# Large Scale Sequence Analysis with Applications to Genomics

Gunnar Rätsch

Friedrich Miescher Laboratory, Max Planck Society
Tübingen, Germany

Talk at CSML, University College London

March 18, 2009

# Discovery of the Nuclein (Friedrich Miescher, 1869)












## Discovery of Nuclein:

- from lymphocyte & salmon
- "multi-basic acid" ($\geq 4$)

"If one . . . wants to assume that a single substance . . . is the specific cause of fertilization, then one should undoubtedly first and foremost consider nuclein" (Miescher, 1874)

# $\frac{1}{150\,000}$ of the DNA of *C. elegans*



```
>CHROMOSOME_I
GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGC
CTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCT
AAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAA
GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGC
CTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCT
AAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAA
GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGC
CTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCT
AAGCCTAAGCCTAAGCCTAAGCCTAAAAAATTGAGATAAGAAAA
CATTTTACTTTTTCAAAATTGTTTTCATGCTAAATTCAAAACGTTTTTTT
TTTAGTGAAGCTTCTAGATATTTGGCGGGTACCTCTAATTTTGCCTGCCT
GCCAACCTATATGCTCCTGTGTTTAGGCCTAATACTAAGCCTAAGCCTAA
GCCTAATACTAAGCCTAAGCCTAAGACTAAGCCTAATACTAAGCCTAAGC
CTAAGACTAAGCCTAAGACTAAGCCTAAGACTAAGCCTAATACTAAGCCT
...
```







# Research Topics



**Machine Learning**

1. Inference methods for structured data
   ⇒ *Develop fast and accurate learning methods*
2. Convergence properties of iterative algorithms
   ⇒ *Boosting-like algorithms and semi-infinite LPs*

**Molecular Biology**

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*

# Research Topics



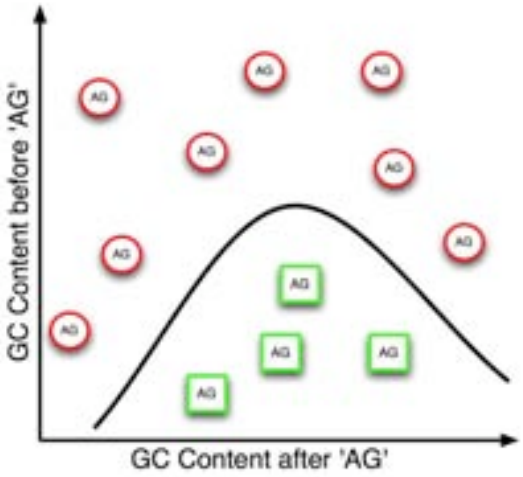
**Machine Learning**

1. Inference methods for structured data
   ⇒ *Develop fast and accurate learning methods*
2. Convergence properties of iterative algorithms
   ⇒ *Boosting-like algorithms and semi-infinite LPs*

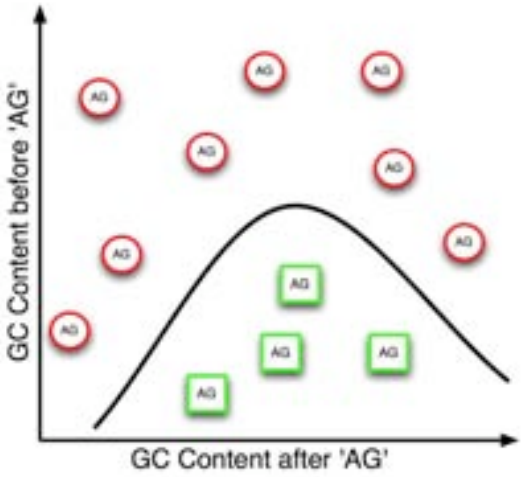
**Molecular Biology**

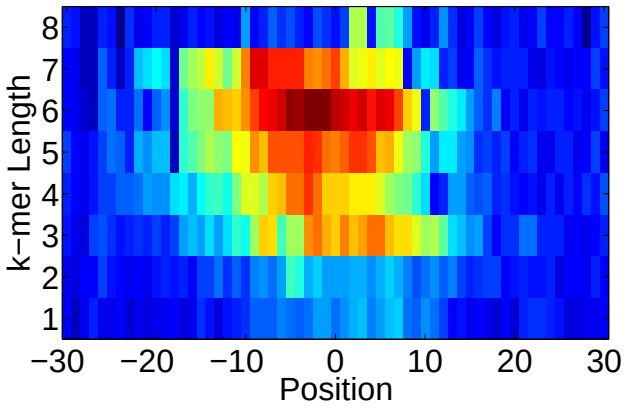
3. Genome annotation
   ⇒ *Predict features encoded on DNA*
4. Biological networks
   ⇒ *Understand interactions between gene products*
5. Analysis of polymorphisms
   ⇒ *Discover polymorphisms and associate with phenotypes*

1. Large scale sequence classification
   with Sonnenburg (Fraunhofer, Berlin) & Schölkopf (MPI Biol. Cybernetics)
2. Analysis and explanation of learning results
   with Sonnenburg (Fraunhofer, Berlin)
3. Sequence segmentation & structure prediction
   with Altun (MPI Biol. Cybernetics)



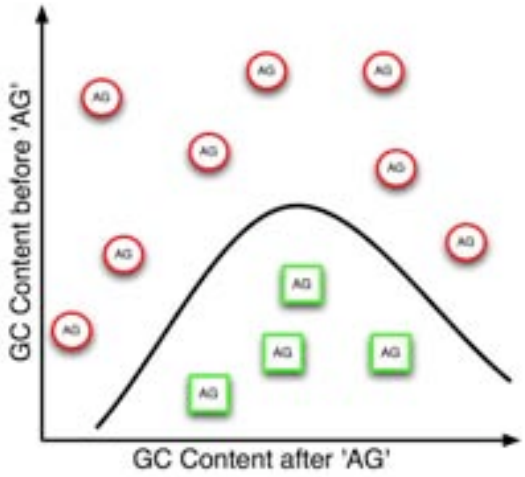
1. Large scale sequence classification
   with Sonnenburg (Fraunhofer, Berlin) & Schölkopf (MPI Biol. Cybernetics)
2. Analysis and explanation of learning results
   with Sonnenburg (Fraunhofer, Berlin)
3. Sequence segmentation & structure prediction
   with Altun (MPI Biol. Cybernetics)

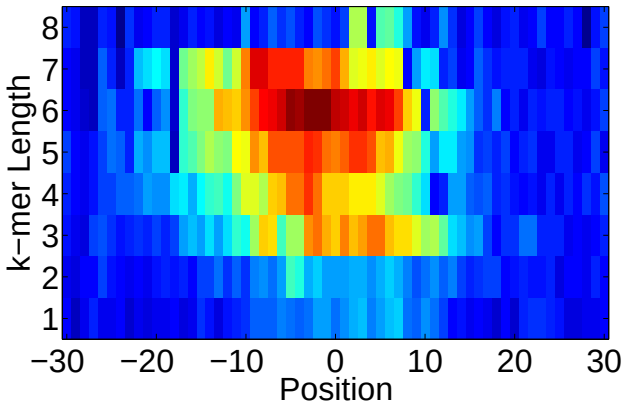


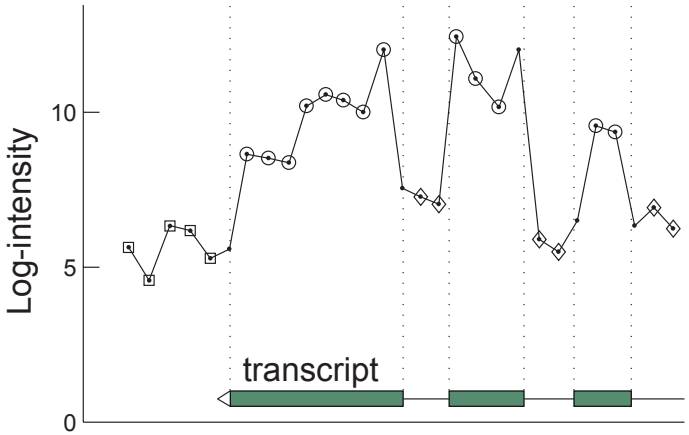



Friedrich Miescher Laboratory
of the Max Planck Society

1. Large scale sequence classification
   with Sonnenburg (Fraunhofer, Berlin) & Schölkopf (MPI Biol. Cybernetics)
2. Analysis and explanation of learning results
   with Sonnenburg (Fraunhofer, Berlin)
3. Sequence segmentation & structure prediction
   with Altun (MPI Biol. Cybernetics)

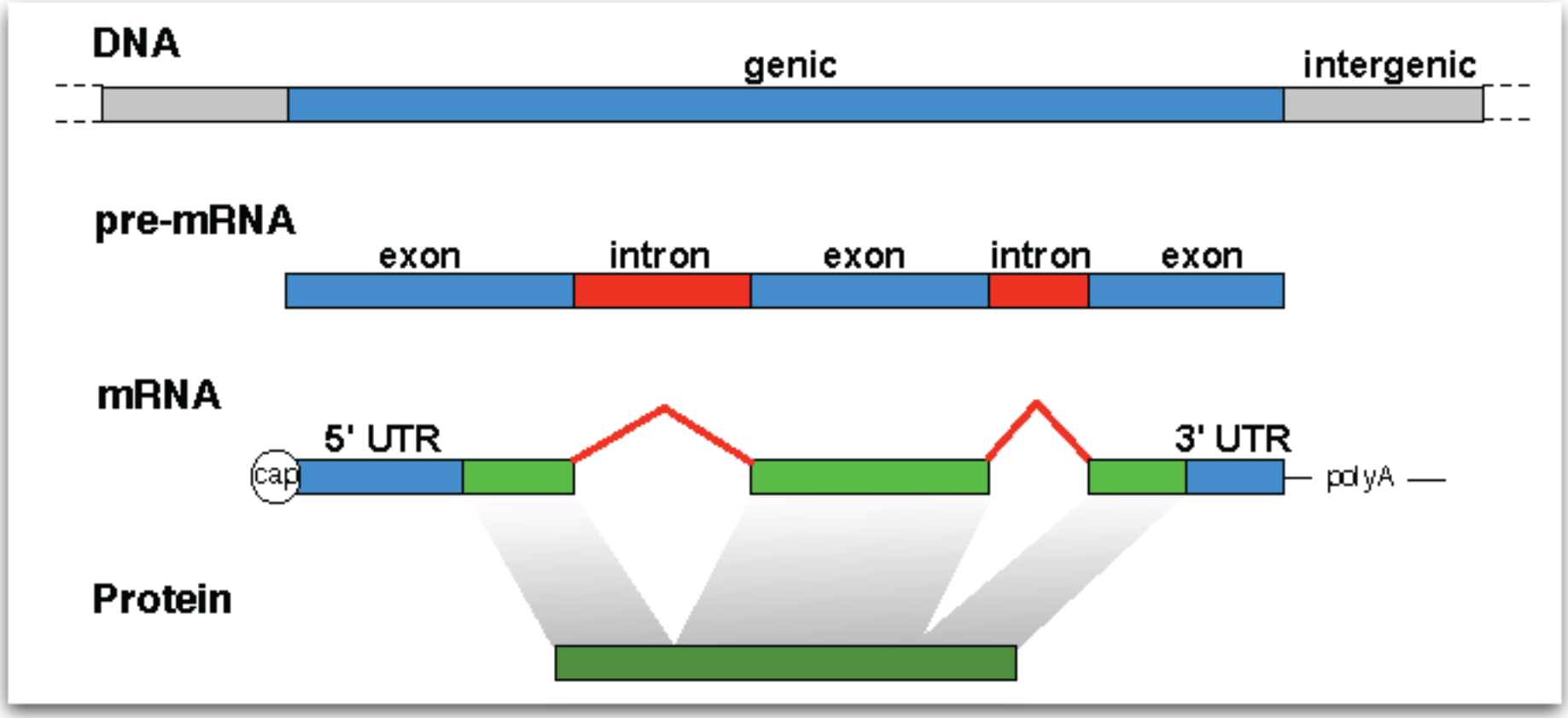


# Computational Genome Annotation



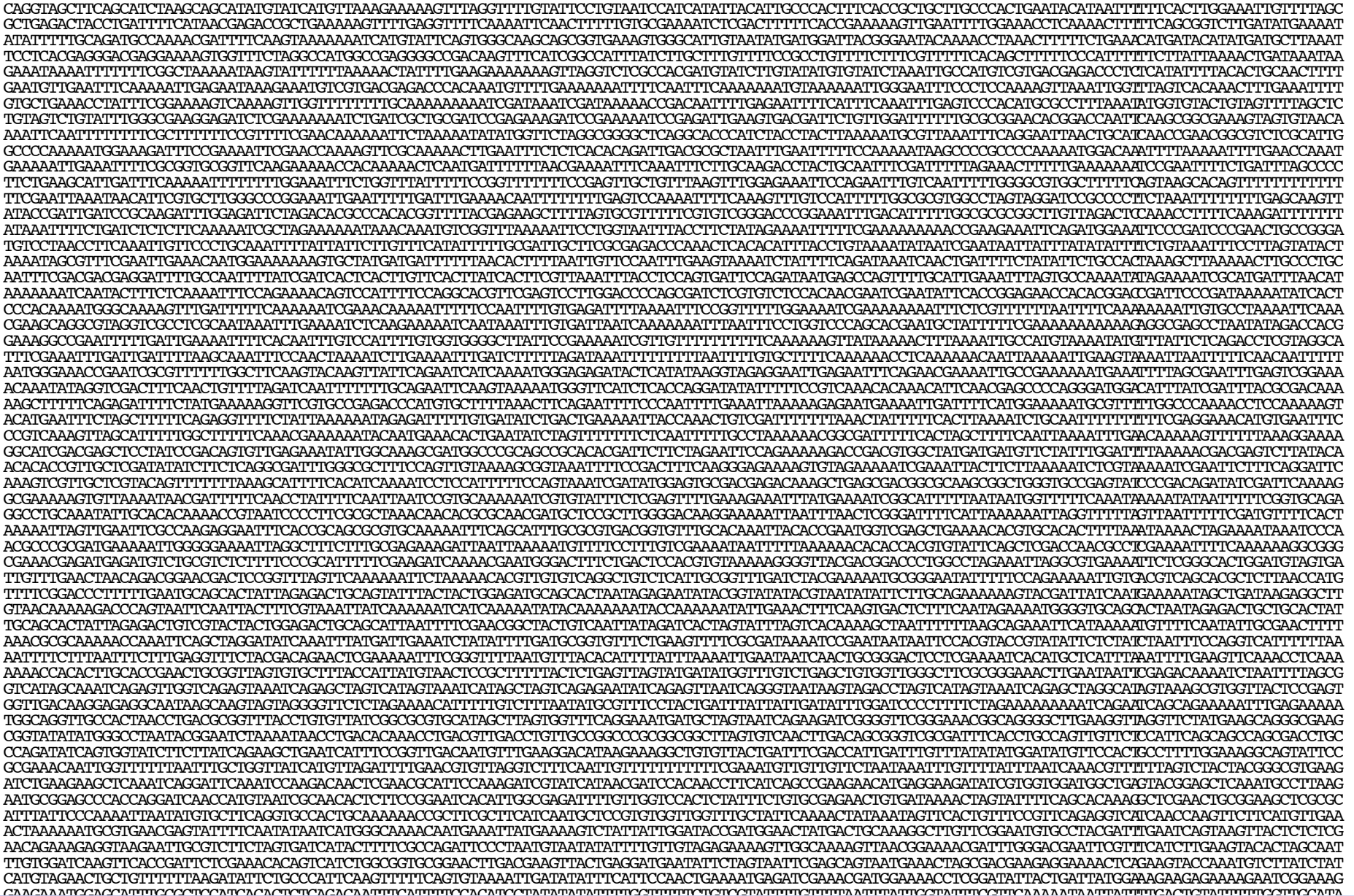



Simplest formulation:

- Given a DNA sequence $\mathbf{x} \in \{\text{'A','C','G','T'}\}^L$
- Find the correct **label sequence y** $= y_1 y_2 \ldots y_L$
  ($y_i \in Y = \{$"intergenic", "5' UTR", "coding", "intron", $\ldots\}$)

# Example: *C. elegans* (I: 43,500 - 52,050)

Friedrich Miescher Laboratory
of the Max Planck Society



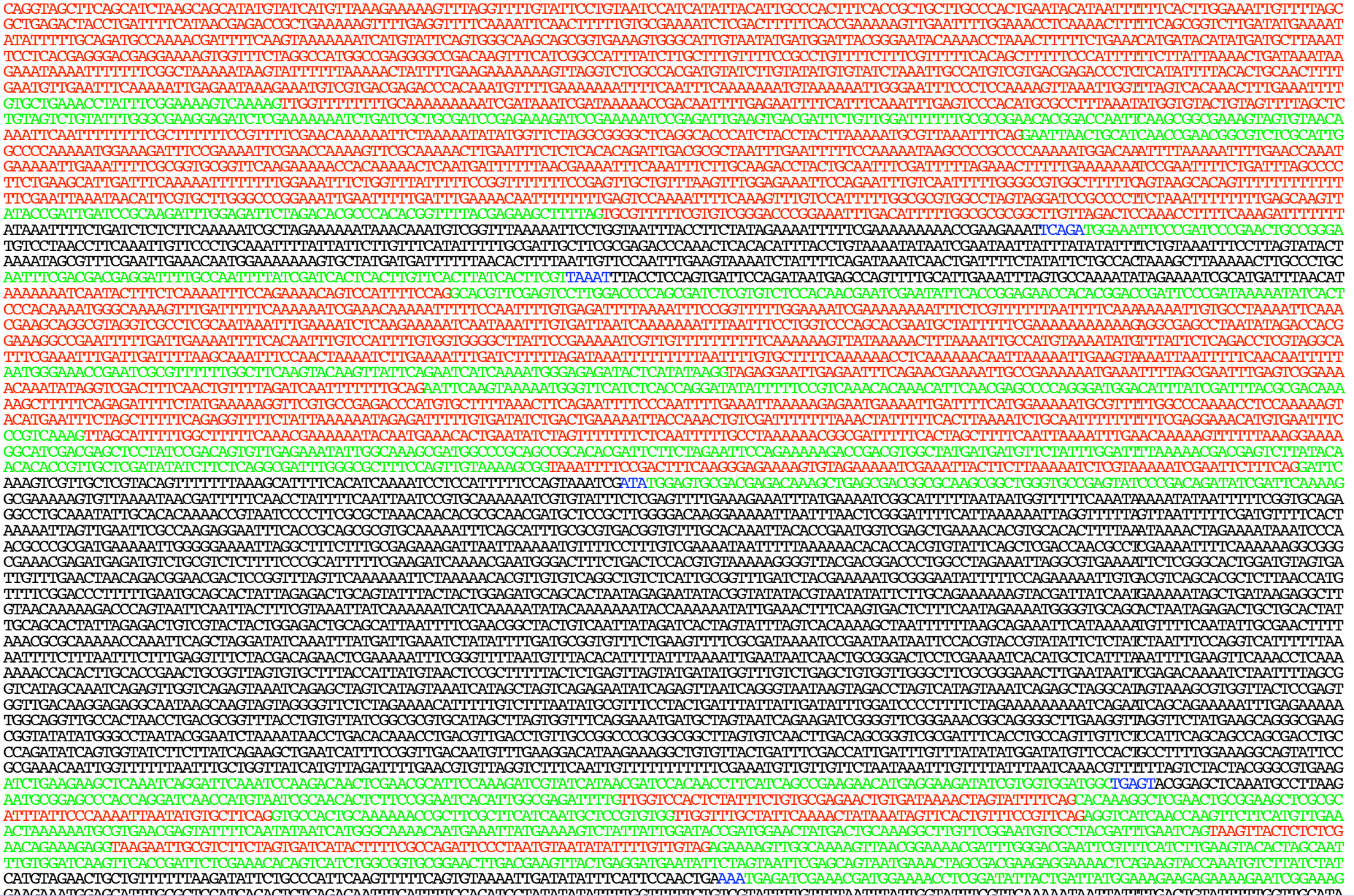
# Example: *C. elegans* (I: 43,500 - 52,050)

Friedrich Miescher Laboratory
of the Max Planck Society



# Example: *C. elegans* (I: 43,500 - 52,050)

Friedrich Miescher Laboratory
of the Max Planck Society



# Some Problem Characteristics



- Genome sequence of 100Mb (*C. elegans*; yet relatively small)
  - Can be interpreted in both directions
  - The human genome is 35× larger
- Segment boundaries exhibit specific sequence patterns
  - Almost every position is a potential segment start
  - Many "examples" to classify
- Statistics within different segments differs
  - Score segments of different length
- Segments are known to appear in a certain order

**Summary:** BIG label sequence learning problem

# Some Problem Characteristics



- Genome sequence of 100Mb (*C. elegans*; yet relatively small)
  - Can be interpreted in both directions
  - The human genome is 35× larger
- Segment boundaries exhibit specific sequence patterns
  - Almost every position is a potential segment start
  - Many "examples" to classify
- Statistics within different segments differs
  - Score segments of different length
- Segments are known to appear in a certain order

**Summary:** BIG label sequence learning problem

# Some Problem Characteristics



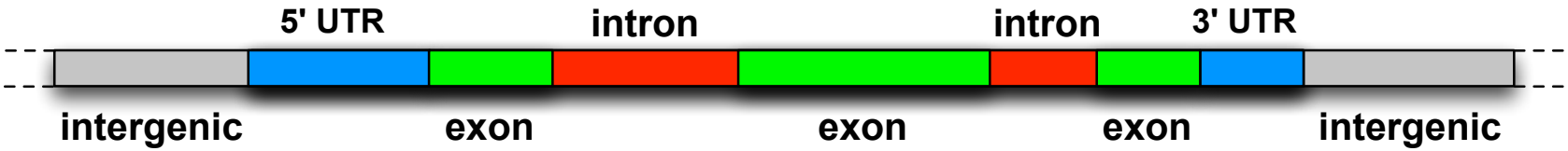
- Genome sequence of 100Mb (*C. elegans*; yet relatively small)
  - Can be interpreted in both directions
  - The human genome is 35× larger
- Segment boundaries exhibit specific sequence patterns
  - Almost every position is a potential segment start
  - Many "examples" to classify
- Statistics within different segments differs
  - Score segments of different length
- Segments are known to appear in a certain order

**Summary:** BIG label sequence learning problem

# Some Problem Characteristics



- Genome sequence of 100Mb (*C. elegans*; yet relatively small)
  - Can be interpreted in both directions
  - The human genome is 35× larger
- Segment boundaries exhibit specific sequence patterns
  - Almost every position is a potential segment start
  - Many "examples" to classify
- Statistics within different segments differs
  - Score segments of different length
- Segments are known to appear in a certain order

