Machine Learning for Predictive Sequence Analysis

Gunnar Rätsch

Friedrich Miescher Laboratory, Tübingen

8th Course on Bioinformatics and Systems Biology
Bertinoro, Italy

19th of March, 2008

http://www.fml.mpg.de/raetsch/lectures/bertinoro08

About the Title

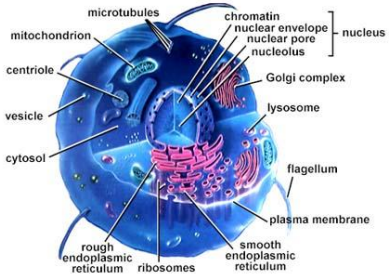
Machine Learning for Predictive Sequence Analysis

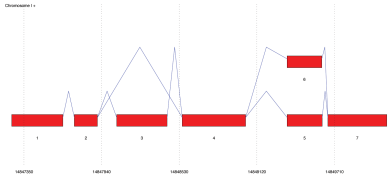
- **Sequence Analysis**
- **Prediction methods**
- **Machine Learning**

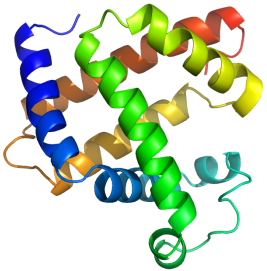
Sequence Analysis

A few examples:

- Annotate Sequences
  - Motif finding
  - Protein Subcellular Localization
  - Gene finding
- Alignment
  - EST to DNA
  - Remote Homology
- Infer Structure
  - RNA secondary structure
  - Protein secondary structure







Why machine learning?



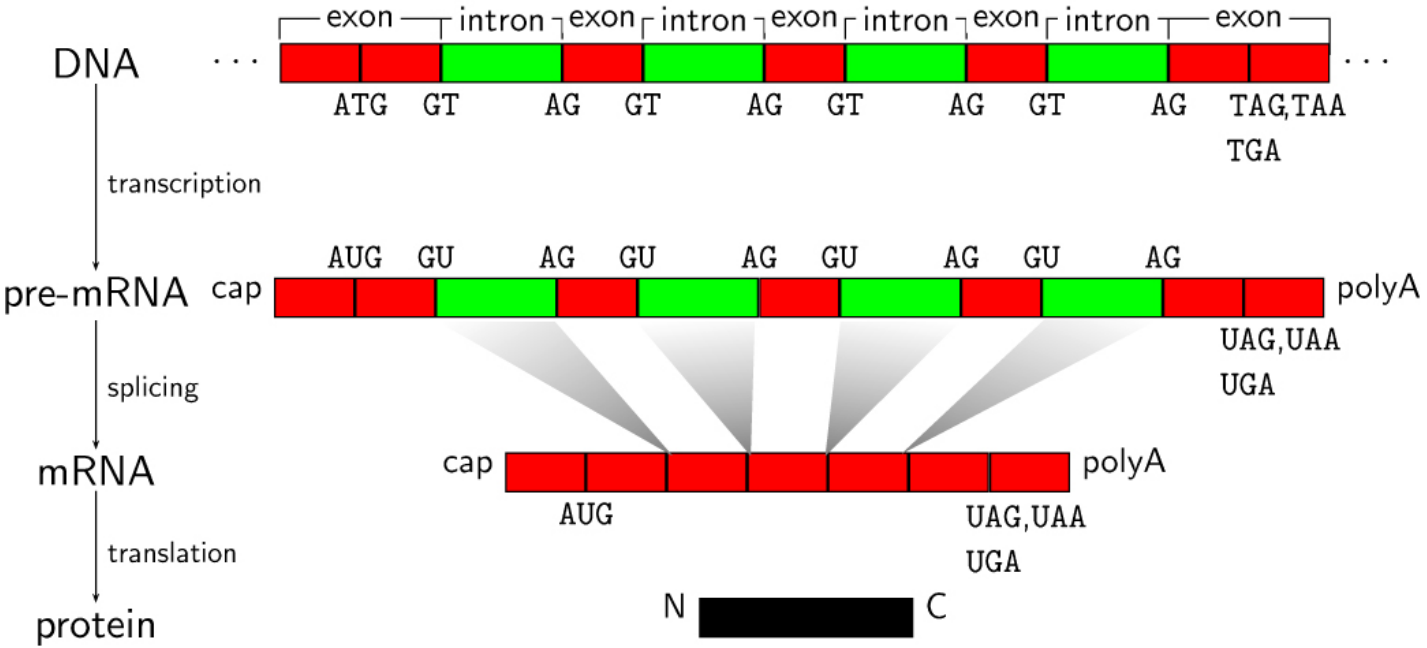
- A lot of data
- Data is noisy
- No clear biological theory
- Large number of features
- Complex relationships

Let the data do the talking!

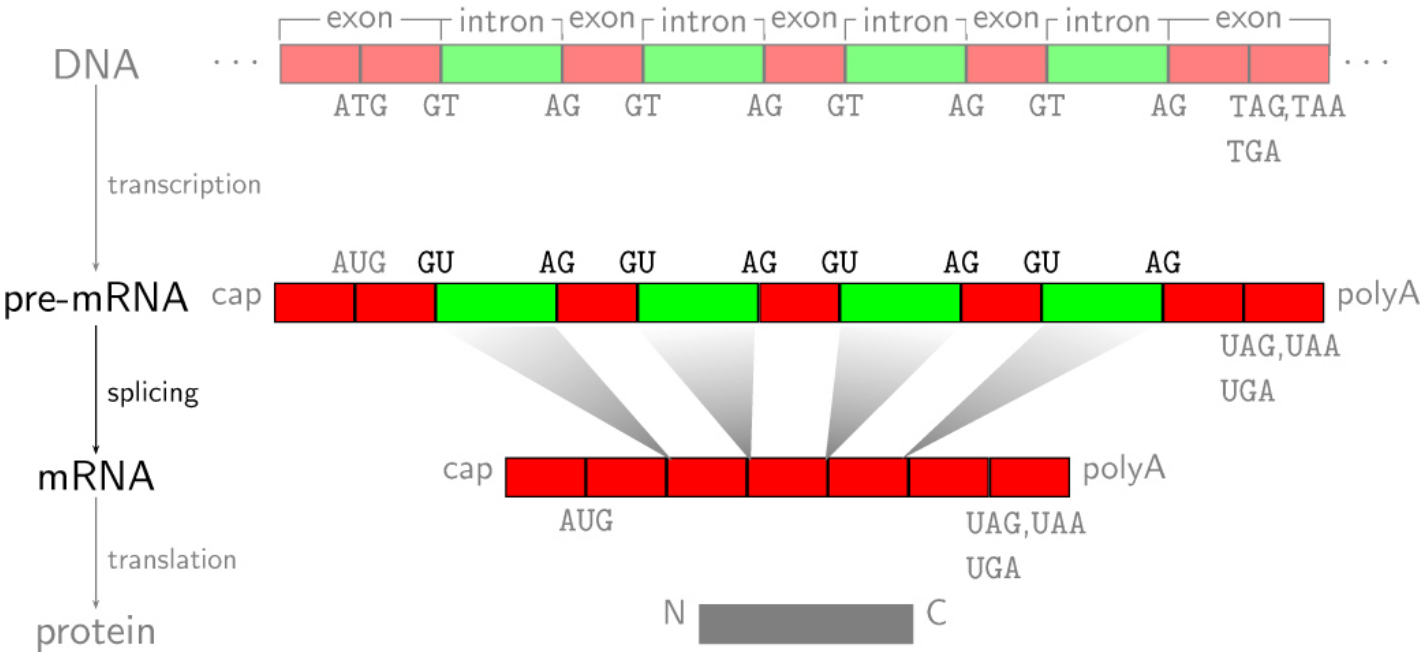
Tutorial Outline

- Basic concepts in Machine Learning
  - Classification, regression & co.
  - Generalization performance & model selection
- Support vector machines
  - “Large Margins” & the “Optimization Problem”
  - Learning nonlinearities
- Learning with Kernels
  - Kernels for sequences
  - Efficient data structures
- Applications in Computational Biology
  - Prediction of splicing events
  - Whole genome tiling arrays

Running Example: Splicing



Running Example: Splicing

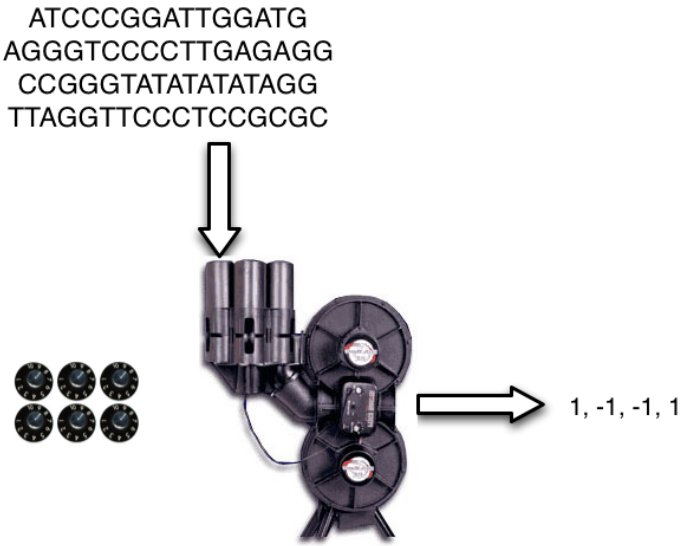


- Almost all *donor splice sites* exhibit `GU`
- Almost all *acceptor splice site* exhibit `AG`
- Not all `GU`'s and `AG`'s are used as splice site

Classification of Sequences

Example: Recognition of splice sites

- Every 'AG' is a possible acceptor splice site
- Computer has to learn what splice sites look like
  - given some known genes/splice sites . . .
- Prediction on unknown DNA



