Large Scale Genomic Sequence SVM Classifiers

Sören Sonnenburg[†], Gunnar Rätsch[♮], Bernhard Schölkopf[*]

Fraunhofer FIRST Institut Rechnerarchitektur und Software technik

MAX-PLANCK-GESELLSCHAFT

[†] Fraunhofer FIRST.IDA, Kekuléstr. 7, 12489 Berlin, Germany
[♮] Friedrich Miescher Laboratory of the Max Planck Society,
[*] Max Planck Institute for Biological Cybernetics,
Spemannstr. 37-39, 72076 Tübingen, Germany

Soeren.Sonnenburg@first.fraunhofer.de,
{Gunnar.Raetsch,Bernhard.Schoelkopf}@tuebingen.mpg.de

Roadmap:

- Large Scale Problems
- Algorithm
- A Real World Large Scale Dataset
- Results
- Outlook and Conclusion

Large Scale Problems

- Text Classification
  - Task: Given N documents, with class label ± 1, predict text type.
  - Approach: Words-in-a-bag kernel, n-gram kernel + SVM
- Biology, e.g. Promotor, Splice Site Prediction
  - Task: Given N sequences around Promotor/Splice Site (label $+1$) and fake examples (label -1), predict whether there is a Promotor/Splice Site in the middle
  - Approach: String kernel + SVM

Properties:

$\Rightarrow$ **large N is needed to achieve high accuracy (i.e. $N = 1,000,000$)**
$\Rightarrow$ **kernel is inner product of sparse feature vectors**

Formally

- Given:
  - N training examples $(\boldsymbol{x}_i, y_i) \in (\mathcal{X}, \pm 1)$, $i = 1 \ldots N$
  - kernel $K(\boldsymbol{x}, \boldsymbol{x}') = \Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}')$
  - where $\mathcal{X}$ discrete space and $\Phi(x)$ sparse
- Examples:
  - words-in-a-bag-kernel
  - k-mer based kernels (Spektrum, Weighted Degree)
- Task: Train SVM on Large Scale Datasets, e.g. $N = 10^6$

Speeding up SVM training

FIRST

How?

- optimize kernel (i.e. find $O(L)$ formulation, where $L = dim(\mathcal{X})$)
- tune SVM training

$\Rightarrow$ **We will do both!**

SVM training:

- Kernel Caching infeasable
  (for $N = 10^6$ only 125 kernel rows fit in 1GiB memory)
- **Proposed Method is faster + needs no kernel caching**

Derivation I

Analyzing SVM^{light}:

training algorithm (chunking):

while optimality conditions are violated **do**
  select q variables for the working set.
  solve reduced problem on the working set.
end while

- At each iteration, the vector $\boldsymbol{f}$, $f_i = \sum_{j=1}^{N} \alpha_j y_j \, \mathrm{k}(x_i, x_j) \;\; i = 1 \ldots N$ is needed for checking termination criteria and selecting new working set (based on gradient w.r.t. $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}$).

- Avoiding to recompute $\boldsymbol{f}$, most time is spend computing "linear updates" on $\boldsymbol{f}$ on the working set W

$$f_i \leftarrow f_i^{old} + \sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j \, \mathrm{k}(x_i, x_j)$$

Derivation II

The update rule: $f_i \leftarrow f_i^{old} + \sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j \, \mathrm{k}(x_i, x_j)$

Exploiting $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$ and $\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\boldsymbol{x}_i)$:

$$f_i \leftarrow f_i^{old} + \sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) = f_i^{old} + \boldsymbol{w}^W \cdot \Phi(\boldsymbol{x}_i)$$

($\boldsymbol{w}^W$ normal on working set)

Key Idea: Store $\boldsymbol{w}^W$ and compute $\boldsymbol{w}^W \cdot \Phi(\boldsymbol{x})$ *efficiently*

When is that possible ?

1. $\boldsymbol{w}^W$ has low dimensionality, sparse (e.g. 4^8 for Feature map of Spectrum Kernel of order 8 DNA)

2. $\boldsymbol{w}^W$ has very few nonzero entries, although high dimensional (e.g. 10^{14} for Weighted Degree Kernel of order 20 on DNA sequences of length 100)

TECHNICAL REMARK

FIRST

Treating $\boldsymbol{w}$

- $\boldsymbol{w}$ must be accessible by some index u (i.e. $u = 1 \ldots 4^8$ for 8-mers of Spectrum Kernel on DNA or word index for word-in-a-bag kernel)

- Needed Operations
  - `Clear:` $\boldsymbol{w} = \mathbf{0}$
  - `Add:` $w_u \leftarrow w_u + v$ (only needed $|W|$ times per iteration)
  - `Lookup:` obtain w_u (must be highly efficient)