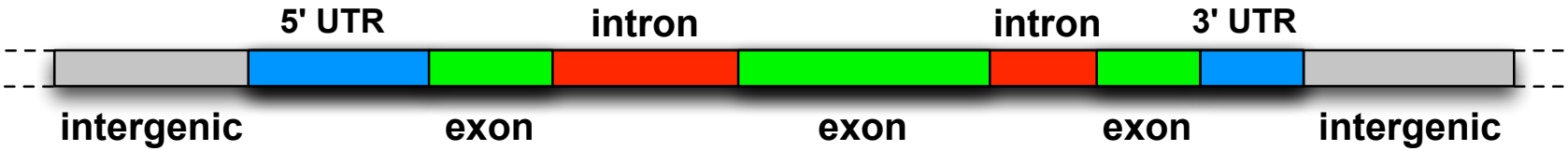



**Summary:** BIG label sequence learning problem

# Some Problem Characteristics



- Genome sequence of 100Mb (*C. elegans*; yet relatively small)
  - Can be interpreted in both directions
  - The human genome is 35× larger
- Segment boundaries exhibit specific sequence patterns
  - Almost every position is a potential segment start
  - Many "examples" to classify
- Statistics within different segments differs
  - Score segments of different length
- Segments are known to appear in a certain order




**Summary:** BIG label sequence learning problem

# Max-Margin Structured Output Learning

- Learn function $f(\mathbf{y}|\mathbf{x})$ scoring segmentations $\mathbf{y}$ for input $\mathbf{x}$
- Maximize $f(\mathbf{y}|\mathbf{x})$ w.r.t. $\mathbf{y}$ for prediction:

$$\operatorname*{argmax}_{\mathbf{y}\in\Upsilon^*} f(\mathbf{y}|\mathbf{x})$$

- Idea: $f(\mathbf{y}|\mathbf{x}) \gg f(\hat{\mathbf{y}}|\mathbf{x})$ for wrong labels $\hat{\mathbf{y}} \neq \mathbf{y}$
- Approach:
  - Given $N$ sequence pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$ for training
  - Solve using column-generation techniques:

$$\begin{aligned} \min_f \quad & C\sum_{n=1}^{N}\xi_n + \mathbf{P}[f] \\ \text{w.r.t.} \quad & f(\mathbf{y}_n|\mathbf{x}_n) - f(\mathbf{y}|\mathbf{x}_n) \geq \ell(\mathbf{y}_n, \mathbf{y}) - \xi_n \\ & \text{for all } \mathbf{y}_n \neq \mathbf{y} \in \Upsilon^*, n = 1, \ldots, N \end{aligned}$$

**All the remaining details are in $f$, $P[f]$, and $\ell(\cdot,\cdot)$.**

- Learn function $f(\mathbf{y}|\mathbf{x})$ scoring segmentations $\mathbf{y}$ for input $\mathbf{x}$
- Maximize $f(\mathbf{y}|\mathbf{x})$ w.r.t. $\mathbf{y}$ for prediction:

$$\operatorname*{argmax}_{\mathbf{y}\in\Upsilon^*} f(\mathbf{y}|\mathbf{x})$$

- Idea: $f(\mathbf{y}|\mathbf{x}) \gg f(\hat{\mathbf{y}}|\mathbf{x})$ for wrong labels $\hat{\mathbf{y}} \neq \mathbf{y}$
- Approach:
  - Given $N$ sequence pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$ for training
  - Solve using column-generation techniques:

$$\begin{aligned} \min_f \quad & C\sum_{n=1}^{N}\xi_n + \mathbf{P}[f] \\ \text{w.r.t.} \quad & f(\mathbf{y}_n|\mathbf{x}_n) - f(\mathbf{y}|\mathbf{x}_n) \geq \ell(\mathbf{y}_n, \mathbf{y}) - \xi_n \\ & \text{for all } \mathbf{y}_n \neq \mathbf{y} \in \Upsilon^*, n = 1, \ldots, N \end{aligned}$$

**All the remaining details are in $f$, $P[f]$, and $\ell(\cdot,\cdot)$.**

- Learn function $f(\mathbf{y}|\mathbf{x})$ scoring segmentations $\mathbf{y}$ for input $\mathbf{x}$
- Maximize $f(\mathbf{y}|\mathbf{x})$ w.r.t. $\mathbf{y}$ for prediction:

$$\operatorname*{argmax}_{\mathbf{y}\in\Upsilon^*} f(\mathbf{y}|\mathbf{x})$$

- Idea: $f(\mathbf{y}|\mathbf{x}) \gg f(\hat{\mathbf{y}}|\mathbf{x})$ for wrong labels $\hat{\mathbf{y}} \neq \mathbf{y}$
- Approach:
  - Given $N$ sequence pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$ for training
  - Solve using column-generation techniques:

$$\begin{aligned} \min_f \quad & C\sum_{n=1}^{N}\xi_n + \mathbf{P}[f] \\ \text{w.r.t.} \quad & f(\mathbf{y}_n|\mathbf{x}_n) - f(\mathbf{y}|\mathbf{x}_n) \geq \ell(\mathbf{y}_n, \mathbf{y}) - \xi_n \\ & \text{for all } \mathbf{y}_n \neq \mathbf{y} \in \Upsilon^*, n = 1, \ldots, N \end{aligned}$$

**All the remaining details are in $f$, $P[f]$, and $\ell(\cdot,\cdot)$.**

# Parametrization of $f$



**Requirements:**

- Must allow efficient computation of $\operatorname{argmax}_{\mathbf{y}\in\Upsilon^*} f(\mathbf{y}|\mathbf{x})$
- Better has a small number of parameters

**Plausible model:**

- Represent segmentation as sequence of segments:

$$(p_i, q_i', y_i), \quad \text{for } i = 1, \ldots, l$$

- Model is additive in segment properties (Semi-Markov)

$$f(\mathbf{y}|\mathbf{x}) := \sum_{i=1}^{l} (f_{y_i}^{l}(\mathbf{x}_{\sim p_i \sim}) + f_{y_i}^{s}(\mathbf{x}_{p_i \sim q_i}) + f_{y_i}^{r}(\mathbf{x}_{\sim q_i \sim}))$$

Need to learn to score strings $\mathbf{x}_*$!

$\Rightarrow$ **String kernels!??** No! ... Yes!

# Parametrization of $f$



**Requirements:**

- Must allow efficient computation of $\operatorname{argmax}_{\mathbf{y}\in\Upsilon^*} f(\mathbf{y}|\mathbf{x})$
- Better has a small number of parameters

**Plausible model:**

- Represent segmentation as sequence of segments:

$$(p_i, q_i', y_i), \quad \text{for } i = 1, \ldots, I$$

- Model is additive in segment properties (Semi-Markov)

$$f(\mathbf{y}|\mathbf{x}) := \sum_{i=1}^{I} (f_{y_i}^{l}(\mathbf{x}_{\sim p_i \sim}) + f_{y_i}^{s}(\mathbf{x}_{p_i \sim q_i}) + f_{y_i}^{r}(\mathbf{x}_{\sim q_i \sim}))$$

Need to learn to score strings $\mathbf{x}_*$!

$\Rightarrow$ **String kernels!??** No! ... Yes!

# Parametrization of $f$



**Requirements:**

- Must allow efficient computation of $\operatorname{argmax}_{\mathbf{y}\in\Upsilon^*} f(\mathbf{y}|\mathbf{x})$
- Better has a small number of parameters

**Plausible model:**

- Represent segmentation as sequence of segments:

$$(p_i, q_i', y_i), \quad \text{for } i = 1, \ldots, l$$

- Model is additive in segment properties (Semi-Markov)

$$f(\mathbf{y}|\mathbf{x}) := \sum_{i=1}^{l} (f_{y_i}^{l}(\mathbf{x}_{\sim p_i \sim}) + f_{y_i}^{s}(\mathbf{x}_{p_i \sim q_i}) + f_{y_i}^{r}(\mathbf{x}_{\sim q_i \sim}))$$

Need to learn to score strings $\mathbf{x}_*$!

$\Rightarrow$ **String kernels!??** No! ... Yes!

# Parametrization of $f$



**Requirements:**

- Must allow efficient computation of $\operatorname{argmax}_{\mathbf{y}\in\Upsilon^*} f(\mathbf{y}|\mathbf{x})$
- Better has a small number of parameters

**Plausible model:**

- Represent segmentation as sequence of segments:

$$(p_i, q_i', y_i), \quad \text{for } i = 1, \ldots, l$$

- Model is additive in segment properties (Semi-Markov)

$$f(\mathbf{y}|\mathbf{x}) := \sum_{i=1}^{l} (f_{y_i}^{l}(\mathbf{x}_{\sim p_i \sim}) + f_{y_i}^{s}(\mathbf{x}_{p_i \sim q_i}) + f_{y_i}^{r}(\mathbf{x}_{\sim q_i \sim}))$$

Need to learn to score strings $\mathbf{x}_*$!

$\Rightarrow$ **String kernels!??** No! ... Yes!

# Solve Problem in two Steps



$$f(\mathbf{y}|\mathbf{x}) := \sum_{i=1}^{I} \left( f^{l}_{y_i}(\mathbf{x}_{\sim p_i \sim}) + f^{s}_{y_i}(\mathbf{x}_{p_i \sim q_i}) + f^{r}_{y_i}(\mathbf{x}_{\sim q_i \sim}) \right)$$

$$f^{*}_{y_i}(\mathbf{x}_*) := h^{*}_{y_i}(g^{*}_{y_i}(\mathbf{x}_*))$$

- Step 1: String analysis, leading to $g_{y_i} : \mathbf{x} \rightarrow \mathbb{R}$
- Step 2: Combination, leading to $h_{y_i} : \mathbb{R} \mapsto \mathbb{R}$

**How to train $g^{*}_{y_i}(\mathbf{x}_*)$?**

- Should be large, if $\mathbf{x}_*$ is part of true label sequence
- Two-class problem: $\mathbf{x}_*$ at every possible position is negative, except at boundaries of true segments

**How to train $h^{*}_{y_i}(\cdot)$?**

- Simple 1-d function, e.g. piece-wise linear function

# Solve Problem in two Steps

$$f(\mathbf{y}|\mathbf{x}) := \sum_{i=1}^{I} (f^{l}_{y_i}(\mathbf{x}_{\sim p_i \sim}) + f^{s}_{y_i}(\mathbf{x}_{p_i \sim q_i}) + f^{r}_{y_i}(\mathbf{x}_{\sim q_i \sim}))$$

$$f^{*}_{y_i}(\mathbf{x}_*) := h^{*}_{y_i}(g^{*}_{y_i}(\mathbf{x}_*))$$

- Step 1: String analysis, leading to $g_{y_i} : \mathbf{x} \to \mathbb{R}$
- Step 2: Combination, leading to $h_{y_i} : \mathbb{R} \mapsto \mathbb{R}$

**How to train $g^{*}_{y_i}(\mathbf{x}_*)$?**

- Should be large, if $\mathbf{x}_*$ is part of true label sequence
- Two-class problem: $\mathbf{x}_*$ at every possible position is negative, except at boundaries of true segments

**How to train $h^{*}_{y_i}(\cdot)$?**

- Simple 1-d function, e.g. piece-wise linear function

# Solve Problem in two Steps

$$f(\mathbf{y}|\mathbf{x}) := \sum_{i=1}^{I} (f^l_{y_i}(\mathbf{x}_{\sim p_i \sim}) + f^s_{y_i}(\mathbf{x}_{p_i \sim q_i}) + f^r_{y_i}(\mathbf{x}_{\sim q_i \sim}))$$

$$f^*_{y_i}(\mathbf{x}_*) := h^*_{y_i}(g^*_{y_i}(\mathbf{x}_*))$$

- Step 1: String analysis, leading to $g_{y_i} : \mathbf{x} \to \mathbb{R}$
- Step 2: Combination, leading to $h_{y_i} : \mathbb{R} \mapsto \mathbb{R}$

**How to train $g^*_{y_i}(\mathbf{x}_*)$?**

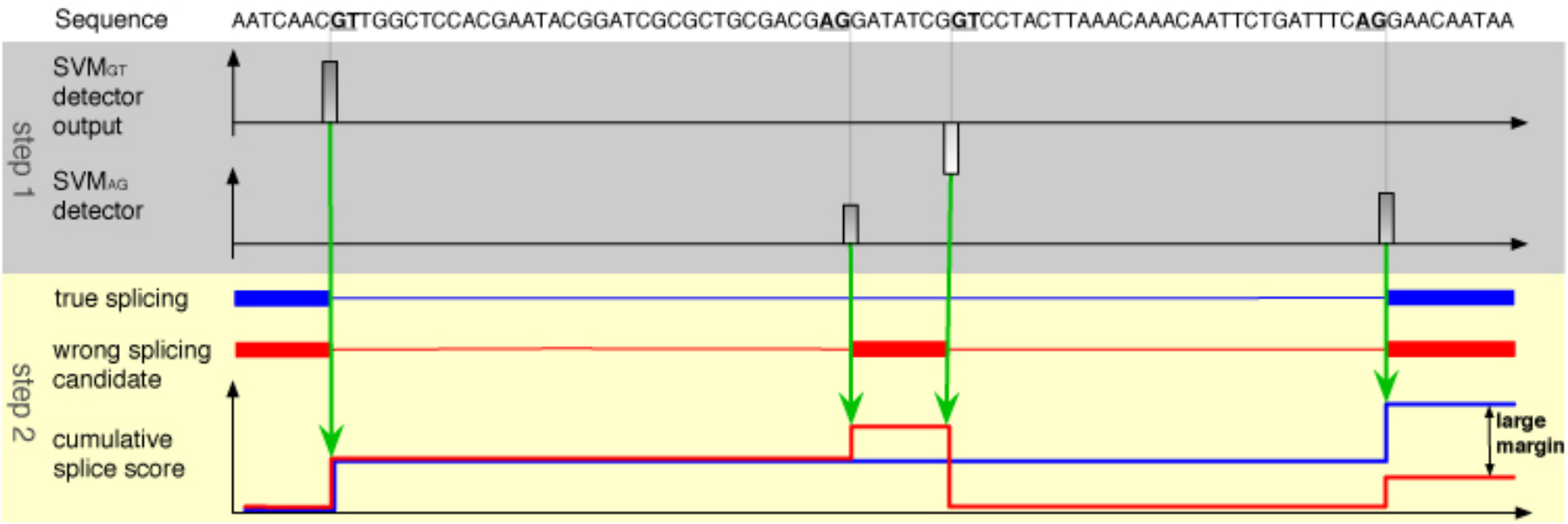
- Should be large, if $\mathbf{x}_*$ is part of true label sequence
- Two-class problem: $\mathbf{x}_*$ at every possible position is negative, except at boundaries of true segments

**How to train $h^*_{y_i}(\cdot)$?**

- Simple 1-d function, e.g. piece-wise linear function

# Discriminative Gene Prediction (simplified)






[Rätsch, Sonnenburg, Srinivasan, Witte, Müller, Sommer, Schölkopf, 2007]

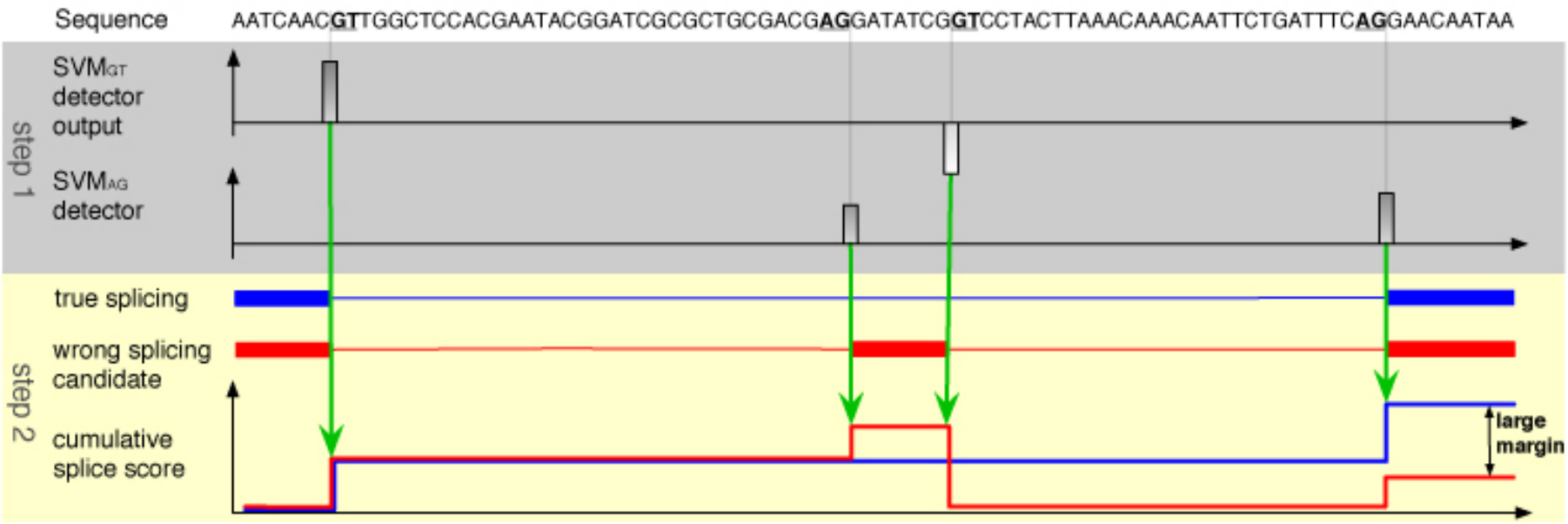
Simplified Model: Score for splice form $\mathbf{y} = \{(p_j, q_j)\}_{j=1}^{J}$:

$$F(\mathbf{y}) := \underbrace{\sum_{j=1}^{J-1} S_{GT}(f_j^{GT}) + \sum_{j=2}^{J} S_{AG}(f_j^{AG})}_{\text{Splice signals}} + \underbrace{\sum_{j=1}^{J-1} S_{L_I}(p_{j+1} - q_j) + \sum_{j=1}^{J} S_{L_E}(q_j - p_j)}_{\text{Segment lengths}}$$

Tune free parameters (in functions $S_{GT}, S_{AG}, S_{L_E}, S_{L_I}$) by solving **linear program** using training set with known splice forms

# Discriminative Gene Prediction (simplified)






[Rätsch, Sonnenburg, Srinivasan, Witte, Müller, Sommer, Schölkopf, 2007]

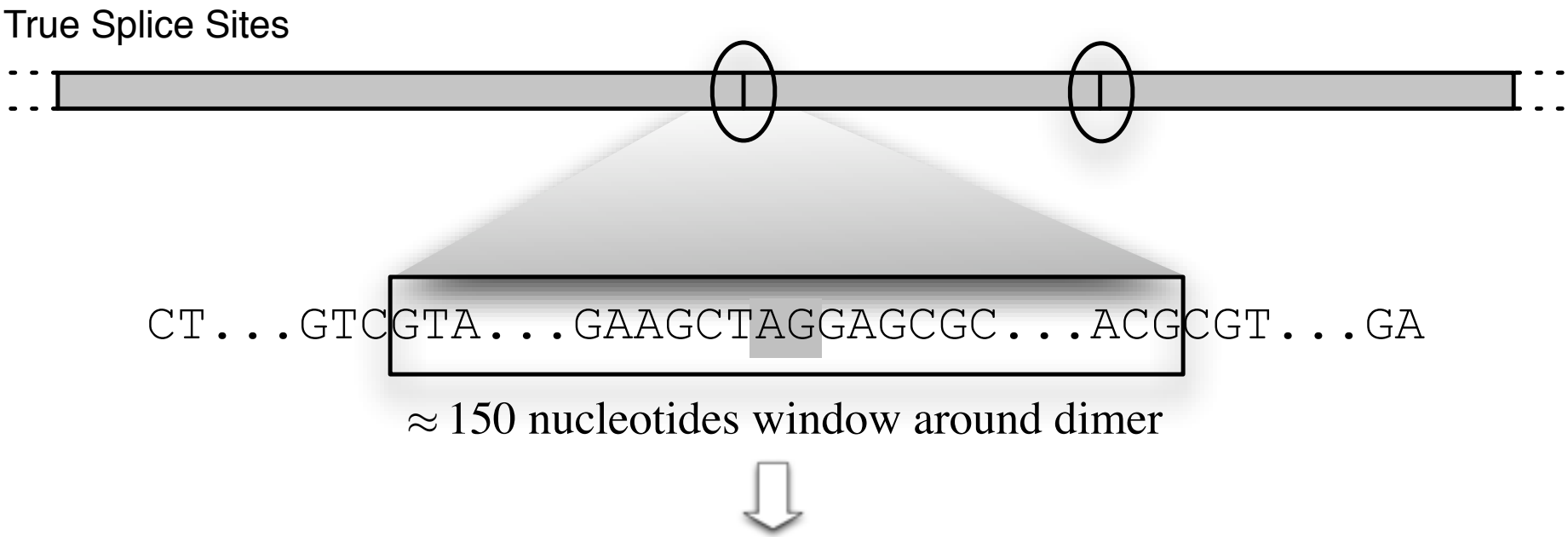
Simplified Model: Score for splice form $\mathbf{y} = \{(p_j, q_j)\}_{j=1}^{J}$:

$$F(\mathbf{y}) := \underbrace{\sum_{j=1}^{J-1} S_{GT}(f_j^{GT}) + \sum_{j=2}^{J} S_{AG}(f_j^{AG})}_{\text{Splice signals}} + \underbrace{\sum_{j=1}^{J-1} S_{L_I}(p_{j+1} - q_j) + \sum_{j=1}^{J} S_{L_E}(q_j - p_j)}_{\text{Segment lengths}}$$

Tune free parameters (in functions $S_{GT}, S_{AG}, S_{L_E}, S_{L_I}$) by solving **linear program** using training set with known splice forms