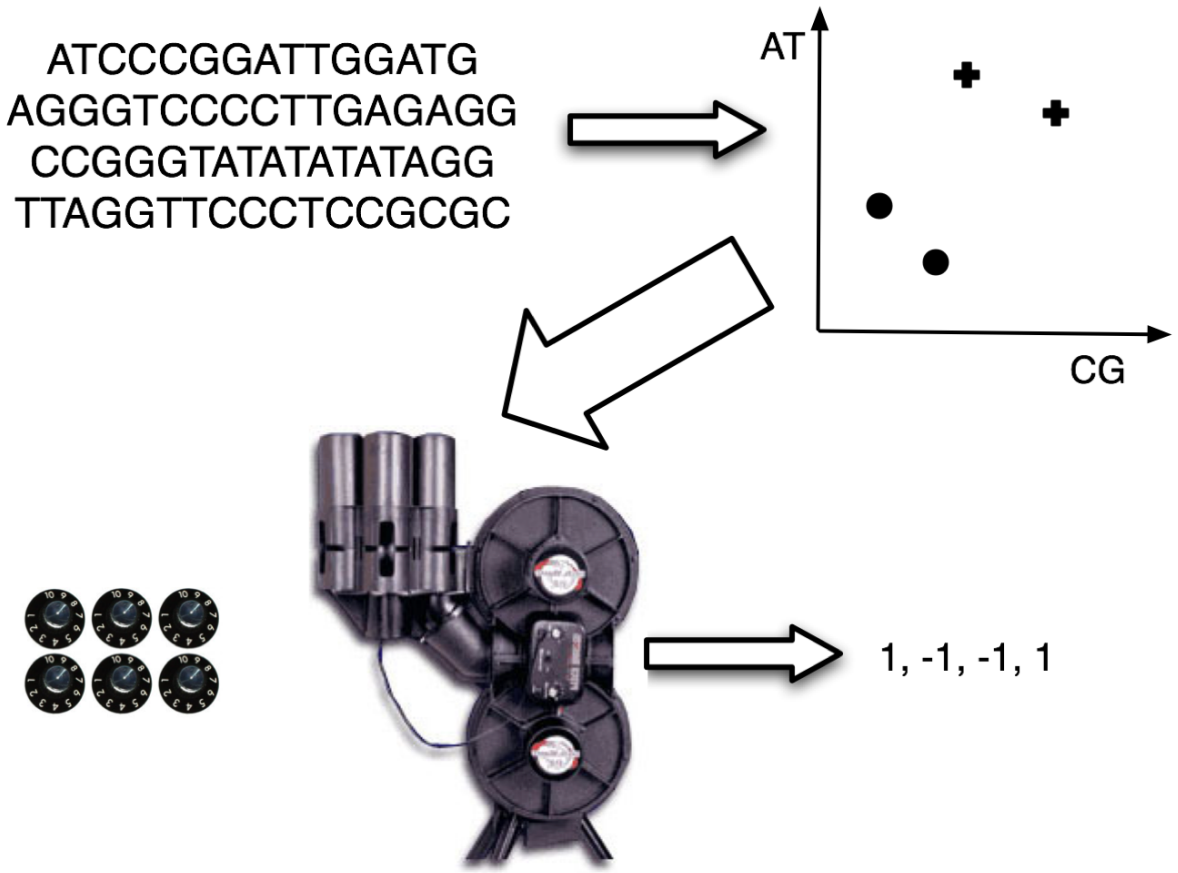
From Sequences to Features

- Many algorithms depend on numerical representations.
  - Each example is a vector of values (features).
- Use background knowledge to design good features.



	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	...
GC before	0.6	0.2	0.4	0.3	0.2	0.4	0.5	0.5	...
GC after	0.7	0.7	0.3	0.6	0.3	0.4	0.7	0.6	...
AG**AG**AAG	0	0	0	1	1	0	0	1	...
TTT**AG**	1	1	1	0	0	1	0	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱
Label	+1	+1	+1	−1	−1	+1	−1	−1	...

Numerical Representation



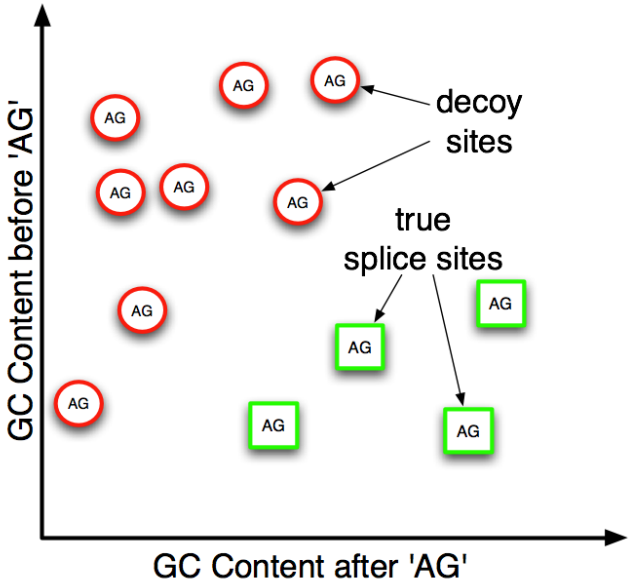
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron exon

- Goal: Rule that distinguishes true from false ones



e.g. exploit that exons have higher GC content

or

that certain motifs are located nearby

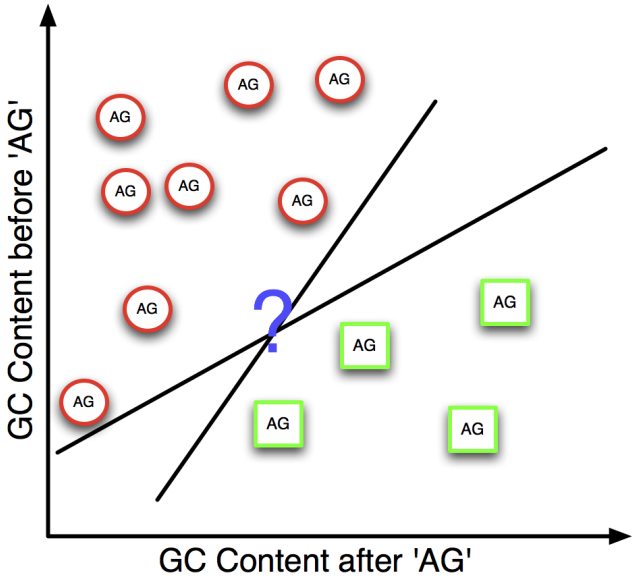
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

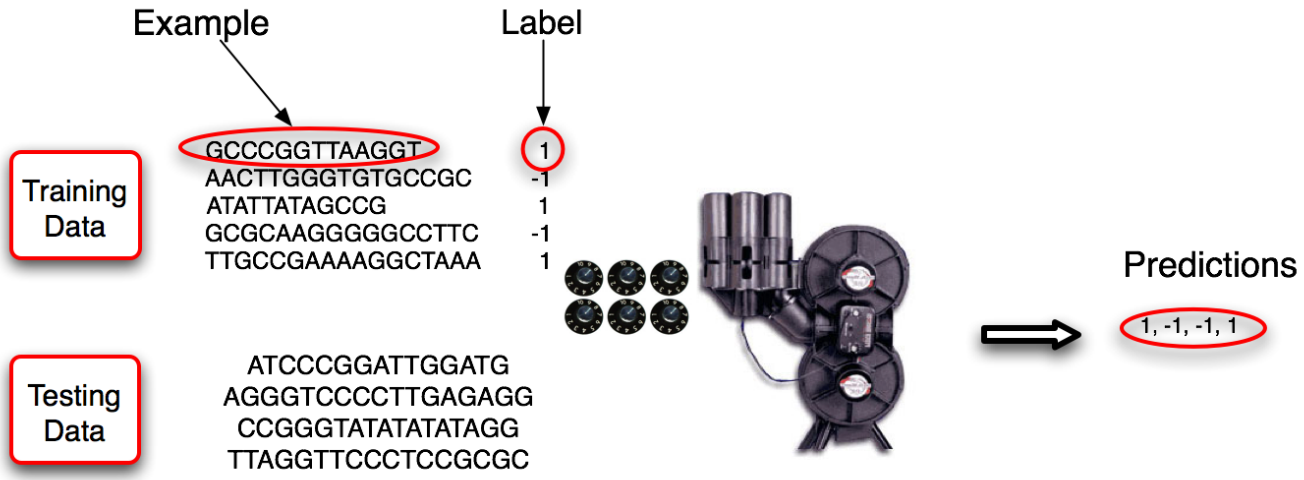
intron **exon**

- Goal: Rule that distinguishes true from false ones



For instance:
Linear classifiers

Empirical Inference (=Learning from Examples)



The machine utilizes information from training data to predict the outputs associated with a particular test example.

- Use training data to “train” the machine.
- Use trained machine to perform prediction on test data.

Words, words, words...

Example $\mathbf{x}_i \in \mathcal{X}$ for example a nucleotide sequence.

Label $y_i \in \mathcal{Y}$ for example whether the sequence contains a splice site at central position

Training Data Data consisting of examples and associated labels, which are used for training the machine.

Testing Data Data consisting only of examples, used for generating predictions.

Predictions Output of the trained machine.

Machine Learning: Main Tasks

Supervised Learning

We have both examples and labels for each example. The aim is to learn about the pattern between examples and labels.

Unsupervised Learning

We do not have labels for the examples, and wish to discover the underlying structure of the data.

Reinforcement Learning

How an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals.

Estimators

Basic Notion

We want to estimate the relationship between the examples $\mathbf{x}_i$ and the associated label y_i.

Formally

We want to choose an estimator

$$f : \mathcal{X} \to \mathcal{Y}.$$

Intuition

We would like a function f which correctly predicts the label y for a given example $\mathbf{x}$.

Question

How do we measure how well we are doing?

Loss Function

Basic Notion

We characterize the quality of an estimator by a loss function.

Formally

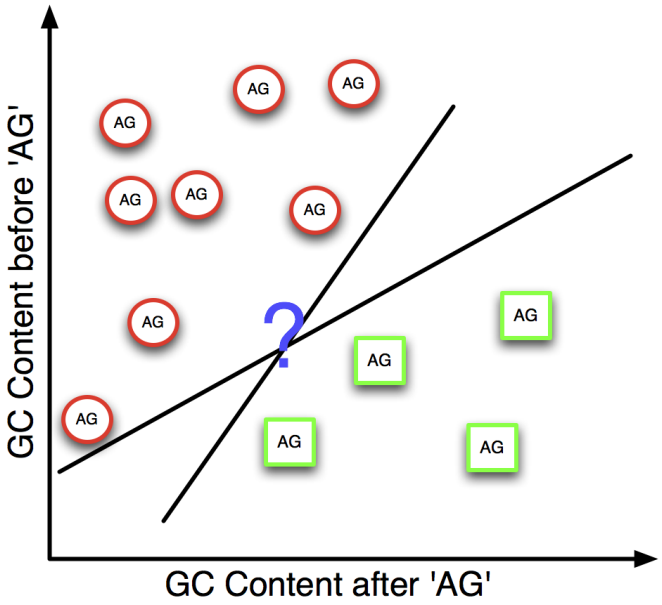
We define a loss function

$$\ell(f(\mathbf{x}_i), y_i) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+.$$

Intuition

For a given label y_i and a given prediction $f(\mathbf{x}_i)$, we want a positive value telling us how much error we have made.

