Kernel Methods for Fusing Heterogeneous Data

Gunnar Rätsch

Friedrich Miescher Laboratory, Max Planck Society
Tübingen, Germany

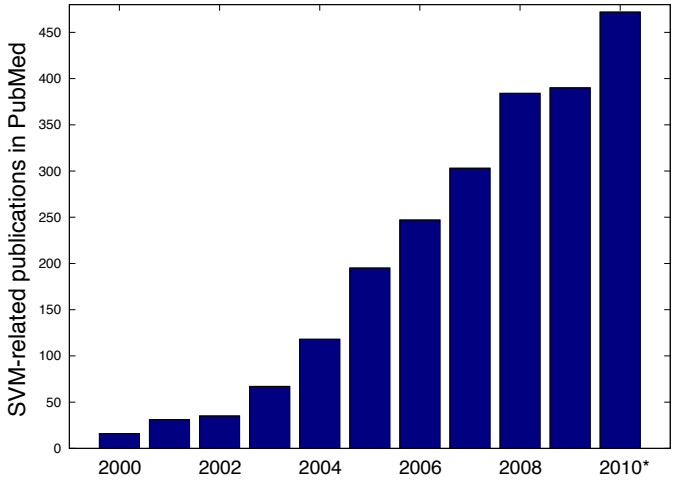
Pre-conference Course, Bio-IT World Europe, Hannover, Germany
October 4, 2010



Roadmap

- Support Vector Machines (SVMs)
- Kernels and the "trick"
- Kernels for non-vectorial data
- Heterogeneous data integration
- Examples
- Software

SVM-related publications



Slides and additional material available at:

http://tinyurl.com/dfbi2010

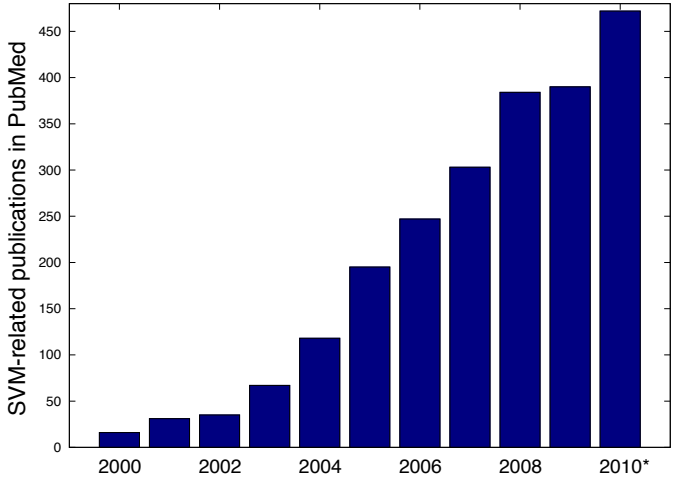
http://fml.mpg.de/raetsch/lectures/datafusion-bio-it-2010

fml

Roadmap

- Support Vector Machines (SVMs)
- Kernels and the "trick"
- Kernels for non-vectorial data
- Heterogeneous data integration
- Examples
- Software

SVM-related publications



Slides and additional material available at:

http://tinyurl.com/dfbi2010

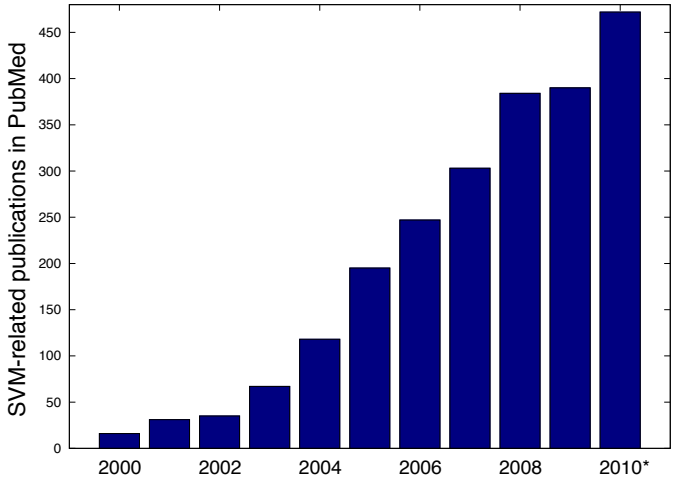
http://fml.mpg.de/raetsch/lectures/datafusion-bio-it-2010

fml

Roadmap

- Support Vector Machines (SVMs)
- Kernels and the "trick"
- Kernels for non-vectorial data
- Heterogeneous data integration
- Examples
- Software

SVM-related publications



Slides and additional material available at:

http://tinyurl.com/dfbi2010

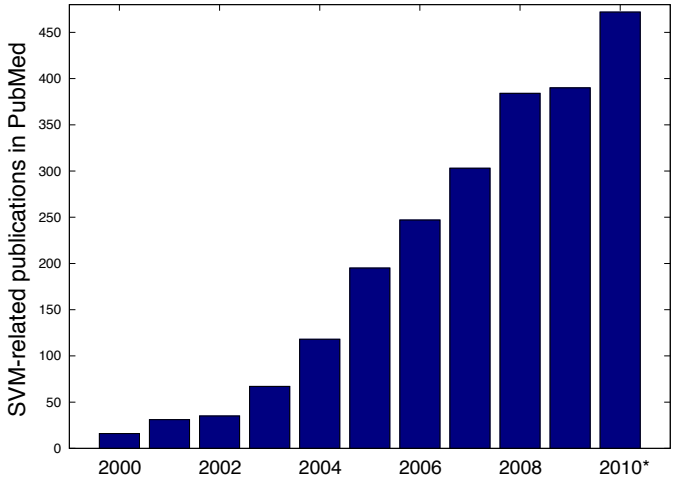
http://fml.mpg.de/raetsch/lectures/datafusion-bio-it-2010

fml

Roadmap

- Support Vector Machines (SVMs)
- Kernels and the "trick"
- Kernels for non-vectorial data
- Heterogeneous data integration
- Examples
- Software

SVM-related publications



Slides and additional material available at:

http://tinyurl.com/dfbi2010

http://fml.mpg.de/raetsch/lectures/datafusion-bio-it-2010

fml

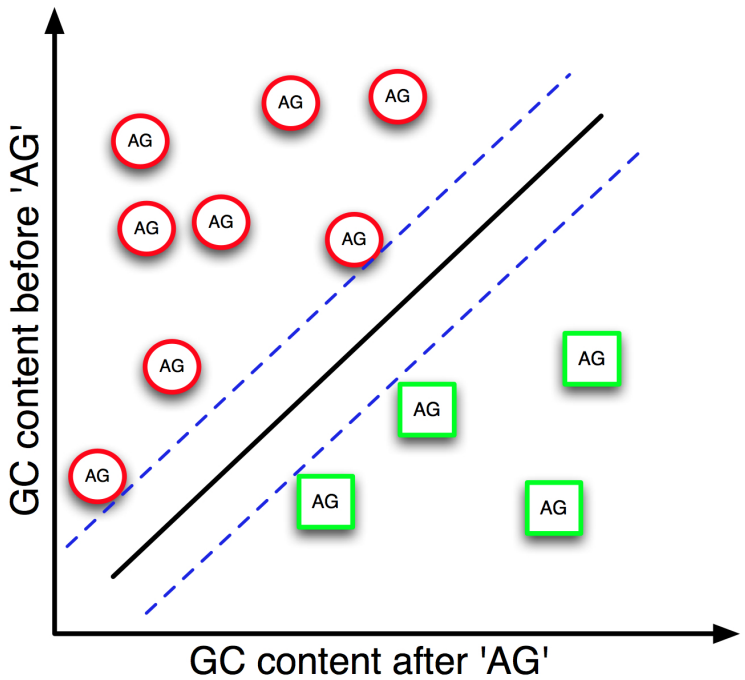
Example: Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

- Goal: Rule that distinguishes true from false ones

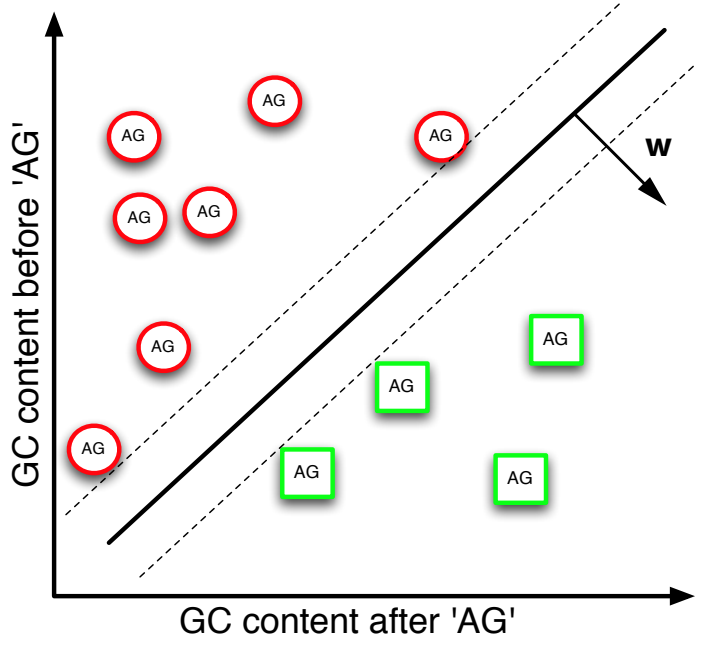


Linear Classifiers
with large **margin**

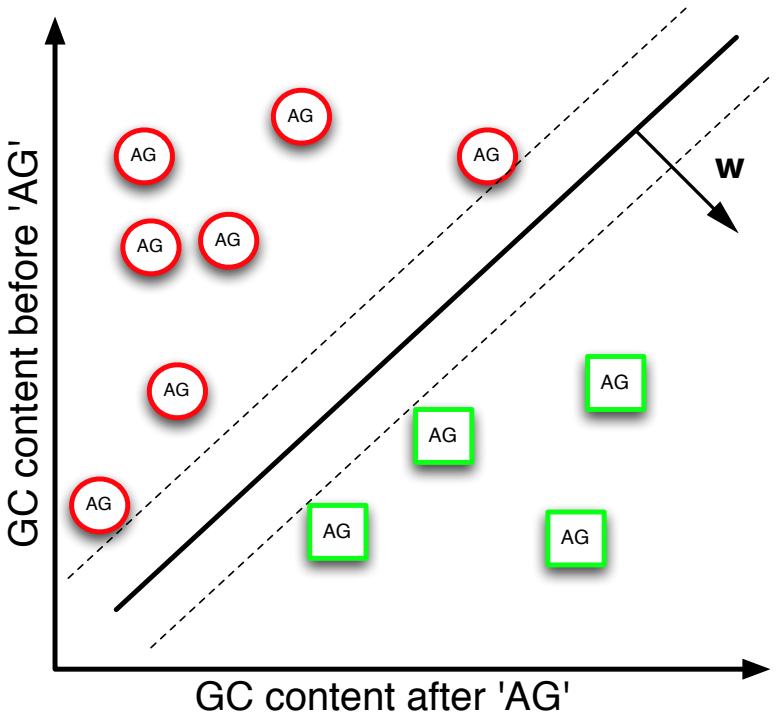
fml

SVMs: Maximize the margin! Why?

- Intuitively, it feels the safest.
- For a small error in the separating hyperplane, we do not suffer too many mistakes.
- Empirically, it works well.
- Learning theory indicates that it is the right thing to do.



How to Maximize the Margin?



Maximize

$$\underbrace{\rho}_{=\text{margin}} - C\sum_{i=1}^{n}\xi_i$$

Subject to

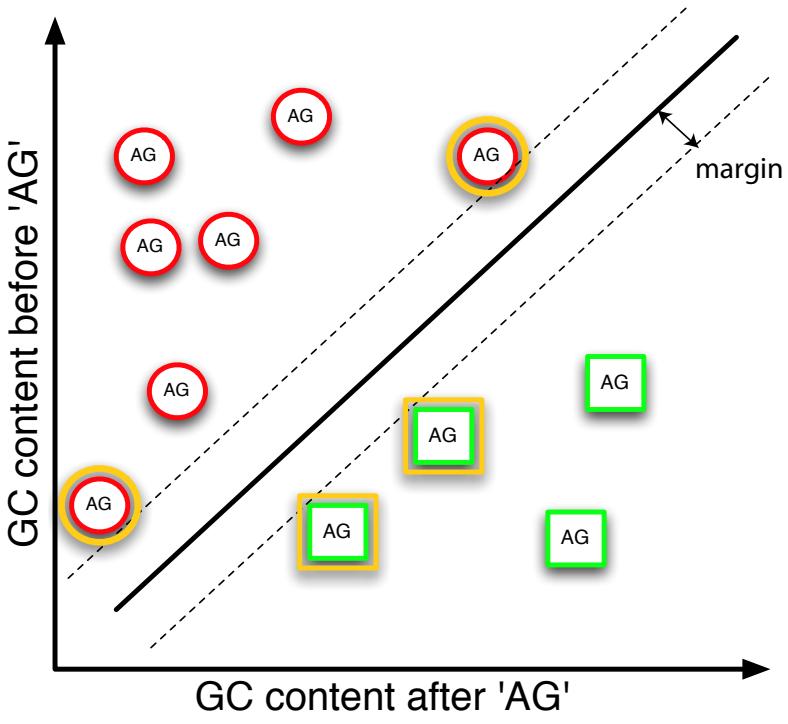
$$y_i\langle \mathbf{w}, \mathbf{x}_i\rangle \geqslant \rho - \xi_i$$
$$\xi_i \geqslant 0$$
for all $i = 1, \ldots, n$,
$\|\mathbf{w}\| = 1$.

- Examples on the margin are called support vectors [Vapnik, 1995]
- Soft margin SVMs [Cortes and Vapnik, 1995]
- Hyperplane only depends on distances between examples:

$$d(\mathbf{x}, \mathbf{x}')^2 = \|\mathbf{x} - \mathbf{x}'\|^2 = \langle \mathbf{x}, \mathbf{x}\rangle - \underbrace{\langle \mathbf{x}, \mathbf{x}'\rangle}_{\text{scalar product}} + \langle \mathbf{x}', \mathbf{x}'\rangle$$

fml

How to Maximize the Margin?



