Regularization Methods for Sequence Analysis

Gunnar Rätsch

Friedrich Miescher Laboratory, Max Planck Society
Tübingen, Germany

Workshop “Regularisierte Schätzung zur Modellierung molekularer Daten”
GMDS Jahrestagung in Mannheim

September 6, 2010
Group’s Research Topics

1. Statistical inference methods for structured data
   ⇒ Develop fast and accurate learning methods

2. Convergence properties of iterative algorithms
   ⇒ Boosting-like algorithms and semi-infinite LPs

3. Genome annotation
   ⇒ Predict features encoded on DNA

4. Biological networks
   ⇒ Understand interactions between gene products

5. Analysis of polymorphisms
   ⇒ Discover polymorphisms and associate with phenotypes
Group’s Research Topics

1. Statistical inference methods for structured data
   ⇒ Develop fast and accurate learning methods

2. Convergence properties of iterative algorithms
   ⇒ Boosting-like algorithms and semi-infinite LPs

3. Genome annotation
   ⇒ Predict features encoded on DNA

4. Biological networks
   ⇒ Understand interactions between gene products

5. Analysis of polymorphisms
   ⇒ Discover polymorphisms and associate with phenotypes
Learning about the Transcriptome

Machine Learning View
- How to learn to predict what these processes accomplish?
- How well can we predict it from the available information?

Biological View
- What can we not predict yet? What is missing?
- Can we derive a deeper understanding of these processes?
Learning about the Transcriptome

Machine Learning View

- How to learn to predict what these processes accomplish?
- How well can we predict it from the available information?

Biological View

- What can we not predict yet? What is missing?
- Can we derive a deeper understanding of these processes?
Learning about the Transcriptome

Machine Learning View

- How to learn to predict what these processes accomplish?
- How well can we predict it from the available information?

Biological View

1. What can we not predict yet? What is missing?
2. Can we derive a deeper understanding of these processes?
Learning about the Transcriptome

Machine Learning View
- How to learn to predict what these processes accomplish?
- How well can we predict it from the available information?

Biological View
1. What can we not predict yet? What is missing?
2. Can we derive a deeper understanding of these processes?
Simplest formulation:

- Given a DNA sequence \( x \in \{ 'A', 'C', 'G', 'T' \}^L \)
- Find the correct label sequence \( y = y_1 y_2 \ldots y_L \)  
  \( (y_i \in Y = \{ \text{"intergenic"}, \text{"5' UTR"}, \text{"coding"}, \text{"intron"}, \ldots \} ) \)
Traditionally HMMs, here: **Max-Margin Structured Output Learning**

Learn function $f(y|x)$ scoring segmentations $y$ for $x$
Maximize $f(y|x)$ w.r.t. $y$ for prediction:

$$\arg\max_{y \in \Upsilon^*} f(y|x)$$

Idea: $f(y|x) \gg f(\hat{y}|x)$ for wrong labels $\hat{y} \neq y$
Max-margin learning: [Altun et al., 2003, Rätsch and Sonnenburg, 2007]

Given $N$ sequence pairs $(x_1, y_1), \ldots, (x_N, y_N)$ for training
Solve optimization problem:

$$\max_{\rho, f \text{ subject to}} \rho$$

w.r.t. $f(y_n|x_n) - f(y|x_n) \geq \rho - \xi_n$ for all $y \neq y_n \in \Upsilon^*, n = 1, \ldots, N$
Discriminative Segmentation Learning

- Traditionally HMMs, here: **Max-Margin Structured Output Learning**
- Learn function $f(y|x)$ scoring segmentations $y$ for $x$
- Maximize $f(y|x)$ w.r.t. $y$ for prediction:
  \[
  \arg\max_{y \in \mathcal{Y}^*} f(y|x)
  \]

- **Idea**: $f(y|x) \gg f(\hat{y}|x)$ for wrong labels $\hat{y} \neq y$
- Max-margin learning: [Altun et al., 2003, Rätsch and Sonnenburg, 2007]
  - Given $N$ sequence pairs $(x_1, y_1), \ldots, (x_N, y_N)$ for training
  - Solve optimization problem:
  \[
  \max_{\rho, f, \xi \in \mathbb{R}_+^N} \rho - C \sum_{n=1}^N \xi_n \\
  \text{w.r.t. } f(y_n|x_n) - f(y|x_n) \geq \rho - \xi_n \quad \text{for all } y \neq y_n \in \mathcal{Y}^*, n = 1, \ldots, N
  \]
Discriminative Segmentation Learning

- Traditionally HMMs, here: **Max-Margin Structured Output Learning**
- Learn function $f(y|x)$ scoring segmentations $y$ for $x$
- Maximize $f(y|x)$ w.r.t. $y$ for prediction:
  \[
  \text{argmax}_{y \in \Upsilon^*} f(y|x)
  \]
- Idea: $f(y|x) \gg f(\hat{y}|x)$ for wrong labels $\hat{y} \neq y$
- Max-margin learning: [Altun et al., 2003, Rätsch and Sonnenburg, 2007]
  - Given $N$ sequence pairs $(x_1, y_1), \ldots, (x_N, y_N)$ for training
  - Solve optimization problem:
  \[
  \max_{\rho, f, \xi \in \mathbb{R}_+^N} \rho - C \sum_{n=1}^{N} \xi_n \\
  \text{w.r.t. } f(y_n|x_n) - f(y|x_n) \geq \rho - \xi_n
  \]
  for all $y \neq y_n \in \Upsilon^*$, $n = 1, \ldots, N$
Discriminative Segmentation Learning

- Traditionally HMMs, here: Max-Margin Structured Output Learning
- Learn function \( f(y|x) \) scoring segmentations \( y \) for \( x \)
- Maximize \( f(y|x) \) w.r.t. \( y \) for prediction:

\[
\arg\max_{y \in \Upsilon^*} f(y|x)
\]