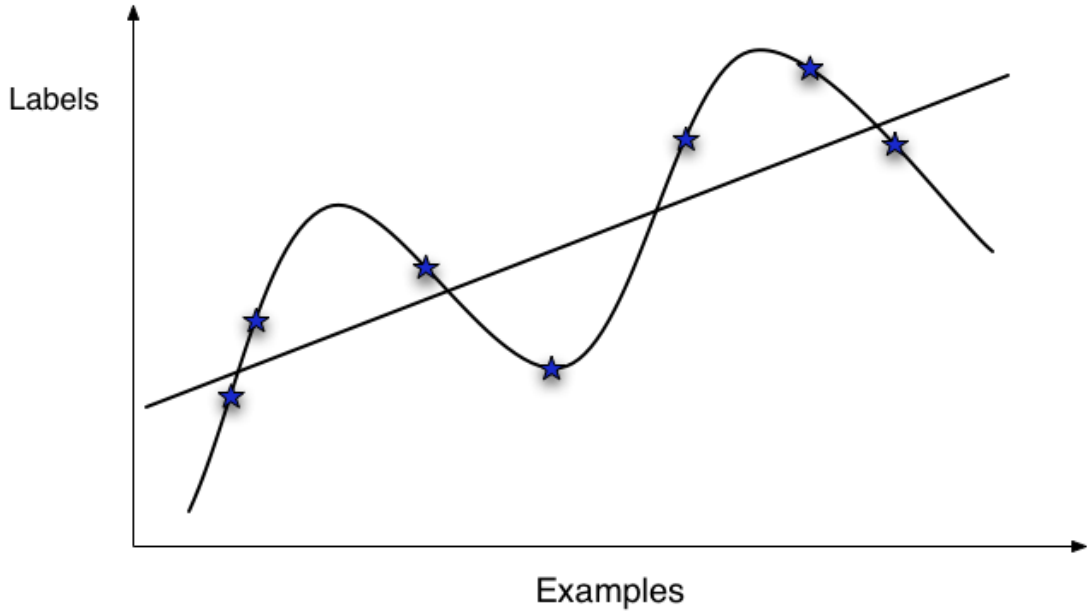
Classification



In the case of classification ($y = \{-1, +1\}$), we can have the loss:

$$\ell(f(\mathbf{x}_i), y_i) = \begin{cases} 0 & \text{if } f(\mathbf{x}_i) = y_i \\ 1 & \text{if } f(\mathbf{x}_i) \neq y_i \end{cases}$$

Regression



In the case of regression ($\mathcal{Y} = \mathbb{R}$), the loss function can be the square loss

$$\ell(f(\mathbf{x}_i), y_i) = (f(\mathbf{x}_i) - y_i)^2.$$

Expected vs. Empirical Risk

Expected Risk

Is the average loss of unseen examples. We would like to have it as small as possible, but hard to compute

Empirical Risk

We can find the average on training data. We define the empirical risk to be:

$$\mathbf{R}_{emp}(f, X, Y) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i).$$

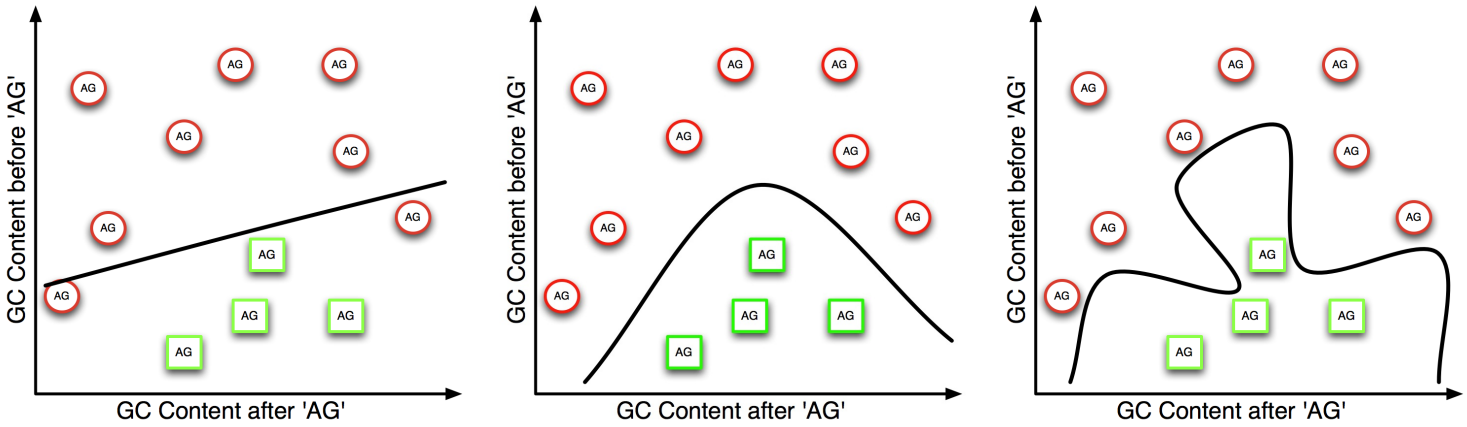
Basic Notion

Instead of minimizing the expected risk, we minimize the empirical risk. This is called empirical risk minimization.

Question

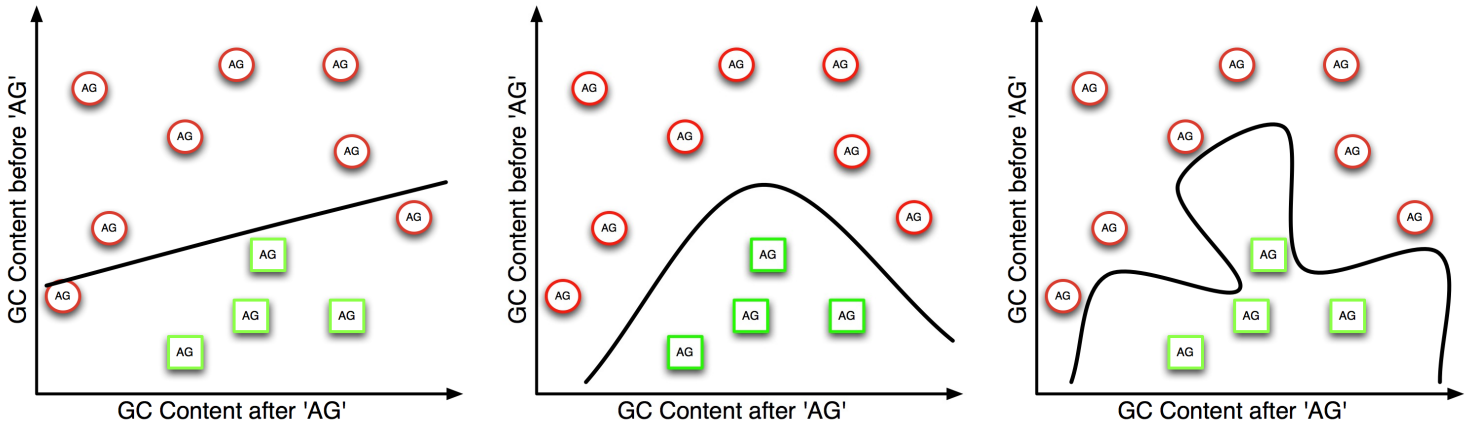
How do we know that our estimator will perform well on unseen data?

Simple vs. Complex Functions



Which function is preferable?

Simple vs. Complex Functions



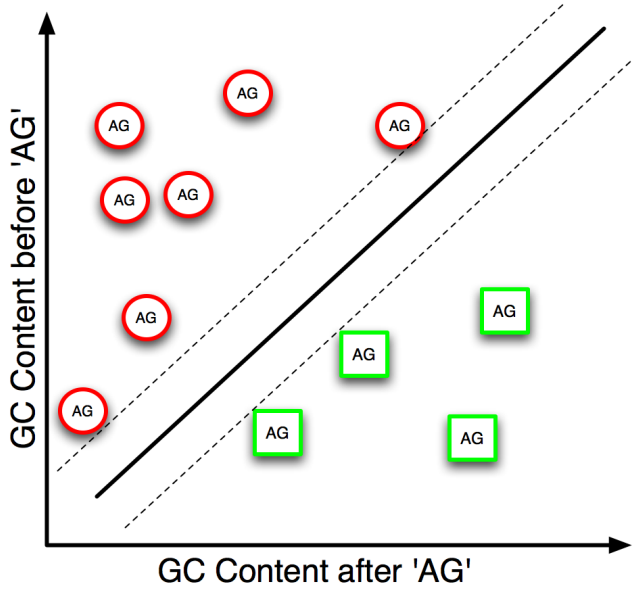
Which function is preferable?



Occam's razor (William Occam, 14th century):

"Entities should not be multiplied beyond necessity"

Complexity of Hyperplanes?



What is the complexity of a hyperplane classifier?

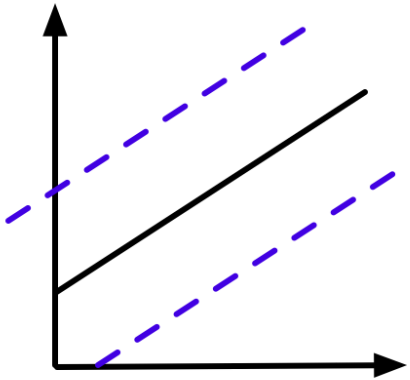


Vapnik [1995]:
Vapnik-Chervonenkis (VC) dimension

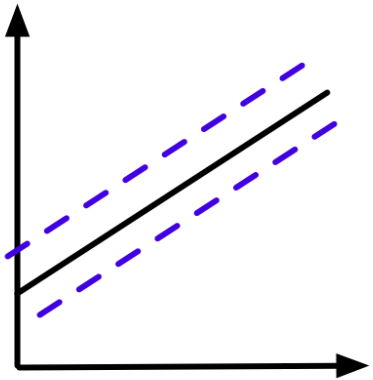
Larger Margin $\Rightarrow$ Less Complex

Large Margin $\Rightarrow$ Small VC dimension

Hyperplane classifiers with large margin have small VC dimension [Vapnik, 1995].



VC dim. small



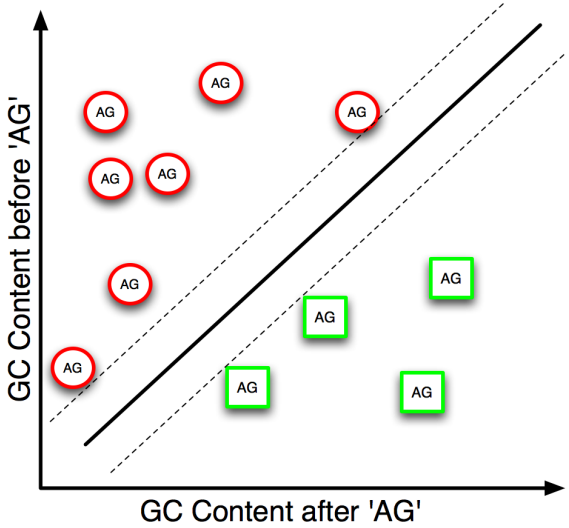
VC dim. large

Maximum Margin $\Rightarrow$ Minimum Complexity

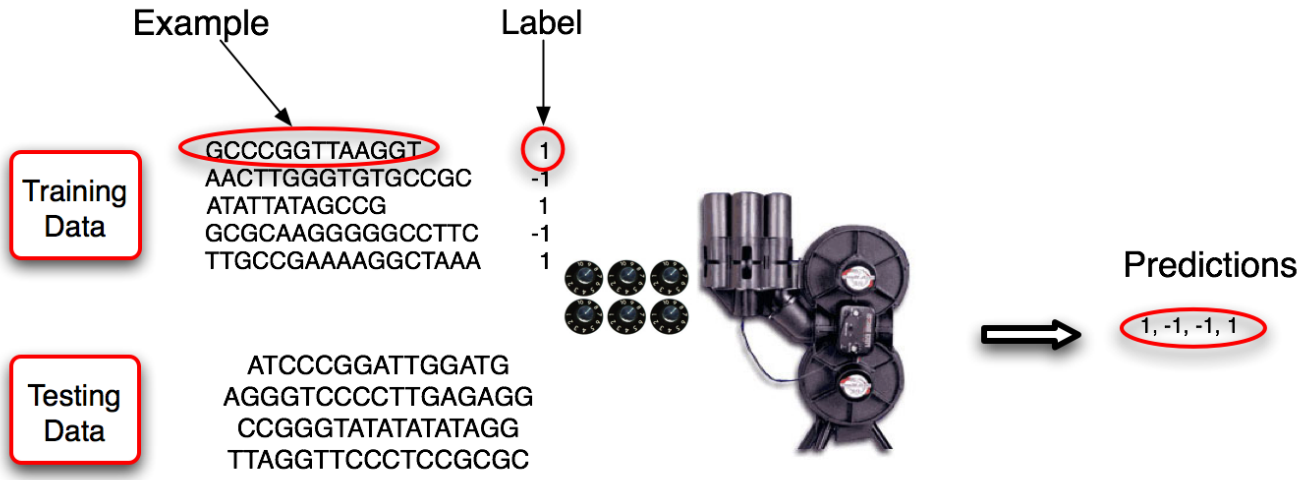
Minimize complexity by maximizing margin (irrespective of the dimension of the space).