Maximize

$$\underbrace{\rho}_{=\text{margin}} - C\sum_{i=1}^{n}\xi_i$$

Subject to

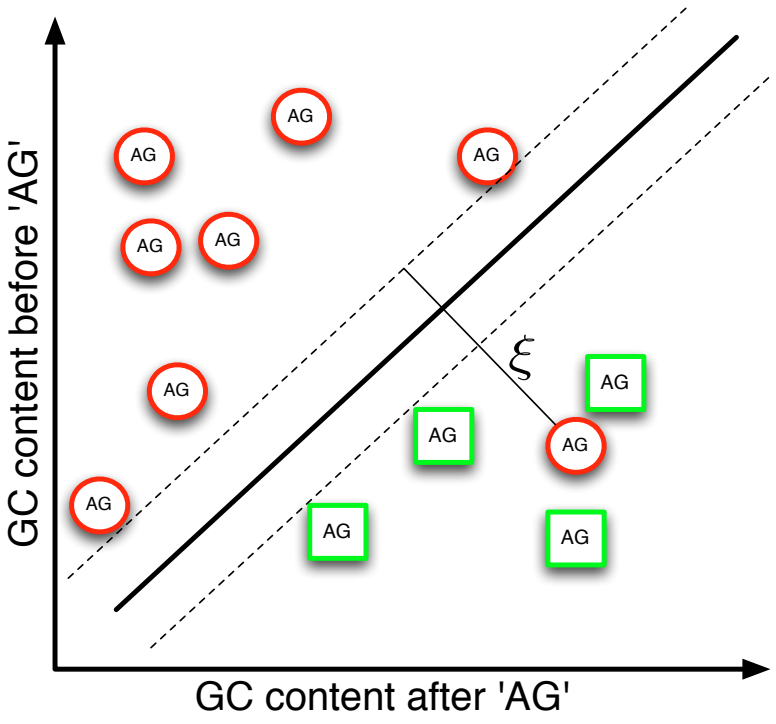
$$y_i\langle \mathbf{w}, \mathbf{x}_i\rangle \geqslant \rho - \xi_i$$
$$\xi_i \geqslant 0$$
for all $i = 1, \ldots, n$,
$\|\mathbf{w}\| = 1$.

- Examples on the margin are called **support vectors** [Vapnik, 1995]
- Soft margin SVMs [Cortes and Vapnik, 1995]
- Hyperplane only depends on distances between examples:

$$d(\mathbf{x}, \mathbf{x}')^2 = \|\mathbf{x} - \mathbf{x}'\|^2 = \langle \mathbf{x}, \mathbf{x}\rangle - \underbrace{\langle \mathbf{x}, \mathbf{x}'\rangle}_{\text{scalar product}} + \langle \mathbf{x}', \mathbf{x}'\rangle$$

fml

How to Maximize the Margin?



Maximize

$$\underbrace{\rho}_{=\text{margin}} - C \sum_{i=1}^{n} \xi_i$$

Subject to

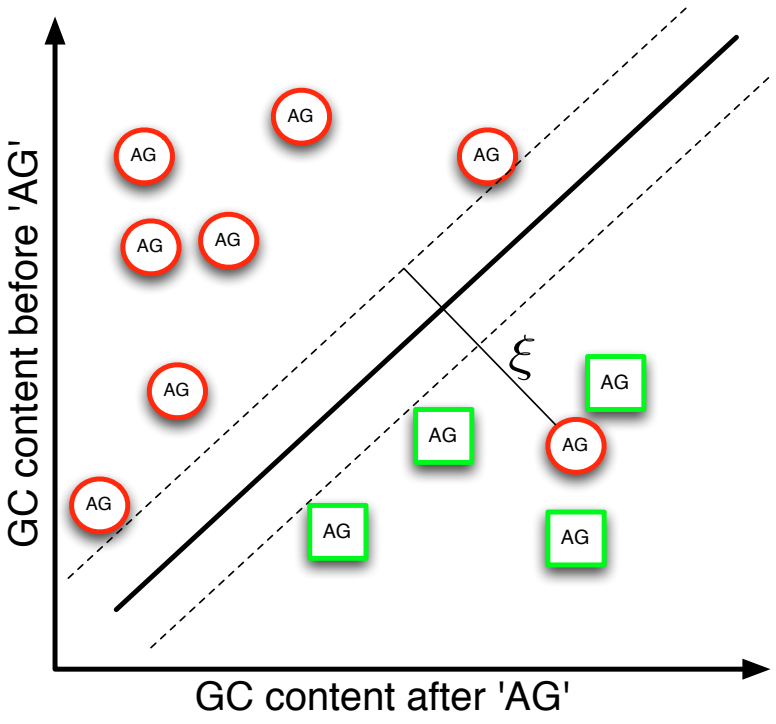
$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geqslant \rho - \xi_i$$
$$\xi_i \geqslant 0$$
for all $i = 1, \ldots, n$,
$$\|\mathbf{w}\| = 1.$$

- Examples on the margin are called **support vectors** [Vapnik, 1995]
- Soft margin SVMs [Cortes and Vapnik, 1995]
- Hyperplane only depends on distances between examples:

$$d(\mathbf{x}, \mathbf{x}')^2 = \|\mathbf{x} - \mathbf{x}'\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle - \underbrace{\langle \mathbf{x}, \mathbf{x}' \rangle}_{\text{scalar product}} + \langle \mathbf{x}', \mathbf{x}' \rangle$$

fml

How to Maximize the Margin?



Maximize

$$\underbrace{\rho}_{=\text{margin}} - C \sum_{i=1}^{n} \xi_i$$

Subject to

$$y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geqslant \rho - \xi_i$$
$$\xi_i \geqslant 0$$
for all $i = 1, \ldots, n$,
$$\|\mathbf{w}\| = 1.$$

- Examples on the margin are called **support vectors** [Vapnik, 1995]
- Soft margin SVMs [Cortes and Vapnik, 1995]
- Hyperplane only depends on distances between examples:

$$d(\mathbf{x}, \mathbf{x}')^2 = \|\mathbf{x} - \mathbf{x}'\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle - \underbrace{\langle \mathbf{x}, \mathbf{x}' \rangle}_{\text{scalar product}} + \langle \mathbf{x}', \mathbf{x}' \rangle$$

fml

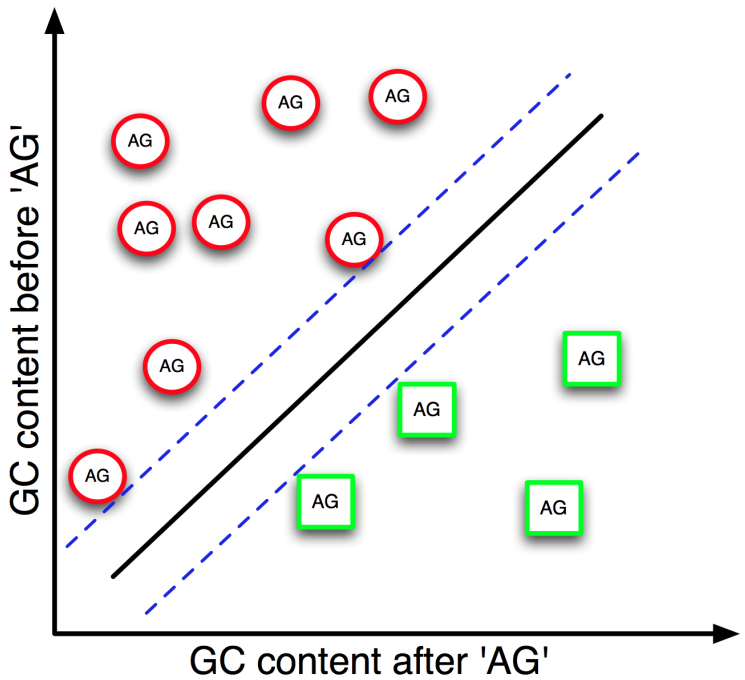
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

- Goal: Rule that distinguishes true from false ones



Linear Classifiers with large **margin**

fml

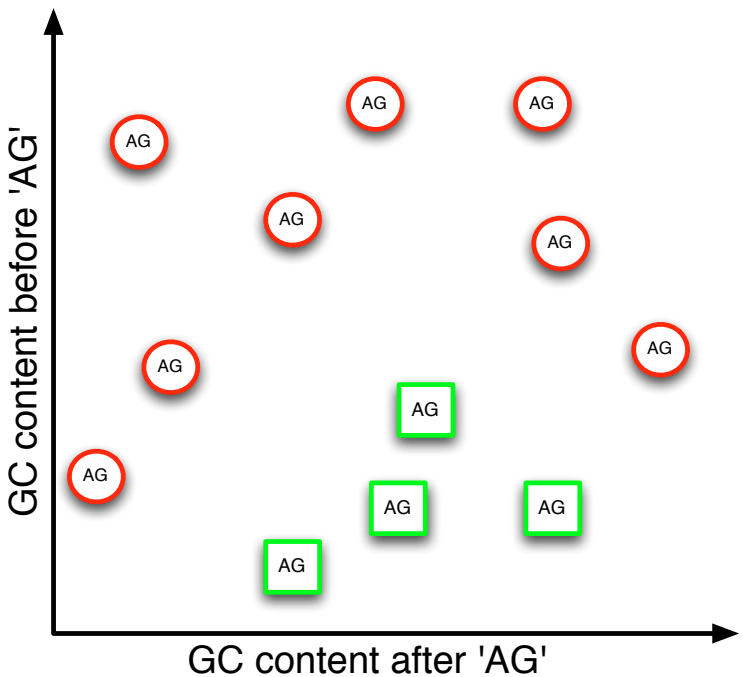
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

- Goal: Rule that distinguishes true from false ones



More realistic problem?

- Not linearly separable!
- Need nonlinear separation?
- Need more features?

fml

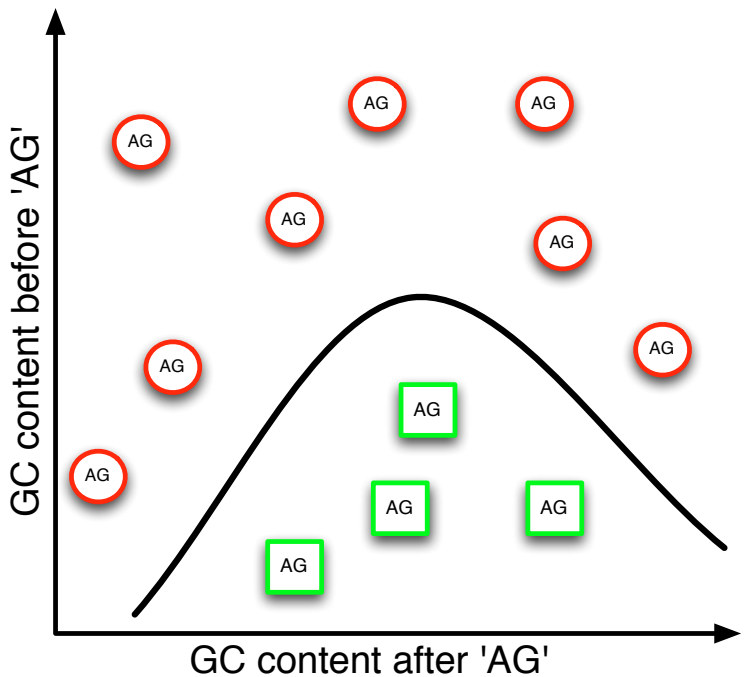
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

- Goal: Rule that distinguishes true from false ones



More realistic problem?

- Not linearly separable!
- **Need nonlinear separation?**
- Need more features?

fml

Nonlinear Separations

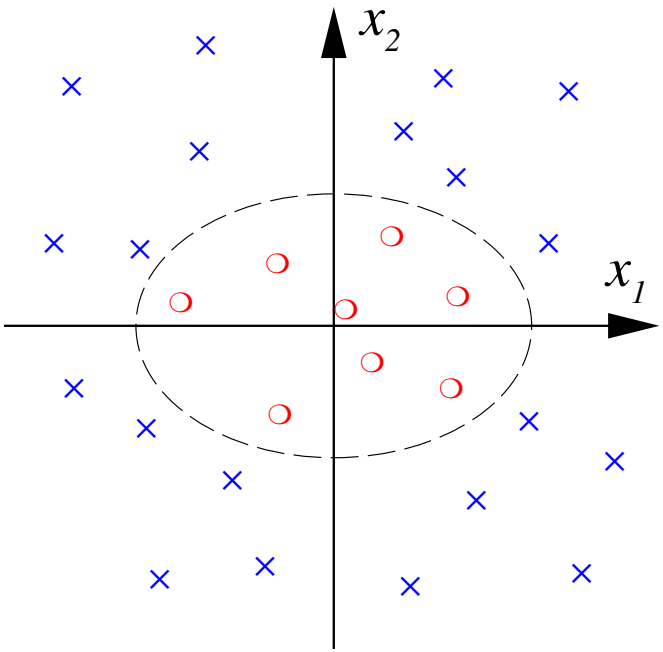
Linear separation might not be sufficient!