- Idea: \( f(y|x) \gg f(\hat{y}|x) \) for wrong labels \( \hat{y} \neq y \)

Max-margin learning: [Altun et al., 2003, Rätsch and Sonnenburg, 2007]

- Given \( N \) sequence pairs \((x_1, y_1), \ldots, (x_N, y_N)\) for training
- Solve optimization problem:

\[
\max_{\rho, f, \xi \in \mathbb{R}_+^N} \quad \rho - C \sum_{n=1}^{N} \xi_n
\]

w.r.t. \( f(y_n|x_n) - f(y|x_n) \geq \rho - \xi_n \)

for all \( y \neq y_n \in \Upsilon^* \), \( n = 1, \ldots, N \)
Optimization Strategy \( f(y|x) = \langle w, \Phi(x, y) \rangle \)

\[
\min_{w, \xi \geq 0} \|w\|_2^2 + C \sum_{n=1}^{N} \xi_n
\]

w.r.t. \( \langle w, \Phi(x, y_n) - \Phi(x, y) \rangle \geq 1 - \xi_n \)

for all \( y_n \neq y \in \Upsilon^*, n = 1, \ldots, N \)

- Big: one constraint per example and wrong labeling
- Iterative solution (a.k.a. “column generation”, “cutting planes”)
  - Begin with small set of wrong labellings
  - Solve reduced optimization problem
  - Find labellings that violate constraints
  - Add constraints, resolve

**Iteration bounds:**

- \( \| \cdot \|_2: R/\epsilon^2 \), see [Tsochantaridis et al., 2005]
- \( \| \cdot \|_1: \log(N)/\epsilon^2 \) [Rätsch and Warmuth, 2005], \( O(1/\epsilon) \) [Warmuth et al., 2009]
Optimization may require many iterations

Number of constraints increases linearly

- When using kernels, solving optimization problems can become easily infeasible (number of variables also increases)

\[ w = \sum_{n=1}^{N} \sum_{y \in \mathcal{Y}^*} \alpha_{n,y} \Phi(x_n, y) \]

- Approximation algorithms needed (or better optimization alg.)

- Idea: Decompose problem
  - First part uses kernels, can be precomputed
  - Second part without kernels and only combines ingredients

⇒ Solve problems with 10,000 examples, instead of just \( \leq 500 \)
Problems

- Optimization may require many iterations
- Number of constraints increases linearly
- When using kernels, solving optimization problems can become easily infeasible (number of variables also increases)

\[ w = \sum_{n=1}^{N} \sum_{y \in \Upsilon^*} \alpha_{n,y} \Phi(x_n, y) \]

- Approximation algorithms needed (or better optimization alg.)
- Idea: Decompose problem
  - First part uses kernels, can be precomputed
  - Second part without kernels and only combines ingredients

⇒ Solve problems with 10,000 examples, instead of just ≤ 500
Optimization may require many iterations
Number of constraints increases linearly
When using kernels, solving optimization problems can become easily infeasible (number of variables also increases)

\[ w = \sum_{n=1}^{N} \sum_{y \in \Upsilon^*} \alpha_{n,y} \Phi(x_n, y) \]

Approximation algorithms needed (or better optimization alg.)
Idea: Decompose problem
  - First part uses kernels, can be precomputed
  - Second part without kernels and only combines ingredients

⇒ Solve problems with 10,000 examples, instead of just \( \leq 500 \)
Problems

- Optimization may require many iterations
- Number of constraints increases linearly
- When using kernels, solving optimization problems can become easily infeasible (number of variables also increases)

\[ w = \sum_{n=1}^{N} \sum_{y \in \mathcal{Y}^*} \alpha_{n,y} \Phi(x_n, y) \]

- Approximation algorithms needed (or better optimization alg.)
- Idea: Decompose problem
  - First part uses kernels, can be precomputed
  - Second part without kernels and only combines ingredients

⇒ Solve problems with 10,000 examples, instead of just ≤ 500
Given a piece of DNA sequence
Predict gene products including the intermediate processing steps
• Predict signals used during processing
- Predict signals used during processing
Predict signals used during processing

Predict the correct corresponding label sequence with labels “intergenic”, “exon”, “intron”, “5’ UTR”, etc.
Example: Splice Site Recognition

True Splice Sites
Example: Splice Site Recognition

True Splice Sites

CT...GTCGTA...GAAGCTAGGAGCGC...ACGCGT...GA

≈ 150 nucleotides window around dimer
Example: Splice Site Recognition

Potential Splice Sites

\[
\text{CT...GTCGTA...GAAGCTAGGAGCGC...ACGCGT...GA}
\]

\[\approx 150 \text{ nucleotides window around dimer}\]
Example: Splice Site Recognition

Potential Splice Sites

≈ 150 nucleotides window around dimer

CT...GTCGTA...GAAGCTAGGAGCGC...ACGCGT...GA

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

⇒ Millions of examples from EST databases
Example: Splice Site Recognition

Potential Splice Sites

CT...GTCGTA...GAAGCTAGGAGCGC...ACGCGT...GA

≈ 150 nucleotides window around dimer

Basic idea:

For instance, exploit that exons have higher GC content
or
that specific motifs appear near splice sites.