Why maximize the margin?

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

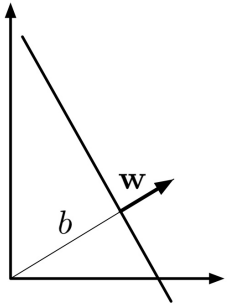


How to Maximize the Margin?

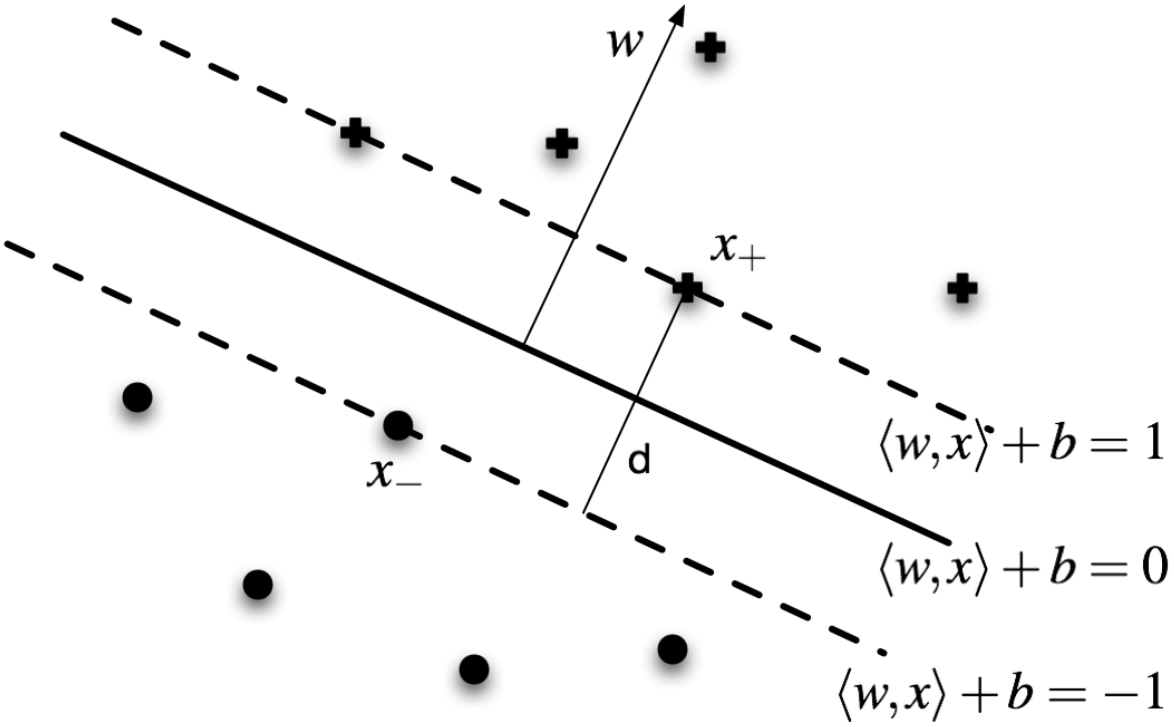


Consider linear hyperplanes with parameters $\mathbf{w}, b$:

$$f(\mathbf{x}) = \sum_{j=1}^{d} w_j x_j + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

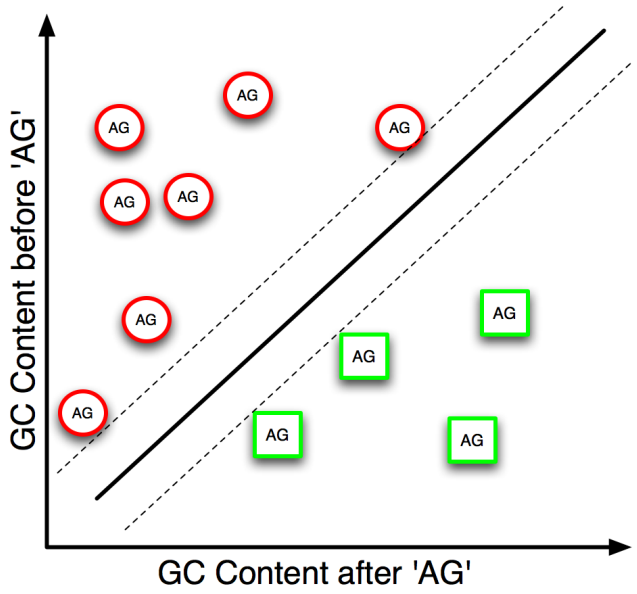


How to Maximize the Margin?



Margin maximization is equivalent to minimizing $\|\mathbf{w}\|$.

How to Maximize the Margin?



Minimize

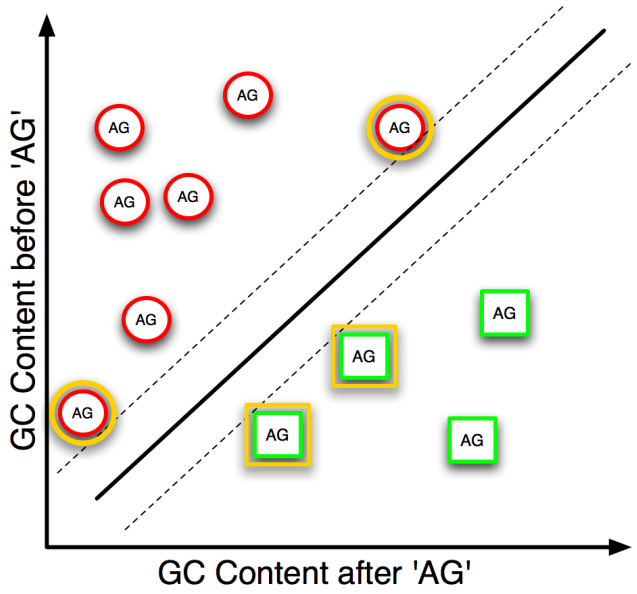
$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

Subject to

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

- The examples on the margin are called support vectors [Vapnik, 1995]
- Called the soft margin SVM [Cortes and Vapnik, 1995]

How to Maximize the Margin?



Minimize

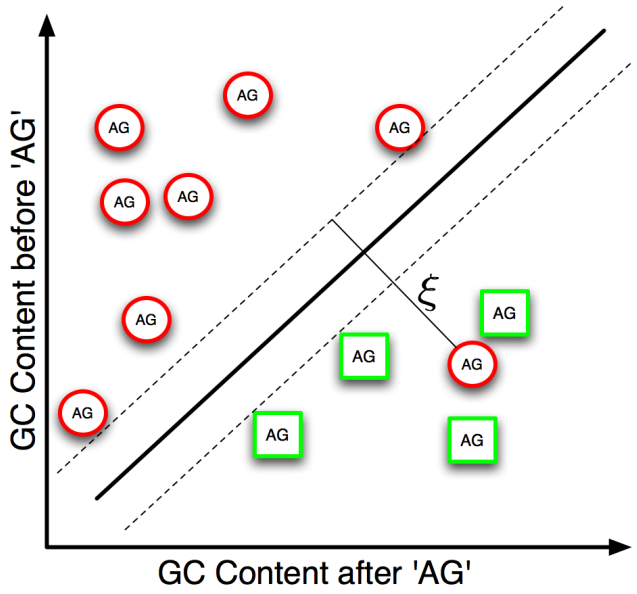
$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

Subject to

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

- The examples on the margin are called support vectors [Vapnik, 1995]
- Called the soft margin SVM [Cortes and Vapnik, 1995]

How to Maximize the Margin?



Minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

Subject to

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

- The examples on the margin are called support vectors [Vapnik, 1995]
- Called the soft margin SVM [Cortes and Vapnik, 1995]

Support Vector Machines

We have to solve an "Optimization Problem":
Find $\mathbf{w}$, b and ξ's such that they

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geqslant 1 - \xi_i \text{ for all } i = 1, \ldots, n. \\ & \xi_i \geqslant 0 \text{ for all } i = 1, \ldots, n \end{aligned}$$

Note: $(\mathbf{x}_i, y_i)$ and parameter C are given in advance

Optimization Problem:

- Conveniently formulate what should come out
  - Here: Maximum Margin hyperplane
- Do not need to care about the technical details
  - Quite heavy mathematical machinery for solution

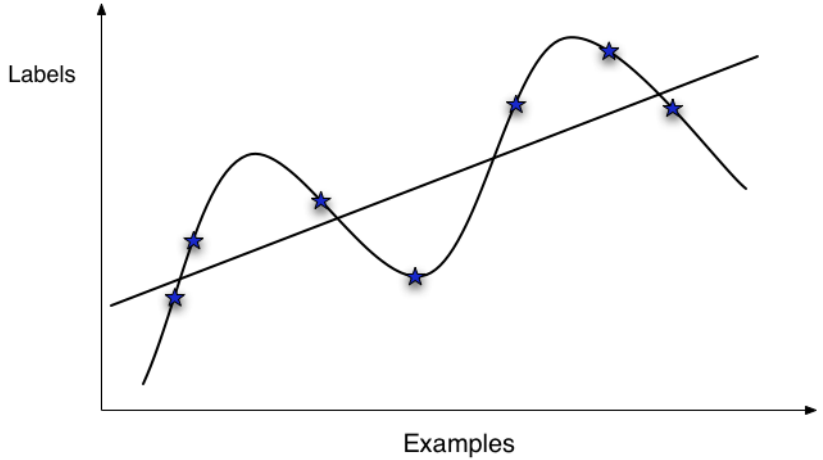
Key ingredients

- Which features $\mathbf{x}$ and which model $f_{\mathbf{w}}(\mathbf{x})$?
  - Often: More features are better.
- How to control complexity for this model?
  - There are a few available standard choices.
- Which loss function should be minimized?
  - Depends on the task at hand.
  - Do not optimize one loss, if you are interested in another one!

$$\min_{\mathbf{w}} \quad \underbrace{\sum_{i=1}^{N} \ell(f_{\mathbf{w}}(\mathbf{x}_i), y_i)}_{\text{empirical risk/loss}} \quad + \quad \underbrace{P[\mathbf{w}]}_{\text{regularization term}}$$

In applications the “What” and not the “How” is important!