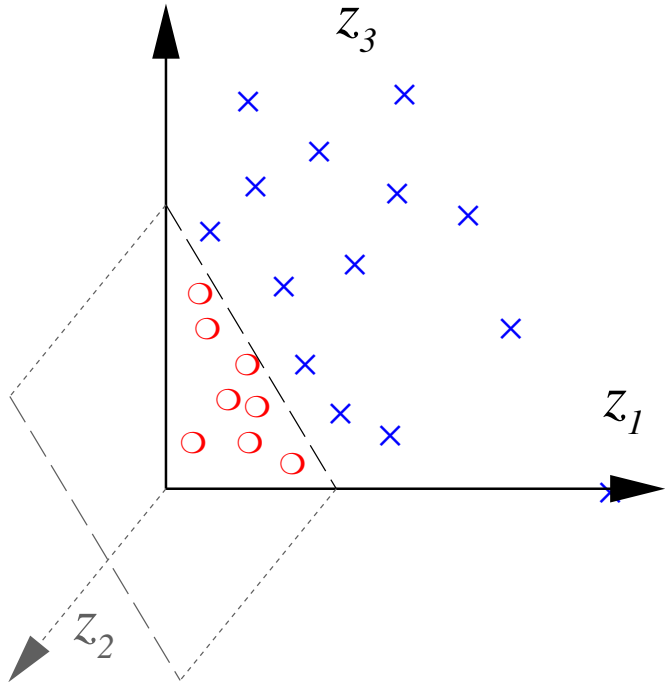
$\Rightarrow$ Map into a higher dimensional feature space

Example:

[Schölkopf and Smola, 2002]

$$\begin{aligned} \Phi : \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2) \end{aligned}$$





Kernel "Trick"

Example: $\mathbf{x} \in \mathbb{R}^2$ and $\Phi(\mathbf{x}) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$ [Boser et al., 1992]

$$\begin{aligned}
\langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}) \rangle &= \left\langle (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2), (\hat{x}_1^2, \sqrt{2}\, \hat{x}_1 \hat{x}_2, \hat{x}_2^2) \right\rangle \\
&= \langle (x_1, x_2), (\hat{x}_1, \hat{x}_2) \rangle^2 \\
&= \langle \mathbf{x}, \hat{\mathbf{x}} \rangle^2 \\
&=: \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}})
\end{aligned}$$

- Scalar product in feature space (here $\mathbb{R}^3$) can be computed in input space (here $\mathbb{R}^2$)!
- Also works for higher orders and dimensions
  $\Rightarrow$ relatively low-dimensional input spaces
  $\Rightarrow$ very high-dimensional feature spaces

fml

Common Kernels

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

Notes:

- These kernels are good for real-valued examples
- Kernels may be combined in case of heterogeneous data

[Vapnik, 1995, Müller et al., 2001, Schölkopf and Smola, 2002]

fml

Common Kernels

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

Notes:

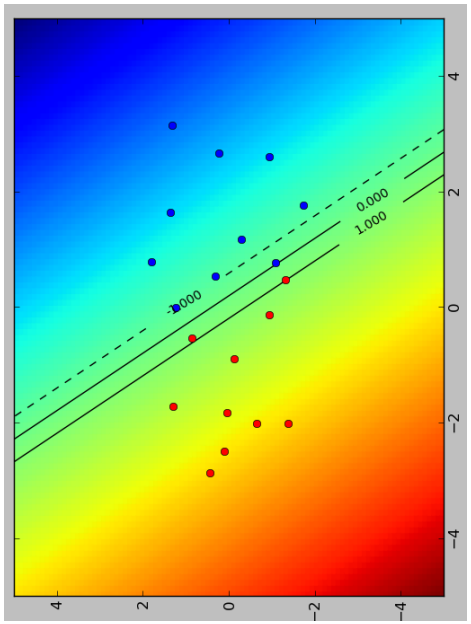
- These kernels are good for real-valued examples
- Kernels may be combined in case of heterogeneous data

[Vapnik, 1995, Müller et al., 2001, Schölkopf and Smola, 2002]

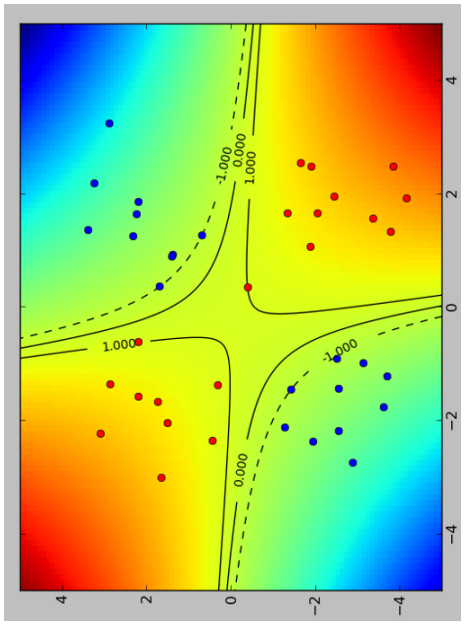
fml

Illustration of Common Kernels

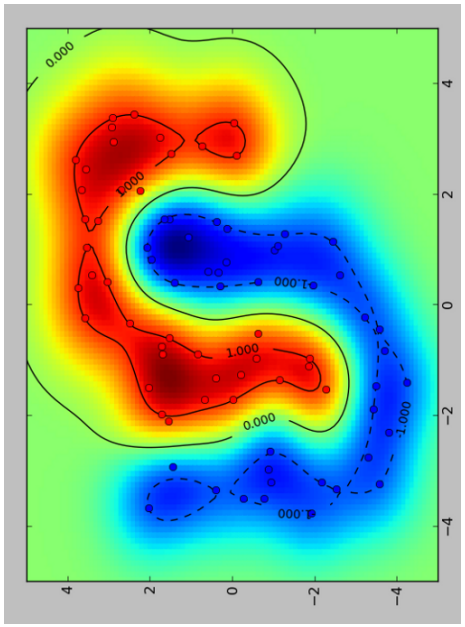
- Kernels for real-valued data



(a) Linear



(b) Polynomial



(c) Gaussian

GC-Content-based Splice Site Recognition

Results for the illustrative example:

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.

Kernel Summary

- Nonlinear separation $\Leftrightarrow$ linear separation of nonlinearly Φ-mapped examples
- Mapping Φ defines a kernel by

$$k(\mathbf{x}, \hat{\mathbf{x}}) := \langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}) \rangle$$

- Choice of kernel has to match the data at hand
- RBF kernel often works pretty well

Kernels for Non-vectorial Data

Examples:

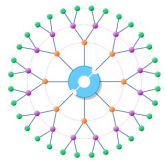
- Biological sequences (DNA, proteins, . . .)



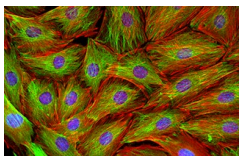
- Natural language (abstracts, gene mentioning, . . .)



- Graphs/Networks (interactions, co-expression, . . .)



- Images (cell imaging, tissue classification, . . .)



- Structured data (gene structures, patient information, . . .)

General ideas:

- Kernels compute similarity between examples
- Good kernels use domain-specific knowledge

fml

Kernels for Non-vectorial Data

Examples:

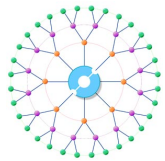
- Biological sequences (DNA, proteins, . . .)



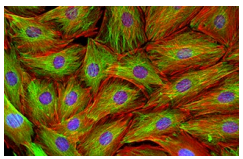
- Natural language (abstracts, gene mentioning, . . .)



- Graphs/Networks (interactions, co-expression, . . .)



- Images (cell imaging, tissue classification, . . .)



- Structured data (gene structures, patient information, . . .)

General ideas:

- Kernels compute similarity between examples
- Good kernels use domain-specific knowledge

fml

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

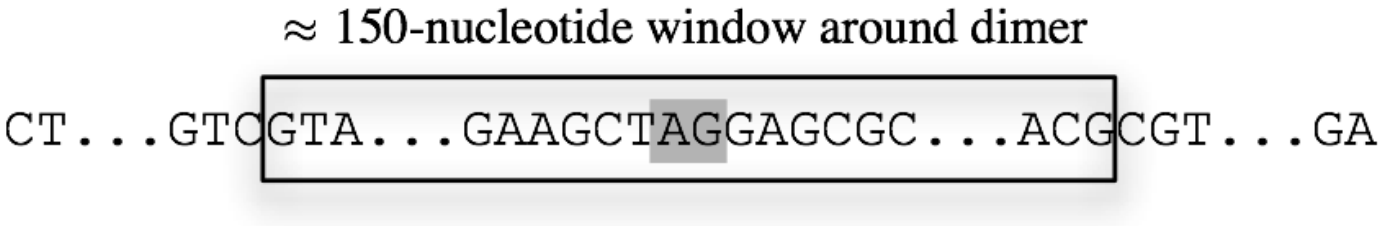
Variations

- Allow gaps or mismatches
- Include wildcards or physico-chemical properties
- Motif kernels
- Consider position in the sequence

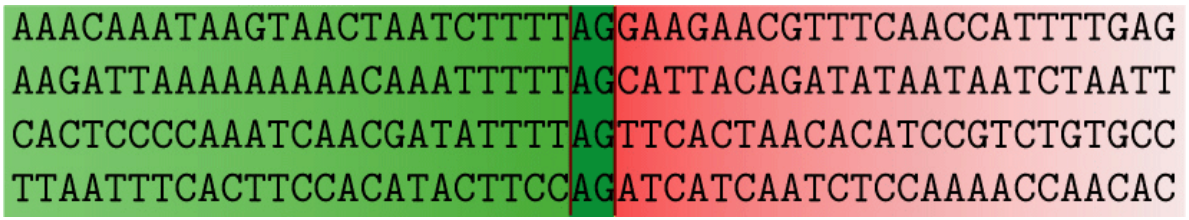
[Haussler, 1999, Zien et al., 2000, Leslie et al., 2002, 2003, Liao and Noble, 2002, Leslie and Kuang, 2004]

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



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

Types of Signal Detection Problems

Problem categorization

(based on positional variability of motifs)

Position-Dependent

→ Motifs very stiff, almost always at same position,

```
AAACAAATAAGTAACTAATCTTTTAAGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAAACAAATTTTTAACATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAATTCACTAACACATCCGTCTGTGCC
```

for instance, splice site identification

Types of Signal Detection Problems

Problem categorization

(based on positional variability of motifs)

Mixture of Position-Dependent/-Independent

$\longrightarrow$ variable but still positional information

```
AAACAAATAAGTAACTAATCTTTTAAAGAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTCATTAAATACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTAAATTTCACTAACACATCCGTCTGTGC
```

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
  - Count n-mers in sequence $\mathbf{x}$ and sequence $\mathbf{x}'$.
  - Spectrum Kernel is sum of product of counts (for same n-mer)

Example $n = 3$:

x AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG

x' TACCTAATTATGAAATTAAATTTCAGTGTGCTGATGGAAACGGAGAAGTC

3-mer	AAA	AAC	...	CCA	CCC	...	TTT
# in $\mathbf{x}$	2	4	...	1	0	...	3
# in $\mathbf{x}'$	3	1	...	0	0	...	1

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

fml

Weighted Degree Kernel

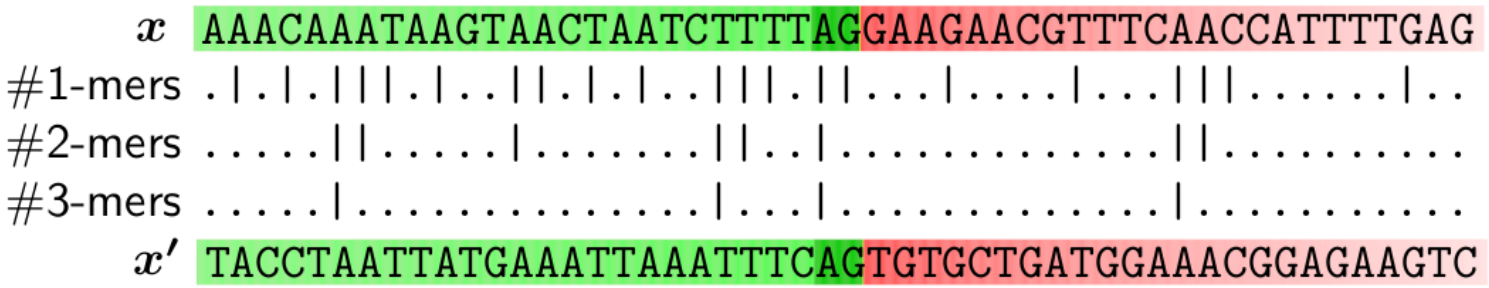
= Spectrum kernels for each position

To make use of position-dependent motifs:

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

- $L :=$ 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 $d = 3$:



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

[Rätsch and Sonnenburg, 2004, Sonnenburg et al., 2007]

fml

Weighted Degree Kernel

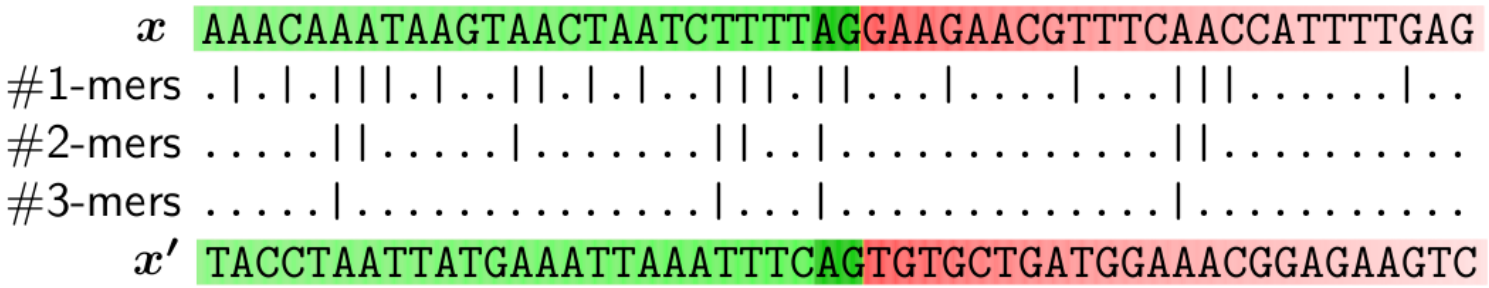
= Spectrum kernels for each position

To make use of position-dependent motifs:

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

- $L :=$ 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 $d = 3$:



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

[Rätsch and Sonnenburg, 2004, Sonnenburg et al., 2007]

fml

Sequence-based Splice Site Recognition

Kernel	auROC
Best vectorial	91.4%
Spectrum $\ell = 1$	94.0%
Spectrum $\ell = 3$	96.4%
Spectrum $\ell = 5$	94.5%
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 and weighted degree kernels for the acceptor splice site recognition task for different substring lengths ℓ.

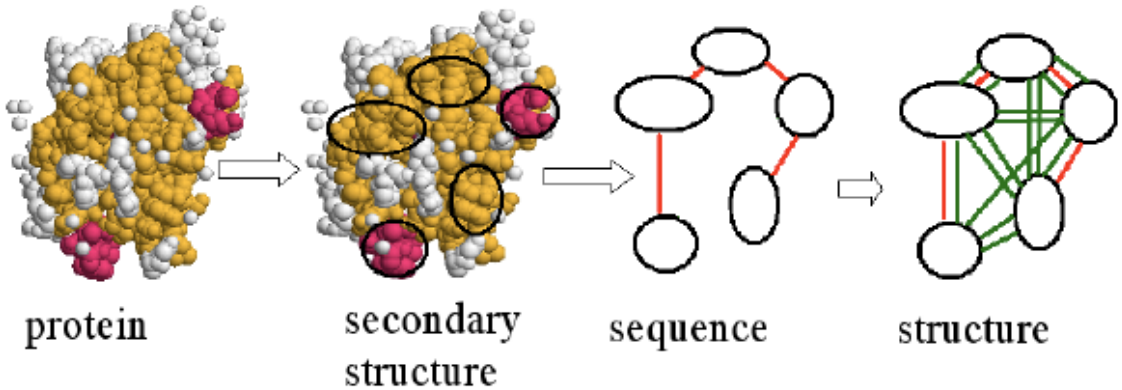
(Alternatives: mixed spectrum, WD with shifts, oligo kernel, ...)

