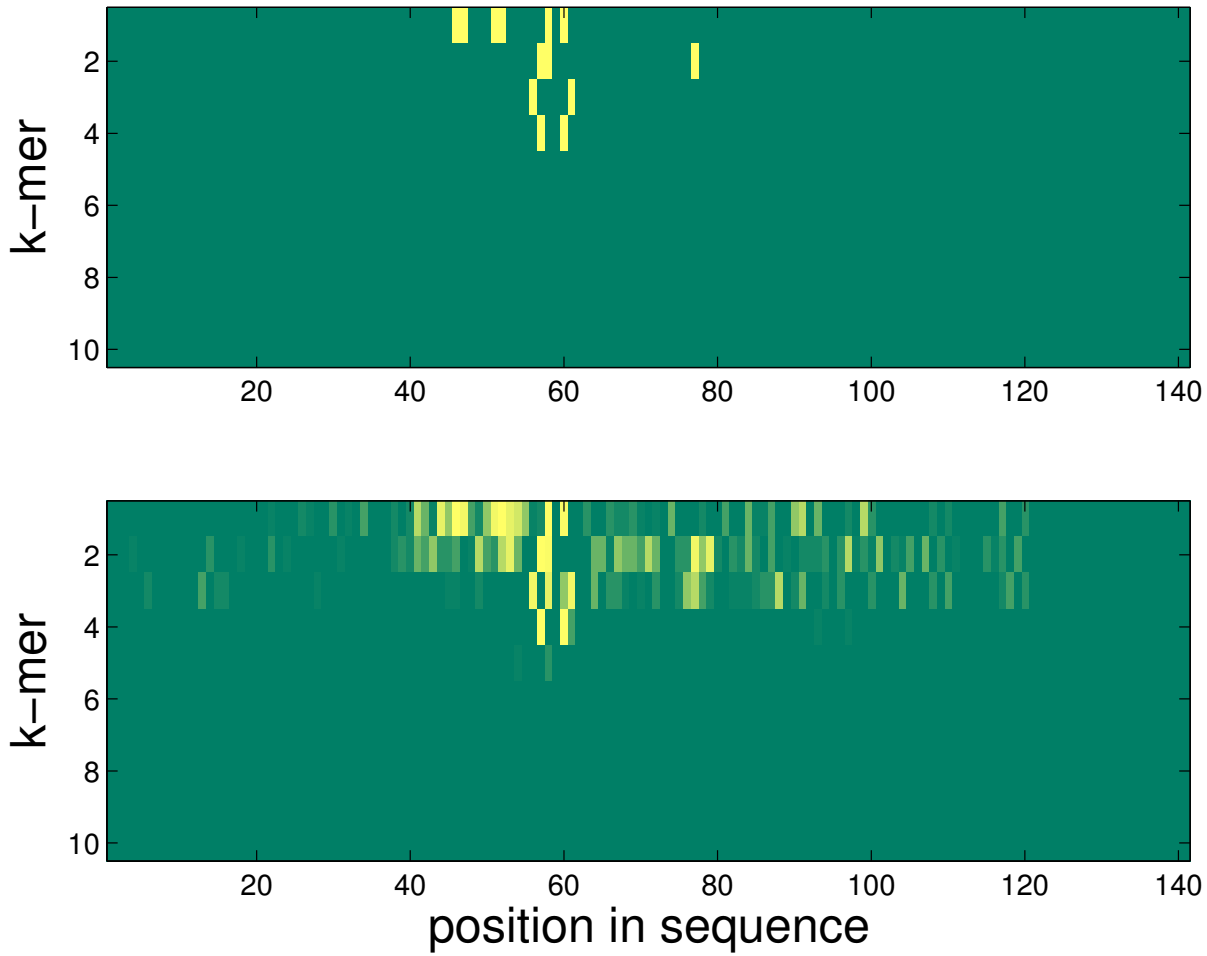
We used a statistical test to investigate significance of weights

Toy Dataset



- DNA sequences of length 50 with hidden motifs at 10-16 and 30-36
- 8×50 string kernels with max. word- length 8
- compute significance level by bootstrapping
- columns $\equiv$ noise level
- subplot columns weights used at certain position, rows oligomer length

Application to Acceptor splice sites



- *C.elegans* splice dataset (AG at position 61 & 62)
- subplot columns weights used at certain position, rows oligomer length

Conclusion

Conclusion:

- suitable for large scale (WD kernel of degree 20 trained on 200000 examples)
- fast, because $t(N, M, \epsilon) \approx 5 \cdot 10^{-11} M^{2.22} N^{1.68} \log_2(1/\epsilon)^{2.52}$
- for certain kernels: allows for interpreting SVM result

Future Work:

- feature selection
- apply to regression

Acknowledgements: Partial support from the PASCAL Network of Excellence (EU #506778), DFG grants JA379/13-2 and MU987/2-1. Thanks to Alexander Zien, Gökhan H. Bakır and K.-R. Müller.