Example: Linear Regression

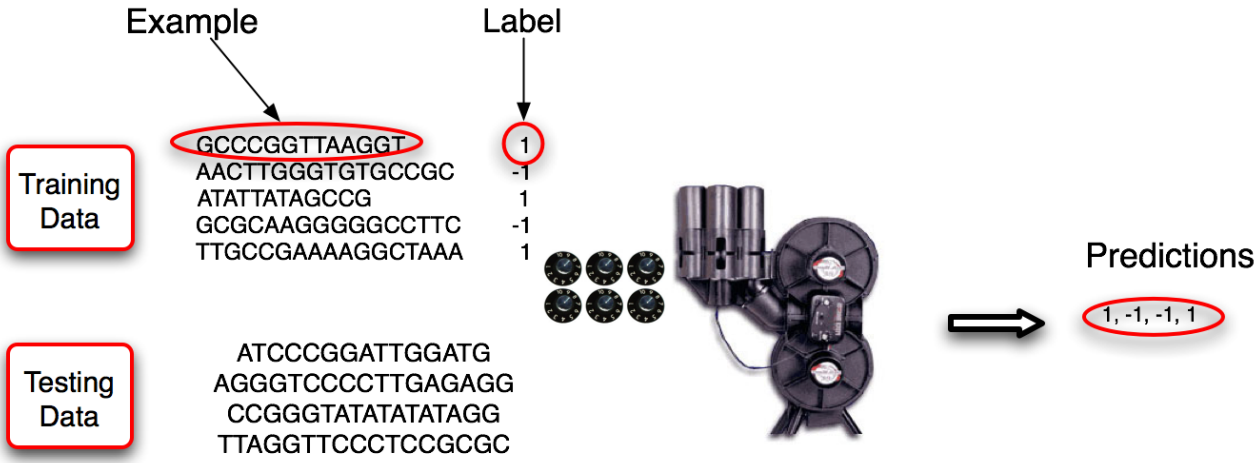


Choices:

- One feature, linear model
- Quadratic regularization (standard)
- Quadratic loss (frequent choice in regression)

$$\min_{w,b} \underbrace{\sum_{i=1}^{N} \left(\overbrace{w \cdot x_i + b}^{f(x)} - y_i \right)^2}_{\text{empirical risk/loss}} + \underbrace{Cw^2}_{\text{regularization term}}$$

Summary: Generalization



Estimator $f : \mathcal{X} \to \mathcal{Y}$, the function we want to learn.

Loss $\ell(f(\mathbf{x}_i), y_i)$, the error for a particular example.

Risk $\mathbf{R}(f)$, The expected loss for all data, including unseen.

Empirical Risk $\mathbf{R}_{emp}(f)$. The average loss we observe.

Theory: Small empirical risk & large margin $\Rightarrow$ small risk

How to Measure Performance?

Important not just to memorize the training examples!
Use some of the labeled examples for validation:

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	...
GC before	0.6	0.2	0.4	0.3	0.2	0.4	0.5	0.5	...
GC after	0.7	0.7	0.3	0.6	0.3	0.4	0.7	0.6	...
AG**AG**AAG	0	0	0	1	1	0	0	1	...
TTT**AG**	1	1	1	0	0	1	0	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱
Label	+1	+1	+1	−1	−1	+1	−1	−1	...
	Training						Testing		

We assume that the future examples are similar to our labeled examples.

How to Measure Performance?

What to do in practice

We split the data into training and validation sets, and use the error on the validation set to estimate the expected error.

A. k-fold cross validation

Split data into k disjoint parts, and use each subset as the validation set, while using the rest as the training set.

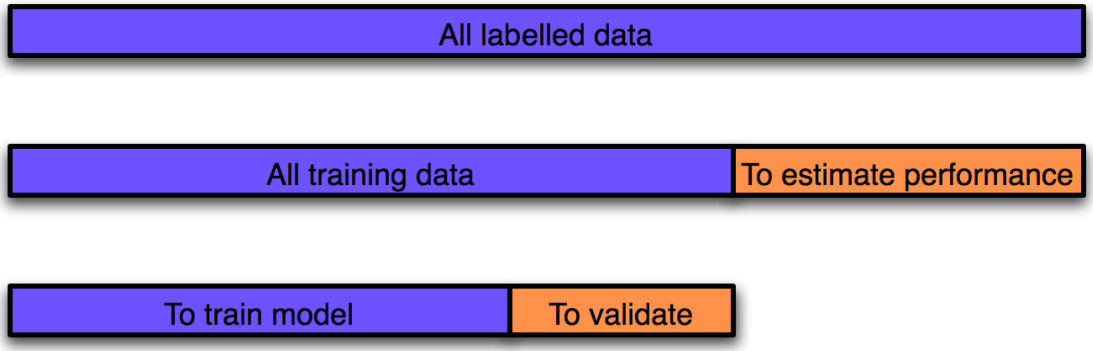
B. Random splits

Randomly split the data set into two parts, for example 80% of the data for training and 20% for validation.

Typical Mistakes

- Training & validation data are overlapping/identical
- Training somehow depends on validation data
- Validation data has different properties

How to Measure Performance?



Model Selection = Find best parameters

- SVM parameter C
- Additional parameters (e.g. of “kernel”)

Do not train on the test set!

- Use subset of data for training
- From subset, further split to select model

Common Performance Measures

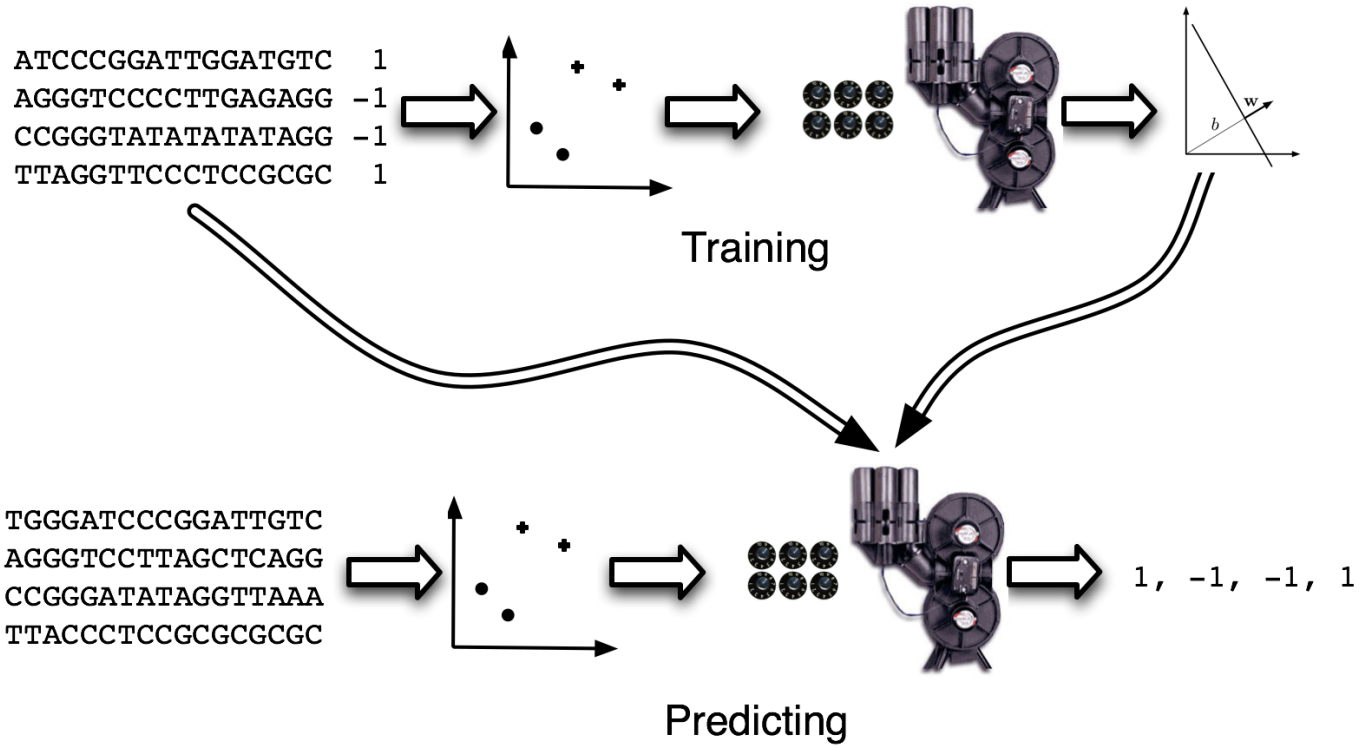
Classification

- Misclassification error: $\frac{1}{n}\sum_{i=1}^{n}\mathrm{Id}(y_i \neq \mathrm{sign}(f(\mathbf{x}_i)))$
- Receiver Operator Curve
  - *True positive rate* against *false positive rate*
- Precision Recall Curve
  - *Positive predictive value* against *true positive rate*

Regression

- Mean Squared Error: $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2$
- Mean Absolute Error: $\frac{1}{n}\sum_{i=1}^{n}|y_i - f(\mathbf{x}_i)|$

Summary: Empirical Inference



Duda et al. [2001], Schölkopf and Smola [2002], Shawe-Taylor and Cristianini [2004]

Demonstration using Galaxy

Task: Distinguish sequences like

`aacgagtACGTgcta`

from

`cgataggtatgtagc`

Represent sequences as binary vectors:

$$\texttt{a} \rightarrow \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \texttt{c} \rightarrow \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \texttt{g} \rightarrow \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \texttt{t} \rightarrow \begin{pmatrix}0\\0\\0\\1\end{pmatrix}.$$

Steps: (Do it yourself at `http://galaxy.fml.tuebingen.mpg.de`)

- Generate two labeled sets
  (use `Toy Data`→ `MotifGen (ARFF)`)
- Train SVM on first and evaluate on second dataset
  (use `SVM Toolbox`→`Train and Test SVM`, WD kernel, degree=1, shift=0)
- Evaluate predictions on second dataset
  (use `SVM Toolbox`→`Evaluate`)

Tutorial Outline

- Basic concepts in Machine Learning
  - Classification, regression & co.
  - Generalization performance & model selection
- Support vector machines
  - “Large Margins” & the “Optimization Problem”
  - Learning nonlinearities
- Learning with Kernels
  - Kernels for sequences
  - Efficient data structures
- Applications in Computational Biology
  - Prediction of splicing events
  - Whole genome tiling arrays

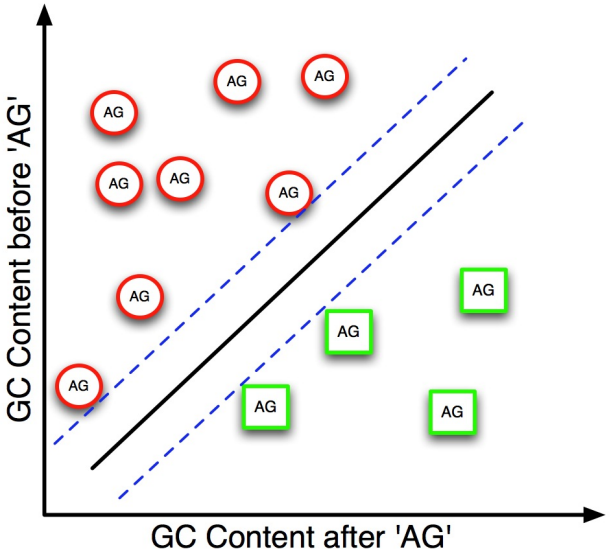
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

- Goal: Rule that distinguishes true from false ones



Linear Classifiers
with large **margin**

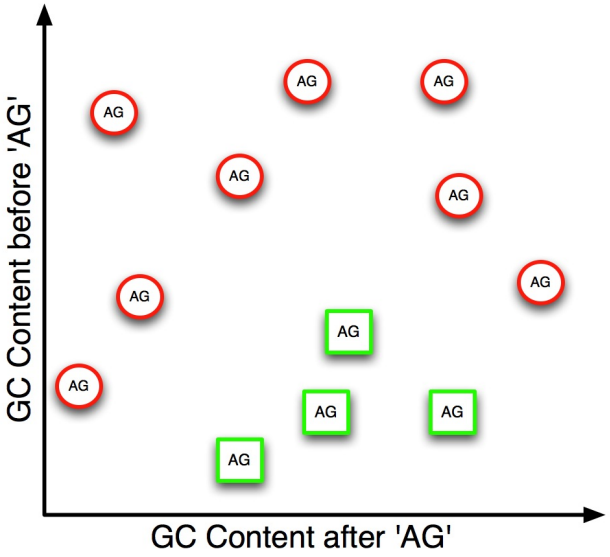
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

- Goal: Rule that distinguishes true from false ones



More realistic problem!?