[Sonnenburg et al., 2007b]
Example: Splice Site Recognition

Potential Splice Sites

CT...GTCGTA...GAAGCTAGGAGCGC...ACGCGT...GA
≈ 150 nucleotides window around dimer

Basic idea:

In practice: Use one feature per possible substring (e.g. ≤20) at all positions

\[150 \cdot (4^1 + \ldots + 4^{20}) \approx 2 \cdot 10^{14}\] features

- Needs efficient algorithms
- Leads to most accurate predictors that currently exist
Example: Splice Site Recognition

Potential Splice Sites

CT...GTCGTA...GAAGCTAGGAGCGC...ACGCGT...GA

≈ 150 nucleotides window around dimer

Basic idea:

In practice: Use one feature per possible substring (e.g. \( \leq 20 \)) at all positions

\[ 150 \cdot (4^1 + \ldots + 4^{20}) \approx 2 \cdot 10^{14} \text{ features} \]

- Needs efficient algorithms
- Leads to most accurate predictors that currently exist
Kernels and String Data Structures

Luckily, with $\ell_2$ regularization, we can use kernels that \textit{implicitly} consider features:

$$k(s_1, s_2) = w_7 + w_1 + w_2 + w_2 + w_3$$

\text{[Rätsch et al., 2007, Rätsch and Sonnenburg, 2004]}

\textbf{Drawback:} Training too expensive for millions of examples

Needs more sophisticated algorithms:

- Exploit that features are sparse and can be explicitly represented
- String indexing data structures \text{[Sonnenburg et al., 2007a]}
- Use novel optimization techniques \text{[Franc and Sonnenburg, 2009]}

Implemented in a freely available software package

www.shogun-toolbox.org \text{[Sonnenburg et al., 2010]}
Luckily, with $\ell_2$ regularization, we can use kernels that *implicitly* consider features:

$$k(s_1, s_2) = w_7 + w_1 + w_2 + w_2 + w_3$$

**Drawback:** Training is too expensive for millions of examples

Needs more sophisticated algorithms:
- Exploit that features are sparse and can be explicitly represented
- String indexing data structures
- Use novel optimization techniques
- Use subgradient-based optimization

Implemented in a freely available software package: [www.shogun-toolbox.org](http://www.shogun-toolbox.org)
Luckily, with $\ell_2$ regularization, we can use kernels that *implicitly* consider features:

$$k(s_1, s_2) = w_7 + w_1 + w_2 + w_2 + w_3$$

**Drawback:** Training was too expensive for millions of examples

Needs more sophisticated algorithms:
- Exploit that features are sparse and can be explicitly represented
- String indexing data structures
- Use novel optimization techniques
- Use subgradient-based optimization

Implemented in a freely available software package

www.shogun-toolbox.org

[Rätsch et al., 2007, Rätsch and Sonnenburg, 2004]
Results on Splice Site Recognition

<table>
<thead>
<tr>
<th></th>
<th>Worm</th>
<th>Fly</th>
<th>Cress</th>
<th>Fish</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>Don</td>
<td>Acc</td>
<td>Don</td>
<td>Acc</td>
</tr>
<tr>
<td>Markov Chain</td>
<td>92.1</td>
<td>90.0</td>
<td>80.3</td>
<td>78.5</td>
<td>87.4</td>
</tr>
<tr>
<td>auPRC(%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>95.9</td>
<td>95.3</td>
<td>86.7</td>
<td>87.5</td>
<td>92.2</td>
</tr>
<tr>
<td>auPRC(%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Sonnenburg, Schweikert, Philips, Behr, Rätsch, 2007]
Performance (=Area under PRC curve) keeps increasing.

- Pays off to use all available data!
- Regularizer less important for accuracy when data abundant
- Efficient optimization for quadratic regularizer

[Sonnenburg et al., 2007b]
Performance (=Area under PRC curve) keeps increasing.

- Pays off to use all available data!
- Regularizer less important for accuracy when data abundant
- Efficient optimization for quadratic regularizer

[Sonnenburg et al., 2007b]
Performance (=Area under PRC curve) keeps increasing.

- Pays off to use all available data!
- Regularizer less important for accuracy when data abundant
- Efficient optimization for quadratic regularizer

[Sonnenburg et al., 2007b]
Example: Predictions in UCSC Browser

[Rätsch et al., 2007, Schweikert et al., 2009c]
Example: Predictions in UCSC Browser

[Rätsch et al., 2007, Schweikert et al., 2009c]
Example: Predictions in UCSC Browser

Signals have to appear in the right order

Based on known genes, learn how to combine predictions for accurate gene prediction.

⇒ Prediction of “Structured Outputs”

[Rätsch et al., 2007, Schweikert et al., 2009c]
Simplified Model: Score for splice form \( y = \{(p_j, q_j)\}_{j=1}^{J} \):

\[
F(y) := \sum_{j=1}^{J-1} S_{GT}(f_{GT}^j) + \sum_{j=2}^{J} S_{AG}(f_{AG}^j) + \sum_{j=1}^{J-1} S_{Li}(p_{j+1} - q_j) + \sum_{j=1}^{J} S_{LE}(q_j - p_j)
\]

Splice signals \quad Segment lengths

Tune free parameters by solving structured output problem.
Simplified Model: Score for splice form $y = \{(p_j, q_j)\}_{j=1}^{J}$:

$$F(y) := \sum_{j=1}^{J-1} S_{GT}(f^G_j) + \sum_{j=2}^{J} S_{AG}(f^A_j) + \sum_{j=1}^{J-1} S_{Li}(p_{j+1} - q_j) + \sum_{j=1}^{J} S_{LE}(q_j - p_j)$$

- Splice signals
- Segment lengths

Tune free parameters by solving **structured output problem**.
Problem-specific Regularizers

- Given the structure of the problem, we can assume that functions transforming signal predictions are monotonic, i.e.
  \[ f(x) \leq f(x'), \text{ if } x \leq x' \]

- Functions should be smooth, e.g. small

\[
\int_{x_{\text{min}}}^{x_{\text{max}}} f'(x)^2 \, dx \approx \sum_{i=1}^{N-1} \left( \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \right)^2
\]