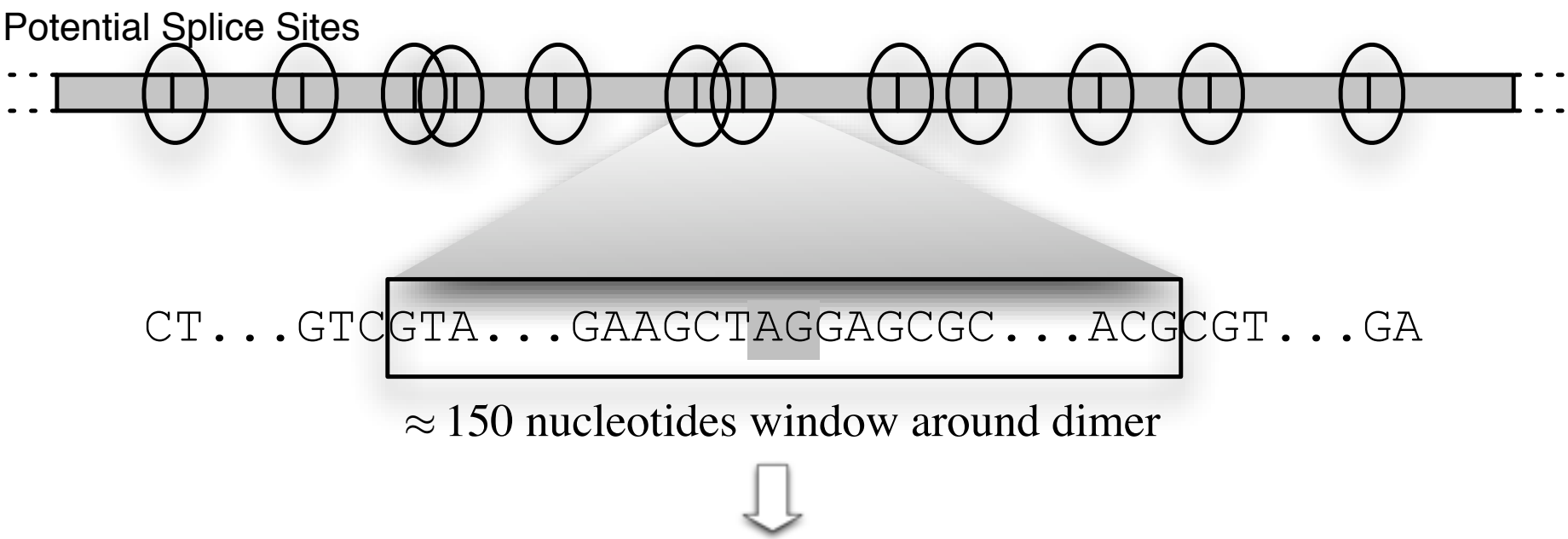
# Example: Intron/Exon Boundary





# Example: Intron/Exon Boundary

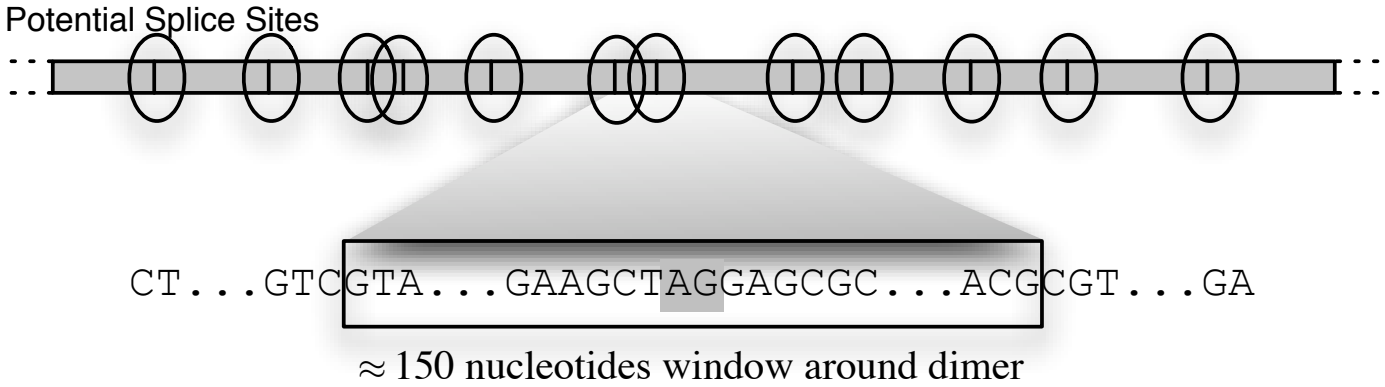


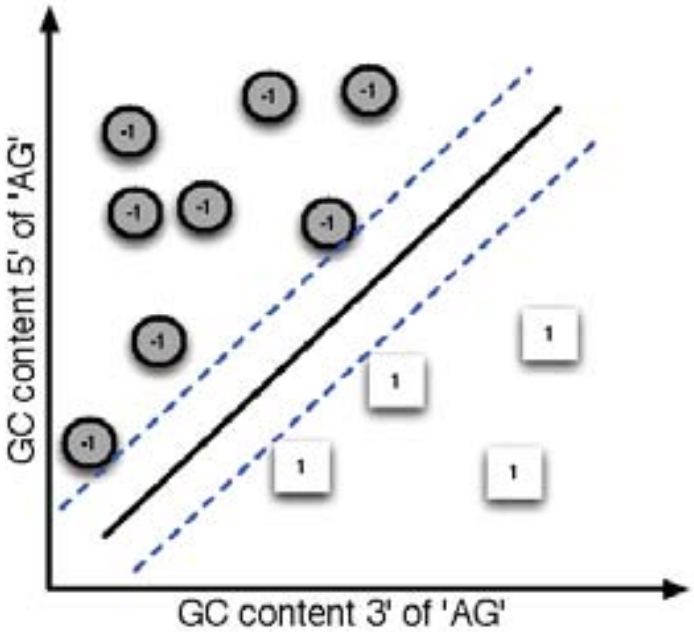



.

# Example: Intron/Exon Boundary






**Basic idea:**

For instance, exploit:

- Exons have more G's and C's
- Certain motifs near boundary

Sonnenburg, Schweikert et al. 2007

# Substring Kernels



- **General idea**
  - Count common substrings in two strings
  - Sequences are deemed the more similar, the more common substrings they contain
- **Variations**
  - Allow for gaps
  - Include wildcards
  - Allow for mismatches
  - Include substitutions
  - Motif Kernels
  - Assign weights to substrings

# Spectrum Kernel

- **General idea** [Leslie et al., 2002]
  - For each $k$-mer $s \in \Sigma^k$, the coordinate indexed by $s$ will be the number of times $s$ occurs in sequence $x$.
  - Then the $k$-spectrum feature map is

$$\Phi_k^{\text{Spectrum}}(\mathbf{x}) = (\phi_s(\mathbf{x}))_{s \in \Sigma^k}$$

  - Here $\phi_s(\mathbf{x})$ is the # occurrences of $s$ in $x$.
  - The spectrum kernel is now the inner product in the feature space defined by this map:

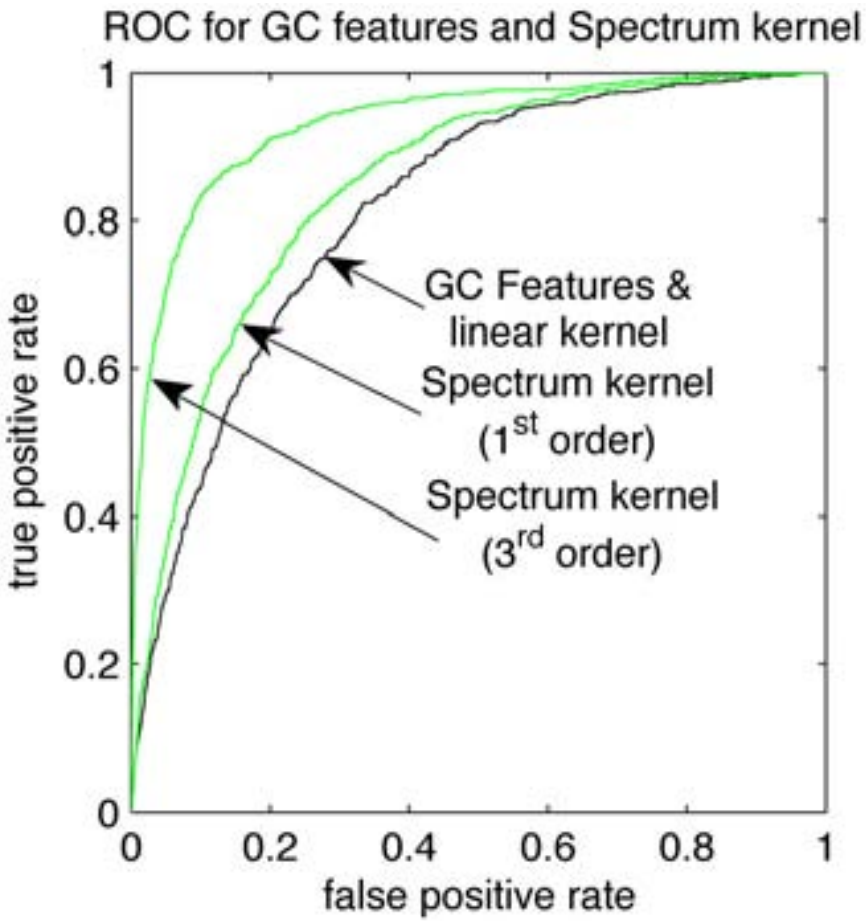
$$k^{\text{Spectrum}}(\mathbf{x}, \mathbf{x}') = \langle \Phi_k^{\text{Spectrum}}(\mathbf{x}), \Phi_k^{\text{Spectrum}}(\mathbf{x}') \rangle$$

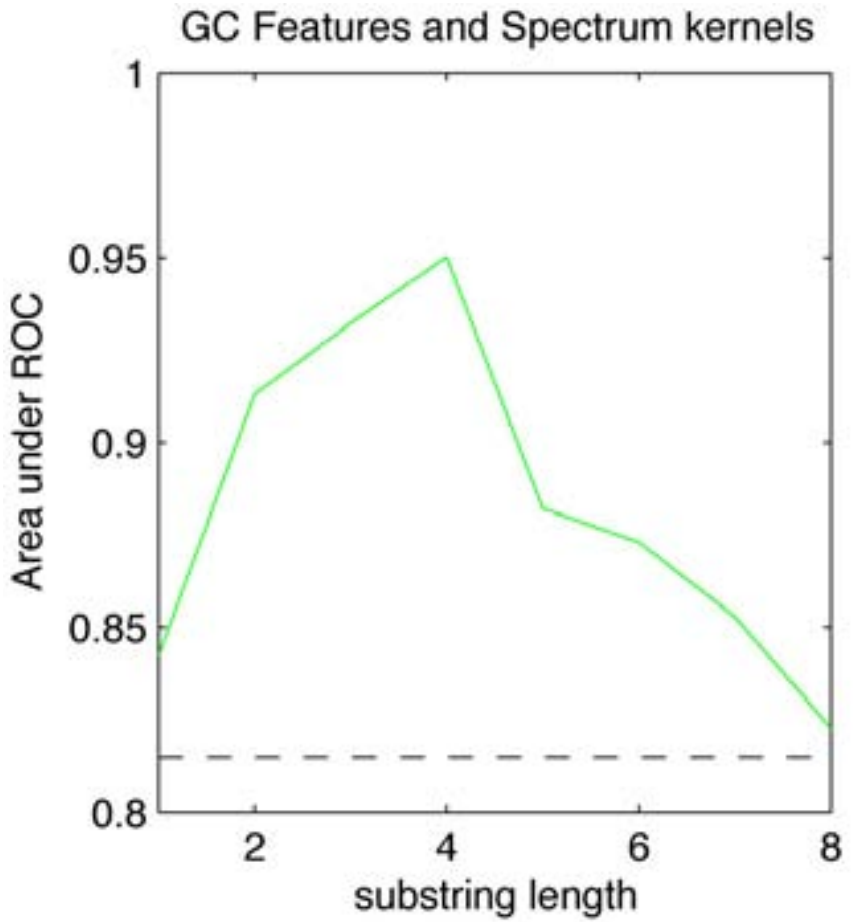
  - Dimensionality: Exponential in $k$: $|\Sigma|^k$

# Simulation Example (Acceptor Splice Sites)



- Linear Kernel on GC-content features
- Spectrum kernel $\mathrm{k}_k^{\mathrm{Spectrum}}(\mathbf{x}, \mathbf{x}')$



# Position Dependence



- Given: Potential acceptor splice sites




- Goal: Rule that distinguishes true from false ones
- Position of motif is important ('T' rich just before 'AG')
- Spectrum kernel is *blind* w.r.t. positions
- New kernels for sequences with constant length
  - Substring kernel per position (sum over positions)
    - Oligo kernel
    - Weighted Degree kernel
  - Can detect motifs at specific positions
    - weak if positions vary
    - Extension: allow "shifting"

# Weighted Degree Kernel [Rätsch and Sonnenburg, 2004]

- Equivalent to a mixture of spectrum kernels (up to order $K$) at every position for appropriately chosen $\beta$'s:

$$\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{K} \sum_{l=1}^{L-k+1} \beta_k \mathrm{k}_k^{\mathrm{Spectrum}}(\mathbf{u}_{l:l+k}(\mathbf{x}_i), \mathbf{u}_{l:l+k}(\mathbf{x}_j))$$

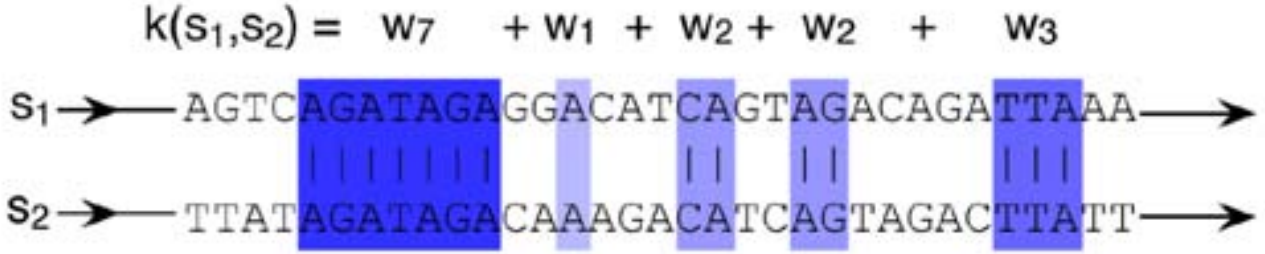
  where $\beta_k = \frac{K-k+1}{\sum_k (K-k+1)} = 2\frac{K-k+1}{k\cdot(k+1)}$.

- Can be equivalently computed by

$$\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{K} \sum_{l=1}^{L-k+1} \beta_k \mathrm{I}(\mathbf{u}_{l:l+k}(\mathbf{x}_i) = \mathbf{u}_{l:l+k}(\mathbf{x}_j))$$

  for appropriately chosen $\beta_k$.

- **Without shifts:** Compare two sequences by identifying the largest matching blocks:




where a matching block of length $k$ implies many shorter matches:

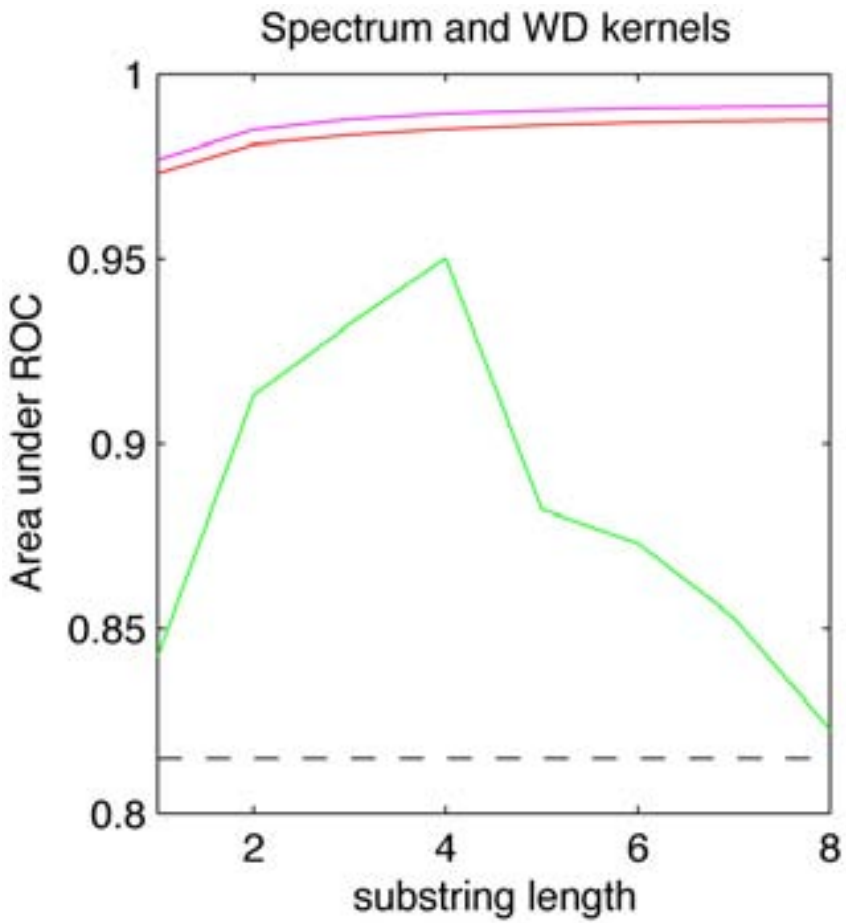
$$w_k = \sum_{j=1}^{\min(k,K)} \beta_j \cdot (k - j + 1).$$

- **With shifts:** Allows matching subsequences with offsets



# Substring Kernel Comparison






- Linear kernel on GC-content features
- Spectrum kernel
- Weighted degree kernel
- Weighted degree kernel with shifts

**Remark:** Higher order substring kernels typically exploit that correlations appear locally and not between arbitrary parts of the sequence (other than e.g. the polynomial kernel).

# Fast string kernels?



- Use index structures to speed up computation
  - Single kernel computation $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$
  - Kernel (sub-)matrix $k(\mathbf{x}_i, \mathbf{x}_j), i \in I, j \in J$
  - Linear combination of kernel elements

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}) = \left\langle \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \right\rangle$$

- Idea: Exploit that $\Phi(\mathbf{x})$ and also $\sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i)$ is sparse:
  - Explicit maps
  - Sorted lists
  - (Suffix) trees/tries/arrays

# Efficient data structures

- $\mathbf{v} = \Phi(\mathbf{x})$ is very sparse
- Computation with $\mathbf{v}$ requires efficient operations on single dimensions, e.g.

$$\texttt{lookup } v_s \quad \text{or } \texttt{update } v_s = v_s + \alpha$$

- Use trees or arrays to store only non-zero elements
  $\Rightarrow$ Substring is the index into the tree or array
- Leads to more efficient optimization algorithms:
  - Precompute $v = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i)$
  - Compute $\sum_{i=1}^{N} \alpha_i \mathrm{k}(\mathbf{x}_i, \mathbf{x})$ by

$$\sum_{s \text{ substring in } \mathbf{x}} v_s$$

# Explicit Maps



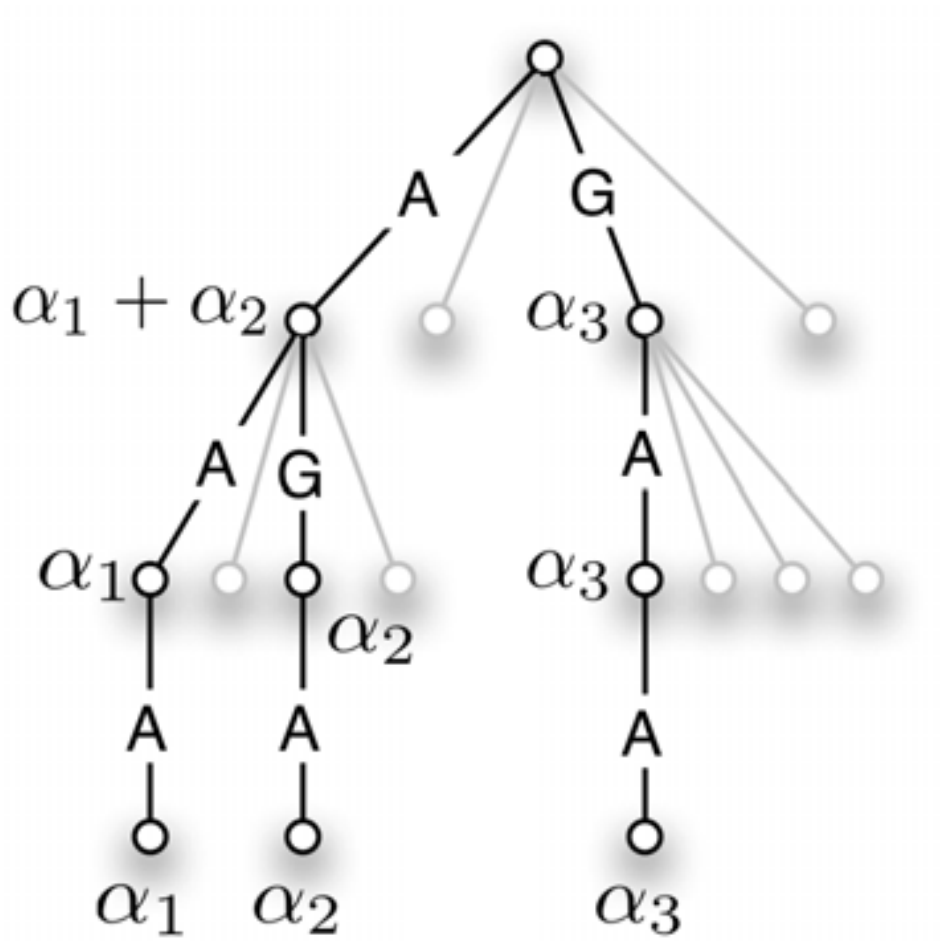
- Require $\mathcal{O}(|\Sigma|^k)$ memory
- Explicitly store $\mathbf{w} = \sum_i \alpha_i \Phi(\mathbf{x}_i)$
- `lookup` and `update` operations are $\mathcal{O}(1)$
- Updating all $f(\mathbf{x}_i)$ takes $\mathcal{O}(Q \cdot L \cdot k + N \cdot L \cdot k)$
- Very efficient, but only work for small $k$

# Sorted Lists

- Generate a sorted list with pairs $(\mathbf{u}, \alpha)$ of length $Q \cdot L$
  - $\mathcal{O}(Q \cdot L \cdot \log(Q \cdot L))$
- Requires $\mathcal{O}(Q \cdot L \cdot k)$ memory
- Iterate trough list and $k$-mer list of example (pre-sorted)
  - identify co-occuring $k$-mers
- Single $f(\mathbf{x}_i)$ requires $\mathcal{O}((Q \cdot L \cdot \log(Q \cdot L) + L) \cdot k)$
- All $f(\mathbf{x}_i)$ require $\mathcal{O}((Q \cdot L \cdot \log(Q \cdot L) + N \cdot L \cdot \log(N \cdot L)) \cdot k)$
- Requires additional sorting of long lists
- Also works for large $k$

# Example: Trees & Tries






Tree (trie) data structure stores sparse weightings on sequences (and their subsequences).