- Storage
  - Explicit Map (store dense $\boldsymbol{w}$); `Lookup` in $\mathcal{O}(1)$
  - Sorted Array (word-in-bag-kernel: all words sorted with value attached) `Lookup` in $\mathcal{O}(\log(\sum_u I(w_u \neq 0)))$
  - Suffix Trees; `Lookup` in $\mathcal{O}(K)$

Algorithm

Recall we need to compute updates on **f** (effort $c_1|W|LN$):

$$f_i \leftarrow f_i^{old} + \sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j \, \mathrm{k}(x_i, x_j) \text{ for all } i = 1 \ldots N$$

Modified SVM^{light} using "LinAdd" algorithm (effort $c_2 \ell LN$, ℓ `lookup` cost)

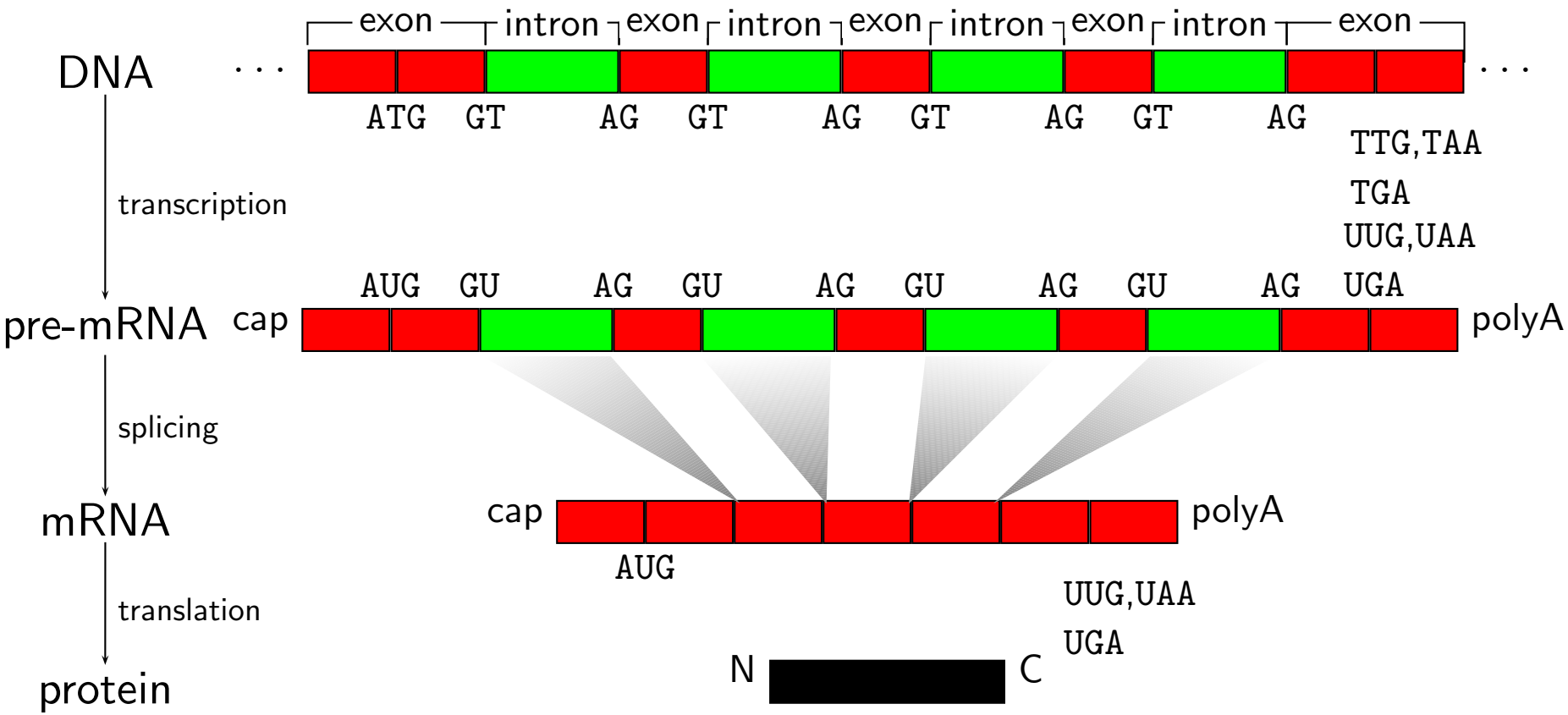
$f_i = 0$, $\alpha_i = 0$ for $i = 1, \ldots, N$
for $t = 1, 2, \ldots$ **do**
  Check optimality conditions and stop if optimal, select working set W
  based on $\mathbf{f}$ and $\boldsymbol{\alpha}$, store $\boldsymbol{\alpha}^{old} = \boldsymbol{\alpha}$
  solve reduced problem W and update $\boldsymbol{\alpha}$

  clear $\boldsymbol{w}$
  $\boldsymbol{w} \leftarrow \boldsymbol{w} + (\alpha_j - \alpha_j^{old}) y_j \Phi(\boldsymbol{x}_j)$ for all $j \in W$
  update $f_i = f_i + \boldsymbol{w} \cdot \Phi(\boldsymbol{x}_i)$ for all $i = 1, \ldots, N$
end for