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

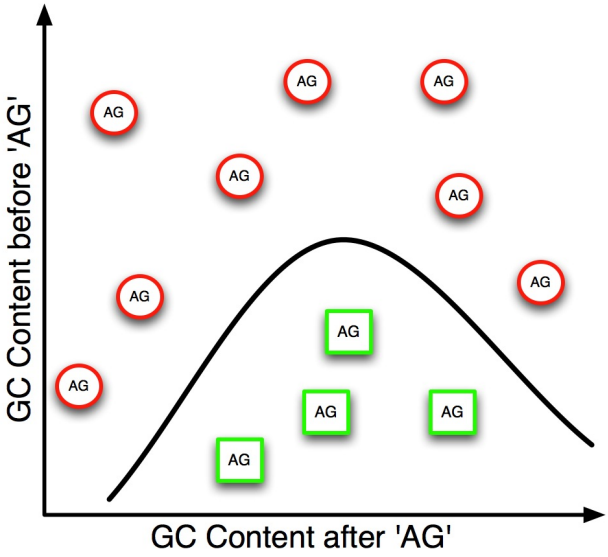
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

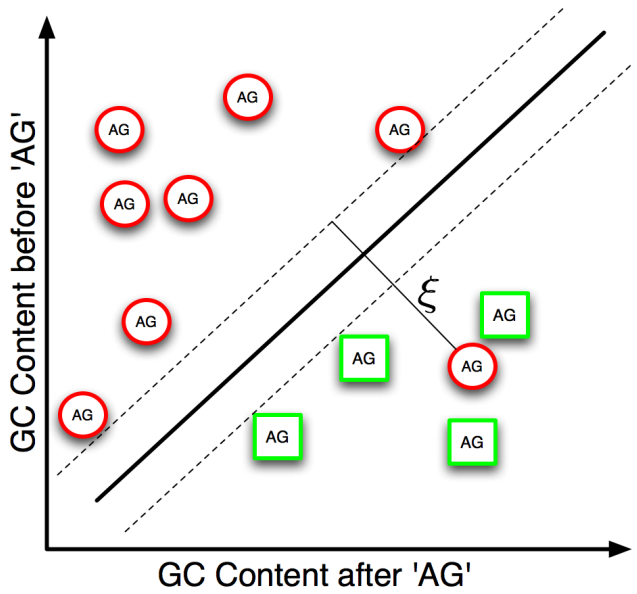
- Goal: Rule that distinguishes true from false ones



More realistic problem!?

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

How to Maximize the Margin?



Minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

Subject to

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

- The examples on the margin are called support vectors [Vapnik, 1995]
- Called the soft margin SVM [Cortes and Vapnik, 1995]

An Important Detail

We have to solve an "Optimization Problem":

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geqslant 1 - \xi_i \text{ for all } i = 1, \ldots, n. \\ & \xi_i \geqslant 0 \text{ for all } i = 1, \ldots, n \end{aligned}$$

Theorem: The optimal $\mathbf{w}$ can be written as a linear combination of the examples (for appropriate α's):

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i,$$

Corollary: Hyperplane only depends on the scalar products of the examples

$$\langle \mathbf{x}, \hat{\mathbf{x}} \rangle = \sum_{d=1}^{D} x_d \hat{x}_d$$

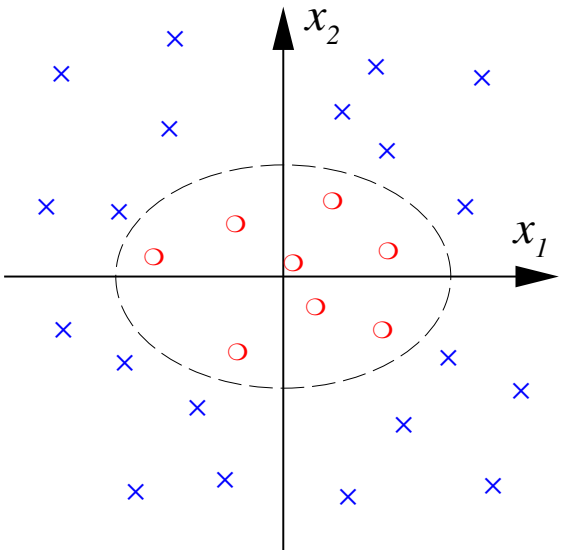
Nonlinear Separations

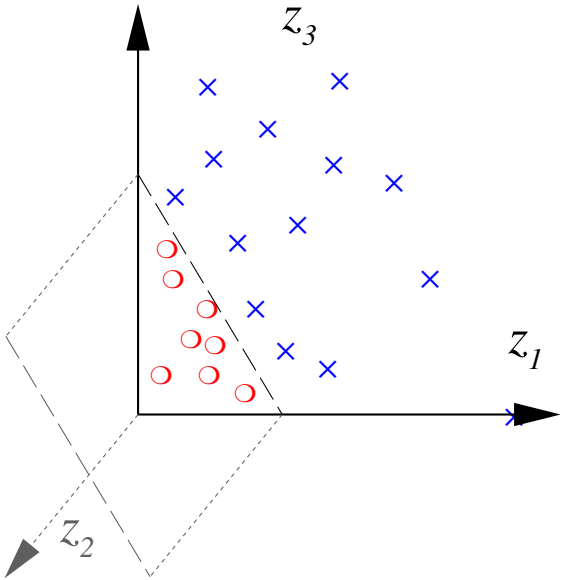
Linear separation might be not sufficient!

$\Rightarrow$ Map into a higher dimensional feature space

Example: all second order monomials

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





Kernel "Trick"

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

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

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

Putting things together ...

- Use $\Phi(\mathbf{x})$ instead of $\mathbf{x}$
- Use linear classifier on the $\Phi(\mathbf{x})$'s
- From Theorem: $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i)$.
- Nonlinear separation:

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b \\ &= \sum_{i=1}^{n} \alpha_i \underbrace{\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle}_{\mathrm{k}(\mathbf{x}_i, \mathbf{x})} + b \end{aligned}$$

- Trick: $\mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) = \langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}) \rangle$, i.e. **do not use Φ, but k!**

See e.g. Vapnik [1995], Müller et al. [2001], Schölkopf and Smola [2002] for details.

Kernel $\approx$ Similarity Measure

Distance:

$$\|\Phi(\mathbf{x}) - \Phi(\hat{\mathbf{x}})\|^2 = \|\Phi(\mathbf{x})\|^2 - 2\langle\Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}})\rangle + \|\Phi(\hat{\mathbf{x}})\|$$

Scalar product: $\langle\Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}})\rangle$

- If $\|\Phi(\mathbf{x})\|^2 = \|\Phi(\hat{\mathbf{x}})\|^2 = 1$, then

  scalar product = $2-$distance

- Angle between vectors, i.e.

$$\frac{\langle\Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}})\rangle}{\|\Phi(\mathbf{x})\| \, \|\Phi(\hat{\mathbf{x}})\|} = \cos(\Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}))$$

Technical detail: kernel functions have to satisfy certain conditions (Mercer conditions).

How to construct a kernel

At least two ways to get to a kernel:

- Construct Φ and think about efficient ways to compute scalar product $\langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}) \rangle$
- Construct similarity measure (show Mercer condition) and think about what it means

What can you do if kernel is not positive definite?

- Optimization problem is not convex!
- Add constant to diagonal (cheap)
- Exponentiate kernel matrix (all eigenvalues become positive)
- SVM-pairwise use similarity as features

Common Kernels

Frequently used kernels:

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

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

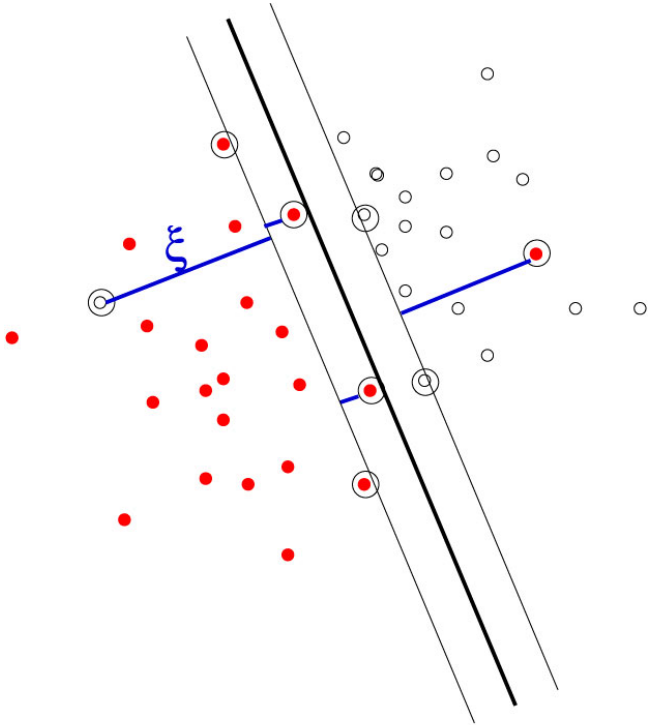
Notes:

- Kernels may be combined in case of heterogeneous data
- These kernels are good for real-valued examples
- Sequences need special care (comes soon!)

Toy Examples

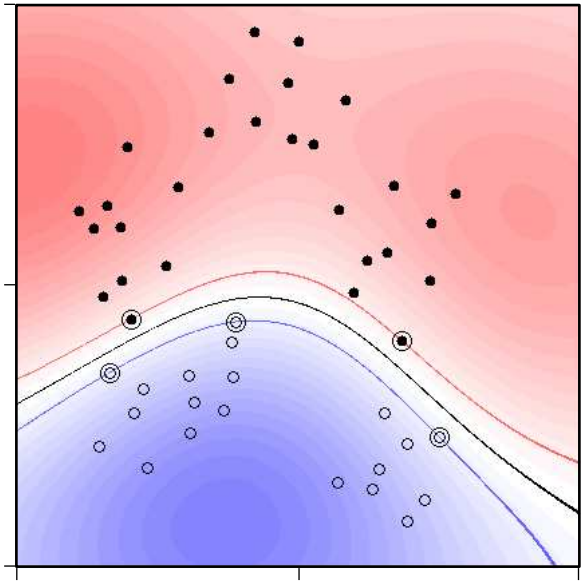
Linear kernel

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



RBF kernel

$$k(\mathbf{x}, \hat{\mathbf{x}}) = \exp(-\|\mathbf{x} - \hat{\mathbf{x}}\|^2 / 2\sigma)$$



Kernel Summary

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

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

- (Mercer) Kernel defines a mapping Φ (nontrivial)
- Choice of kernel has to match the data at hand
- RBF kernel often works pretty well

Tutorial Outline

- Basic concepts in Machine Learning
  - Classification, regression & co.
  - Generalization performance & model selection
- Support vector machines
  - “Large Margins” & the “Optimization Problem”
  - Learning nonlinearities
- Learning with Kernels
  - Kernels for sequences
  - Efficient data structures
- Applications in Computational Biology
  - Prediction of splicing events
  - Whole genome tiling arrays

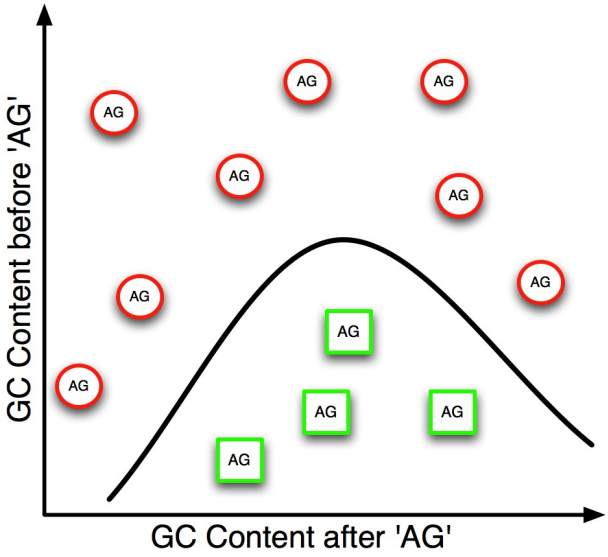
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

- Goal: Rule that distinguishes true from false ones



More realistic problem!?