**Illustration:** Three sequences AAA, AGA, GAA were added to a trie ($\alpha$'s are the weights of the sequences).

- Building tree: $\mathcal{O}(Q \cdot L \cdot k)$
- Compute all $f(\mathbf{x}_i)$: $\mathcal{O}(N \cdot L \cdot k)$

# Solving the SVM Dual



$$\begin{array}{ll} \text{maximize}_\alpha & \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathrm{k}(x_i, x_j) \\ \text{s.t.} & \sum_{i=1}^N \alpha_i y_i = 0 \\ & 0 \leqslant \alpha_i \leqslant C \text{ for } i = 1, 2, \ldots, N. \end{array}$$

Requires $N^2$ kernel computations

- expensive to compute ($\mathcal{O}(k \cdot L \cdot N^2)$)
- expensive to store matrix ($\mathcal{O}(N^2)$)

Solving QP using interior point methods is expensive: $\mathcal{O}(N^3)$

**Idea:** Chunking based methods: Iterate

- Select small number of variables
- Optimize w.r.t. to these variables
- Stop if converged

# Chunking



$$\begin{aligned} F(\boldsymbol{\alpha}) := \quad & \sum_{i=1}^{N} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathrm{k}(x_i, x_j) \\ \text{s.t.} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & 0 \leqslant \alpha_i \leqslant C \text{ for } i = 1, 2, \ldots, N. \end{aligned}$$

- Select $Q$ variables $i_1, \ldots, i_Q$
  - Random (inefficient)
  - Sequential (inefficient)
  - Heuristic selection motivated by KKT conditions
    - Requires $f(\mathbf{x}_j) = \sum_{i=1}^{N} \alpha_i \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)$ for all $j$
    - Points that have too small margin, but $\alpha_i < C$
    - Points that are outside margin area, but $\alpha_i > 0$
    - Points with $0 \leq \alpha_i \leq C$
- Solve QP of size Q ($\mathcal{O}(Q^3)$)
- Update $f(\mathbf{x}_j)$ if necessary

# Chunking



- What do we need per iteration
  - Compute $f(\mathbf{x}_j) = \sum_{i=1}^{N} \alpha_i \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)$ for all $j$
  - Solve QP of size Q
- Complexity: $\mathcal{O}(Q \cdot N + Q^3)$
- First part very expensive for large $N$

- Can we speedup computing $f(\mathbf{x}_j)$?
- So far for string kernels: $\mathcal{O}(Q \cdot N \cdot L \cdot k)$
- With new data structures: $\mathcal{O}(N \cdot L \cdot k)$

# Algorithm



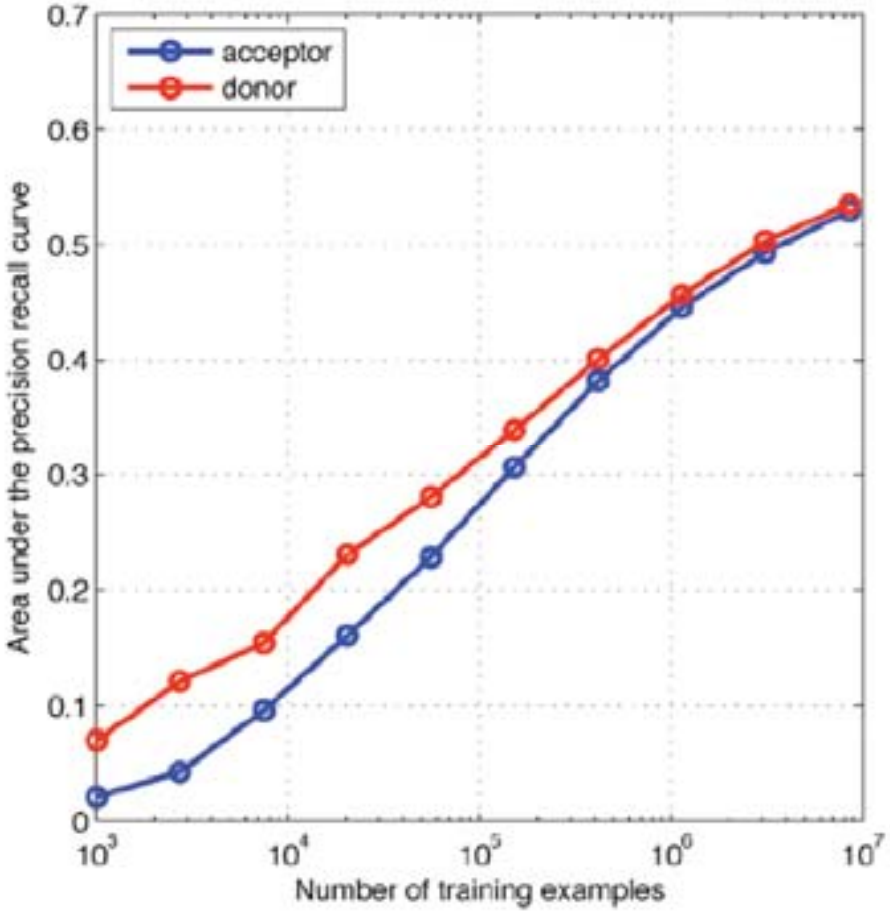
INITIALIZATION
$f_i = 0$, $\alpha_i = 0$ for $i = 1, \ldots, N$
LOOP UNTIL CONVERGENCE
`For` $t = 1, 2, \ldots$
  Check optimality conditions and stop if optimal
  Select working set W based on $\mathbf{g}$ and $\boldsymbol{\alpha}$, store $\boldsymbol{\alpha}^{old} = \boldsymbol{\alpha}$
  Solve reduced QP and update $\boldsymbol{\alpha}$
  `clear` $\mathbf{w}$
  $\mathbf{w} \leftarrow \mathbf{w} + (\alpha_j - \alpha_j^{old}) y_j \Phi(\mathbf{x}_j)$ for all $j \in W$
  Update $f_i = f_i + \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle$ for all $i = 1, \ldots, N$

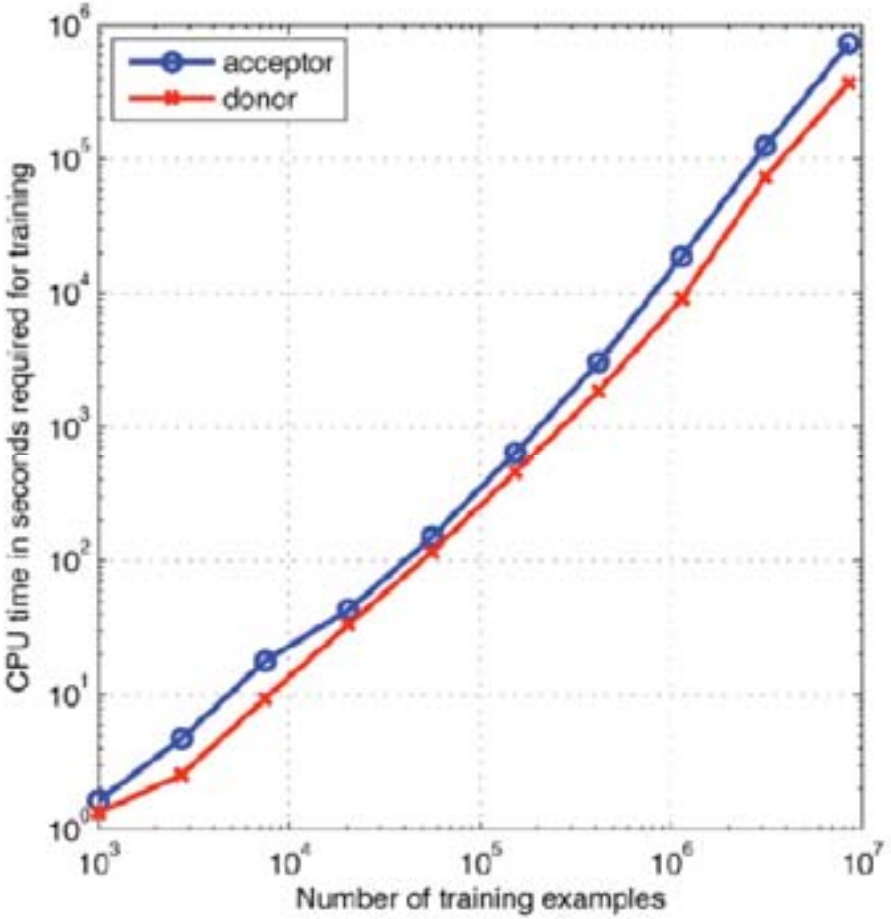
See Sonnenburg et al. [2007a] for more details.

All implemented in Shogun toolbox
(`http://www.shogun-toolbox.org`)

# Human Splice Sites with WD Kernel

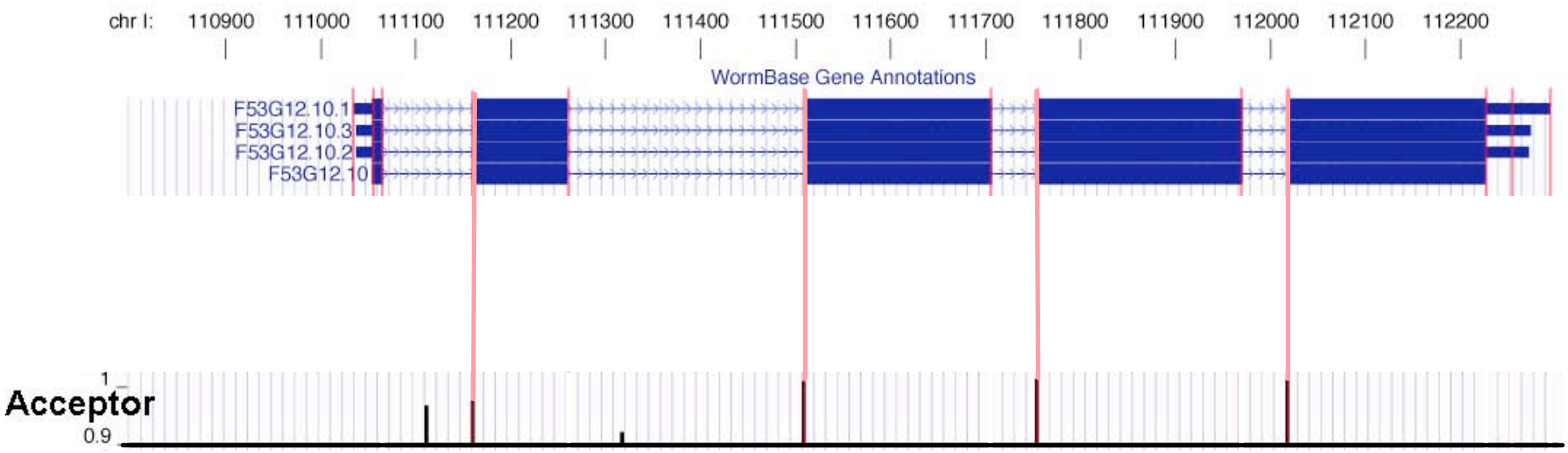




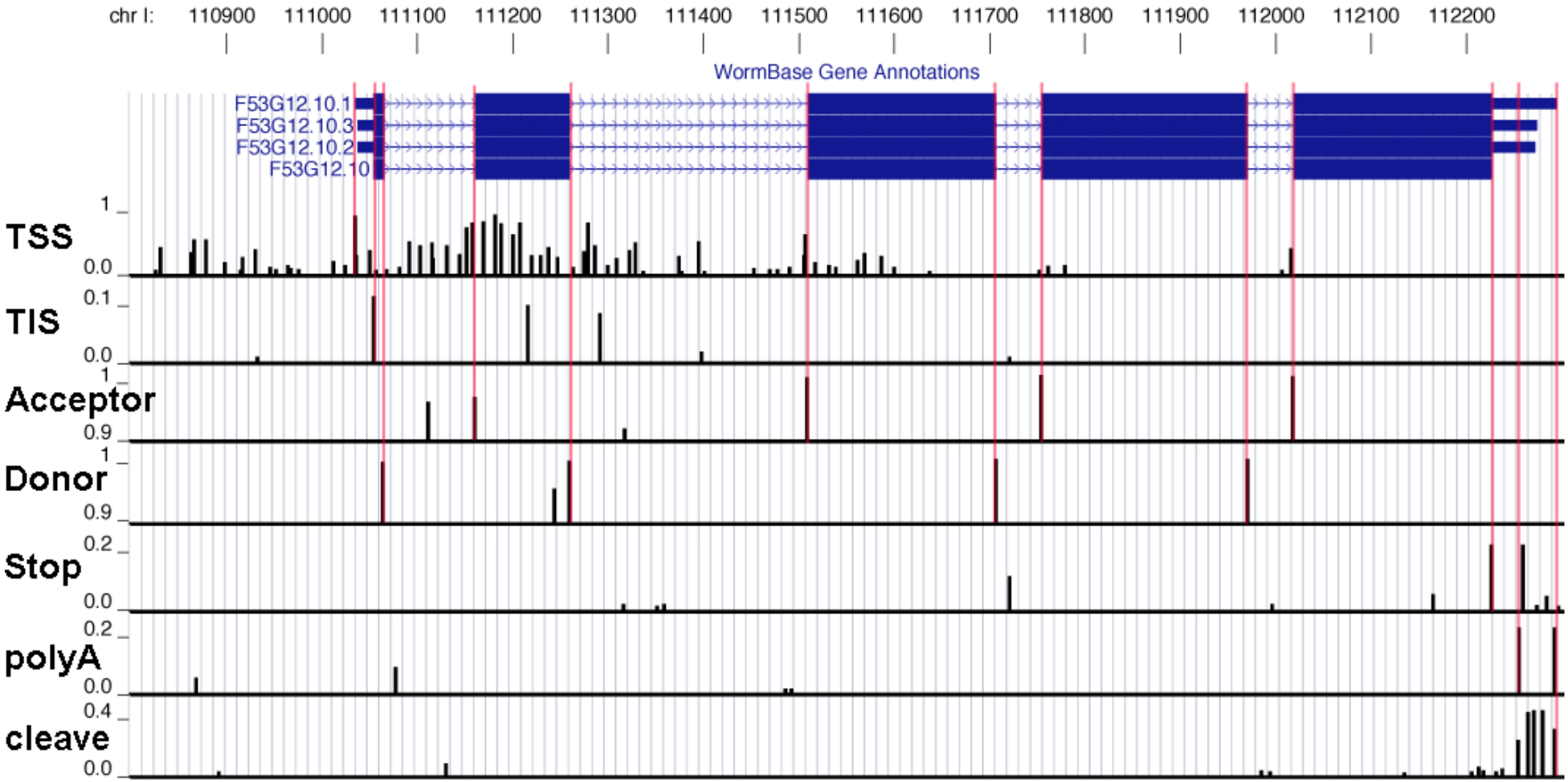
# Example: Predictions in UCSC Browser





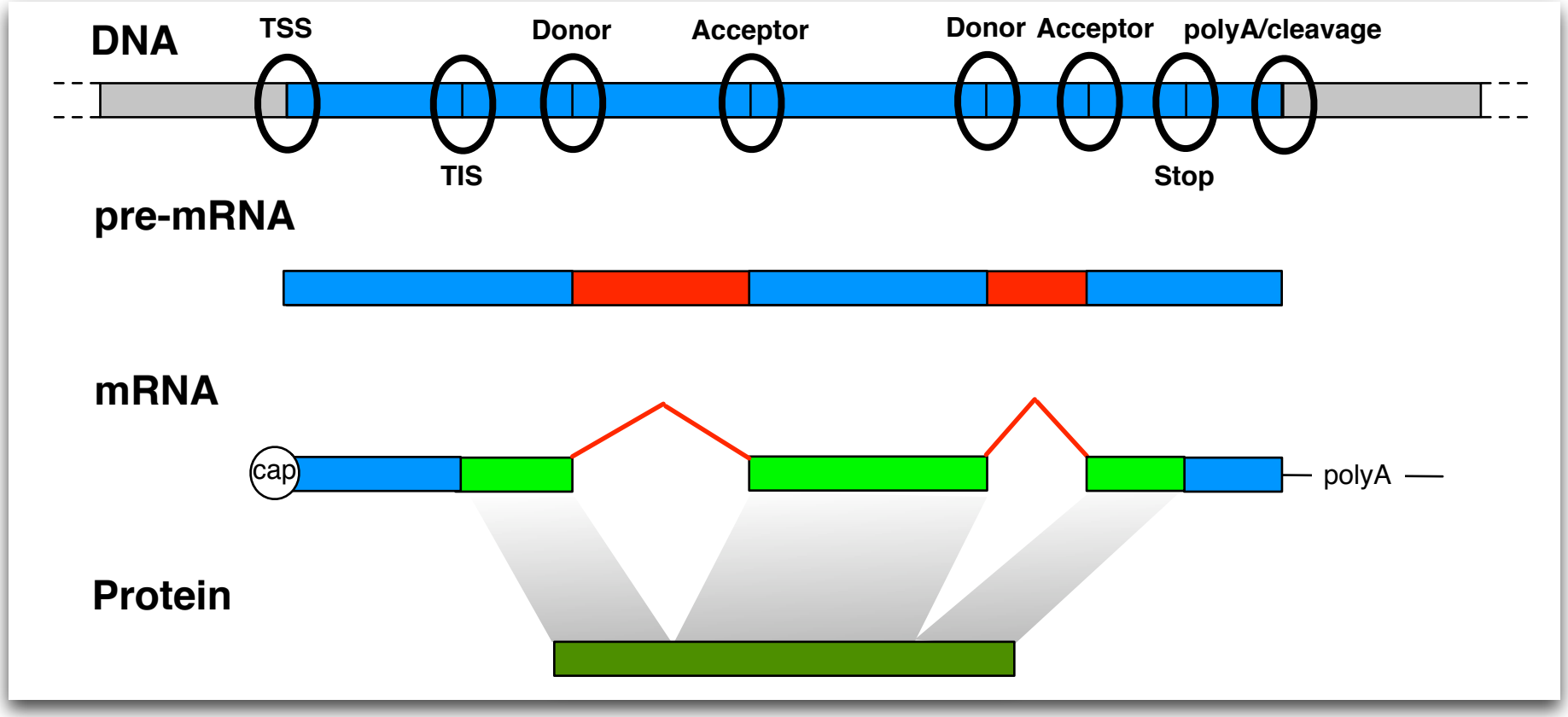
# Example: Predictions in UCSC Browser

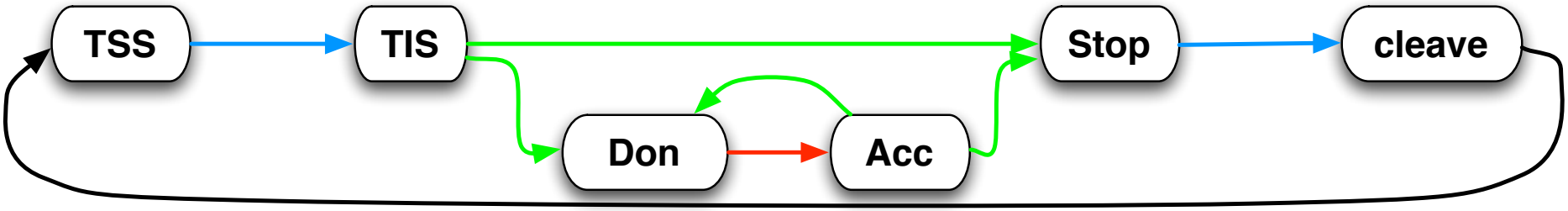




# Integration of Signals





# nGASP Competition

**Find the most accurate gene finder for annotation of new nematode genomes:**

- Highly controlled competition conditions
- 4 Categories:
  - Cat 1: *Ab initio* gene finders
  - Cat 2: Dual/Multi-genome gene finders
  - Cat 3: Gene finders that use EST/cDNA alignments
  - Cat 4: Combining algorithms
- 47 submitted predictions from 17 different groups, including Fgenesh, Augustus, N-SCAN
- Evaluation on *gold standard set* of genes

# Results: nGASP

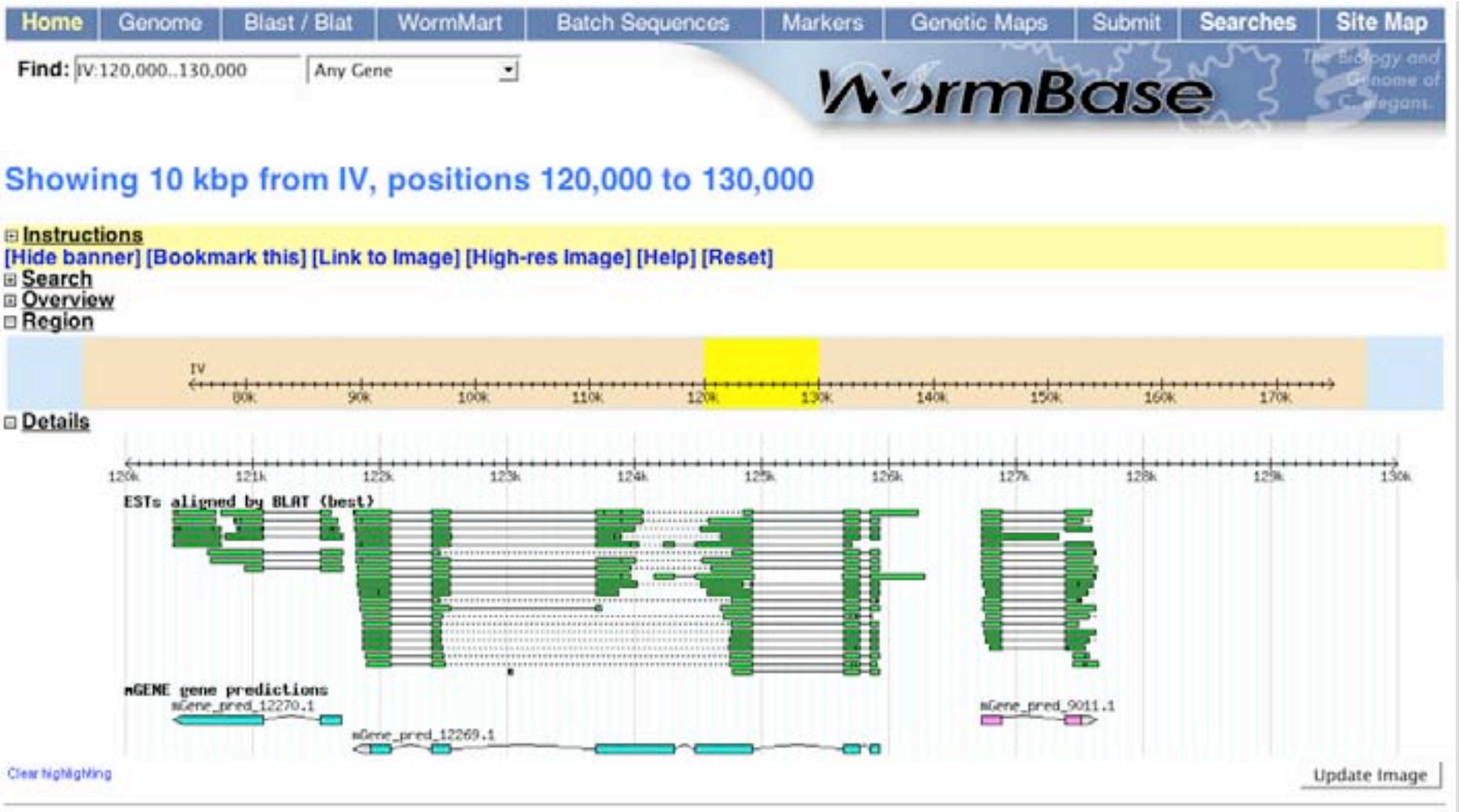


| Cat. | Method | Nucleotide Avg | Exon Avg | Transcript Avg | Gene Avg |
|---|---|---|---|---|---|
| 1 | mGene.init | **93.83** | **82.64** | **45.92** | **51.49** |
| 1 | Craig | 93.23 | 79.16 | 35.57 | 39.58 |
| 1 | EuGene | 91.72 | 76.64 | 38.64 | 44.16 |
| 1 | Fgenesh | 92.65 | 79.96 | 40.61 | 45.90 |
| 1 | Augustus | 93.01 | 79.34 | 40.77 | 49.42 |
| 2 | mGene.multi | **94.31** | **82.06** | **46.10** | **53.2** |
| 2 | N-SCAN | 92.73 | 77.17 | 33.43 | 38.23 |
| 2 | EuGene | 91.86 | 77.79 | 40.22 | 46.51 |
| 3 | mGene.seq | 94.46 | **87.83** | 58.97 | **66.30** |
| 3 | Gramene | **96.81** | 80.11 | 32.04 | 40.62 |
| 3 | Fgenesh++ | 93.64 | 85.68 | **59.27** | 65.30 |
| 3 | Augustus | 94.74 | 86.33 | 57.77 | 65.46 |