[Zien et al., 2000, Jaakkola et al., 2000, Tsuda et al., 2002, Liao and Noble, 2002]

[Meinicke et al., 2004, Vert et al., 2004, Rätsch et al., 2005, Schultheiss et al., 2008]

Kernels for Graphs and Images

- Kernels on Graphs



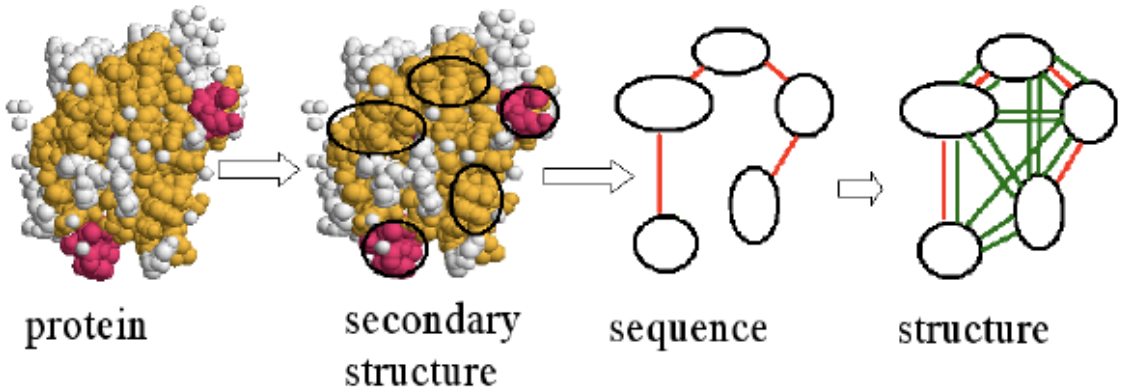
Idea: Two graphs are similar, when they contain many similar paths or sub-graphs [Ralaivola et al., 2005, Borgwardt et al., 2005]

- Kernels on Images

Idea: Two images are similar, when they contain a similar distribution of interest points [Nowak et al., 2006, Gehler and Nowozin, 2009]

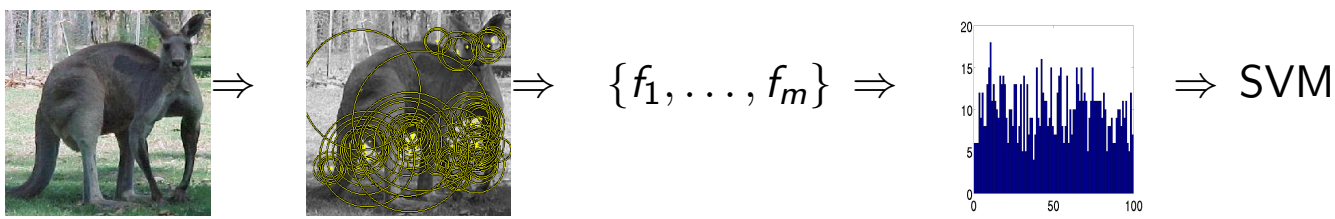
Kernels for Graphs and Images

- Kernels on Graphs



Idea: Two graphs are similar, when they contain many similar paths or sub-graphs [Ralaivola et al., 2005, Borgwardt et al., 2005]

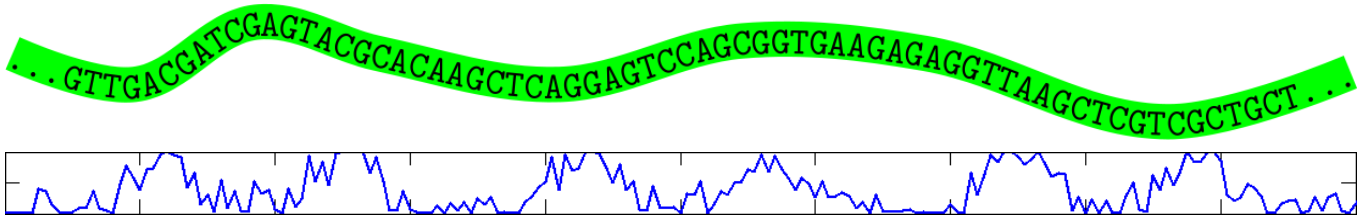
- Kernels on Images



Idea: Two images are similar, when they contain a similar distribution of interest points [Nowak et al., 2006, Gehler and Nowozin, 2009]

Combining Heterogeneous Data

- Consider data from different domains: e.g DNA-strings, structure, binding energies, conservation, ...



- Several options: [Lanckriet et al., 2004]
  - Define a new kernel based on both information pieces. For instance,
    - based on prior knowledge, or
    - by adding or multiplying existing kernels:

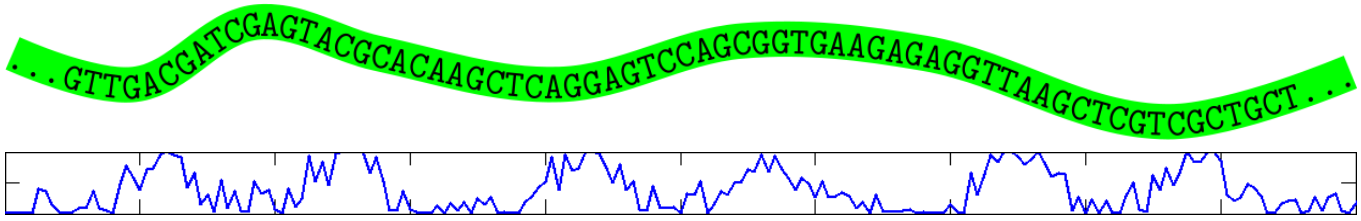
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad \text{or} \quad k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}')$$

  - Train classifiers independently and appropriately combine them in second step ("late integration")

fml

Combining Heterogeneous Data

- Consider data from different domains: e.g DNA-strings, structure, binding energies, conservation, ...



- Several options: [Lanckriet et al., 2004]
  - Define a new kernel based on both information pieces. For instance,
    - based on prior knowledge, or
    - by adding or multiplying existing kernels:

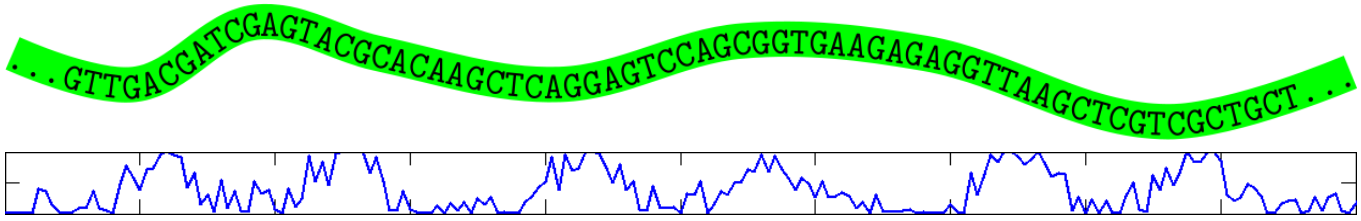
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad \text{or} \quad k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}')$$

  - Train classifiers independently and appropriately combine them in second step ("late integration")

fml

Combining Heterogeneous Data

- Consider data from different domains: e.g DNA-strings, structure, binding energies, conservation, ...



- Several options: [Lanckriet et al., 2004]
  - Define a new kernel based on both information pieces. For instance,
    - based on prior knowledge, or
    - by adding or multiplying existing kernels:

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad \text{or} \quad k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}')$$

  - Train classifiers independently and appropriately combine them in second step ("late integration")

fml

Combinations of Kernels: Multiple Kernel Learning

Possible solution: We can add the two kernels,

$$k(\mathbf{x}, \mathbf{x}') := k_{sequence}(\mathbf{x}, \mathbf{x}') + k_{structure}(\mathbf{x}, \mathbf{x}').$$

Better solution: We can mix the two kernels,

$$k(\mathbf{x}, \mathbf{x}') := (1 - t)k_{sequence}(\mathbf{x}, \mathbf{x}') + tk_{structure}(\mathbf{x}, \mathbf{x}'),$$

where t is estimated from the training data

In general: use the data to find the best convex combination.

$$k(\mathbf{x}, \mathbf{x}') = \sum_{p=1}^{K} \beta_p k_p(\mathbf{x}, \mathbf{x}').$$

Algorithms

- SDP/QCQP [Lanckriet et al., 2004, Bach et al., 2004]
- SILP (part of shogun toolbox) [Sonnenburg et al., 2006a]
- SimpleMKL [Rakotomamonjy et al., 2008]

fml

Combinations of Kernels: Multiple Kernel Learning

Possible solution: We can add the two kernels,

$$k(\mathbf{x}, \mathbf{x}') := k_{sequence}(\mathbf{x}, \mathbf{x}') + k_{structure}(\mathbf{x}, \mathbf{x}').$$

Better solution: We can mix the two kernels,

$$k(\mathbf{x}, \mathbf{x}') := (1 - t)k_{sequence}(\mathbf{x}, \mathbf{x}') + tk_{structure}(\mathbf{x}, \mathbf{x}'),$$

where t is estimated from the training data

In general: use the data to find the best convex combination.

$$k(\mathbf{x}, \mathbf{x}') = \sum_{p=1}^{K} \beta_p k_p(\mathbf{x}, \mathbf{x}').$$

Algorithms

- SDP/QCQP [Lanckriet et al., 2004, Bach et al., 2004]
- SILP (part of shogun toolbox) [Sonnenburg et al., 2006a]
- SimpleMKL [Rakotomamonjy et al., 2008]

fml

Combinations of Kernels: Multiple Kernel Learning

Possible solution: We can add the two kernels,

$$k(\mathbf{x}, \mathbf{x}') := k_{sequence}(\mathbf{x}, \mathbf{x}') + k_{structure}(\mathbf{x}, \mathbf{x}').$$

Better solution: We can mix the two kernels,

$$k(\mathbf{x}, \mathbf{x}') := (1 - t)k_{sequence}(\mathbf{x}, \mathbf{x}') + tk_{structure}(\mathbf{x}, \mathbf{x}'),$$

where t is estimated from the training data

In general: use the data to find the best convex combination.

$$k(\mathbf{x}, \mathbf{x}') = \sum_{p=1}^{K} \beta_p k_p(\mathbf{x}, \mathbf{x}').$$

Algorithms

- SDP/QCQP [Lanckriet et al., 2004, Bach et al., 2004]
- SILP (part of shogun toolbox) [Sonnenburg et al., 2006a]
- SimpleMKL [Rakotomamonjy et al., 2008]

fml

Combinations of Kernels: Multiple Kernel Learning

Possible solution: We can add the two kernels,

$$k(\mathbf{x}, \mathbf{x}') := k_{sequence}(\mathbf{x}, \mathbf{x}') + k_{structure}(\mathbf{x}, \mathbf{x}').$$

Better solution: We can mix the two kernels,

$$k(\mathbf{x}, \mathbf{x}') := (1 - t)k_{sequence}(\mathbf{x}, \mathbf{x}') + tk_{structure}(\mathbf{x}, \mathbf{x}'),$$

where t is estimated from the training data

In general: use the data to find the best convex combination.

$$k(\mathbf{x}, \mathbf{x}') = \sum_{p=1}^{K} \beta_p k_p(\mathbf{x}, \mathbf{x}').$$

Algorithms

- SDP/QCQP [Lanckriet et al., 2004, Bach et al., 2004]
- SILP (part of shogun toolbox) [Sonnenburg et al., 2006a]
- SimpleMKL [Rakotomamonjy et al., 2008]

fml

Example 1: Human Transcript Start Site Recognition

BIOINFORMATICS

Vol. 22 no. 14 2006, pages e472–e480
doi:10.1093/bioinformatics/btl250

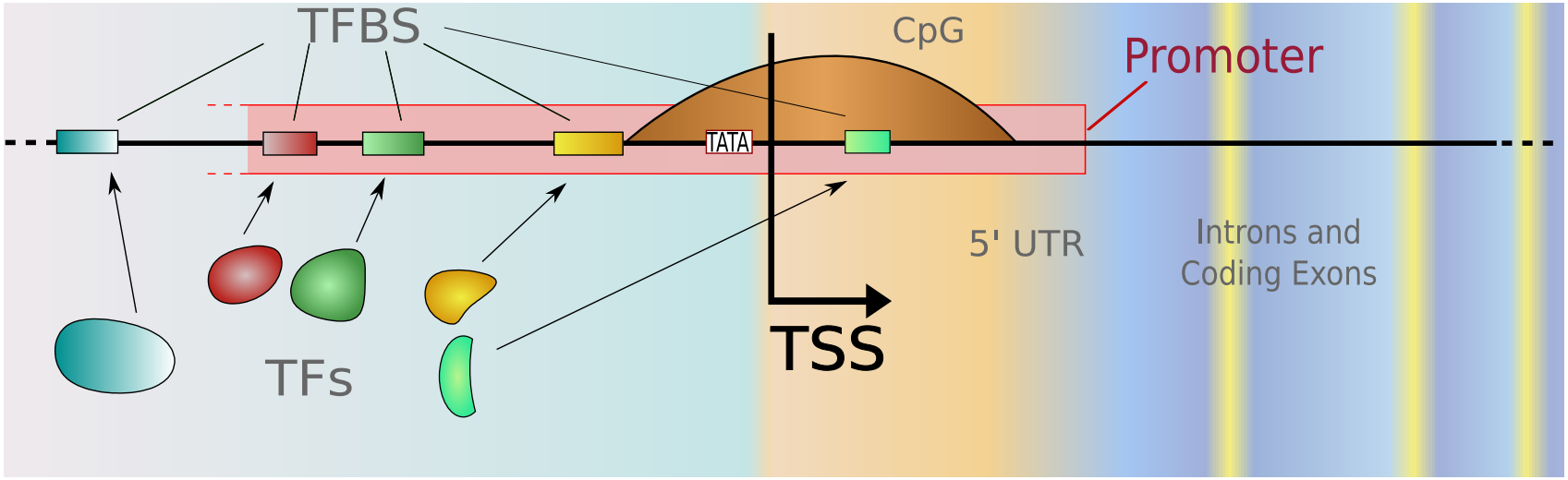
ARTS: accurate recognition of transcription starts in human

Sören Sonnenburg[1], Alexander Zien[2,3] and Gunnar Rätsch[3,*]

[1]Fraunhofer Institute FIRST, Kekuléstr. 7, Berlin, Germany, [2]Max Planck Institute for Biological Cybernetics, Spemannstr. 38, Tübingen, Germany and [3]Friedrich Miescher Laboratory, Max Planck Society, Spemannstr. 39, Tübingen, Germany

[Sonnenburg et al., 2006b]

Detecting Transcription Start Sites (TSS)