- We use piece-wise linear functions (PLiFs)
  - Can be linearly parametrized by \( f(x) = \langle w, \phi(x) \rangle \)
  - Constraints can be easily implemented by \( w_i \leq w_{i+1} \)
  - Regularization term:

\[
\sum_{i=1}^{N-1} (w_{i+1} - w_i)^2
\]
Given the structure of the problem, we can assume that functions transforming signal predictions are monotonic, i.e.
\[ f(x) \leq f(x'), \text{ if } x \leq x' \]

functions should be smooth, e.g. small
\[
\int_{x_{\min}}^{x_{\max}} f'(x)^2 dx \approx \sum_{i=1}^{N-1} \left( \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \right)^2
\]

We use piece-wise linear functions (PLiFs)
- Can be linearly parametrized by \( f(x) = \langle w, \phi(x) \rangle \)
- Constraints can be easily implemented by \( w_i \leq w_{i+1} \)
- Regularization term:
\[
\sum_{i=1}^{N-1} (w_{i+1} - w_i)^2
\]
Results using mGene

- Most accurate ab initio method in the nGASP genome annotation challenge for *C. elegans* [Coghlan et al., 2008]

- Validation of gene predictions for *C. elegans*: [Schweikert et al., 2009c]

<table>
<thead>
<tr>
<th></th>
<th>No. of genes</th>
<th>No. of genes analyzed</th>
<th>Frac. of genes w/ expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>New genes</td>
<td>2,197</td>
<td>57</td>
<td>≈ 42%</td>
</tr>
<tr>
<td>Missing unconf. genes</td>
<td>205</td>
<td>24</td>
<td>≈ 8%</td>
</tr>
</tbody>
</table>

- Annotation of other nematode genomes: [Schweikert et al., 2009c]

<table>
<thead>
<tr>
<th>Genome</th>
<th>Genome size [Mbp]</th>
<th>No. of genes</th>
<th>No. exons/gene (mean)</th>
<th>mGene accuracy</th>
<th>best other accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>C. remanei</em></td>
<td>235.94</td>
<td>31503</td>
<td>5.7</td>
<td>96.6%</td>
<td>93.8%</td>
</tr>
<tr>
<td><em>C. japonica</em></td>
<td>266.90</td>
<td>20121</td>
<td>5.3</td>
<td>93.3%</td>
<td>88.7%</td>
</tr>
<tr>
<td><em>C. brenneri</em></td>
<td>453.09</td>
<td>41129</td>
<td>5.4</td>
<td>93.1%</td>
<td>87.8%</td>
</tr>
<tr>
<td><em>C. briggsae</em></td>
<td>108.48</td>
<td>22542</td>
<td>6.0</td>
<td>87.0%</td>
<td>82.0%</td>
</tr>
</tbody>
</table>

(Evaluation on exon level)
Results using mGene

- Most accurate *ab initio* method in the nGASP genome annotation challenge for *C. elegans* [Coghlan et al., 2008]

- Validation of gene predictions for *C. elegans*: [Schweikert et al., 2009c]

<table>
<thead>
<tr>
<th></th>
<th>No. of genes</th>
<th>No. of genes analyzed</th>
<th>Frac. of genes w/ expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>New genes</td>
<td>2,197</td>
<td>57</td>
<td>≈ 42%</td>
</tr>
<tr>
<td>Missing unconf. genes</td>
<td>205</td>
<td>24</td>
<td>≈ 8%</td>
</tr>
</tbody>
</table>

Annotation of other nematode genomes: [Schweikert et al., 2009c]

<table>
<thead>
<tr>
<th>Genome</th>
<th>Genome size [Mbp]</th>
<th>No. of genes</th>
<th>No. exons/gene (mean)</th>
<th>mGene accuracy</th>
<th>best other accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>C. remanei</em></td>
<td>235.94</td>
<td>31503</td>
<td>5.7</td>
<td>96.6%</td>
<td>93.8%</td>
</tr>
<tr>
<td><em>C. japonica</em></td>
<td>266.90</td>
<td>20121</td>
<td>5.3</td>
<td>93.3%</td>
<td>88.7%</td>
</tr>
<tr>
<td><em>C. brenneri</em></td>
<td>453.09</td>
<td>41129</td>
<td>5.4</td>
<td>93.1%</td>
<td>87.8%</td>
</tr>
<tr>
<td><em>C. briggsae</em></td>
<td>108.48</td>
<td>22542</td>
<td>6.0</td>
<td>87.0%</td>
<td>82.0%</td>
</tr>
</tbody>
</table>

(Evaluation on exon level)
Results using mGene

- Most accurate *ab initio* method in the nGASP genome annotation challenge for *C. elegans* [Coghlan et al., 2008]

- Validation of gene predictions for *C. elegans*: [Schweikert et al., 2009c]

<table>
<thead>
<tr>
<th></th>
<th>No. of genes</th>
<th>No. of genes analyzed</th>
<th>Frac. of genes w/ expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>New genes</td>
<td>2,197</td>
<td>57</td>
<td>≈ 42%</td>
</tr>
<tr>
<td>Missing unconf. genes</td>
<td>205</td>
<td>24</td>
<td>≈ 8%</td>
</tr>
</tbody>
</table>

- Annotation of other nematode genomes: [Schweikert et al., 2009c]

<table>
<thead>
<tr>
<th>Genome</th>
<th>Genome size [Mbp]</th>
<th>No. of genes</th>
<th>No. exons/gene (mean)</th>
<th>mGene accuracy</th>
<th>best other accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>C. remanei</em></td>
<td>235.94</td>
<td>31503</td>
<td>5.7</td>
<td>96.6%</td>
<td>93.8%</td>
</tr>
<tr>
<td><em>C. japonica</em></td>
<td>266.90</td>
<td>20121</td>
<td>5.3</td>
<td>93.3%</td>
<td>88.7%</td>
</tr>
<tr>
<td><em>C. brenneri</em></td>
<td>453.09</td>
<td>41129</td>
<td>5.4</td>
<td>93.1%</td>
<td>87.8%</td>
</tr>
<tr>
<td><em>C. briggsae</em></td>
<td>108.48</td>
<td>22542</td>
<td>6.0</td>
<td>87.0%</td>
<td>82.0%</td>
</tr>
</tbody>
</table>

(Evaluation on exon level)
• Gene finding (by far) still not perfect
  • New *ab initio* techniques more accurate
  • Many genes mis-predicted (*C. elegans*: ≈ 50%; human: ≈ 80%)
  • What is missing?
    • Not enough examples for training?
    • Model complexity?
    • Other information?