*mGene*: Most accurate method in the nGASP genome annotation challenge for *C. elegans*

# Results: *mGene* on Wormbase





# Results: Genome-wide Predictions



Annotation of other nematode genomes: (Schweikert et al., 2009)

| Genome | Genome size [Mbp] | No. of genes | No. exons/gene (mean) | *mGene* accuracy | best other accuracy |
|---|---|---|---|---|---|
| *C. remanei* | 235.94 | 31503 | 5.7 | **96.6%** | 93.8% |
| *C. japonica* | 266.90 | 20121 | 5.3 | **93.3%** | 88.7% |
| *C. brenneri* | 453.09 | 41129 | 5.4 | **93.1%** | 87.8% |
| *C. briggsae* | 108.48 | 22542 | 6.0 | **87.0%** | 82.0% |

- *C.elegans* model works well for closely related species.
- For intermediately distant organisms one can employ techniques to transfer learnt information. (Schweikert et al. 2009)
- For distantly related organisms retraining necessary:
  → Galaxy based web service `http://mgene.org/webservice`

# Results: Genome-wide Predictions



Annotation of other nematode genomes: (Schweikert et al., 2009)

| Genome | Genome size [Mbp] | No. of genes | No. exons/gene (mean) | *mGene* accuracy | best other accuracy |
|---|---|---|---|---|---|
| *C. remanei* | 235.94 | 31503 | 5.7 | **96.6%** | 93.8% |
| *C. japonica* | 266.90 | 20121 | 5.3 | **93.3%** | 88.7% |
| *C. brenneri* | 453.09 | 41129 | 5.4 | **93.1%** | 87.8% |
| *C. briggsae* | 108.48 | 22542 | 6.0 | **87.0%** | 82.0% |

- *C.elegans* model works well for closely related species.
- For intermediately distant organisms one can employ techniques to transfer learnt information. (Schweikert et al. 2009)
- For distantly related organisms retraining necessary:
  → Galaxy based web service `http://mgene.org/webservice`

# Results: Genome-wide Predictions



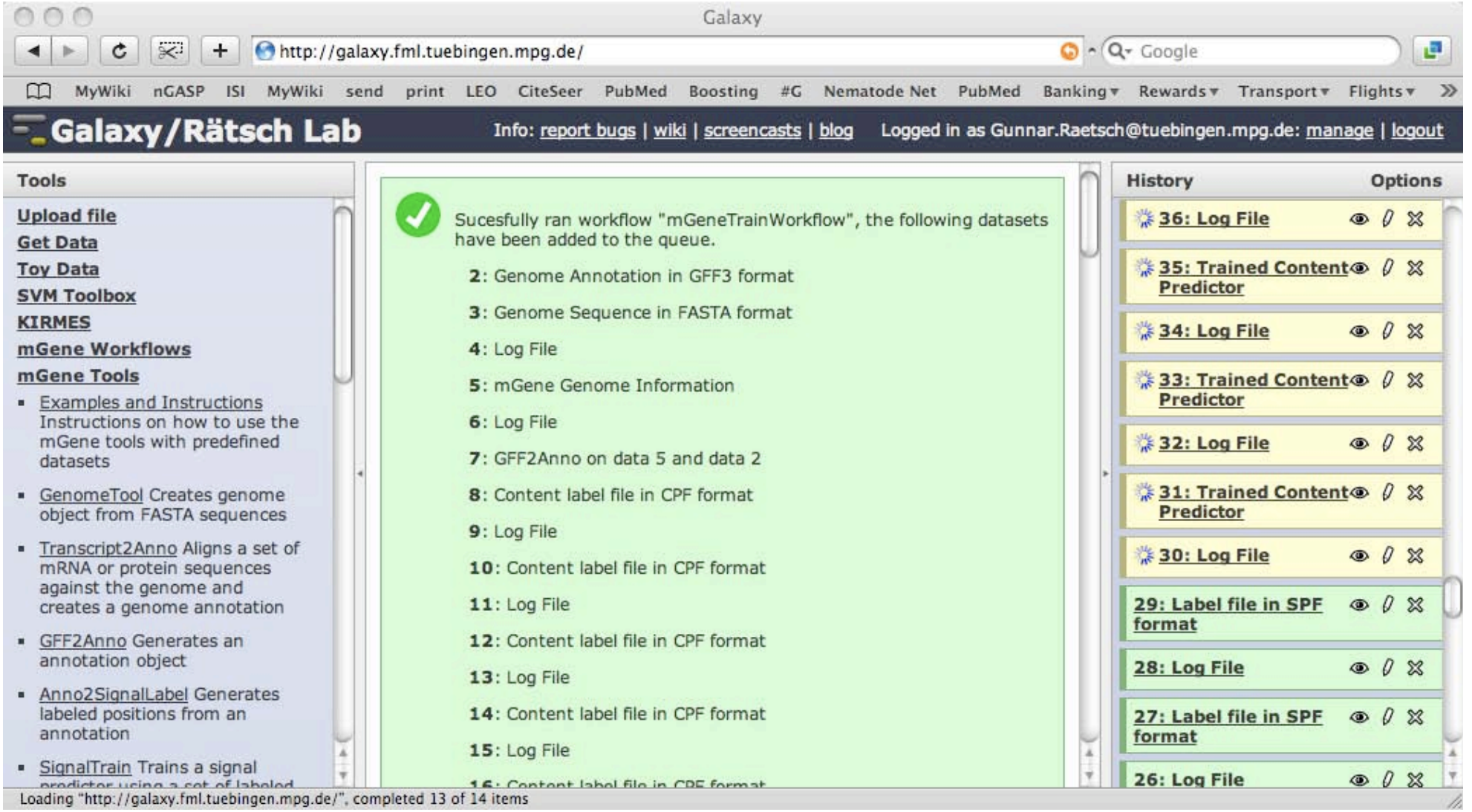
Annotation of other nematode genomes: (Schweikert et al., 2009)

| Genome | Genome size [Mbp] | No. of genes | No. exons/gene (mean) | *mGene* accuracy | best other accuracy |
|---|---|---|---|---|---|
| *C. remanei* | 235.94 | 31503 | 5.7 | **96.6%** | 93.8% |
| *C. japonica* | 266.90 | 20121 | 5.3 | **93.3%** | 88.7% |
| *C. brenneri* | 453.09 | 41129 | 5.4 | **93.1%** | 87.8% |
| *C. briggsae* | 108.48 | 22542 | 6.0 | **87.0%** | 82.0% |

- *C.elegans* model works well for closely related species.
- For intermediately distant organisms one can employ techniques to transfer learnt information. (Schweikert et al. 2009)
- For distantly related organisms retraining necessary:
  $\longrightarrow$ Galaxy based web service `http://mgene.org/webservice`

# **mGene.web**: Gene Finding for Everybody ;-)

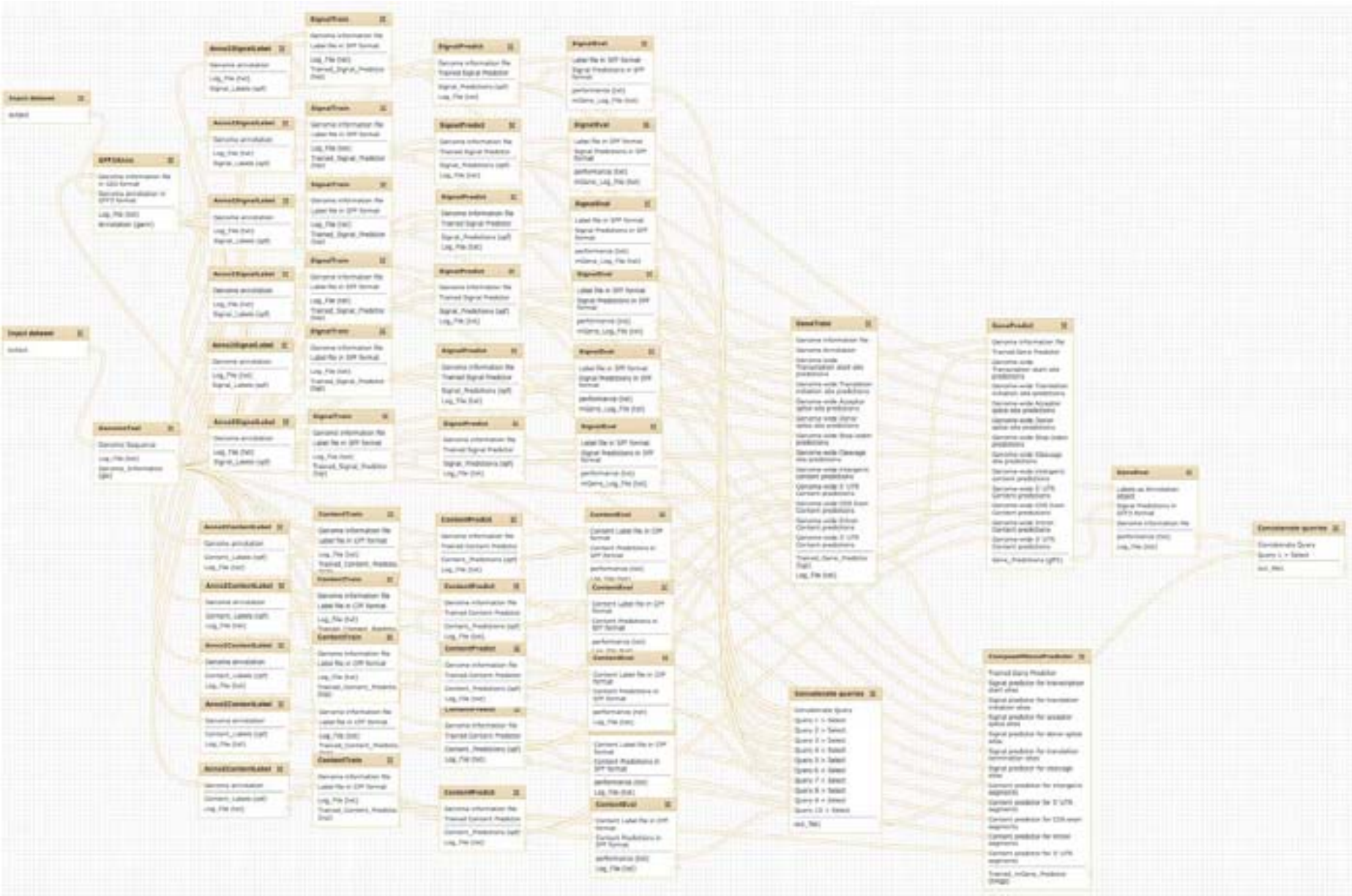
Friedrich Miescher Laboratory
of the Max Planck Society




http://mgene.org/webservice

# **mGene.web**: Gene Finding for Everybody ;-)

Friedrich Miescher Laboratory
of the Max Planck Society



# Limitations/Extensions



- Gene finding accuracy still far from perfect
  - Misses genes, predicts incorrect gene models
  - Does not (yet) predict alternative transcripts
  - Cannot predict when transcripts are expressed/modified/degraded...
- Accurate enough to accurately predict the effects of SNPs?
  - Annotate all the newly sequenced variations of genomes
  - Consensus site changes just the first step [Clark et al., 2007]
- Needs to be adapted to new genomes
  - Requires sufficient number of known gene models for "training"
  - Develop methods that exploit *evolutionary information* and gene models from other genomes [Schweikert et al., 2008]

$\Rightarrow$ Model and understand the differences in transcription, RNA processing & translation

# Limitations/Extensions



- Gene finding accuracy still far from perfect
  - Misses genes, predicts incorrect gene models
  - Does not (yet) predict alternative transcripts
  - Cannot predict when transcripts are expressed/modified/degraded...
- Accurate enough to accurately predict the effects of SNPs?
  - Annotate all the newly sequenced variations of genomes
  - Consensus site changes just the first step [Clark et al., 2007]
- Needs to be adapted to new genomes
  - Requires sufficient number of known gene models for "training"
  - Develop methods that exploit *evolutionary information* and gene models from other genomes [Schweikert et al., 2008]

  $\Rightarrow$ Model and understand the differences in transcription, RNA processing & translation.

# Limitations/Extensions



- Gene finding accuracy still far from perfect
  - Misses genes, predicts incorrect gene models
  - Does not (yet) predict alternative transcripts
  - Cannot predict when transcripts are expressed/modified/degraded...
- Accurate enough to accurately predict the effects of SNPs?
  - Annotate all the newly sequenced variations of genomes
  - Consensus site changes just the first step [Clark et al., 2007]
- Needs to be adapted to new genomes
  - Requires sufficient number of known gene models for "training"
  - Develop methods that exploit *evolutionary information* and gene models from other genomes [Schweikert et al., 2008]

$\Rightarrow$ Model and understand the differences in transcription, RNA processing & translation.

# Limitations/Extensions

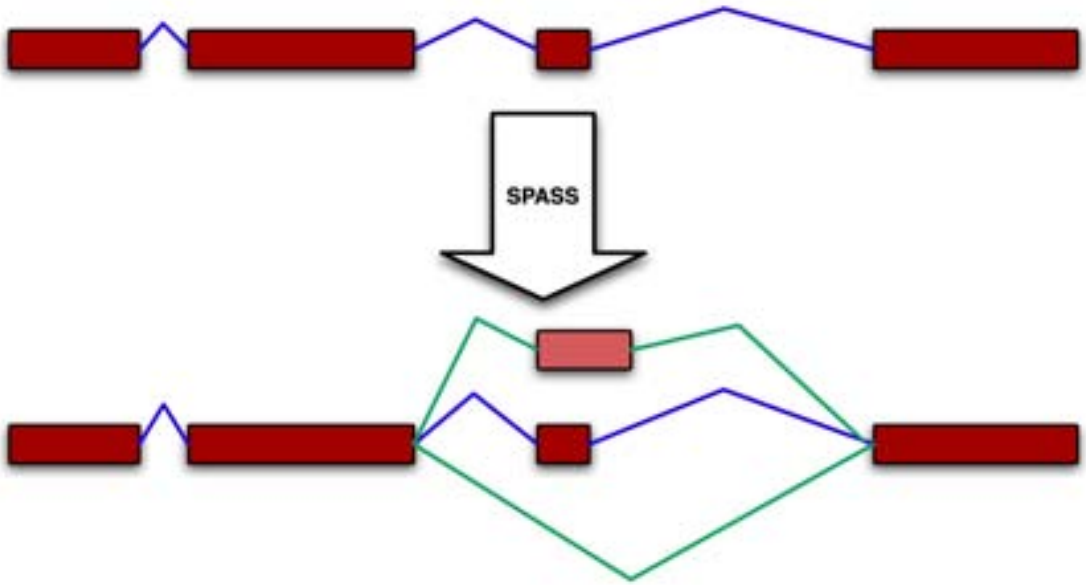


- Gene finding accuracy still far from perfect
  - Misses genes, predicts incorrect gene models
  - Does not (yet) predict alternative transcripts
  - Cannot predict when transcripts are expressed/modified/degraded...
- Accurate enough to accurately predict the effects of SNPs?
  - Annotate all the newly sequenced variations of genomes
  - Consensus site changes just the first step [Clark et al., 2007]
- Needs to be adapted to new genomes
  - Requires sufficient number of known gene models for "training"
  - Develop methods that exploit *evolutionary information* and gene models from other genomes [Schweikert et al., 2008]

  $\Rightarrow$ Model and understand the differences in transcription, RNA processing & translation.

# Alternative Splicing: First Steps

Predictions of alternative splicing




- Predict novel alternative splicing as independent events
- Use only information available to splicing machinery (Rätsch et. al, ISMB'05)
- Quite accurate for frequently appearing patterns
- Requires known gene structures

# Alternative Splicing: More Steps



- Combine gene finding with prediction of single alternative splicing events
  - Predict the **splice graph** of a gene
  - Machine learning challenge:
    - Input: DNA sequence
    - Output: Splice graph
- mSplicer approach can be extended to
  - Include predictions of alternative splicing events
  - Predict **"simple splice graphs"**
- Predicting arbitrary graphs is considerably harder

# Domain Adaptation for Classification



**Motivation:**

- Increasing number of sequenced genomes
- Often newly sequenced genomes are poorly annotated
- However often relatives with good annotation exist
- **Idea:** Transfer knowlege between organisms

**Study on domain adaptation for splice site prediction.**

**Example:** Splice site annotation in nematodes

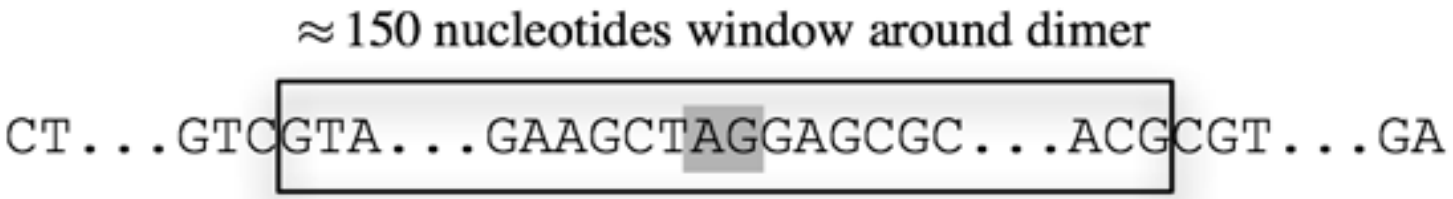
- Newly sequenced organism: *C. brennerei*
  - 590 confirmed splice site pairs
- Well annotated relative: *C. elegans*
  - 36782 confirmed splice site pairs

[Schweikert et al., NIPS'08]

# Splice Site Recognition



*Idea: Discriminate true signal positions against all other positions*




Binary classification problem

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

$\Rightarrow$ We learn a classification model from labeled training examples

# Formal definition of Domain Adaptation



Terminology:

- Well annotated organisms: Source domain
- Poorly annotated organisms: Target domain

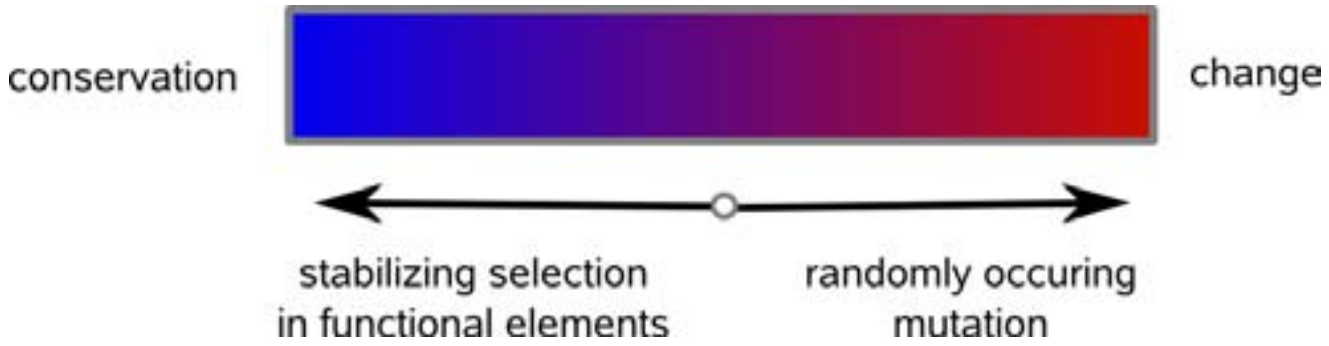
Distributional point of view:

- In Supervised Learning, example-label pairs are drawn from $P(X, Y)$
- $P_S(X, Y)$ might differ from $P_T(X, Y)$

- Factorization: $P(X, Y) = P(Y|X) \cdot P(X)$
  - Covariate Shift: $P_S(X) \neq P_T(X)$
  - Differing Conditionals: $P_S(Y|X) \neq P_T(Y|X)$

# Splice Site Prediction and Domain Adaptation

- Sequence subject to opposing forces:




- $P_S(X) \neq P_T(X)$
  - Assume a splicesite pattern **x** occurs more frequently in a group of genes (e.g. chromosome)
  - Duplication or deletion events could lead to altered $P(X)$
- $P_S(Y|X) \neq P_T(Y|X)$
  - Think of the conditional as underlying mechanism
  - Evolution of splicing machinery