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

More Features?

Some ideas:

- statistics for all four letters (or even dimer/codon usage),
- appearance of certain motifs,
- information content,
- secondary structure, ...

Approaches:

- Manually generate a few *strong* features
  - Requires background knowledge
  - Nonlinear decisions often beneficial
- Include many potentially useful *weak* features
  - Requires more training examples

Best in practice: Combination of both

Spectrum Kernel

General idea [Leslie et al., 2002]

- For each ℓ-mer $s \in \Sigma^\ell$, the coordinate indexed by s will be the number of times s occurs in sequence x.
- Then the ℓ-spectrum feature map is

$$\Phi_\ell^{\text{Spectrum}}(\mathbf{x}) = \left(\phi_s(\mathbf{x})\right)_{s \in \Sigma^\ell}$$

- 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:

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

- Dimensionality: exponential in ℓ: $|\Sigma|^\ell$

Spectrum Kernel

Principle

- Spectrum kernel: Count exactly common ℓ-mers

Protein A: ILVFMC

Protein B: WLVFQC

Common 1-mers
L, V, F, C

Common 2-mers
LV
VF

Common 3-mers
LVF

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

Motif kernels

General idea

- Conserved motif in sequences indicate structural and functional characteristics
- Model sequence as feature vector representing motifs
- i-th vector component is 1 $\Leftrightarrow$ $\mathbf{x}$ contains i-th motif

Motif databases

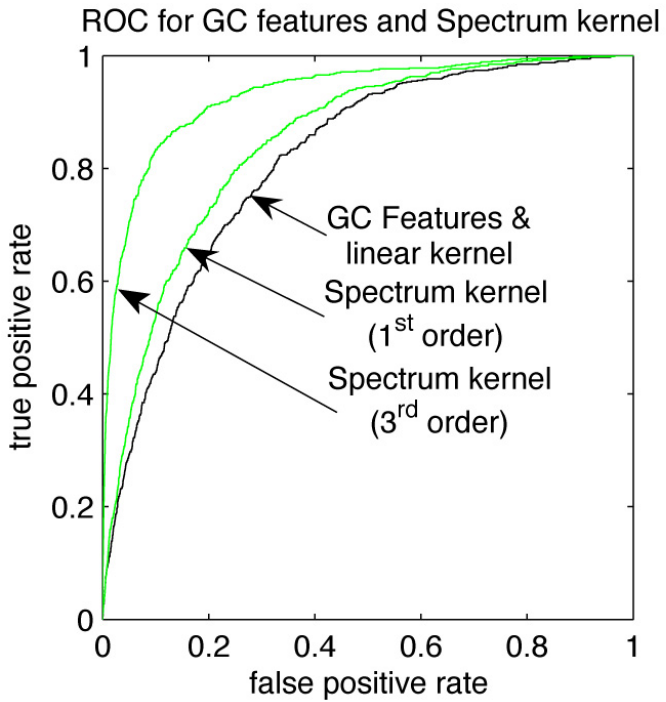
- Protein: Pfam, PROSITE, ...
- DNA: Transfac, Jaspar, ...
- RNA: Rfam, Structures, Regulatory sequences, ...

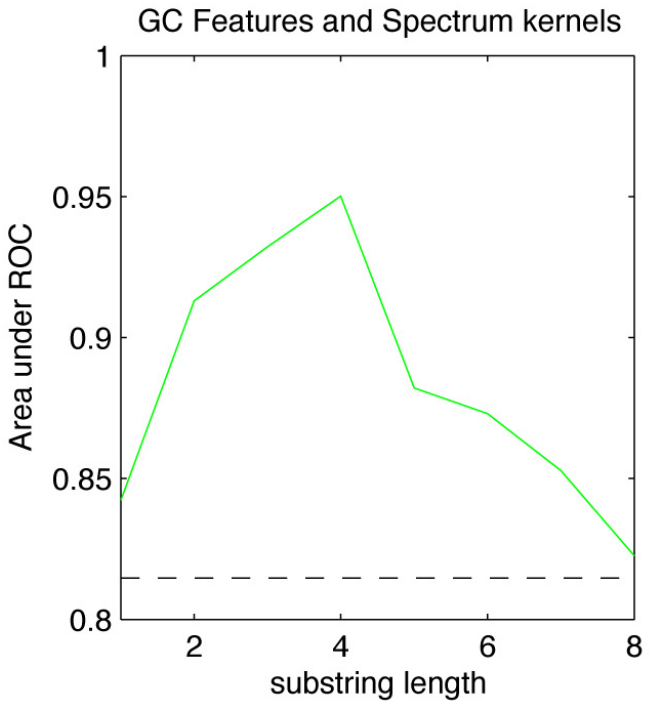
Generated by

- manual construction/prior knowledge
- multiple sequence alignment (do not use test set!)

Simulation Example

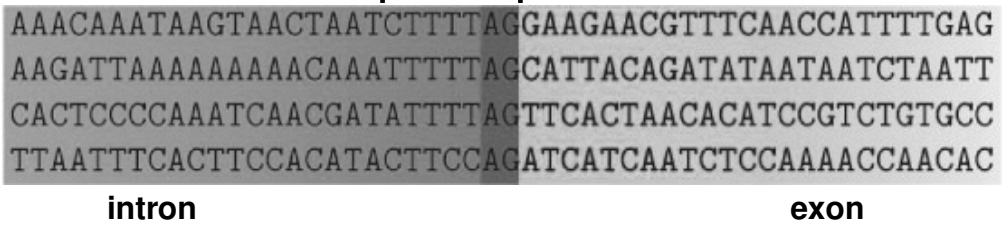
- Linear Kernel on GC-content features
- Spectrum kernel $\mathrm{k}_{\ell}^{\mathrm{Spectrum}}(\mathbf{x}, \mathbf{x}')$





Position Dependence

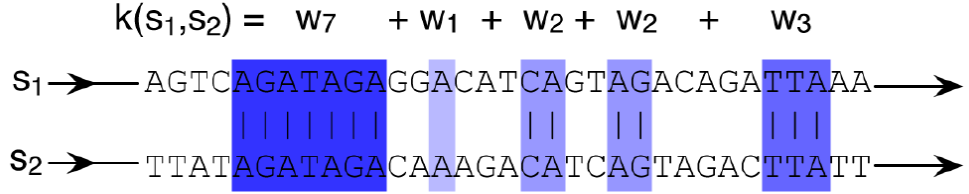
- Given: Potential acceptor splice sites



- 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)
  - Can detect motifs at specific positions
    - weak if positions vary
    - Extension: allow "shifting"

Weighted Degree Kernel

Weighted degree kernel compares two sequences by identifying the largest matching blocks which contribute depending on their length [Rätsch and Sonnenburg, 2004].

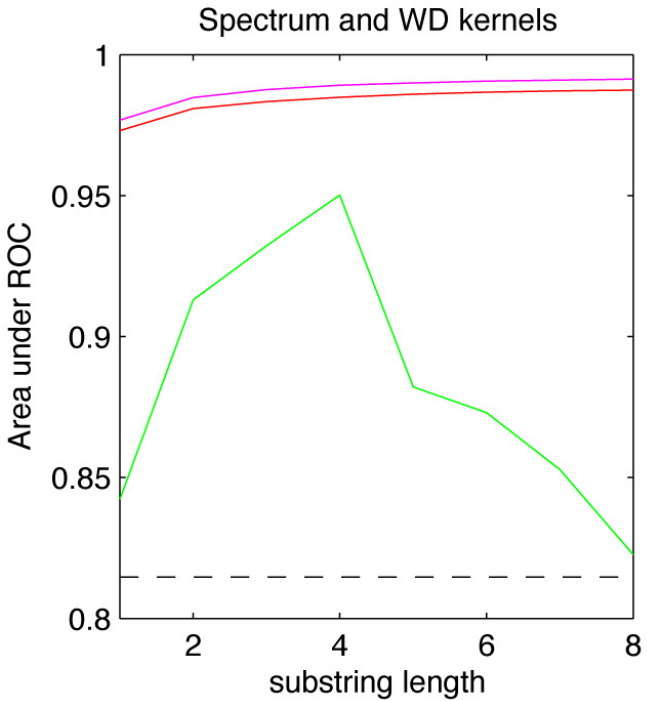


Equivalent to a mixture of spectrum kernels (up to order ℓ) at every position for appropriately chosen weights w (depending on ℓ).

Weighted degree kernel w/ shifts allows matching subsequences to be offset from each other [Rätsch et al., 2005].



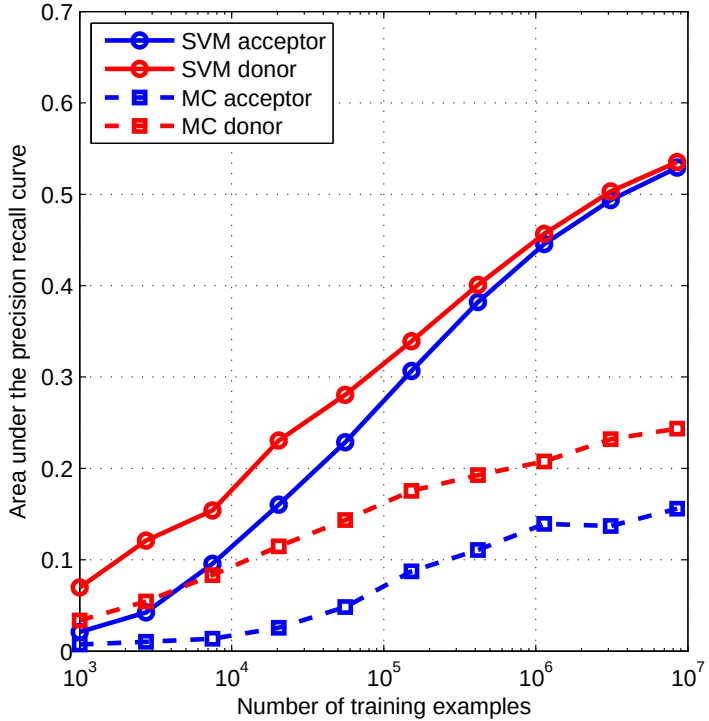
Substring Kernel Comparison

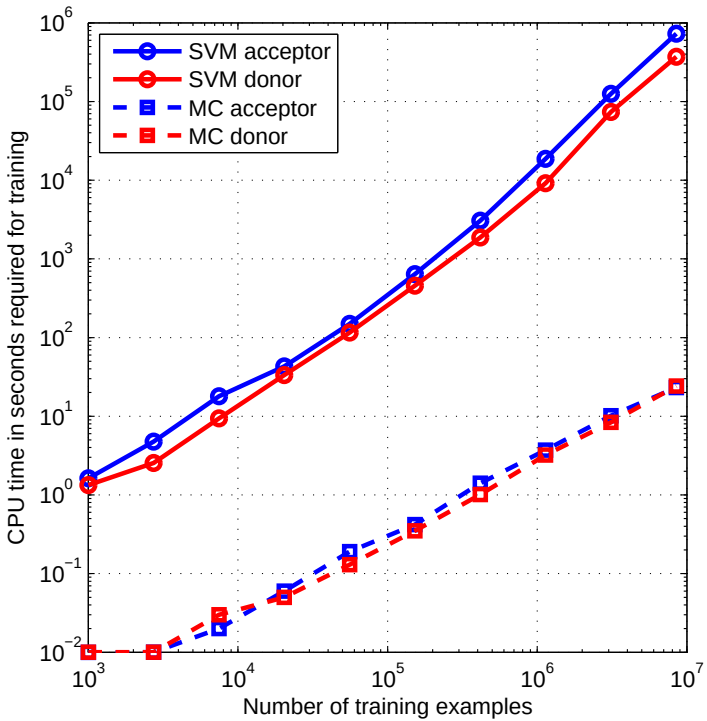


- Linear Kernel on GC-content features
- Spectrum kernel
- Weighted Degree Kernel
- Weighted Degree Kernel with shifts

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).