- Polymerase binds to rather unspec. region of $\approx [-20, +20]$ bp
- Upstream of TSS: promoter containing transcription factor binding sites
- Downstream of TSS: 5' UTR, and further downstream coding regions and introns (different statistics)
- 3D structure of the promoter must allow the transcription factors to bind

$\Rightarrow$ **Many weak features leading to TSS recognition**

The Five Sub-kernels

1. TSS signal (including parts of core promoter with TATA box)
   – use **Weighted Degree Shift kernel**
2. CpG Islands, distant enhancers and TFBS upstream of TSS
   – use **Spectrum kernel** (large window upstream)
3. Model coding sequence TFBS downstream of TSS
   – use another **Spectrum kernel** (small window downstream)
4. Stacking energy of DNA
   – use *btwist* energy of dinucleotides with **Linear kernel**
5. Twistedness of DNA
   – use *btwist* angle of dinucleotides with **Linear kernel**

Combine weak features to build strong promoter predictor

$$k(\mathbf{x}, \mathbf{x}') = k_{TSS}(\mathbf{x}, \mathbf{x}') + k_{CpG}(\mathbf{x}, \mathbf{x}') + k_{coding}(\mathbf{x}, \mathbf{x}') + k_{energy}(\mathbf{x}, \mathbf{x}') + k_{twist}(\mathbf{x}, \mathbf{x}')$$

The Five Sub-kernels

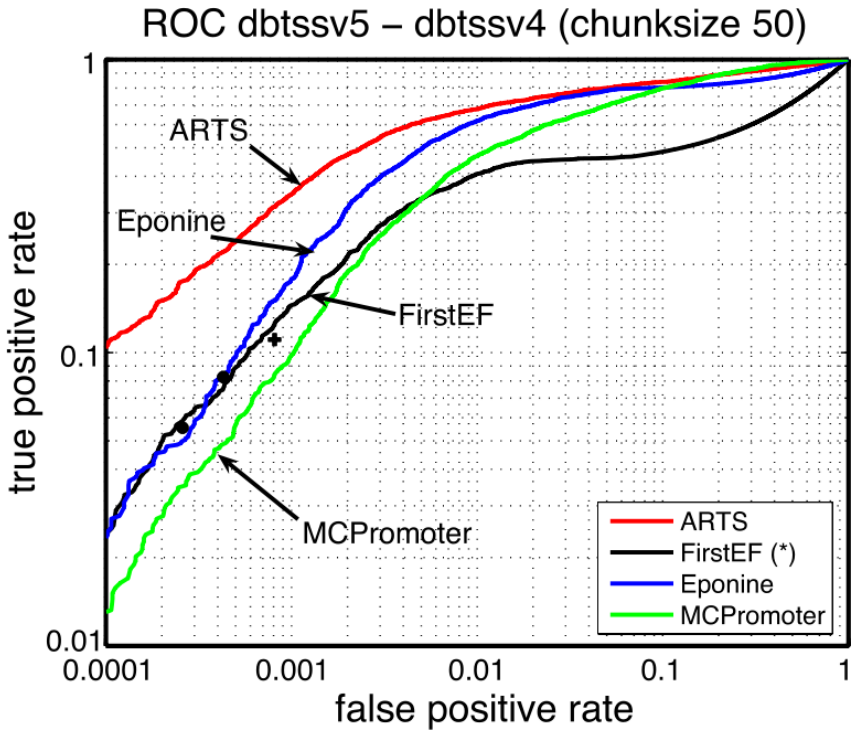
1. TSS signal (including parts of core promoter with TATA box)
   – use **Weighted Degree Shift kernel**
2. CpG Islands, distant enhancers and TFBS upstream of TSS
   – use **Spectrum kernel** (large window upstream)
3. Model coding sequence TFBS downstream of TSS
   – use another **Spectrum kernel** (small window downstream)
4. Stacking energy of DNA
   – use *btwist* energy of dinucleotides with **Linear kernel**
5. Twistedness of DNA
   – use *btwist* angle of dinucleotides with **Linear kernel**

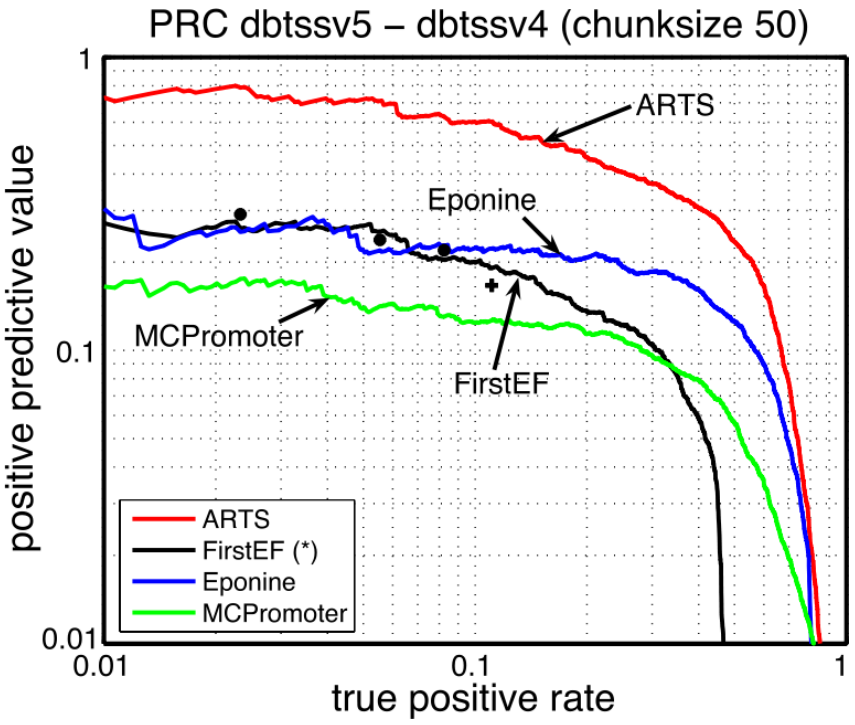
Combine weak features to build strong promoter predictor

$$k(\mathbf{x}, \mathbf{x}') = k_{TSS}(\mathbf{x}, \mathbf{x}') + k_{CpG}(\mathbf{x}, \mathbf{x}') + k_{coding}(\mathbf{x}, \mathbf{x}') + k_{energy}(\mathbf{x}, \mathbf{x}') + k_{twist}(\mathbf{x}, \mathbf{x}')$$

State-of-the-art Performance (in 2006 and still now)

Receiver Operator Characteristic Curve and Precision Recall Curve

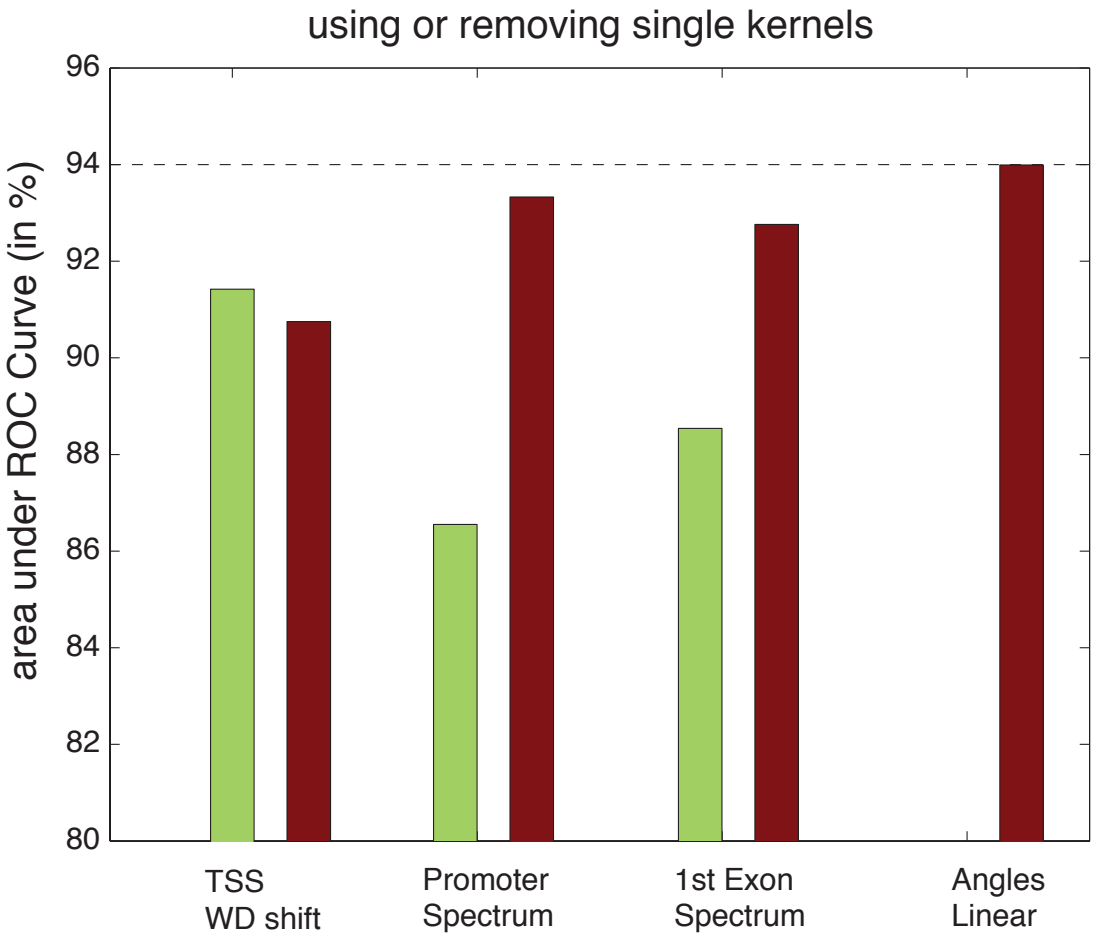




$\Rightarrow$ 35% **true positives at a false positive rate of** 1/1000
(best other method finds about one half (18%**))**

Contributions of the Kernels

Test performance of single kernel (green) and ensemble without the kernel (red).



⇒ **Most important: Weighted Degree Shift kernel modeling the TSS signal**

Example 2: Genome Annotation with Heterogeneous Data

Goal: Predict genes taking advantage of heterogeneous genomic information

Preliminaries:

- Signals (splice sites etc.) can be predicted using SVMs
- How to combine the signal predictions?
- How to integrate, for instance, RNA-seq experimental data to increase accuracy?

More details in my talk tomorrow (11:30am, NGS Data Management).

Example 2: Genome Annotation with Heterogeneous Data

Goal: Predict genes taking advantage of heterogeneous genomic information

Preliminaries:

- Signals (splice sites etc.) can be predicted using SVMs
- How to combine the signal predictions?
- How to integrate, for instance, RNA-seq experimental data to increase accuracy?

More details in my talk tomorrow (11:30am, NGS Data Management).

Example 2: Genome Annotation with Heterogeneous Data

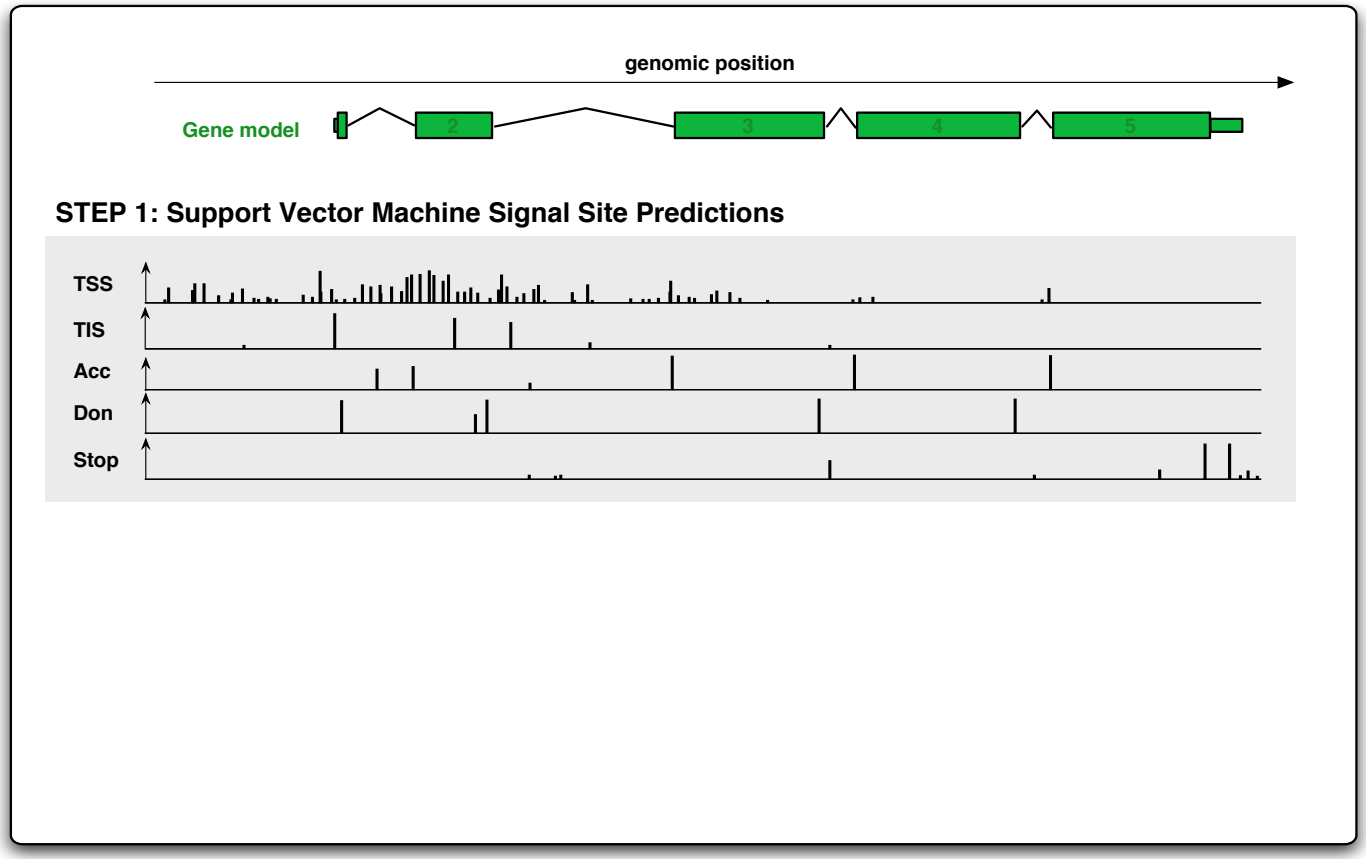
Goal: Predict genes taking advantage of heterogeneous genomic information

Preliminaries:

- Signals (splice sites etc.) can be predicted using SVMs
- How to combine the signal predictions?
- How to integrate, for instance, RNA-seq experimental data to increase accuracy?