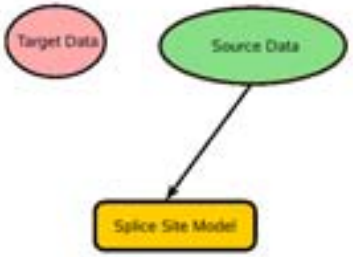
# Domain Adaptation Algorithms Overview



TargetOnly




SourceOnly



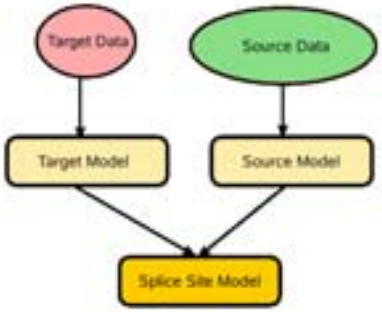

Combined



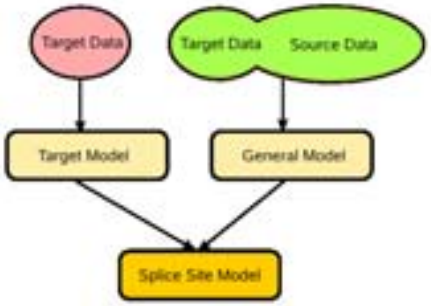

KMM/Shift



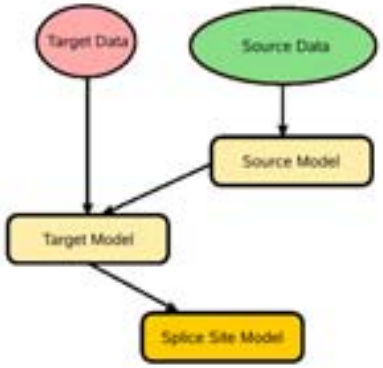

Convex



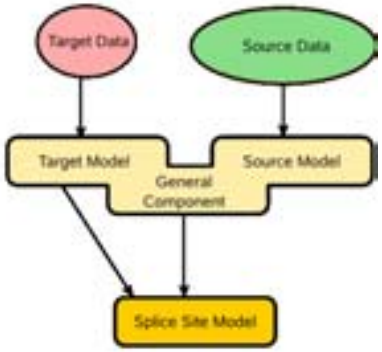

CombinedConvex



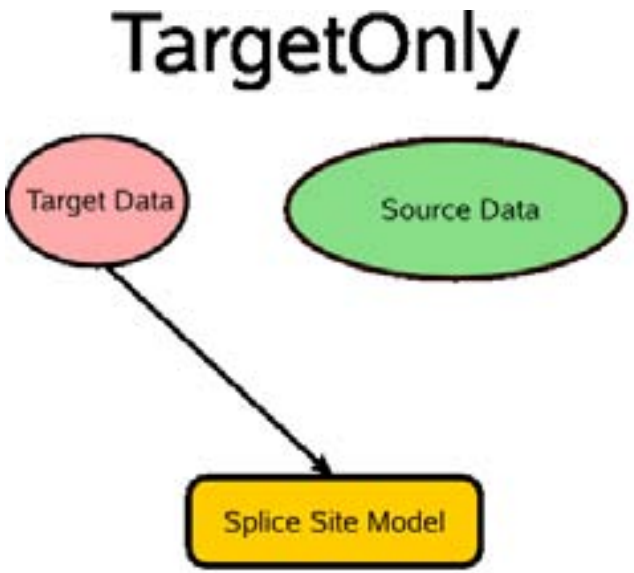

Regtrick

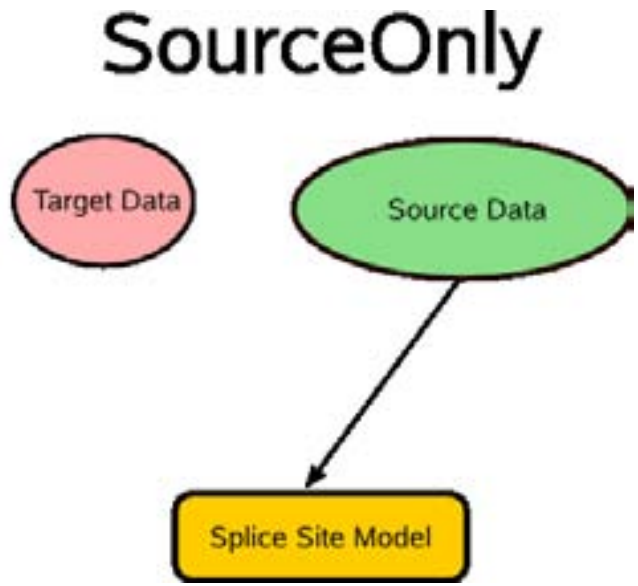



DualTask



# Domain Adaptation Methods




Formula:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) + \xi_i - 1 \geq 0 \;\forall i \in [1,n]$$

$$\xi_i \geq 0 \;\forall i \in [1,n]$$

Resulting Model: $f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} + b)$

# Domain Adaptation Methods




Idea:

- Train on union of source and target
- Set trade-off via loss-term

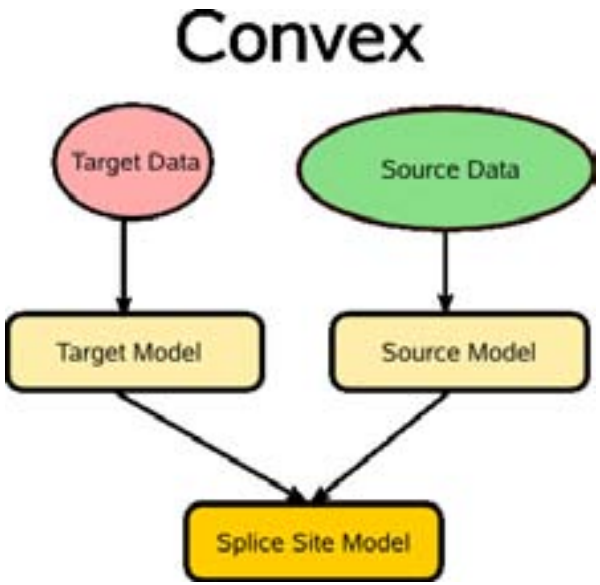
Formula:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_S \sum_{i=1}^{n} \xi_i + C_T \sum_{i=1}^{m} \xi_i$$

$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) + \xi_i - 1 \geq 0 \;\forall i \in [1, n+m]$$

# Domain Adaptation Methods






Idea:

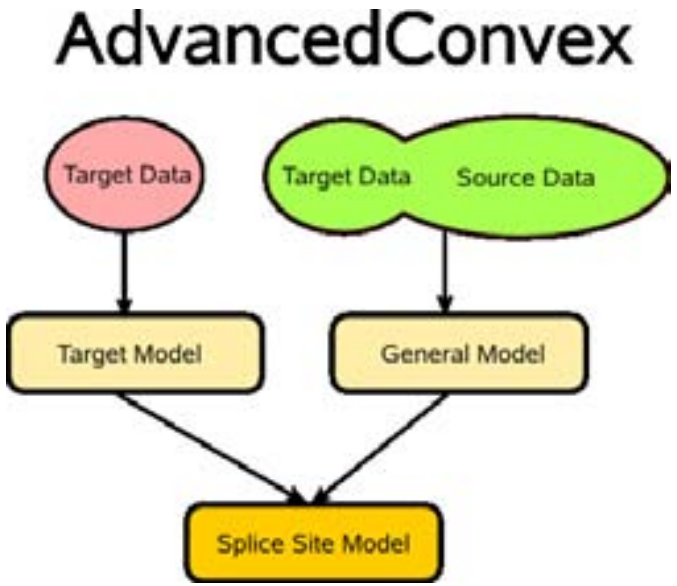
- Combine trained models
- Efficient hyperparameter-optimization

Formula:

$$F(x) = \alpha f_S(x) + (1 - \alpha) f_T(x)$$

# Domain Adaptation Methods





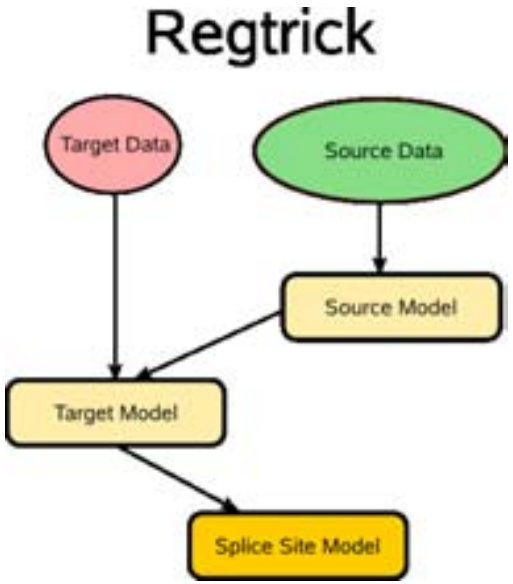

Idea:

- Takes interactions between source and target examples into account
- Two times linear search spaces for individual methods
- Captures 'General' and 'Target-specific' component

Formula:

$$F(x) = \alpha f_C(x) + (1 - \alpha) f_T(x)$$

# Domain Adaptation Methods




Idea:

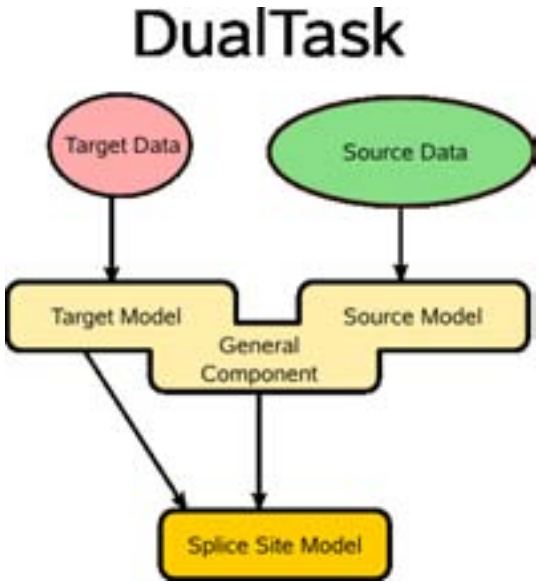
- Previous solution contains prior information
- Modified regularization term

Formula:

$$\min_{\mathbf{w}_T,\xi} \frac{1}{2}\mathbf{w}_T^T\mathbf{w}_T + C\sum_{i=1}^{n}\xi_i - B\mathbf{w}_T^T\mathbf{w}_S$$

$$\text{s.t. } y_i(\mathbf{w}_T^T\mathbf{x}_i + b) + \xi_i - 1 \geq 0 \;\forall i \in [1,n]$$

# Domain Adaptation Methods




Idea:

- Simultaneous optimization of both models
- Similarity between solution enforced

Formula:

$$\min_{\mathbf{w}_S, \mathbf{w}_T, \xi} \quad \frac{1}{2}\|\mathbf{w}_S - \mathbf{w}_T\|^2 + C \sum_{i=1}^{m+n} \xi_i \tag{1}$$

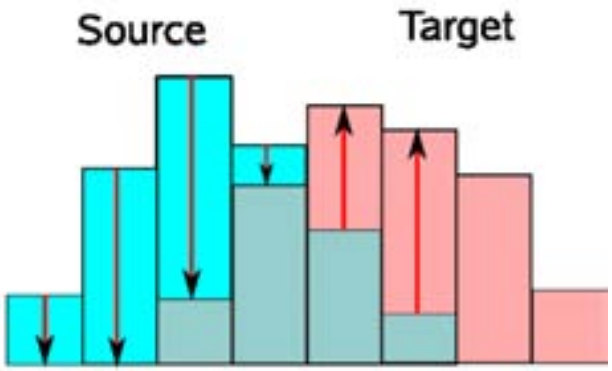
$$\text{s.t.} \quad y_i(\langle \mathbf{w}_S, \Phi(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i \qquad \forall i \in 1, \ldots, m$$

$$y_i(\langle \mathbf{w}_T, \Phi(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i \qquad \forall i \in m+1, \ldots, m+n$$

# Domain Adaption Methods

## Kernel Mean Matching




Idea:

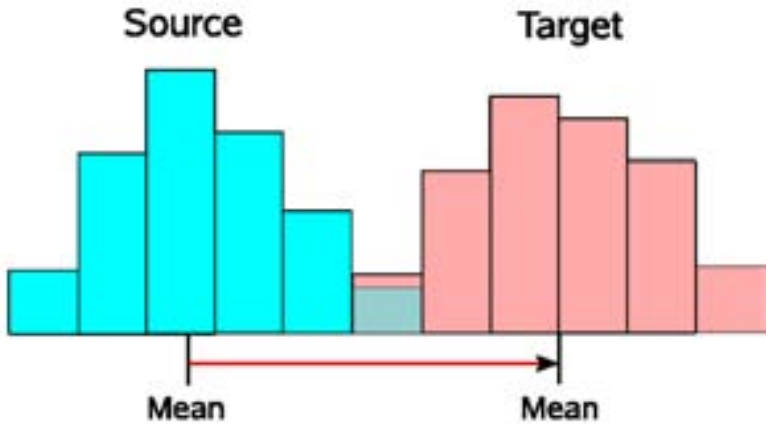
- Match mean of source and target by reweighting examples
- Higher-order moments defined by mean when using a universal kernel

[Huang, Smola, Gretton, Borgwardt, Schölkopf, 2007]

# Domain Adaption Methods

## Shift (KMM)



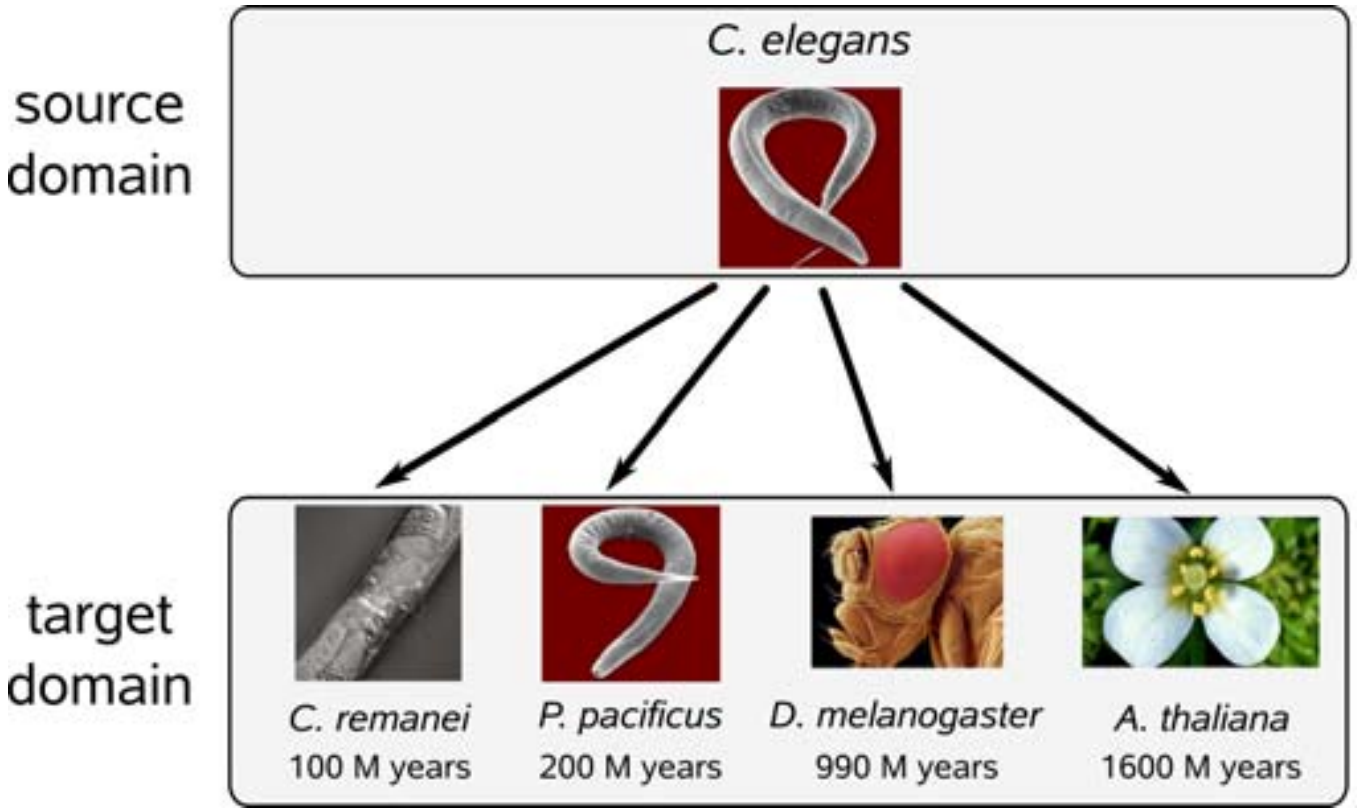




Idea:

- Based on same assumption
- Mean matching via translation rather than reweighting

# Large scale experiments






- Varying distances
- Different data set sizes

[MPI Developmental Biology, Departments 4 & 6 and UCSC Genome Browser]

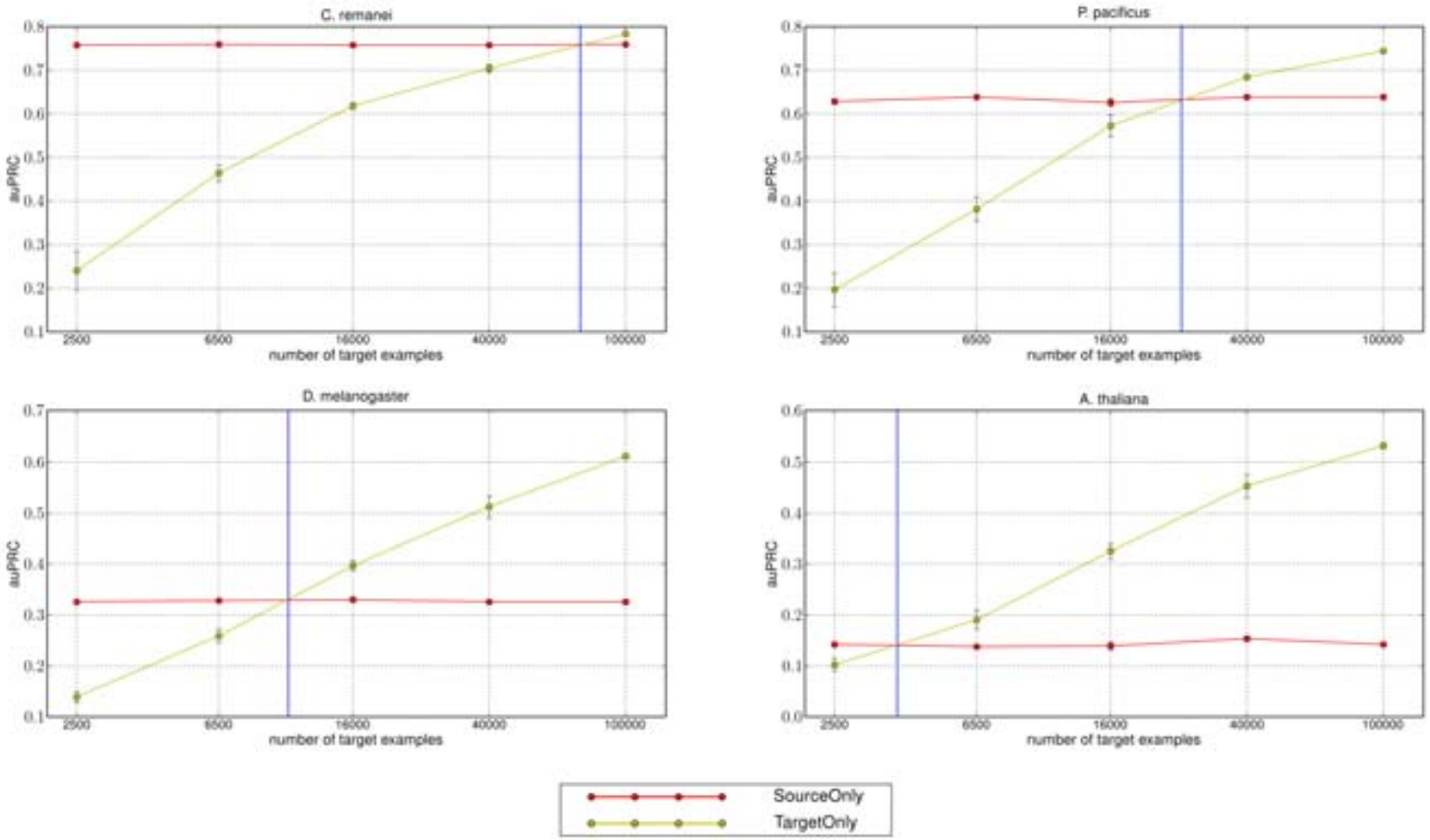
# Experimental Setup



- Source dataset size: always 100k examples
- Target dataset sizes: {2500, 6500, 16000, 64000, 100000}
- Simple kernel (WDK of degree 1)
- Model selection for each method
- auROC/auPRC measured for each setting