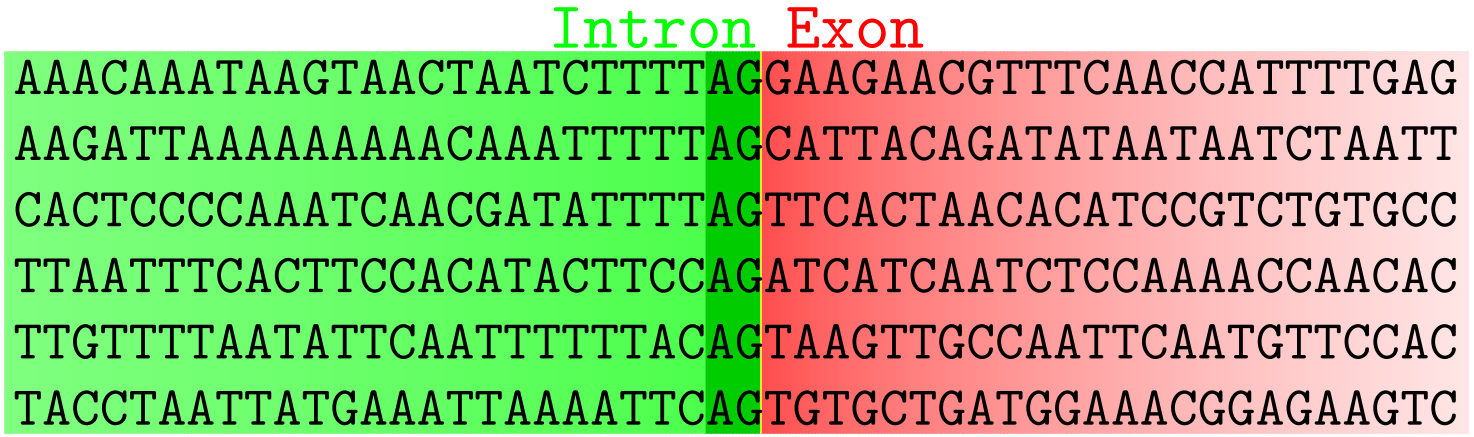
Speedup of factor $\frac{c_1}{c_2 \ell} |W|$

A Real World Large Scale Dataset

Splice sites are locations on DNA at boundaries of
- exons (which code for proteins)
- introns (which do not)



Biology: Detection of Splice Sites



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

$\Rightarrow$ 2-class classification problem

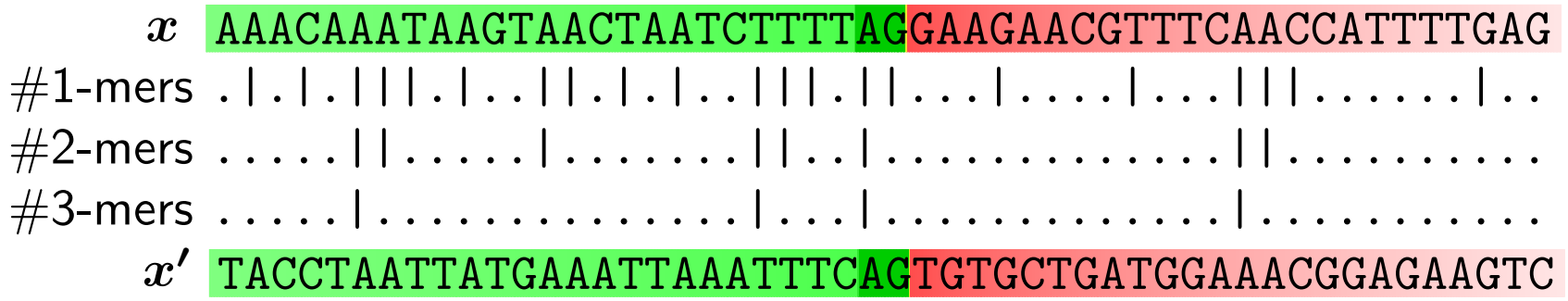
Approach: Weighted Degree Kernel + SVM

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

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

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

Example degree $d = 3$:



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

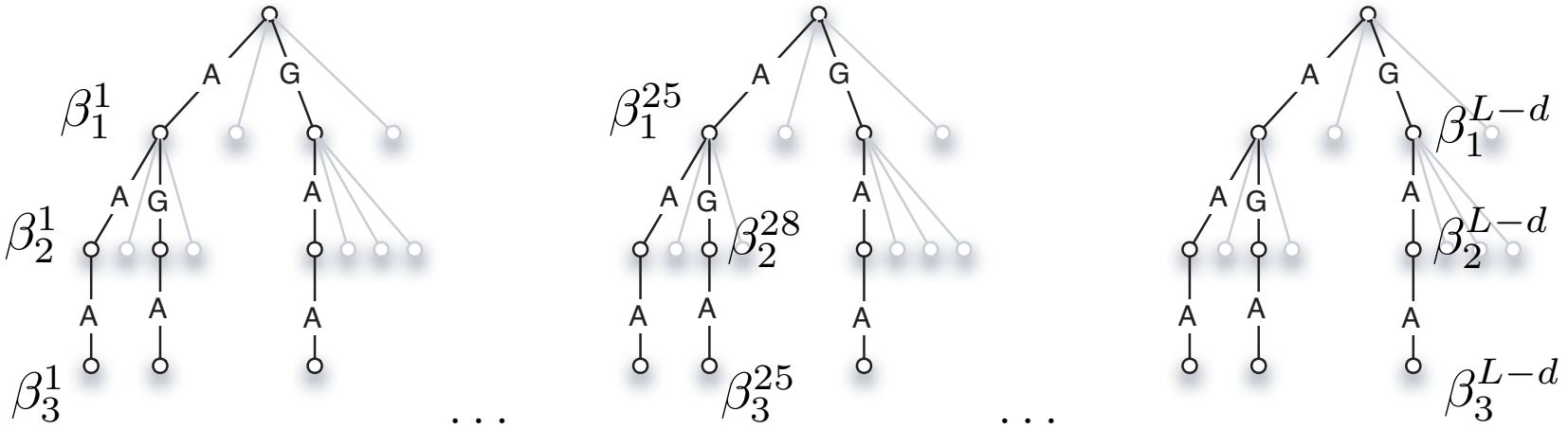
Efficient computation via Suffix Tree

for linear comb. of kernels: $\sum_{j \in W} (\alpha_j - \alpha_j^{old}) y_j \, \mathrm{k}(x_i, x_j)$ $(\mathcal{O}(Ld|W|N))$

AAACTAATTATGAAATTAAATTTCAGAGTGCTGATGGAAACGGAGAAGAA

- use one tree of depth d per position in sequence
- for `lookup` use traverse one tree of depth d per position in sequence

Example $d = 3$:



output for N sequences of length L in $\mathcal{O}(Ld \cdot N)$ (d depth of tree $\equiv$ degree of WD kernel)

Results

N	AUC	rel. AUC Imp.	Test Err	t orig	t linadd
500	96.91%	-	6.03%	1	3
1000	97.82%	29.45%	6.03%	1	5
5000	98.96%	52.29%	3.38%	19	24
10000	99.28%	30.77%	2.40%	58	45
30000	99.58%	41.67%	1.57%	317	**159**
50000	99.65%	16.67%	1.31%	794	**355**
100000	99.73%	22.86%	1.07%	2,507	**761**
200000	99.80%	25.93%	0.92%	8,863	**2,024**
500000	99.84%	20.00%	0.83%	40,632	**9,119**
1000000	99.87%	18.75%	0.71%	131,379	**26,107**

- doubling the training data reduces the AUC gap to 100% by 20%

(high accuracy necessary as classifier will be applied genome wide)

Speedup of factor 5!

Conclusion

Conclusion:

- general method applicable to all kernels that can be written as inner product in some sparse feature space, which can be enumerated + has efficient `clear, add, lookup` operations
- speedup of factor 20 (5) for Spectrum (Weighted Degree) kernel (also speeds up MKL)
- more training data helps (99.87% AUC on $C.\ elegans$ splice sites)
- in paper extensions to Spectrum/Weighted Degree (WD) kernel:
  - WD kernel: $\mathcal{O}(L)$ algorithm; WD kernel with-mismatches; position invariant version
  - predicting using mismatch spectrum kernel in $\mathcal{O}(KL)$

Future: train on *Human Splice Sites:* $3 \cdot 10^7$ examples
(note: dataset already $\approx 6GiB$ in size)