• Transcriptome is the result of many complex processes
  • Current methods also ignore other important information:
    • Chromatin structure and methylation patterns
    • RNA structure and processing regulation . . .
  • *Ab initio* methods cannot predict alternative transcripts

• New experimental data . . .
  • Deep RNA sequencing, DNA/histone modifications, . . .
Gene finding (by far) still not perfect
  - New \textit{ab initio} techniques more accurate
  - Many genes mis-predicted (\textit{C. elegans}: $\approx 50\%$; human: $\approx 80\%$)
  - What is missing?
    - Not enough examples for training?
    - Model complexity?
    - Other information?

Transcriptome is the result of many complex processes
  - Current methods also ignore other important information:
    - Chromatin structure and methylation patterns
    - RNA structure and processing regulation . . .
  - \textit{Ab initio} methods cannot predict alternative transcripts

New experimental data . . .
  - Deep RNA sequencing, DNA/histone modifications, . . .
Gene finding (by far) still not perfect
  - New \textit{ab initio} techniques more accurate
  - Many genes mis-predicted (\textit{C. elegans}: \approx 50\%; human: \approx 80\%)
  - What is missing?
    - Not enough examples for training?
    - Model complexity?
    - Other information?

Transcriptome is the result of many complex processes
  - Current methods also ignore other important information:
    - Chromatin structure and methylation patterns
    - RNA structure and processing regulation . . .

\textit{Ab initio} methods cannot predict alternative transcripts

New experimental data . . .
  - Deep RNA sequencing, DNA/histon modifications, . . .
Gene finding (by far) still not perfect

- New *ab initio* techniques more accurate
- Many genes mis-predicted (*C. elegans*: \( \approx 50\% \); human: \( \approx 80\% \))
- What is missing?
  - Not enough examples for training?
  - Model complexity?
  - Other information?

Transcriptome is the result of many complex processes

- Current methods also ignore other important information:
  - Chromatin structure and methylation patterns
  - RNA structure and processing regulation . . .

- *Ab initio* methods cannot predict alternative transcripts

New experimental data . . .

- Deep RNA sequencing, DNA/histone modifications, . . .
Limitations/Extensions

- Gene finding (by far) still not perfect
  - New *ab initio* techniques more accurate
  - Many genes mis-predicted (*C. elegans*: \( \approx 50\% \); human: \( \approx 80\% \))
  - What is missing?
    - Not enough examples for training?
    - Model complexity?
    - Other information?

- Transcriptome is the result of many complex processes
  - Current methods also ignore other important information:
    - Chromatin structure and methylation patterns
    - RNA structure and processing regulation . . .
  - *Ab initio* methods cannot predict alternative transcripts

- New experimental data . . .
  - Deep RNA sequencing, DNA/histon modifications, . . .
Learning to Integrate Data Sources

STEP 1: SVM Signal Predictions

[Behr et al., 2009]
Learning to Integrate Data Sources

STEP 1: SVM Signal Predictions

STEP 2: Integration

F(x,y)
Learning to Integrate Data Sources

STEP 1: SVM Signal Predictions

STEP 2: Integration

[Behr et al., 2009]
Learning to Integrate Data Sources

STEP 1: SVM Signal Predictions
- tss
- tis
- acc
- don
- stop

STEP 2: Integration

Tiling Array Data

Transform features

Large margin

[Behr et al., 2009]
Regularizer Considerations

Obvious choices:

- In *exonic regions* expression evidence should be scored monotonically increasing
- In *intronic and intergenic regions* expression evidence should be scored monotonically decreasing

More sophisticated for expression-aware model: [Zeller et al., 2008c]

- Replicate whole model $w^{(m)}$, $m = 1, \ldots, M$ for $M$ expression levels
- Regularize sub-models by coupling them:

$$\sum_{m=1}^{M-1} \sum_{i=1}^{N} (w_{i}^{(m+1)} - w_{i}^{(m)})^2$$
Regularizer Considerations

Obvious choices:
- In *exonic regions* expression evidence should be scored monotonically increasing
- In *intronic and intergenic regions* expression evidence should be scored monotonically decreasing

More sophisticated for expression-aware model: [Zeller et al., 2008c]
- Replicate whole model $w^{(m)}$, $m = 1, \ldots, M$ for $M$ expression levels
- Regularize sub-models by coupling them:

$$\sum_{m=1}^{M-1} \sum_{i=1}^{N} (w_i^{(m+1)} - w_i^{(m)})^2$$

Discrete expression level

intergenic  exonic  intronic

1  2  Q
Is there other information we can use to improve accuracy?

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

<table>
<thead>
<tr>
<th>mGene (ab initio) ...</th>
<th>73.3%</th>
</tr>
</thead>
<tbody>
<tr>
<td>... with DNA methylation (1 tissue)</td>
<td>76.1%</td>
</tr>
<tr>
<td>... with Nucleosome position predictions</td>
<td>78.0%</td>
</tr>
<tr>
<td>... with RNA secondary structure predictions</td>
<td>76.7%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>mGene (ab initio) ...</th>
<th>45.0%</th>
</tr>
</thead>
<tbody>
<tr>
<td>... with mass spectra ...</td>
<td>45.0%</td>
</tr>
<tr>
<td>... with tiling arrays ...</td>
<td>45.7%</td>
</tr>
<tr>
<td>... with ESTs (influenced annotation) ...</td>
<td>56.5%</td>
</tr>
<tr>
<td>... with RNA-seq ...</td>
<td>55.8%</td>
</tr>
</tbody>
</table>
Is there other information we can use to improve accuracy?