More details in my talk tomorrow (11:30am, NGS Data Management).

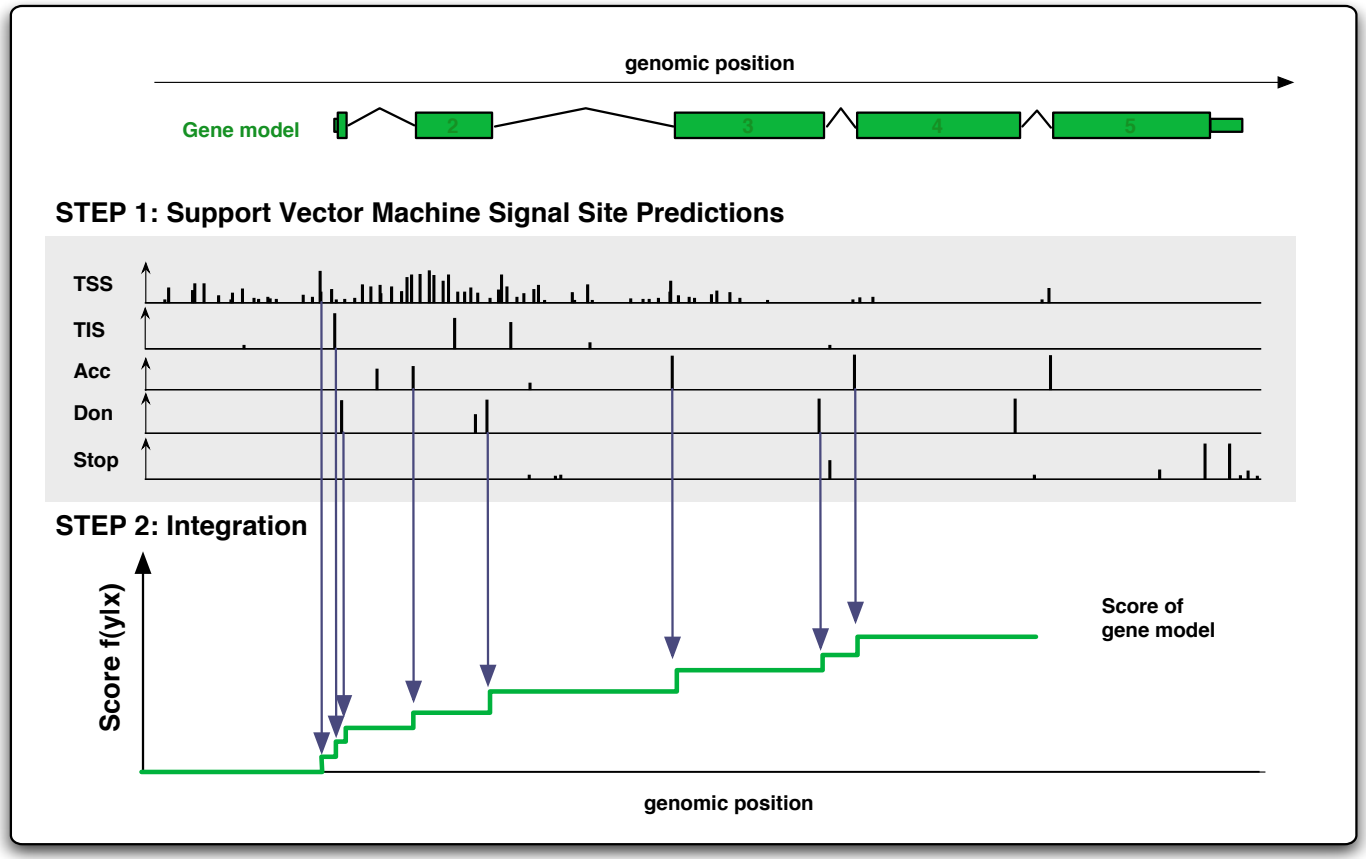
Late Integration of Signal Predictions



- Accurate signal site predictions using SVMs
- SVM-like learning technique performs "late integration"

fml

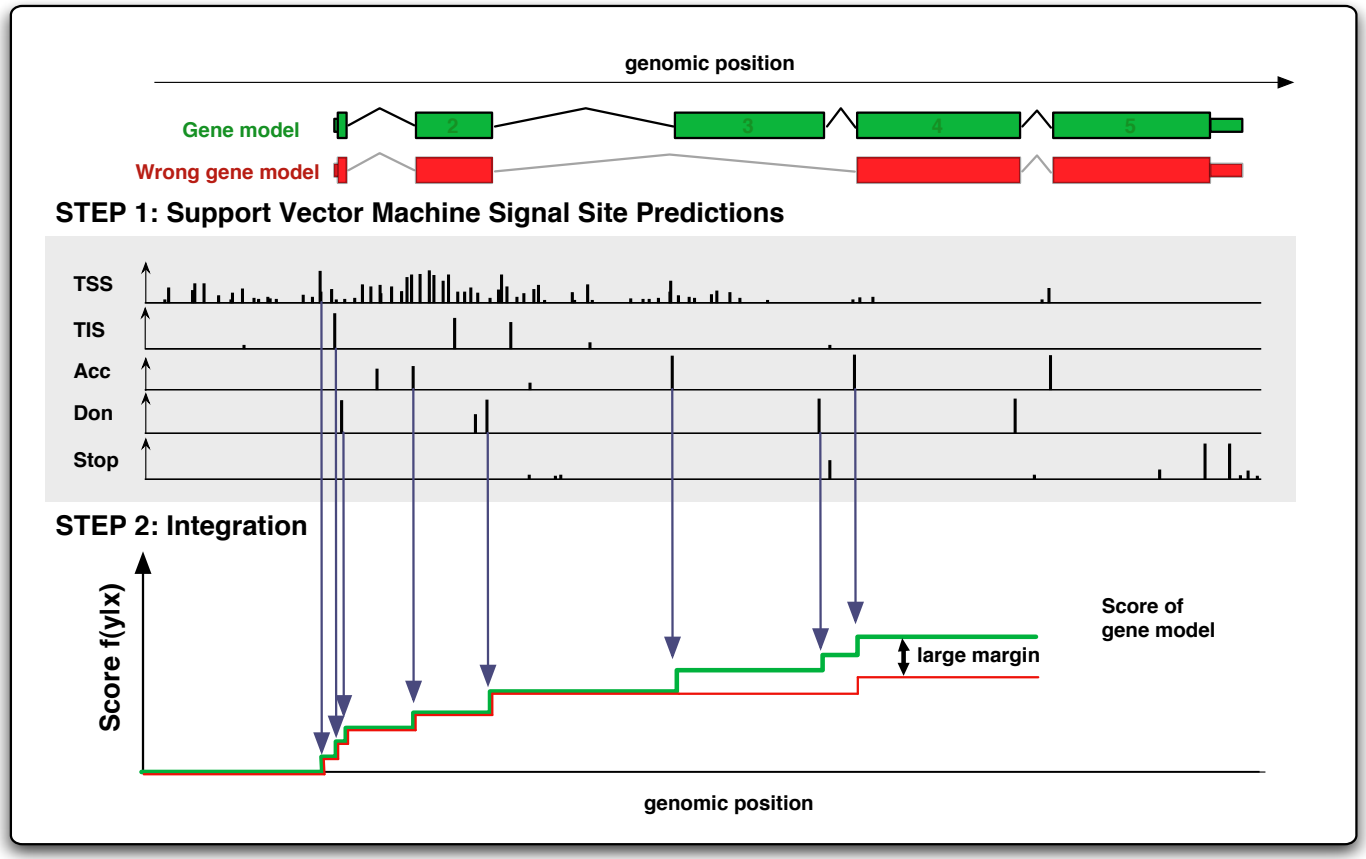
Late Integration of Signal Predictions



- Accurate signal site predictions using SVMs
- SVM-like learning technique performs "late integration"

fml

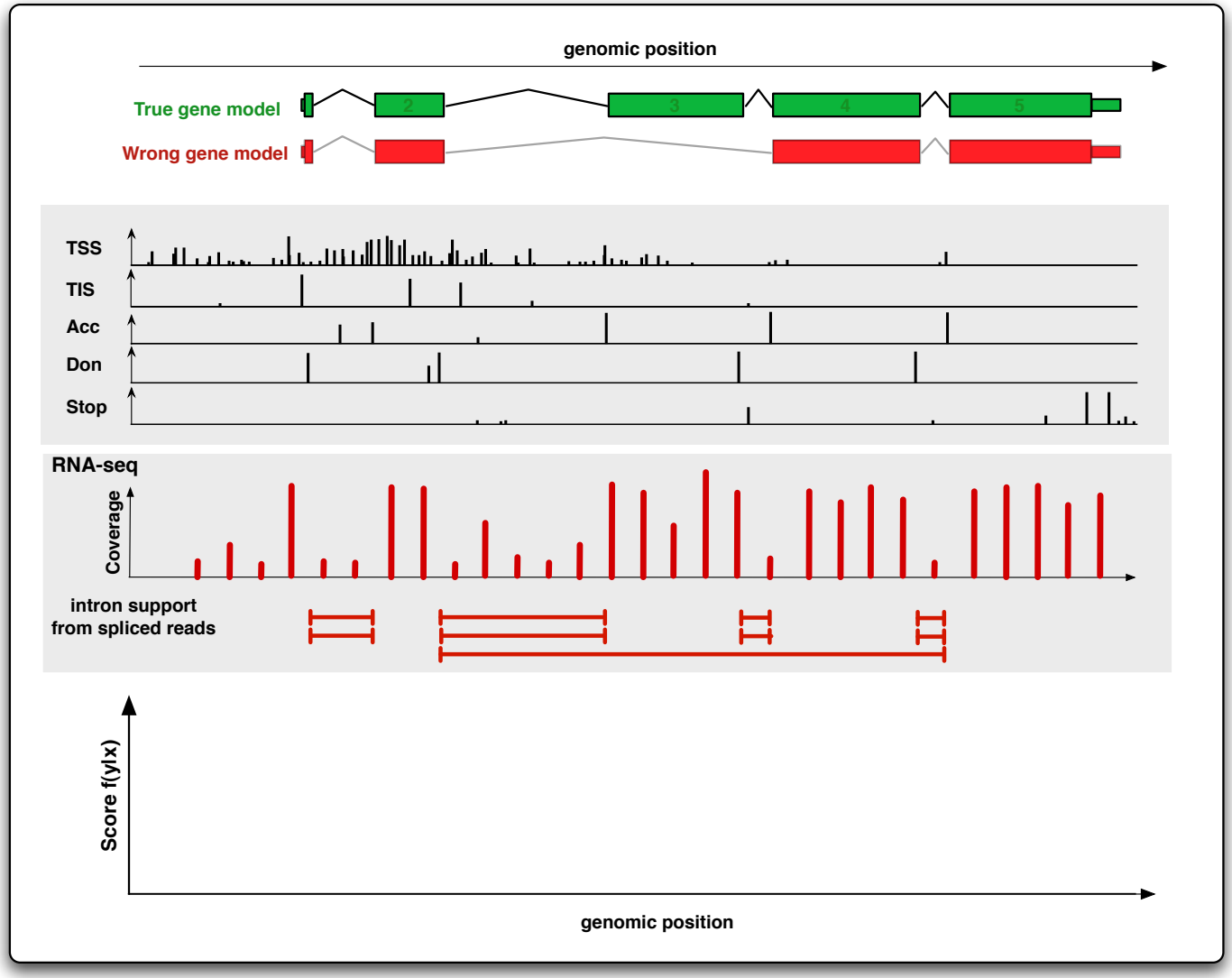
Late Integration of Signal Predictions



- Accurate signal site predictions using SVMs
- SVM-like learning technique performs "late integration"

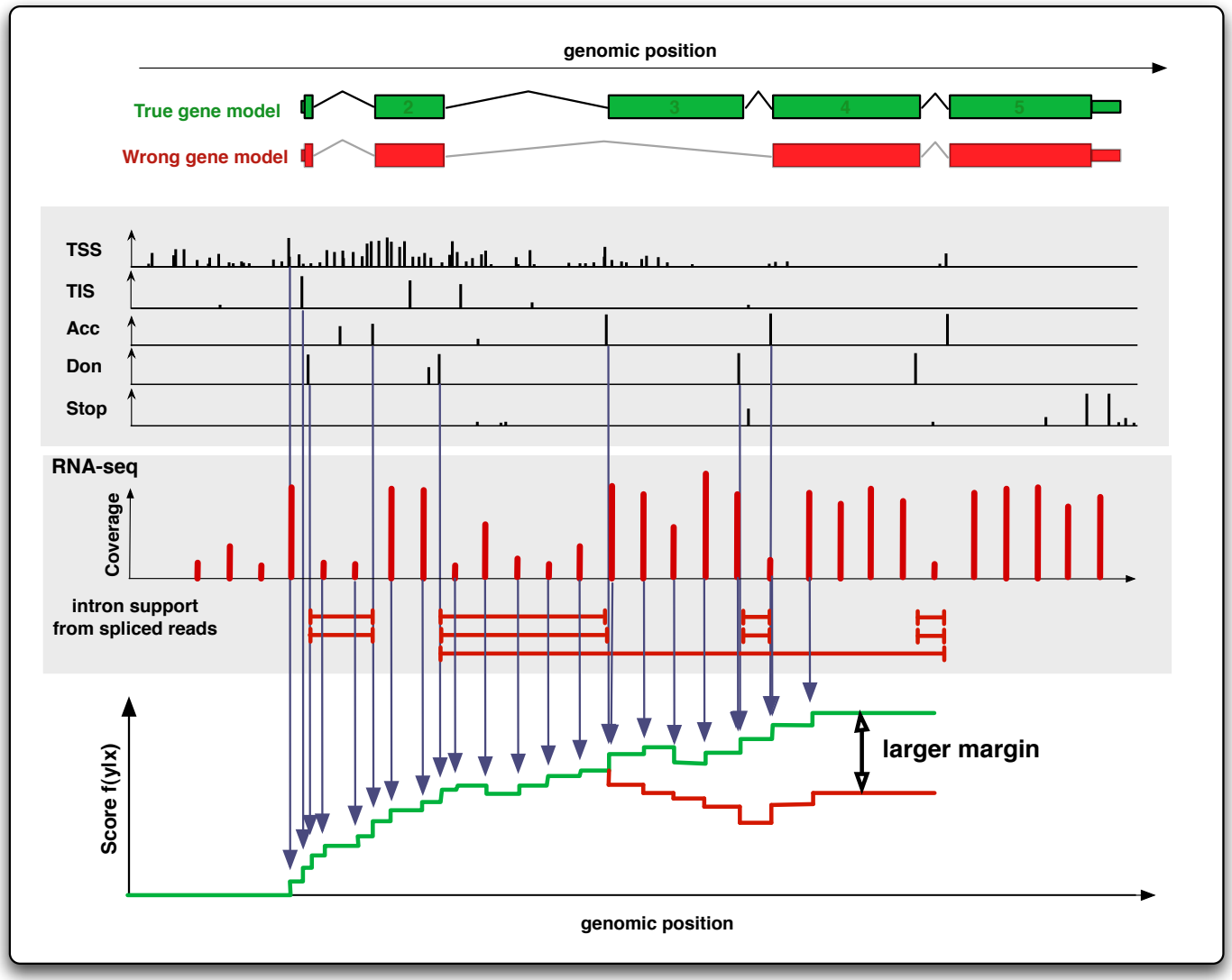
fml

Integration of RNA-Seq Information



- Exploit RNA-Seq coverage and splice junctions

Integration of RNA-Seq Information



- Exploit RNA-Seq coverage and splice junctions

Gene Finding + Heterogeneous Information

Study 1: (*A. thaliana*, [Behr et al., 2009])

	transcript level $(SN + SP)/2$
1. mGene (*ab initio*) ...	**73.3%**
2. ... with DNA methylation (1 tissue)	**76.1%**
3. ... with Nucleosome position predictions	**78.0%**
4. ... with RNA secondary structure predictions	**76.7%**