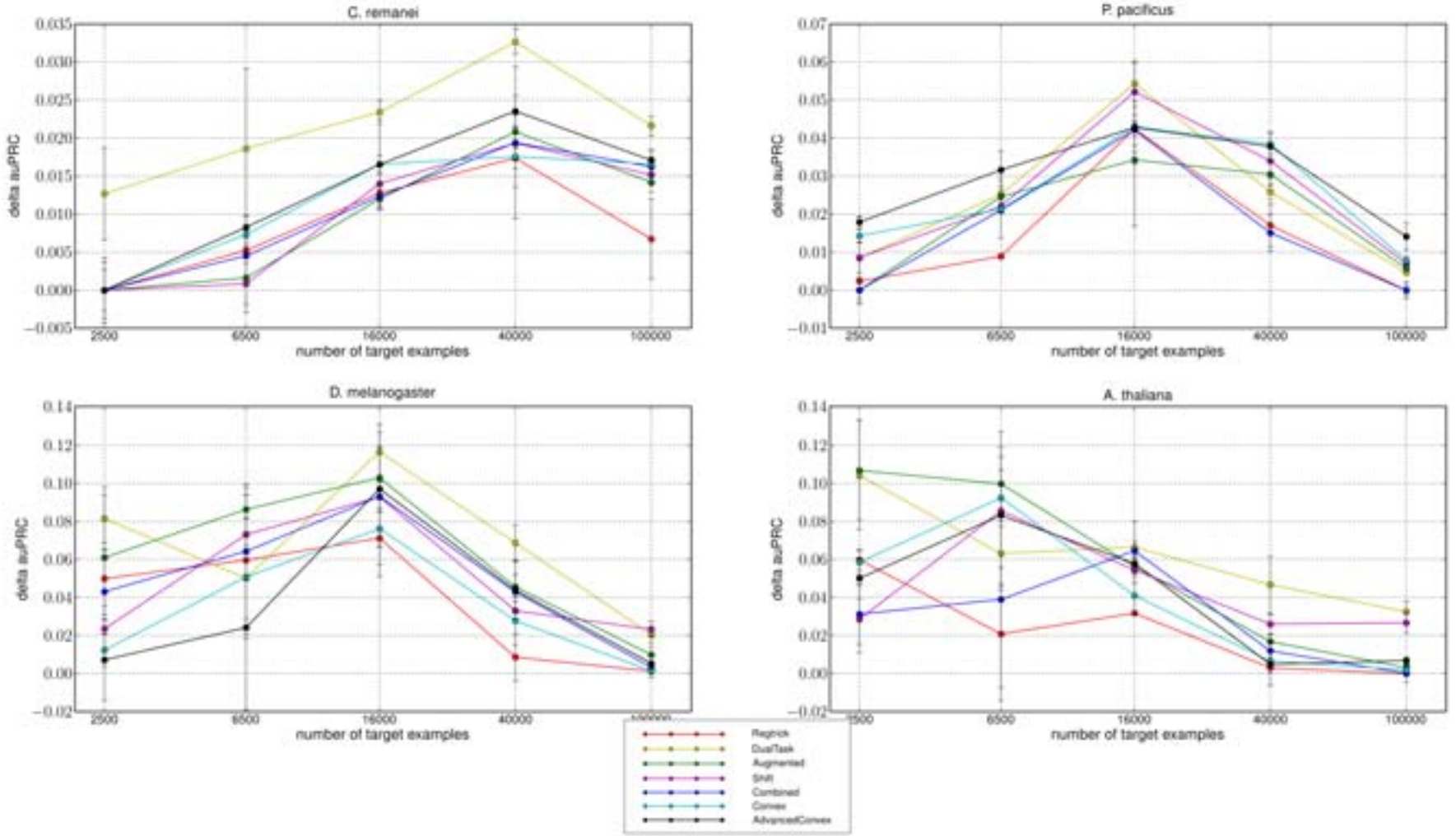
# Results - Baseline Methods





# Results - Improvement over Baseline Methods

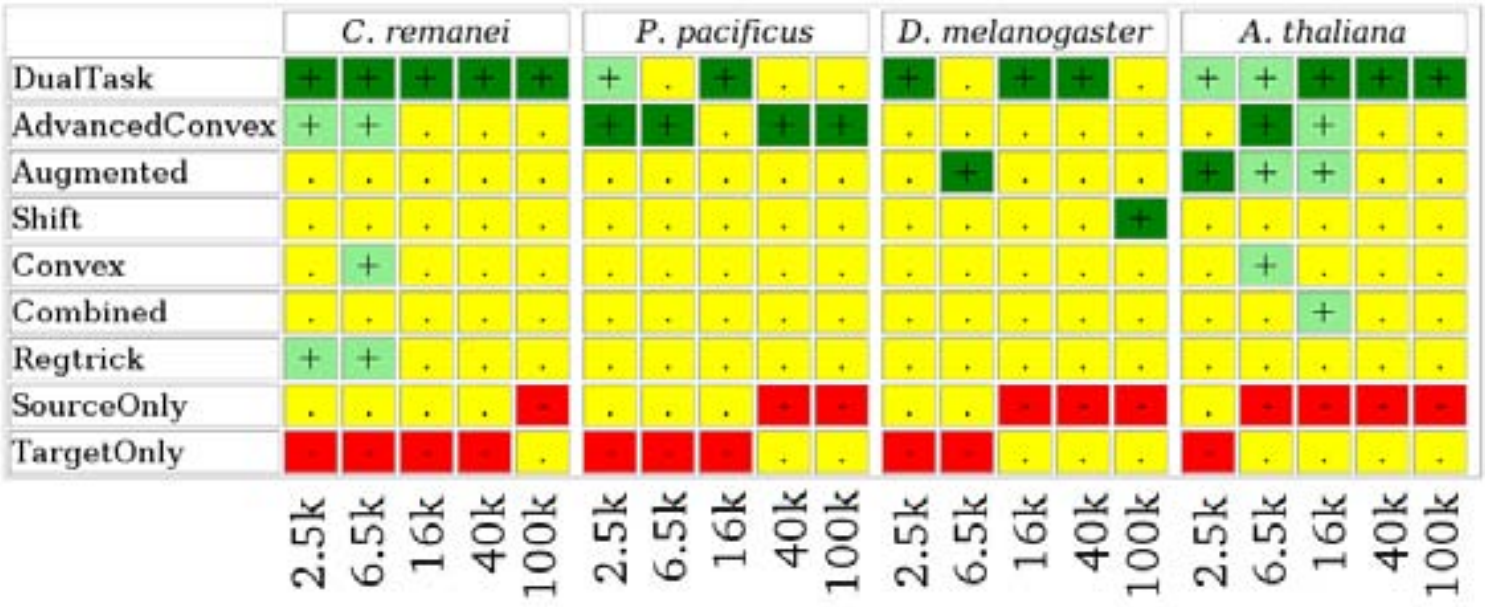




# Results - Summary



- Considerable improvements possible
- Sophisticated domain adaptation methods needed on distantly related organisms
- Best overall performance has *DualTask*
- Most cost effective *Convex/AdvancedConvex*



# Domain Adaptation for LSL



**Problem:**

- Very little data
  - Relatively small amount of sequences available
  - Only ≈50 well analysed genes

Solution:

- Exploit that *P. pacificus* is closely related to *C. elegans*
- Can use *C. elegans* signal and content sensors
- But how can *adapt* the *C. elegans* parameters for gene structure prediction?

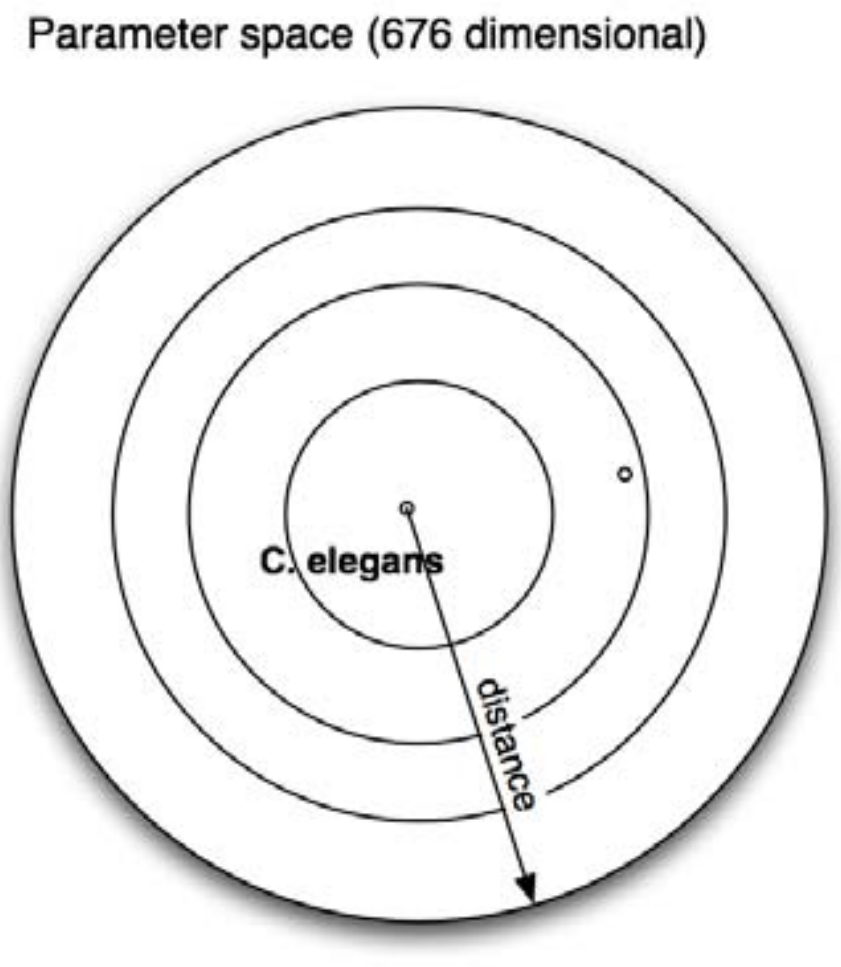
# Domain Adaptation for LSL

**Problem:**

- Very little data
  - Relatively small amount of sequences available
  - Only ≈50 well analysed genes

Solution:

- Exploit that *P. pacificus* is closely related to *C. elegans*
- Can use *C. elegans* signal and content sensors
- But how can *adapt* the *C. elegans* parameters for gene structure prediction?



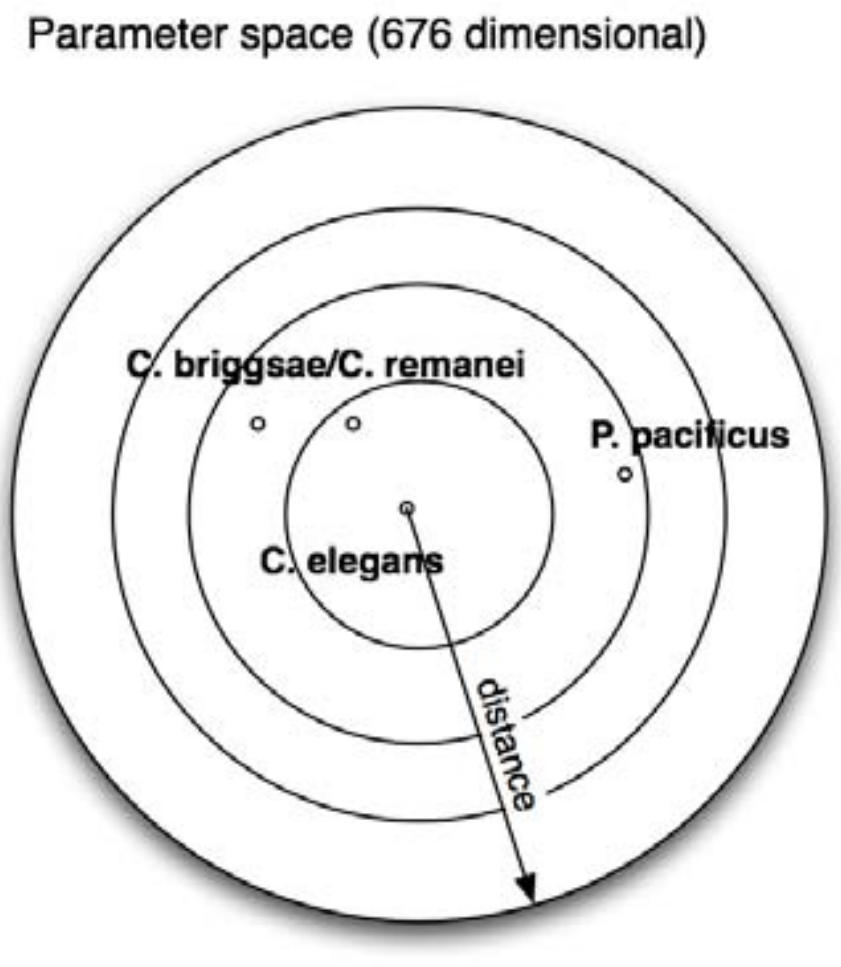
# Domain Adaptation for LSL

**Problem:**

- Very little data
  - Relatively small amount of sequences available
  - Only ≈50 well analysed genes

Solution:

- Exploit that *P. pacificus* is closely related to *C. elegans*
- Can use *C. elegans* signal and content sensors
- But how can *adapt* the *C. elegans* parameters for gene structure prediction?



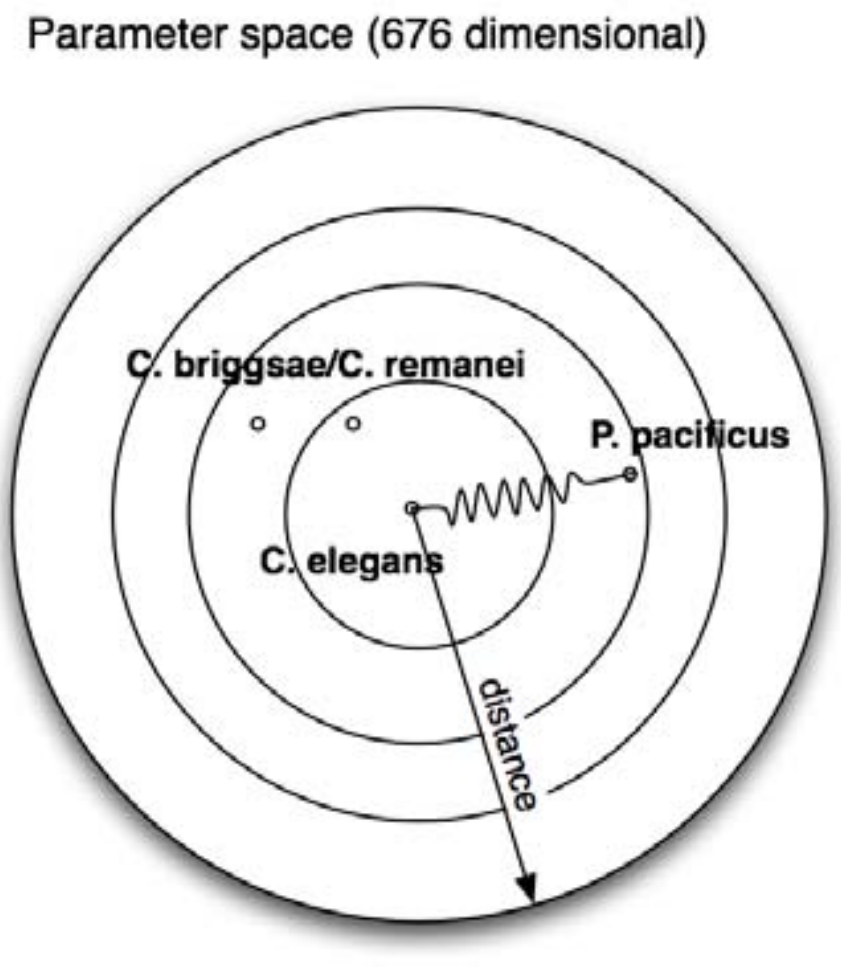
# Domain Adaptation for LSL

**Problem:**

- Very little data
  - Relatively small amount of sequences available
  - Only ≈50 well analysed genes

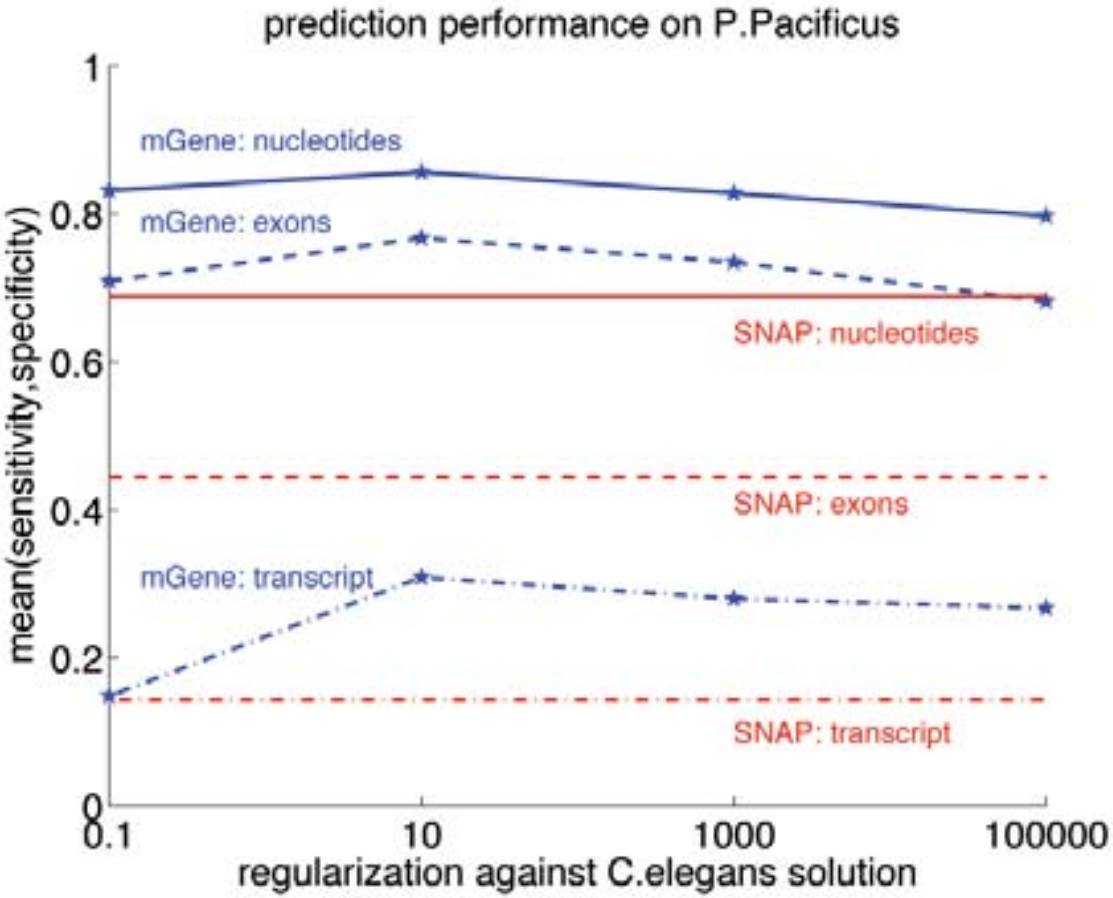
Solution:

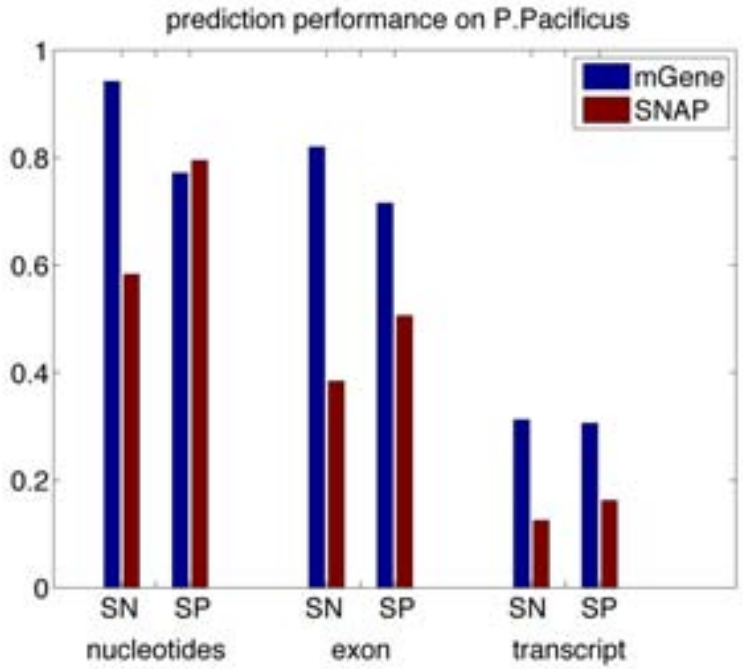
- Exploit that *P. pacificus* is closely related to *C. elegans*
- Can use *C. elegans* signal and content sensors
- But how can *adapt* the *C. elegans* parameters for gene structure prediction?



# Domain Adaptation for LSL





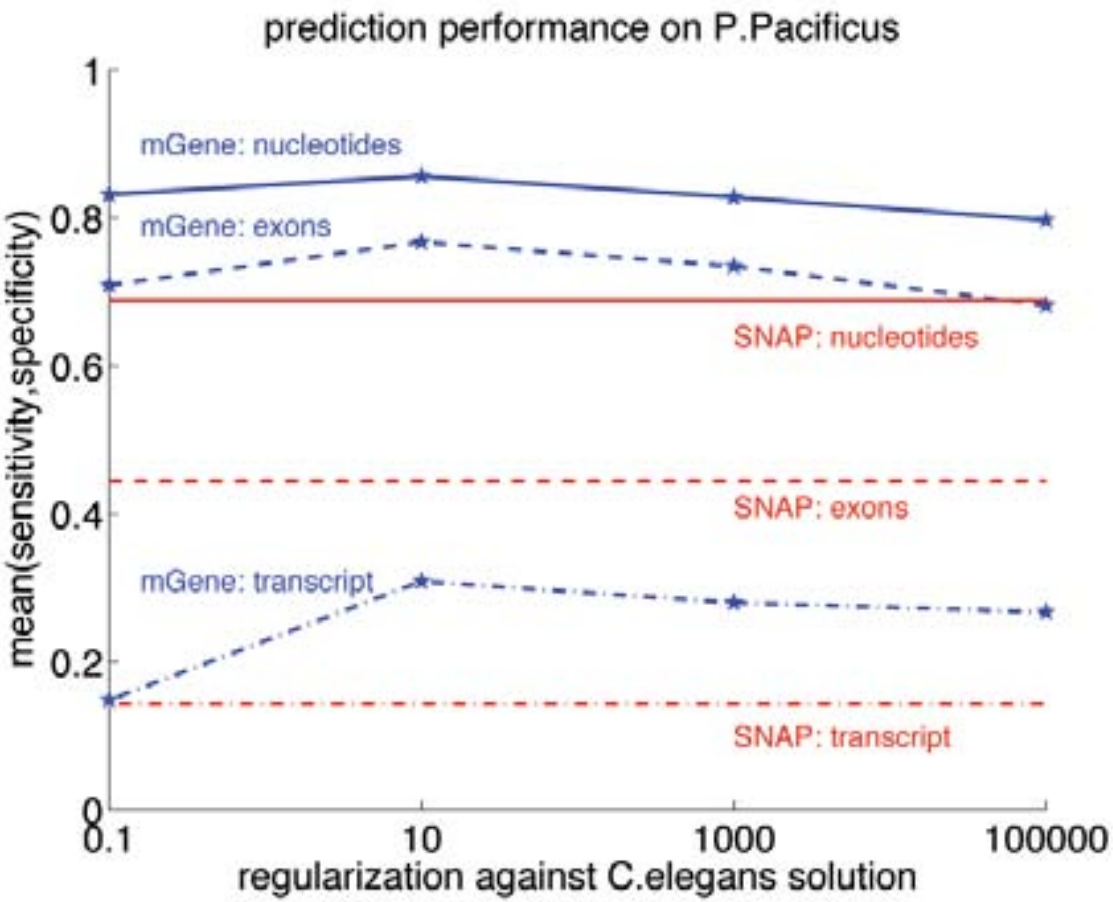



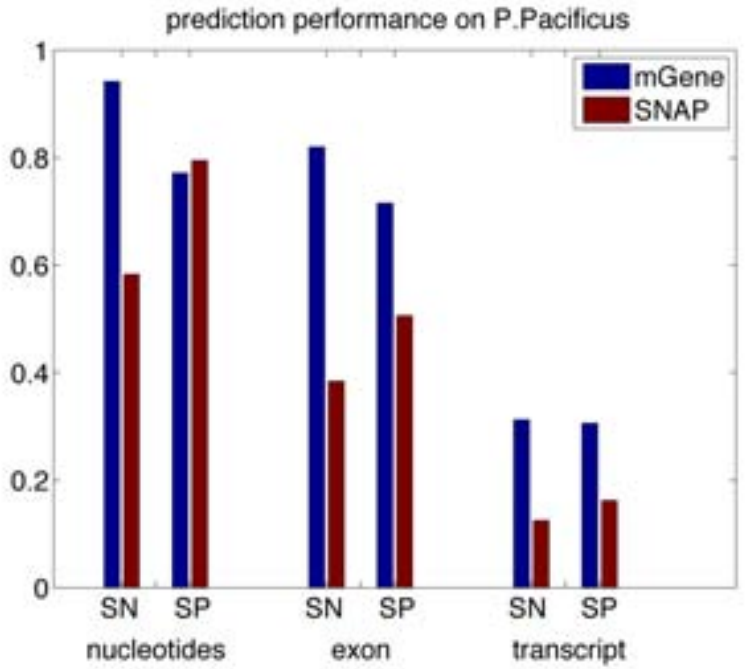

Details:

- Preliminary results!
- Signal predictions from *C. elegans*
- Training using 212 SNAP/EST gene models
- Parameter regularization against *C. elegans* solution
- Testing on 48 regions around known cDNAs (±1000nt)

# Domain Adaptation for LSL






Details:

- Preliminary results!
- Signal predictions from *C. elegans*
- Training using 212 SNAP/EST gene models
- Parameter regularization against *C. elegans* solution
- Testing on 48 regions around known cDNAs ($\pm$1000nt)

# Summary and Future Work



- Genome Annotation is a huge structured output learning problem
- Proposed a two-step learning procedure separating the kernels from the structured output prediction
- Sequence classification already challenging (large!)
- String data structures make training feasible
- Gene prediction is more difficult in reality
  - Predict splice graphs/alternative transcripts
  - Regulation!? Auxiliary data!?
- Domain Adaptation
  - First thorough comparison of Domain Adaptation Algorithms
  - Learn models for multiple organisms simultaneously
  - Develop more efficient training procedure
- Integrate these ideas into the gene finder
  - Annotate the thousands of genomes that are currently being sequenced

# Summary and Future Work



- Genome Annotation is a huge structured output learning problem
- Proposed a two-step learning procedure separating the kernels from the structured output prediction
- Sequence classification already challenging (large!)
- String data structures make training feasible
- Gene prediction is more difficult in reality
  - Predict splice graphs/alternative transcripts
  - Regulation!? Auxiliary data!?
- Domain Adaptation
  - First thorough comparison of Domain Adaptation Algorithms
  - Learn models for multiple organisms simultaneously
  - Develop more efficient training procedure
- Integrate these ideas into the gene finder
  - Annotate the thousands of genomes that are currently being sequenced

# Summary and Future Work



- Genome Annotation is a huge structured output learning problem
- Proposed a two-step learning procedure separating the kernels from the structured output prediction
- Sequence classification already challenging (large!)
- String data structures make training feasible
- Gene prediction is more difficult in reality
  - Predict splice graphs/alternative transcripts
  - Regulation!? Auxiliary data!?
- Domain Adaptation
  - First thorough comparison of Domain Adaptation Algorithms
  - Learn models for multiple organisms simultaneously
  - Develop more efficient training procedure
- Integrate these ideas into the gene finder
  - Annotate the thousands of genomes that are currently being sequenced

# Summary and Future Work



- Genome Annotation is a huge structured output learning problem
- Proposed a two-step learning procedure separating the kernels from the structured output prediction
- Sequence classification already challenging (large!)
- String data structures make training feasible
- Gene prediction is more difficult in reality
  - Predict splice graphs/alternative transcripts
  - Regulation!? Auxiliary data!?
- Domain Adaptation
  - First thorough comparison of Domain Adaptation Algorithms
  - Learn models for multiple organisms simultaneously
  - Develop more efficient training procedure
- Integrate these ideas into the gene finder
  - Annotate the thousands of genomes that are currently being sequenced

# Summary and Future Work



- Genome Annotation is a huge structured output learning problem
- Proposed a two-step learning procedure separating the kernels from the structured output prediction
- Sequence classification already challenging (large!)
- String data structures make training feasible
- Gene prediction is more difficult in reality
  - Predict splice graphs/alternative transcripts
  - Regulation!? Auxiliary data!?
- Domain Adaptation
  - First thorough comparison of Domain Adaptation Algorithms
  - Learn models for multiple organisms simultaneously
  - Develop more efficient training procedure
- Integrate these ideas into the gene finder
  - Annotate the thousands of genomes that are currently being sequenced

# Acknowledgments


Friedrich Miescher Laboratory
of the Max Planck Society

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

**Gene Finding**
Gabi Schweikert (FML/MPI)
Jonas Behr (FML)
Alex Zien (FML & FIRST)
Georg Zeller (FML/MPI)

**Domain Adaptation**
Christian Widmer (FML)
Gabi Schweikert (FML/MPI)
Bernhard Schölkopf (MPI)


## More Information

- http://www.fml.mpg.de/raetsch
- http://www.shogun-toolbox.org
- http://www.mgene.org/webservice
- Slides with references are available online

**Thank you!**

# Acknowledgments


Friedrich Miescher Laboratory
of the Max Planck Society

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

**Gene Finding**
Gabi Schweikert (FML/MPI)
Jonas Behr (FML)
Alex Zien (FML & FIRST)
Georg Zeller (FML/MPI)

**Domain Adaptation**
Christian Widmer (FML)
Gabi Schweikert (FML/MPI)
Bernhard Schölkopf (MPI)


## More Information

- `http://www.fml.mpg.de/raetsch`
- `http://www.shogun-toolbox.org`
- `http://www.mgene.org/webservice`
- Slides with references are available online

Thank you!

# Acknowledgments

Friedrich Miescher Laboratory
of the Max Planck Society

**Sequence Analysis**

Sören Sonnenburg (FML/FIRST)
Gabi Schweikert (FML/MPI)
Alex Zien (FML & FIRST)
Konrad Rieck (FIRST)

**Gene Finding**

Gabi Schweikert (FML/MPI)
Jonas Behr (FML)
Alex Zien (FML & FIRST)
Georg Zeller (FML/MPI)

**Domain Adaptation**

Christian Widmer (FML)
Gabi Schweikert (FML/MPI)
Bernhard Schölkopf (MPI)

**More Information**

- `http://www.fml.mpg.de/raetsch`
- `http://www.shogun-toolbox.org`
- `http://www.mgene.org/webservice`
- Slides with references are available online

**Thank you!**

# References I


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, May 2008.

RM Clark, G Schweikert, C Toomajian, S Ossowski, G Zeller, P Shinn, N Warthmann, TT Hu, G Fu, DA Hinds, H Chen, KA Frazer, DH Huson, B Schölkopf, M Nordborg, G Rätsch, JR Ecker, and D Weigel. Common sequence polymorphisms shaping genetic diversity in arabidopsis thaliana. *Science*, 317(5836):338–342, 2007. ISSN 1095-9203 (Electronic). doi: 10.1126/science.1138632.

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.

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.

G. Schweikert, G. Zeller, A. Zien, J. Behr, C.S. Ong, P. Philips, A. Bohlen, R. Bohnert, F. De Bona, S. Sonnenburg, and G. Rätsch. mGene: Accurate computational gene finding with application to nematode genomes. under revision, March 2009.

# References II


Gabriele Schweikert, Christian Widmer, Bernhard Schölkopf, and Gunnar Rätsch. An empirical analysis of domain adaptation algorithms. In *Proc. NIPS 2008*, Advances in Neural Information Processing Systems, 2008. accepted.

S. Sonnenburg, G. Rätsch, A. Jagota, and K.-R. Müller. New methods for splice-site recognition. In *Proc. International Conference on Artificial Neural Networks*, 2002.

S. Sonnenburg, G. Rätsch, and K. Rieck. *Large Scale Kernel Machines*, chapter Large Scale Learning with String Kernels. MIT Press, 2007a.

S Sonnenburg, G Schweikert, P Philips, J Behr, and G Rätsch. Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, 8 Suppl 10:S7, 2007b. ISSN 1471-2105 (Electronic). doi: 10.1186/1471-2105-8-S10-S7.

Sören Sonnenburg, Alexander Zien, and Gunnar Rätsch. ARTS: Accurate recognition of transcription starts in human. *Bioinformatics*, 22(14):e472–480, 2006.

G Zeller, RM Clark, K Schneeberger, A Bohlen, D Weigel, and G Ratsch. Detecting polymorphic regions in arabidopsis thaliana with resequencing microarrays. *Genome Res*, 18 (6):918–929, 2008a. ISSN 1088-9051 (Print). doi: 10.1101/gr.070169.107.

G. Zeller, S.R. Henz, S. Laubinger, D. Weigel, and G Rätsch. Transcript normalization and segmentation of tiling array data. In *Proceedings Pac. Symp. on Biocomputing*, pages 527–538, 2008b.

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.

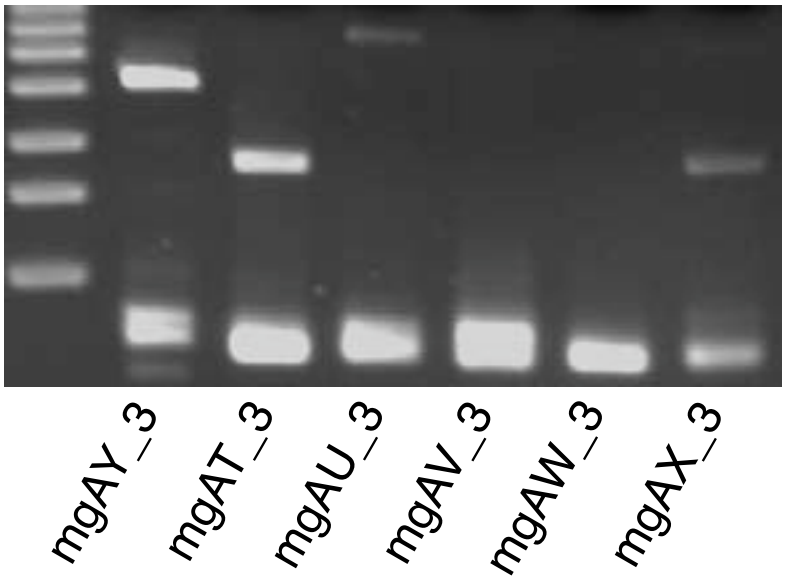
# Results: RT-PCR Validation



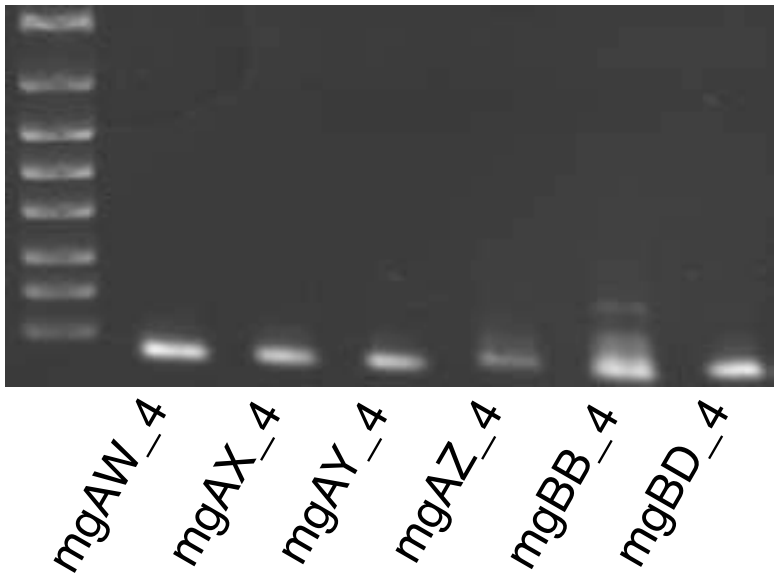
Validation of gene predictions for *C. elegans*: Schweikert et al., 2008

| | No. of genes | No. of genes analyzed | Frac. of genes w/ expression |
|---|---|---|---|
| New genes | 2,197 | 57 | $\approx 42\%$ |
| Missing unconf. genes | 205 | 24 | $\approx 8\%$ |

new genes

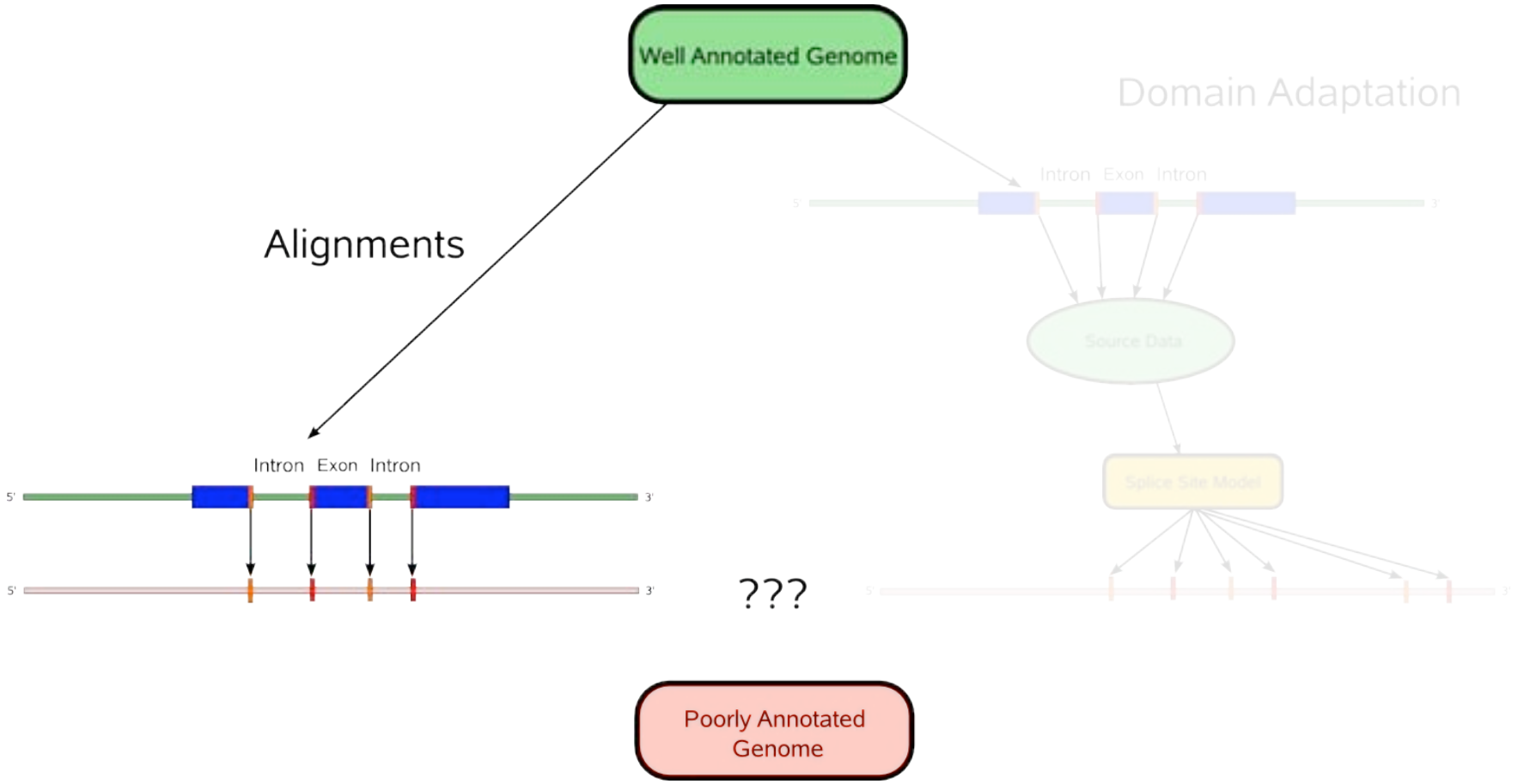



missed genes



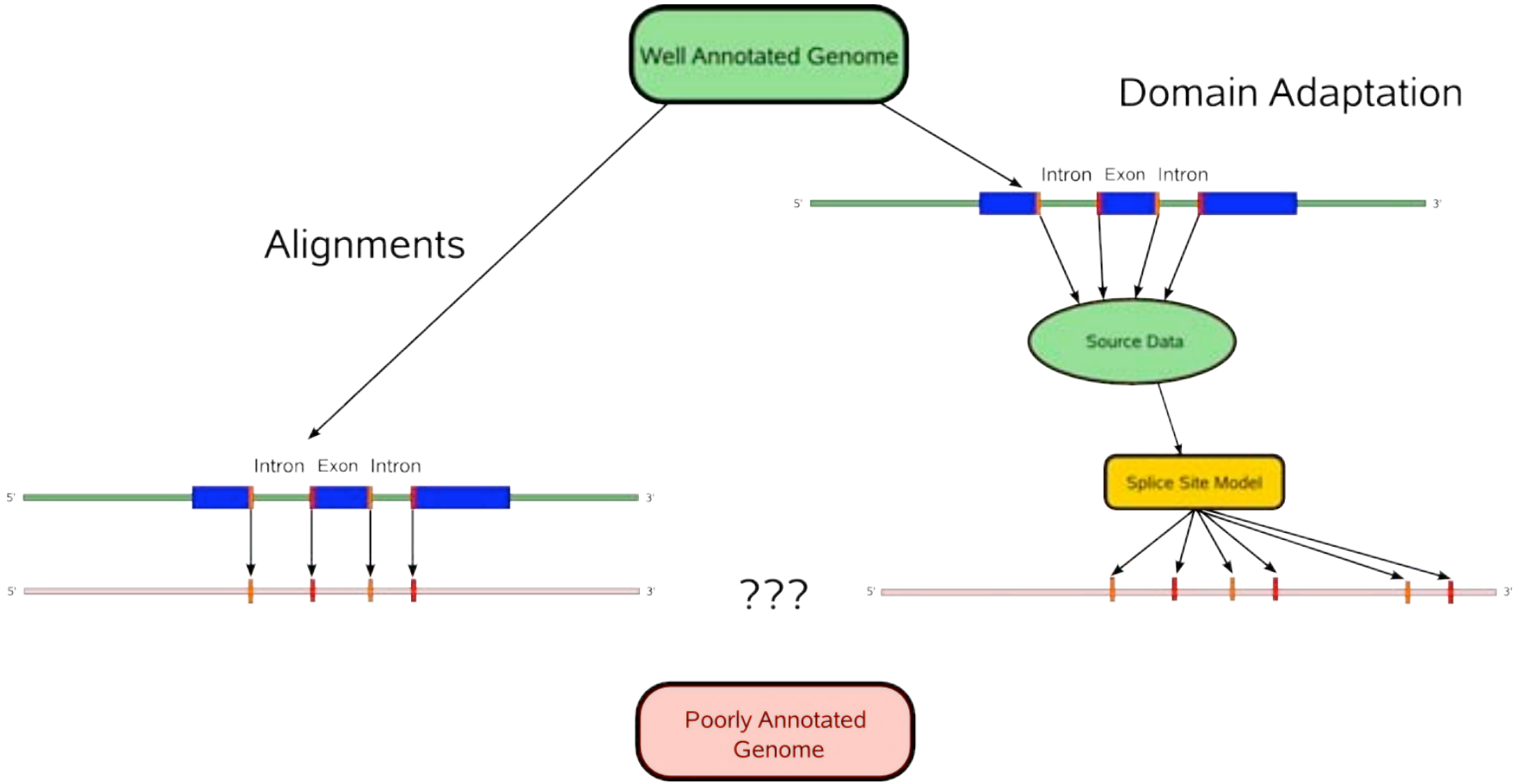
# Domain Adaption by Learning vs. Homology





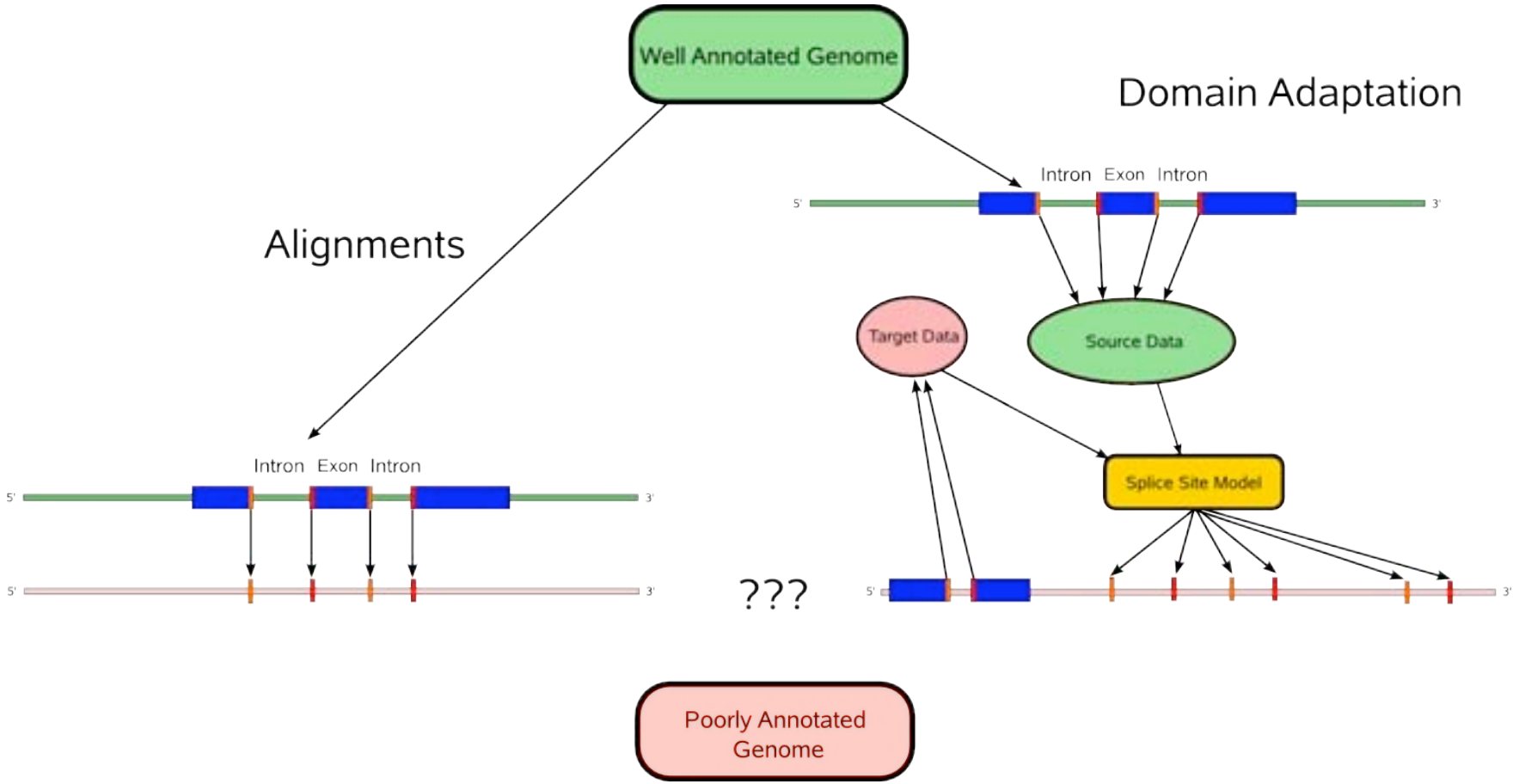
# Domain Adaption by Learning vs. Homology





# Domain Adaption by Learning vs. Homology





# Domain Adaption by Learning vs. Homology