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

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.])*

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%
Is there other information we can use to improve accuracy?

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

1. mGene (ab initio) ...  
   
2. ... with DNA methylation (1 tissue)  
   
3. ... with Nucleosome position predictions  
   
4. ... with RNA secondary structure predictions  

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>mGene (ab initio)</td>
<td>73.3%</td>
</tr>
<tr>
<td>DNA methylation</td>
<td>76.1%</td>
</tr>
<tr>
<td>Nucleosome position predictions</td>
<td>78.0%</td>
</tr>
<tr>
<td>RNA secondary structure predictions</td>
<td>76.7%</td>
</tr>
</tbody>
</table>

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

1. mGene (ab initio) ...  

2. ... with mass spectra ...  

3. ... with tiling arrays ...  

4. ... with ESTs (influenced annotation) ...  

5. ... with RNA-seq ...  

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>mGene (ab initio)</td>
<td>45.0%</td>
</tr>
<tr>
<td>Mass spectra</td>
<td>45.0%</td>
</tr>
<tr>
<td>Tiling arrays</td>
<td>45.7%</td>
</tr>
<tr>
<td>ESTs (influenced annotation)</td>
<td>56.5%</td>
</tr>
<tr>
<td>RNA-seq</td>
<td>55.8%</td>
</tr>
</tbody>
</table>
Adaption to new genomes

Requires sufficient number of known gene models for “training”

⇒ Web service

Use *C. elegans* model for other nematodes:

<table>
<thead>
<tr>
<th>Genome</th>
<th>Genome size [Mbp]</th>
<th>No. of genes</th>
<th>No. exons/gene (mean)</th>
<th>mGene accuracy</th>
<th>best other accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>C. remanei</em></td>
<td>235.94</td>
<td>31503</td>
<td>5.7</td>
<td>96.6%</td>
<td>93.8%</td>
</tr>
<tr>
<td><em>C. japonica</em></td>
<td>266.90</td>
<td>20121</td>
<td>5.3</td>
<td>93.3%</td>
<td>88.7%</td>
</tr>
<tr>
<td><em>C. brenneri</em></td>
<td>453.09</td>
<td>41129</td>
<td>5.4</td>
<td>93.1%</td>
<td>87.8%</td>
</tr>
<tr>
<td><em>C. briggsae</em></td>
<td>108.48</td>
<td>22542</td>
<td>6.0</td>
<td>87.0%</td>
<td>82.0%</td>
</tr>
</tbody>
</table>

What if training data is scarce?

⇒ Develop methods that exploit *evolutionary information* and gene models from other genomes
Adaption to new genomes

Requires sufficient number of known gene models for “training”

⇒ Web service

Use *C. elegans* model for other nematodes:

<table>
<thead>
<tr>
<th>Genome</th>
<th>Genome size [Mbp]</th>
<th>No. of genes</th>
<th>No. exons/gene (mean)</th>
<th>mGene accuracy</th>
<th>best other accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>C. remanei</em></td>
<td>235.94</td>
<td>31503</td>
<td>5.7</td>
<td>96.6%</td>
<td>93.8%</td>
</tr>
<tr>
<td><em>C. japonica</em></td>
<td>266.90</td>
<td>20121</td>
<td>5.3</td>
<td>93.3%</td>
<td>88.7%</td>
</tr>
<tr>
<td><em>C. brenneri</em></td>
<td>453.09</td>
<td>41129</td>
<td>5.4</td>
<td>93.1%</td>
<td>87.8%</td>
</tr>
<tr>
<td><em>C. briggsae</em></td>
<td>108.48</td>
<td>22542</td>
<td>6.0</td>
<td>87.0%</td>
<td>82.0%</td>
</tr>
</tbody>
</table>

What if training data is scarce?

⇒ Develop methods that exploit *evolutionary information* and gene models from other genomes
Adaption to new genomes

- Requires sufficient number of known gene models for “training”
  ⇒ Web service [Schweikert et al., 2009b]

Use *C. elegans* model for other nematodes: [Schweikert et al., 2009c]

<table>
<thead>
<tr>
<th>Genome</th>
<th>Genome size [Mbp]</th>
<th>No. of genes</th>
<th>No. exons/gene (mean)</th>
<th>mGene accuracy</th>
<th>best other accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>C. remanei</em></td>
<td>235.94</td>
<td>31503</td>
<td>5.7</td>
<td>96.6%</td>
<td>93.8%</td>
</tr>
<tr>
<td><em>C. japonica</em></td>
<td>266.90</td>
<td>20121</td>
<td>5.3</td>
<td>93.3%</td>
<td>88.7%</td>
</tr>
<tr>
<td><em>C. brenneri</em></td>
<td>453.09</td>
<td>41129</td>
<td>5.4</td>
<td>93.1%</td>
<td>87.8%</td>
</tr>
<tr>
<td><em>C. briggsae</em></td>
<td>108.48</td>
<td>22542</td>
<td>6.0</td>
<td>87.0%</td>
<td>82.0%</td>
</tr>
</tbody>
</table>

What if training data is scarce?