Study 2: (*C. elegans*, [Behr et al., 2010, in prep.])

	transcript level $(SN + SP)/2$
1. mGene (*ab initio*) ...	**45.0%**
2. ... with mass spectra ...	**45.0%**
3. ... with tiling arrays ...	**45.7%**
4. ... with ESTs (influenced annotation) ...	**56.5%**
5. ... with RNA-seq ...	**55.8%**

fml

Gene Finding + Heterogeneous Information

Study 1: (*A. thaliana*, [Behr et al., 2009])

	transcript level $(SN + SP)/2$
1. mGene (*ab initio*) ...	**73.3%**
2. ...with DNA methylation (1 tissue)	**76.1%**
3. ...with Nucleosome position predictions	**78.0%**
4. ...with RNA secondary structure predictions	**76.7%**

Study 2: (*C. elegans*, [Behr et al., 2010, in prep.])

	transcript level $(SN + SP)/2$
1. mGene (*ab initio*) ...	**45.0%**
2. ...with mass spectra ...	**45.0%**
3. ...with tiling arrays ...	**45.7%**
4. ...with ESTs (influenced annotation) ...	**56.5%**
5. ...with RNA-seq ...	**55.8%**

fml

Gene Finding + Heterogeneous Information

Study 1: (*A. thaliana*, [Behr et al., 2009])

	transcript level $(SN + SP)/2$
1. mGene (*ab initio*) ...	**73.3%**
2. ...with DNA methylation (1 tissue)	**76.1%**
3. ...with Nucleosome position predictions	**78.0%**
4. ...with RNA secondary structure predictions	**76.7%**

Study 2: (*C. elegans*, [Behr et al., 2010, in prep.])

	transcript level $(SN + SP)/2$
1. mGene (*ab initio*) ...	**45.0%**
2. ...with mass spectra ...	**45.0%**
3. ...with tiling arrays ...	**45.7%**
4. ...with ESTs (influenced annotation) ...	**56.5%**
5. ...with RNA-seq ...	**55.8%**

fml

Available SVM Packages

2-Class Classification (35 hits on `http://mloss.org`)
package names sorted by popularity

Multi-Class Classification (7 hits on `http://mloss.org`)

Regression (54 hits on `http://mloss.org`)

More can be found at `http://www.kernel-machines.org`.

Easy-to-use Software

Easysvm an easy-to-use SVM toolbox based on Python and the Shogun toolbox, usable from command line or within Python

PyML an easy-to-use Python-based SVM toolbox, usable from command line or within Python

Shogun toolbox a powerful toolbox for large-scale data analysis, including many SVM implementations with support for Python, R, Matlab, and Octave

LibSVM an SVM library with a graphic interface

SVM-Light an efficient implementation of SVMs in C, usable from command line

Galaxy Web Service a web service for using SVMs, using predefined kernels for real-valued data and string classification (based on Easysvm): `http://galaxy.fml.tuebingen.mpg.de`

fml

Shogun Toolbox

The Shogun toolbox [Sonnenburg et al., 2010] is the result of many years of development and implements most techniques discussed so far.

What can it do?

- Types of problems:
  - Clustering (no labels)
  - **Classification** (binary labels)
  - Regression (real valued labels)
  - Structured Output Learning (structured labels)
- Main focus is on **Support Vector Machines** (SVMs)
- Also implements a number of other ML methods like
  - Hidden Markov Models (HMMs)
  - Linear Discriminant Analysis (LDA)
  - Kernel Perceptrons

http://shogun-toolbox.org

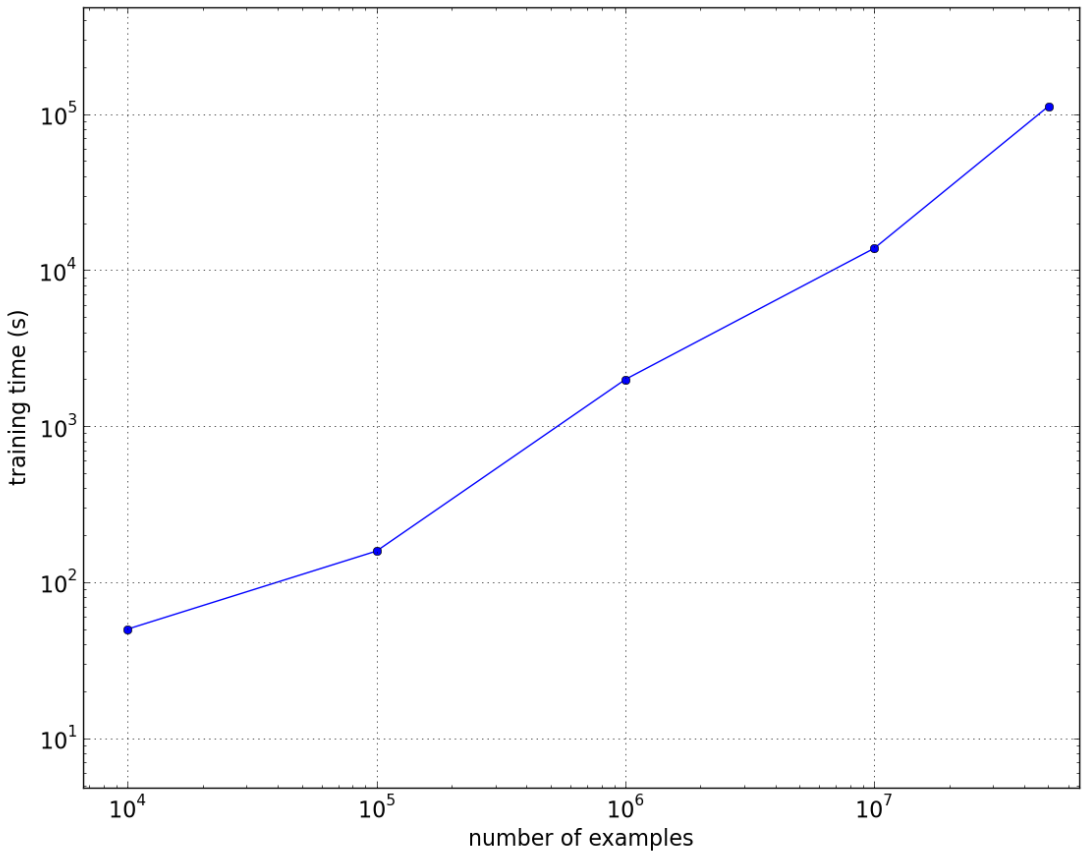
Large-Scale SVM Implementations

- Different SVM solvers employ different strategies
- Provides generic interface to 11 SVM solvers
- Established implementations for solving SVMs with kernels
  - `LibSVM`
  - `SVM`^{light}
- More recent developments: Fast linear SVM solvers
  - `LibLinear`
  - `SvmOCAS` [Franc and Sonnenburg, 2009]
- Support of Multi-Threading