⇒ Develop methods that exploit *evolutionary information* and gene models from other genomes [Schweikert et al., 2008]
Hierarchical structure arises naturally from the Tree of Life

- Taxonomy used to define relationship between tasks
- Closer tasks benefit more from each other
Formal Problem Definition

Multitask Learning

- Consider $M$ tasks $T_i$, where $i \in \{1, \ldots, M\}$
- We are given data $D_i = \{(x_1, y_1), \ldots, (x_{N_i}, y_{N_i})\}$ for each task
- We want to train $M$ predictors $f_1, \ldots, f_M$, each taking into account all available information
- For that we would like to utilize a given taxonomy $\mathcal{T}$, that relates the tasks at hand

⇒ We need algorithms to efficiently exploit $\mathcal{T}$ for transfer learning
Multitask Learning

- Consider $M$ tasks $T_i$, where $i \in \{1, \ldots, M\}$
- We are given data $D_i = \{(x_1, y_1), \ldots, (x_{N_i}, y_{N_i})\}$ for each task
- We want to train $M$ predictors $f_1, \ldots, f_M$, each taking into account all available information
- For that we would like to utilize a given taxonomy $\mathcal{T}$, that relates the tasks at hand

$\Rightarrow$ We need algorithms to efficiently exploit $\mathcal{T}$ for transfer learning
Multitask Learning

- Consider $M$ tasks $T_i$, where $i \in \{1, \ldots, M\}$
- We are given data $D_i = \{(x_1, y_1), \ldots, (x_{N_i}, y_{N_i})\}$ for each task
- We want to train $M$ predictors $f_1, \ldots, f_M$, each taking into account all available information
- For that we would like to utilize a given taxonomy $\mathcal{T}$, that relates the tasks at hand

$\Rightarrow$ We need algorithms to efficiently exploit $\mathcal{T}$ for transfer learning
Two ways of leveraging a given taxonomy $\mathcal{T}$

- Taxonomy
- Transformation
- Task Similarity Matrix

$$\Gamma = \begin{pmatrix} y_{1,1} & \cdots & y_{1,M} \\ \vdots & \ddots & \vdots \\ y_{M,1} & \cdots & y_{M,M} \end{pmatrix}$$

Top-Down

Pairwise & Multitask Kernel

Gunnar Rätsch (FML, Tübingen)
Domain Adaptation by Regularization

Idea: Enforce model similarity via Regularization Term

- Regular SVM

$$\begin{align*}
\min_{w,b} \quad & \frac{1}{2} \|w\|^2 + C \sum_{(x,y) \in D} \ell (\langle \Phi(x), w \rangle + b, y) \\
\text{where } \ell & \text{ is the hinge loss, } \ell(z, y) = \max\{1 - yz, 0\}.
\end{align*}$$

- DA-SVM

$$\begin{align*}
\min_{w,b} \quad & \frac{1}{2} \|w - w_{par}\|^2 + C \sum_{(x,y) \in D} \ell (\langle \Phi(x), w \rangle + b, y),
\end{align*}$$
Domain Adaptation by Regularization

Idea: Enforce model similarity via Regularization Term

- Regular SVM

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{(x,y) \in D} \ell \left( \langle \Phi(x), w \rangle + b, y \right)$$

- DA-SVM

$$\min_{w,b} \frac{1}{2} \|w - w_{par}\|^2 + C \sum_{(x,y) \in D} \ell \left( \langle \Phi(x), w \rangle + b, y \right) ,$$

where $\ell$ is the hinge loss, $\ell(z, y) = \max\{1 - yz, 0\}$.
Hierarchical Top-Down Approach

Idea: Exploit taxonomy $G$ algorithmically

- Initialization: $w_0$ trained on union of task data
- Top-Down for each node $t$:
  - Train on $D_i = \bigcup_{j \leq i} D_j$
  - Regularize $w_i$ against parent predictor $w_{par}$: $\|w_i - w_{par}\|^2$
- Use leaf predictors for classification
Hierarchical Top-Down Approach

**Idea:** Exploit taxonomy \( G \) algorithmically

- Initialization: \( w_0 \) trained on union of task data
- Top-Down for each node \( t \):
  - Train on \( D_i = \bigcup_{j \leq i} D_j \)
  - Regularize \( w_i \) against parent predictor \( w_{par} \):  \( \| w_i - w_{par} \|^2 \)
- Use leaf predictors for classification

\[\text{(a) Top-level training} \quad \text{(b) Inner training} \quad \text{(c) Taxon training}\]

[Widmer et al., 2010]
Hierarchical Top-Down Approach

Idea: Exploit taxonomy $G$ algorithmically

- Initialization: $w_0$ trained on union of task data
- Top-Down for each node $t$:
  - Train on $D_i = \bigcup_{j \lesssim i} D_j$
  - Regularize $w_i$ against parent predictor $w_{par}$: $\|w_i - w_{par}\|^2$
- Use leaf predictors for classification

(a) Top-level training  (b) Inner training  (c) Taxon training

[Widmer et al., 2010]
Pairwise Approach

\[
\min_{\mathbf{w}_1, \ldots, \mathbf{w}_M} \frac{1}{2} \sum_{t=1}^{M} \sum_{s=1}^{M} \gamma_{t,s} \| \mathbf{w}_t - \mathbf{w}_s \|^2 + \sum_{t=1}^{M} C_t \sum_{(x,y) \in D_t} \ell \left( \langle x, \mathbf{w}_t \rangle, y \right).
\]

where \( \ell \) is the hinge loss, \( \ell(z, y) = \max\{1 - yz, 0\} \).

- Train all classifiers \( \mathbf{w}_i \) at the same time
- Loss is evaluated independently on datasets \( D_i \)
- Similarity is enforced via pairwise regularization term
- Task closeness controlled by \( \gamma_{t,s} \)

\[ \Rightarrow \text{efficient solution possible via decomposition} \]

[Evgeniou et al., 2005]
Multitask Kernel Approach

\[
\max_{\alpha} -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \hat{k}(x_i, x_j) + \sum_{i=1}^{n} \alpha_i \\
\text{s.t. } 0 \leq \alpha_i \leq C \quad \forall i \in [1, n] \\
\alpha^T y = 0,
\]

where

\[
\hat{k}((x_i, s), (x_j, t)) = k_{\text{task}}(s, t) \cdot k(x_i, x_j) \cdot \gamma_{t,s}
\]

- Easily implemented by altering existing kernel functions (WDK)
- Reuse existing kernel algorithms (SVM)

[Daumé, 2007, Jacob and Vert, 2008, Widmer et al., 2010]
Method Summary

- Multitask Learning Methods
  - Top-Down
  - Pairwise Regularization
  - Multitask Kernel

- Additional Baselines

---

Plain

Union

\( f_1 \quad f_2 \quad f_3 \quad f_4 \)
Application to splice-site recognition

- Formulate as binary classification problem

  ≈ 150 nucleotides window around dimer
  
  CT...GTCGTA...GAAGCTAGGAGCGC...ACGCGT...GA

- Utilize 15 organisms related by taxonomy
MTL methods outperform baselines
Best performer is Top-Down
Potential increase depends on the particular organism
Summary and Future Work

- **Regularization**
  - Determines convergence rates in optimization
  - Becomes less important, when data is abundant
  - Allows implementation of prior knowledge
  - Helps for solving related learning tasks

- **Genome annotation**
  - Is a huge *structured output learning problem*, which we solve in a two-step learning procedure
  - Gives rise to heterogeneous data integration challenges
  - Is still quite far from perfect

- **Transfer learning**
  - Learn models for multiple organisms simultaneously
  - More accurate, but challenges in joint model optimization

- **New sequencing data gives**
  - Detailed picture on transcriptome
  - Hope to be able to train more sophisticated models
Regularization

- Determines convergence rates in optimization
- Becomes less important, when data is abundant
- Allows implementation of prior knowledge
- Helps for solving related learning tasks

Genome annotation

- Is a huge structured output learning problem, which we solve in a two-step learning procedure
- Gives rise to heterogeneous data integration challenges
- Is still quite far from perfect

Transfer learning

- Learn models for multiple organisms simultaneously
- More accurate, but challenges in joint model optimization

New sequencing data gives

- Detailed picture on transcriptome
- Hope to be able to train more sophisticated models
Summary and Future Work

- **Regularization**
  - Determines convergence rates in optimization
  - Becomes less important, when data is abundant
  - Allows implementation of prior knowledge
  - Helps for solving related learning tasks

- **Genome annotation**
  - Is a huge *structured output learning problem*, which we solve in a two-step learning procedure
  - Gives rise to heterogeneous data integration challenges
  - Is still quite far from perfect

- **Transfer learning**
  - Learn models for multiple organisms simultaneously
  - More accurate, but challenges in joint model optimization
  - New sequencing data gives
    - Detailed picture on transcriptome
    - Hope to be able to train more sophisticated models
Summary and Future Work

- **Regularization**
  - Determines convergence rates in optimization
  - Becomes less important, when data is abundant
  - Allows implementation of prior knowledge
  - Helps for solving related learning tasks

- **Genome annotation**
  - Is a huge *structured output learning problem*, which we solve in a two-step learning procedure
  - Gives rise to heterogeneous data integration challenges
  - Is still quite far from perfect

- **Transfer learning**
  - Learn models for multiple organisms simultaneously
  - More accurate, but challenges in joint model optimization

- **New sequencing data gives**
  - Detailed picture on transcriptome
  - Hope to be able to train more sophisticated models
Acknowledgments

**Sequence Analysis**
- Sören Sonnenburg (FML/FIRST)
- Gabi Schweikert (FML/MPI)
- Alex Zien (Life Biosystems)
- Konrad Rieck (FIRST)

**Gene Finding**
- Jonas Behr (FML)
- Gabi Schweikert (FML/MPI)
- Georg Zeller (FML/MPI)
- Alex Zien (LIFE Biosystems)

**Transfer Learning**
- Christian Widmer (FML)
- Jose Leiva (Spain)
- Yasemin Altun (MPI)
- Bernhard Schölkopf (MPI)

---

More Information
- [http://fml.mpg.de/raetsch](http://fml.mpg.de/raetsch)
- [http://shogun-toolbox.org](http://shogun-toolbox.org)
- [http://mogene.org/web](http://mogene.org/web)
- Funding from DFG
- Slides with references will be available online

---

Thank you!
# Acknowledgments

## Sequence Analysis
- Sören Sonnenburg (FML/FIRST)
- Gabi Schweikert (FML/MPI)
- Alex Zien (Life Biosystems)
- Konrad Rieck (FIRST)

## Gene Finding
- Jonas Behr (FML)
- Gabi Schweikert (FML/MPI)
- Georg Zeller (FML/MPI)
- Alex Zien (LIFE Biosystems)

## Transfer Learning
- Christian Widmer (FML)
- Jose Leiva (Spain)
- Yasemin Altun (MPI)
- Bernhard Schölkopf (MPI)

## More Information
- [http://fml.mpg.de/raetsch](http://fml.mpg.de/raetsch)
- [http://shogun-toolbox.org](http://shogun-toolbox.org)
- [http://mgene.org/web](http://mgene.org/web)
- Funding from DFG
- Slides with references will be available online

Thank you!
## Acknowledgments

### Sequence Analysis
- Sören Sonnenburg (FML/FIRST)
- Gabi Schweikert (FML/MPI)
- Alex Zien (Life Biosystems)
- Konrad Rieck (FIRST)

### Gene Finding
- Jonas Behr (FML)
- Gabi Schweikert (FML/MPI)
- Georg Zeller (FML/MPI)
- Alex Zien (LIFE Biosystems)

### Transfer Learning
- Christian Widmer (FML)
- Jose Leiva (Spain)
- Yasemin Altun (MPI)
- Bernhard Schölkopf (MPI)

## More Information
- [http://fml.mpg.de/raetsch](http://fml.mpg.de/raetsch)
- [http://shogun-toolbox.org](http://shogun-toolbox.org)
- [http://mgene.org/web](http://mgene.org/web)
- Funding from DFG
- Slides with references will be available online

---

Thank you!