$\Rightarrow$ We have trained SVMs with up to 50 million training examples

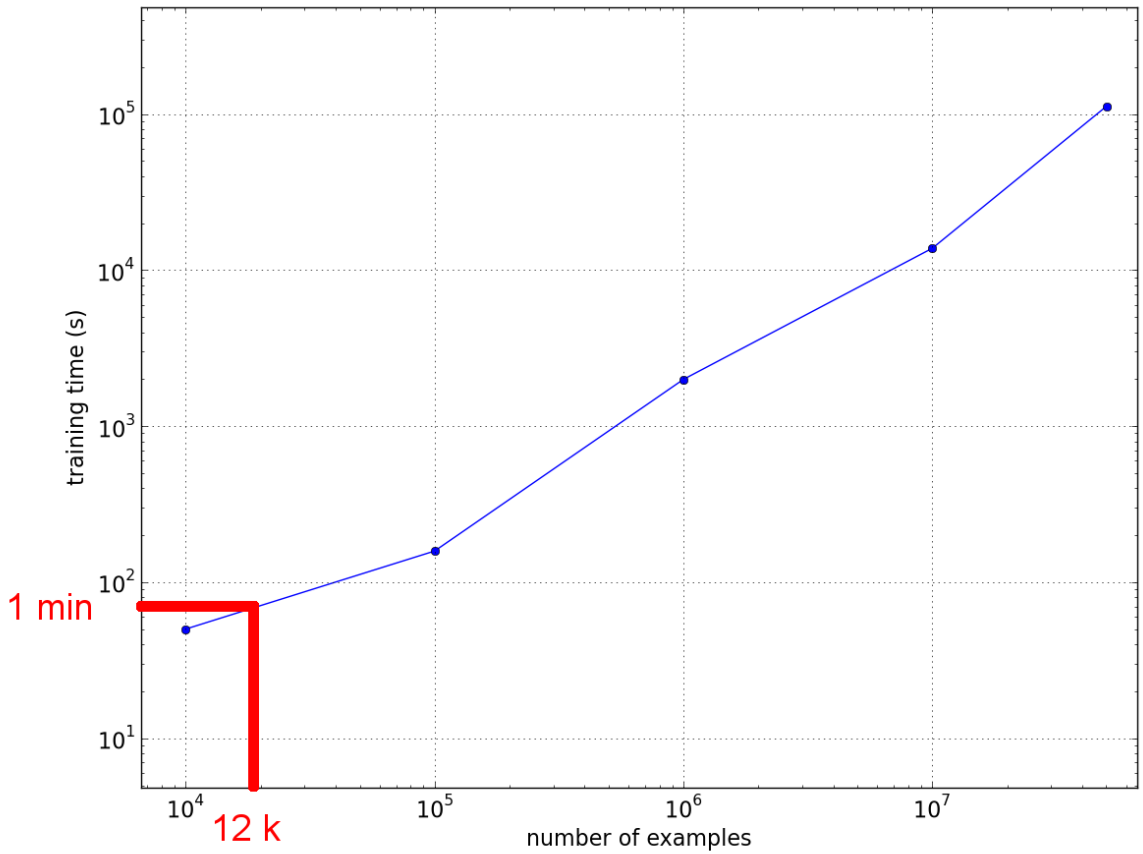
Large Scale Computations

- Training time vs sample size



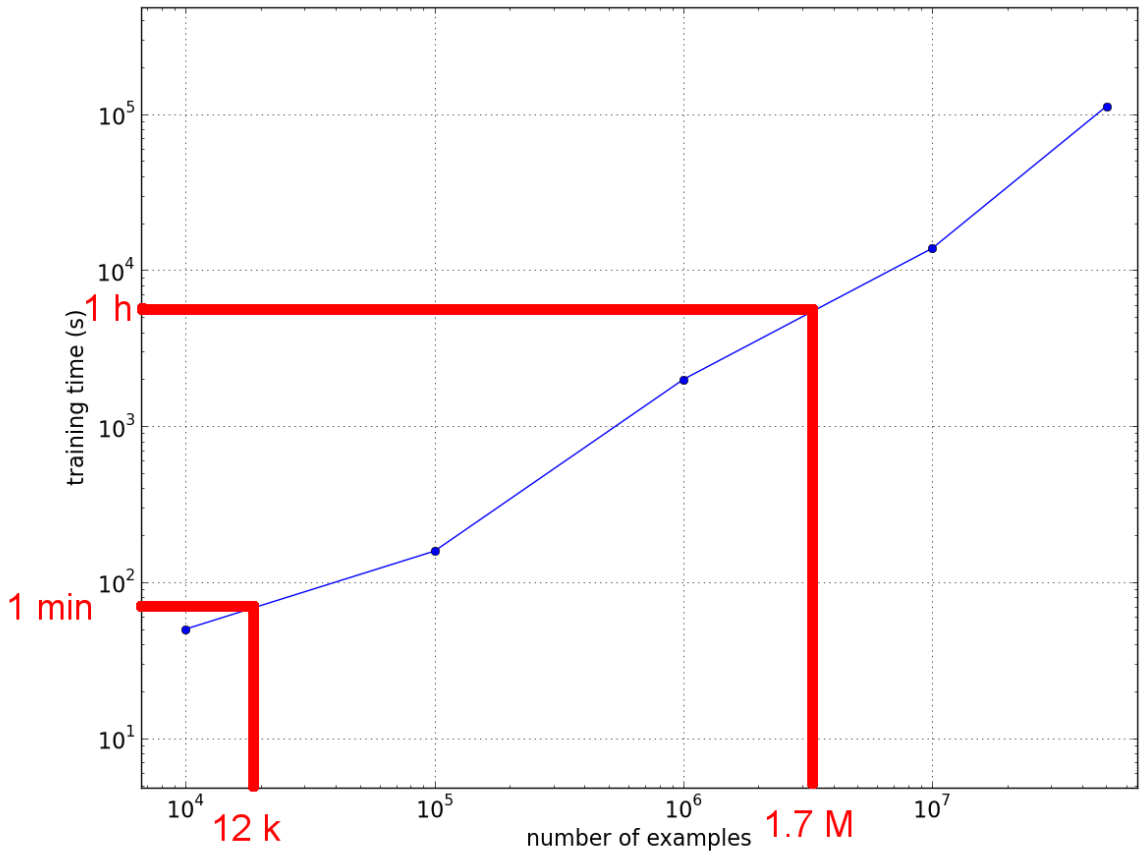
Large Scale Computations

- Training time vs sample size



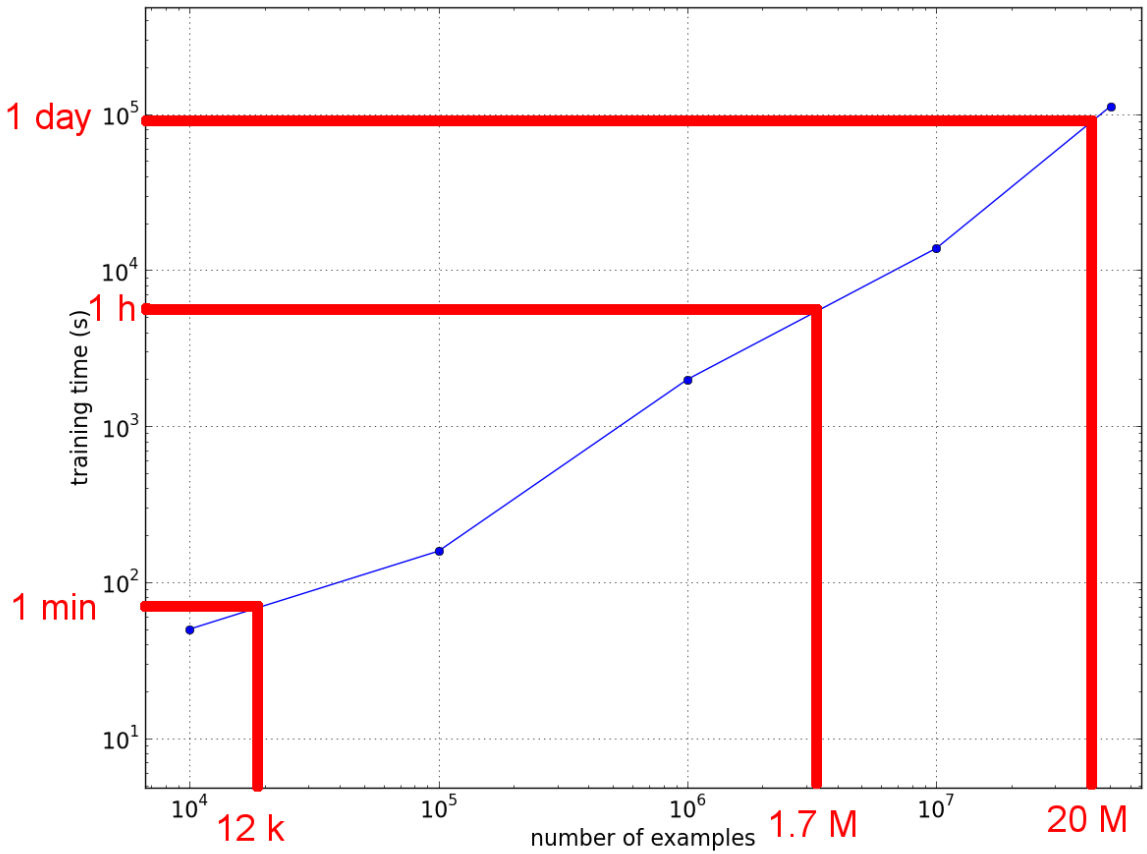
Large Scale Computations

- Training time vs sample size



Large Scale Computations

- Training time vs sample size



Interoperability

- Supports many programming languages
  - Core written in C++ ($> 130,000$ lines of code)
  - Bindings to: R, Python, Matlab, Octave
  - More to come, e.g. Java
- Supports many data formats
  - SVM^{light}, LibSVM, CSV
  - HDF5
- Community
  - Documentation available, many many examples (> 600)
  - Source code is freely available
  - There is a Debian Package, MacOSX
  - Mailing-List, public SVN repository (read-only)
  - $>$ 1000 installations

When is SHOGUN for you?

- You want to work with SVMs (11 solvers to choose from)
- You want to work with Kernels (35 different kernels)
  $\Rightarrow$ Esp.: String Kernels / combinations of Kernels
- You have large scale computations to do (up to 50 million)
- You use one of the following languages:
  R, Python, Octave/MATLAB, C++

Thank you!

For more information, visit:

- Course material `http://tinyurl.com/dfbi2010`
- Demonstration examples
  `http://svmcompbio.tuebingen.mpg.de`
- Shogun `http://www.shogun-toolbox.org`
- Other information about the group
  `http://fml.mpg.de/raetsch`

Acknowledgements:

- Help with slides: Sören Sonnenburg, Cheng Soon Ong, Christian Widmer
- Unpublished gene finding work: Jonas Behr

Thank you for your attention!!

Thank you!

For more information, visit:

- Course material `http://tinyurl.com/dfbi2010`
- Demonstration examples `http://svmcompbio.tuebingen.mpg.de`
- Shogun `http://www.shogun-toolbox.org`
- Other information about the group `http://fml.mpg.de/raetsch`

Acknowledgements:

- Help with slides: Sören Sonnenburg, Cheng Soon Ong, Christian Widmer
- Unpublished gene finding work: Jonas Behr

Thank you for your attention!!

Thank you!

For more information, visit:

- Course material `http://tinyurl.com/dfbi2010`
- Demonstration examples `http://svmcompbio.tuebingen.mpg.de`
- Shogun `http://www.shogun-toolbox.org`
- Other information about the group `http://fml.mpg.de/raetsch`

Acknowledgements:

- Help with slides: Sören Sonnenburg, Cheng Soon Ong, Christian Widmer
- Unpublished gene finding work: Jonas Behr

Thank you for your attention!!

References I

F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

J. Behr, G. Schweikert, J. Cao, F. De Bona, G. Zeller, S. Laubinger, S. Ossowski, K. Schneeberger, D. Weigel, and G. Rätsch. Rna-seq and tiling arrays for improved gene finding. Presented at the CSHL Genome Informatics Meeting, July 2009.

K.M. Borgwardt, C.S. Ong, S. Schonauer, S.V.N. Vishwanathan, A.J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 Suppl 1:i47–56, Jun 2005. ISSN 1367-4803 (Print). doi: 10.1093/bioinformatics/bti1007.

B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20: 273–297, 1995.

References II

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *The Journal of Machine Learning Research*, 10: 2157–2192, 2009.

P.V. Gehler and S. Nowozin. Let the kernel figure it out: Principled learning of pre-processing for kernel classifiers. In *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 06 2009. URL `http://www.cvpr2009.org/`.

D. Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99 - 10, Computer Science Department, UC Santa Cruz, 1999.

T.S. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *J. Comp. Biol.*, 7:95–114, 2000.

G.R.G. Lanckriet, T. De Bie, N. Cristianini, M.I. Jordan, and W.S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 2004.

C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, 2004.

References III

C. Leslie, E. Eskin, and W.S̃. Noble. The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, 2002.

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

L. Liao and W.S. Noble. Combining pairwise sequence similarity and support vector machines. In *Proc. 6th Int. Conf. Computational Molecular Biology*, pages 225–232, 2002.

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

K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *Proc. ECCV'06*, volume 3954 of *Lecture Notes in Computer Science*, page 490, 2006.

References IV

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

L. Ralaivola, S. J Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Netw*, 18(8):1093–1110, Oct 2005. ISSN 0893-6080 (Print). doi: 10.1016/j.neunet.2005.07.009.

G. Rätsch and S. Sonnenburg. Accurate splice site detection for *Caenorhabditis elegans*. In K. Tsuda B. Schoelkopf and J.-P. Vert, editors, *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.

B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

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`.

References V

S. Sonnenburg, G. Rätsch, and K. Rieck. Large scale learning with string kernels. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press, 2007.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, 7:1531–1565, July 2006a.

Sören Sonnenburg, Alexander Zien, and Gunnar Rätsch. ARTS: Accurate Recognition of Transcription Starts in Human. *Bioinformatics*, 22(14): e472–480, 2006b.

Sören Sonnenburg, Gunnar Rätsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio de Bona, Alexander Binder, Christian Gehl, and Vojtech Franc. The SHOGUN machine learning toolbox. *Journal of Machine Learning Research*, 2010. URL `http://www.shogun-toolbox.org`. (accepted).

K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.R. Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14: 2397–2414, 2002.

References VI

V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.

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

A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites. *BioInformatics*, 16(9):799–807, September 2000.

Evaluation Measures for Classification I

The Contingency Table/Confusion Matrix

TP, FP, FN, TN are absolute counts of true positives, false positives, false negatives and true negatives

- N - sample size
- $\mathrm{N}^+ = FN + TP$ number of positive examples
- $\mathrm{N}^- = FP + TN$ number of negative examples
- $\mathrm{O}^+ = TP + FP$ number of positive predictions
- $\mathrm{O}^- = FN + TN$ number of negative predictions

outputs\ labeling	$y = +1$	$y = -1$	Σ
$f(x) = +1$	TP	FP	O^+
$f(x) = -1$	FN	TN	O^-
Σ	N^+	N^-	N

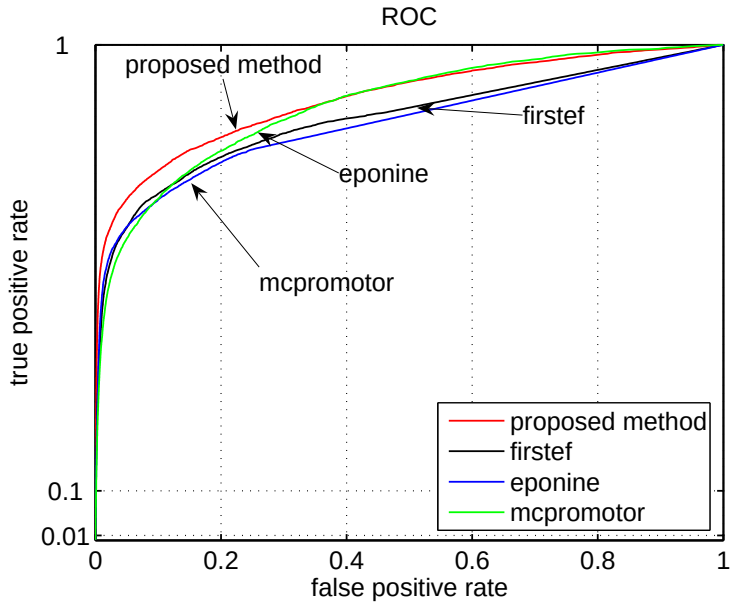
Evaluation Measures for Classification II

Several commonly used performance measures

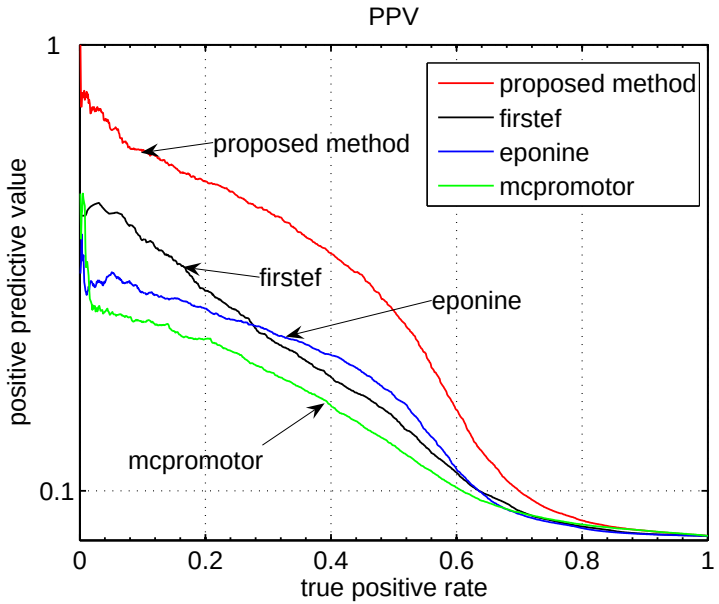
Name	Computation
Accuracy	$\mathrm{ACC} = \frac{TP+TN}{N}$
Error rate (1-accuracy)	$\mathrm{ERR} = \frac{FP+FN}{N}$
Balanced error rate	$\mathrm{BER} = \frac{1}{2}\left(\frac{FN}{FN+TP} + \frac{FP}{FP+TN}\right)$
Weighted relative accuracy	$\mathrm{WRACC} = \frac{TP}{TP+FN} - \frac{FP}{FP+TN}$
F1 score	$\mathrm{F1} = \frac{2*TP}{2*TP+FP+FN}$
Cross-correlation coefficient	$\mathrm{CC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
Sensitivity/recall	$\mathrm{TPR} = TP/N^+ = \frac{TP}{TP+FN}$
Specificity	$\mathrm{TNR} = TN/N^- = \frac{TN}{TN+FP}$
1-sensitivity	$\mathrm{FNR} = FN/N^+ = \frac{FN}{FN+TP}$
1-specificity	$\mathrm{FPR} = FP/N^- = \frac{FP}{FP+TN}$
P.p.v. / precision	$\mathrm{PPV} = TP/O^+ = \frac{TP}{TP+FP}$
False discovery rate	$\mathrm{FDR} = FP/O^+ = \frac{FP}{FP+TP}$

Evaluation Measures for Classification III

[left] Receiver Operating Characteristic (ROC) Curve



[right] Precision Recall Curve



(Obtained by varying bias and recording TPR/FPR or PPV/TPR.)

Use bias-independent scalar evaluation measure

- Area under ROC Curve (auROC)
- Area under Precision Recall Curve (auPRC)

Comparison of String Kernels

Kernel	$l_{\mathbf{x}} \neq l_{\mathbf{x}'}$	$Pr(x\|\boldsymbol{\theta})$	Positional?	Scope	Complexity
linear	no	no	yes	local	$\mathcal{O}(l_{\mathbf{x}})$
polynomial	no	no	yes	global	$\mathcal{O}(l_{\mathbf{x}})$
locality-improved	no	no	yes	local/global	$\mathcal{O}(l \cdot l_{\mathbf{x}})$
sub-sequence	yes	no	yes	global	$\mathcal{O}(n l_{\mathbf{x}} l_{\mathbf{x}'})$
n-gram/Spectrum	yes	no	no	global	$\mathcal{O}(l_{\mathbf{x}})$
WD	no	no	yes	local	$\mathcal{O}(l_{\mathbf{x}})$
WD with *shifts*	no	no	yes	local/global	$\mathcal{O}(s \cdot l_{\mathbf{x}})$
Oligo	yes	no	yes	local/global	$\mathcal{O}(l_{\mathbf{x}} l_{\mathbf{x}'})$
TOP	yes/no	yes	yes/no	local/global	depends
Fisher	yes/no	yes	yes/no	local/global	depends