Results with WD Kernel (human splice sites)

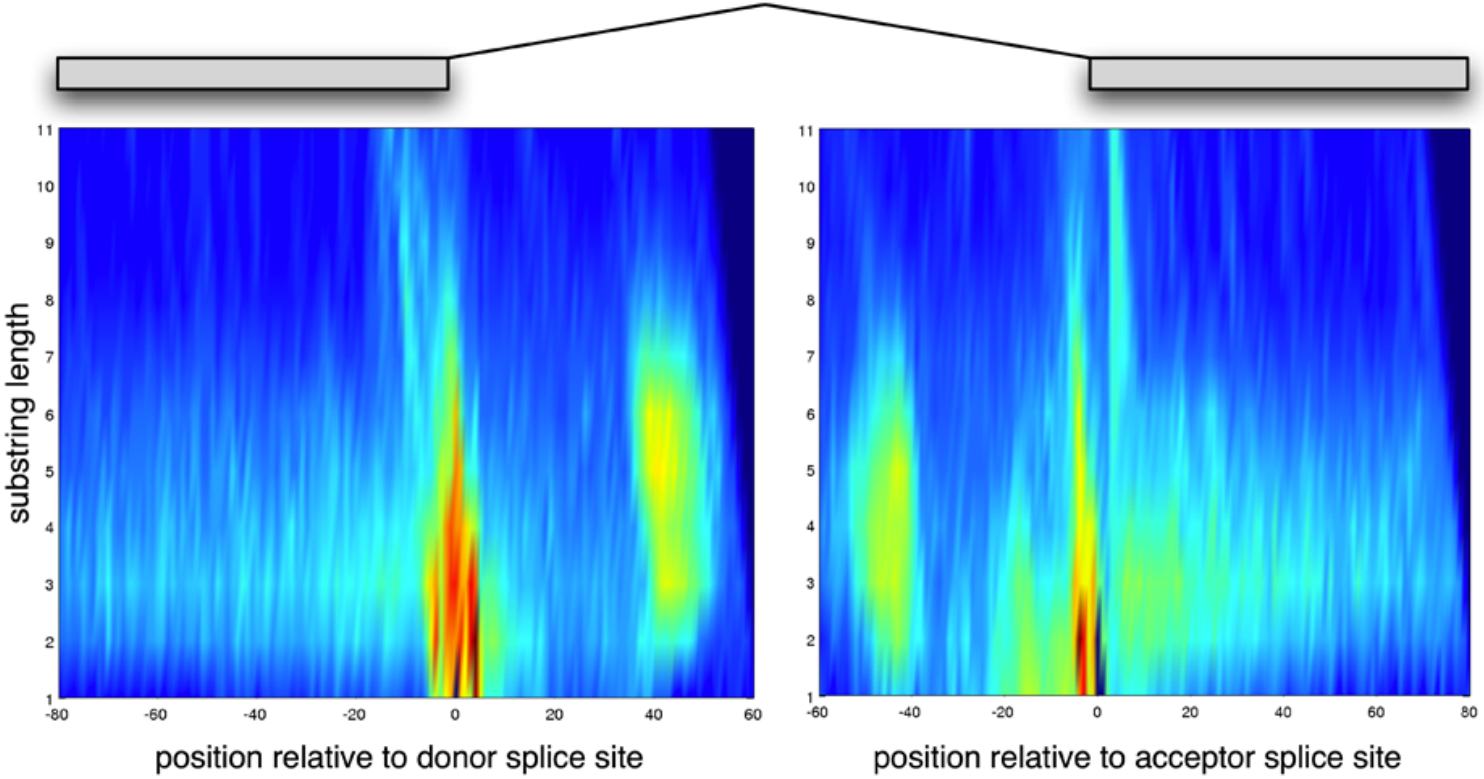




Conclusions:

- Results improve when using more data
- Training time becomes an issue

POIMs for Splicing



Color-coded importance scores of substrings near splice sites. Long substrings are important upstream of the donor and downstream of the acceptor site [Rätsch et al., 2007]

Fast string kernels?

Direct approach is slow

- Number of ℓ-mers grows exponentially with ℓ
- Hence runtime of trivial implementations degenerates

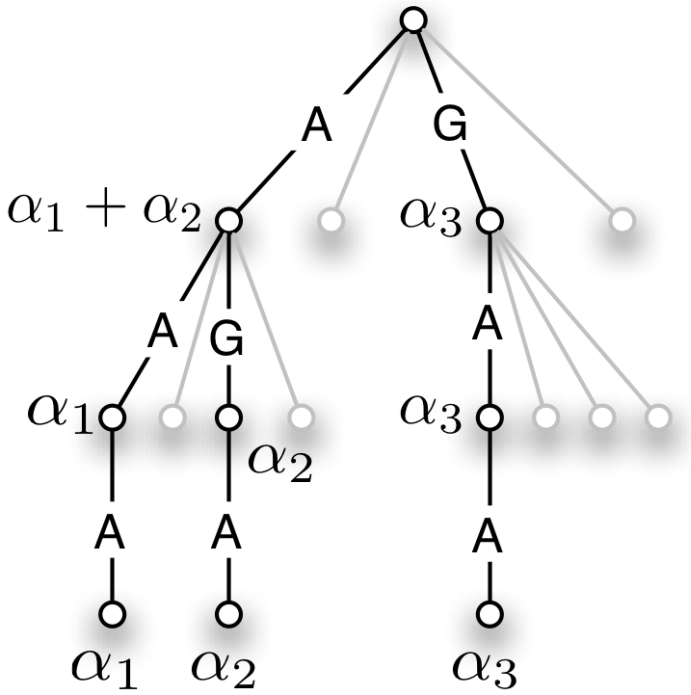
Solution

- Use index structures to speed up computation
  - Single kernel computation $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$
  - 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
  - (Suffix) trees/tries/arrays

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 (α's are the weights of the sequences).

Useful for: [Sonnenburg et al., 2006a]

- Spectrum kernel (tree)
- Mixed order spectrum k. (trie)
- Weighted degree kernel (L tries)

Fisher Kernel

General idea [Jaakkola et al., 2000, Tsuda et al., 2002a]

- Combine probabilistic models and SVMs

Sequence representation

- Arbitrary length sequences s
- Probabilistic model $p(s|\boldsymbol{\theta})$ (e.g. HMM, PSSMs)
- Maximum likelihood estimate $\boldsymbol{\theta}^* \in \mathbb{R}^d$
- Transformation into *Fisher score* features $\Phi(s) \in \mathbb{R}^d$
  - $\Phi(s) = \dfrac{\partial p(s|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$
  - Describes contribution of every parameter to $p(s|\boldsymbol{\theta})$
- $\mathrm{k}(s, s') = \langle \Phi(s), \Phi(s') \rangle$

Example: Fisher Kernel on PSSMs

- Fixed length sequences $s \in \Sigma^N$
- PSSMs: $p(s|\boldsymbol{\theta}) = \prod_{i=1}^{N} \theta_{i,s_i}$
- Fisher scores features: $(\Phi(s))_{i,\sigma} = \frac{\mathrm{d}p(s|\boldsymbol{\theta})}{\mathrm{d}\theta_{i,\sigma}} = \mathrm{Id}(s_i = \sigma)$
- Kernel: $\mathrm{k}(s, s') = \langle \Phi(s), \Phi(s') \rangle = \sum_{i=1}^{N} \mathrm{Id}(s_i = s'_i)$
- Identical to WD kernel with order 1

Note: Marginalized-count kernels [Tsuda et al., 2002b] can be understood as a generalization of Fisher kernels.

Pairwise comparison kernels

General idea [Liao and Noble, 2002]

- Employ empirical kernel map on Smith-Waterman/Blast scores

Advantage

- Utilizes decades of practical experience with Blast

Disadvantage

- High computational cost ($O(N^3)$)

Alleviation

- Employ Blast instead of Smith-Waterman
- Use a smaller subset for empirical map

Local Alignment Kernel

In order to compute the score of an alignment, one needs

- substitution matrix $S \in \mathbb{R}^{\Sigma \times \Sigma}$
- gap penalty $g : \mathbb{N} \to \mathbb{R}$

An alignment π is then scored as follows:

```
CGGSLIAMM----WFGV
|...|||||....||||
C---LIVMMNRLMWFGV
```

$$s_{S,g}(\pi) = S(C,C) + S(L,L) + S(I,I) + S(A,V) + 2S(M,M) + S(W,W) + S(F,F) + S(G,G) + S(V,V) - g(3) - g(4)$$

Smith-Waterman score (not positive definite)

$$SW_{S,g}(\mathbf{x},\mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x},\mathbf{y})} s_{S,g}(\pi)$$

Local Alignment Kernel [Vert et al., 2004]

$$K^{\beta}(\mathbf{x},\mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{x},\mathbf{y})} \exp(\beta s_{S,g}(\pi))$$

Local Alignment Kernel

In order to compute the score of an alignment, one needs

- substitution matrix $S \in \mathbb{R}^{\Sigma \times \Sigma}$
- gap penalty $g : \mathbb{N} \to \mathbb{R}$

An alignment π is then scored as follows:

```
CGGSLIAMM----WFGV
|...|||||....||||
C---LIVMMNRLMWFGV
```

$$s_{S,g}(\pi) = S(C,C) + S(L,L) + S(I,I) + S(A,V) + 2S(M,M)$$
$$+S(W,W) + S(F,F) + S(G,G) + S(V,V) - g(3) - g(4)$$

Smith-Waterman score (not positive definite)

$$SW_{S,g}(\mathbf{x},\mathbf{y}) := \max_{\pi \in \Pi(\mathbf{x},\mathbf{y})} s_{S,g}(\pi)$$

Local Alignment Kernel [Vert et al., 2004]

$$K^{\beta}(\mathbf{x},\mathbf{y}) = \sum_{\pi \in \Pi(\mathbf{x},\mathbf{y})} \exp(\beta s_{S,g}(\pi))$$

Kernel Summary

- Kernel extend SVMs to nonlinear decision boundaries, while keeping the simplicity of linear classification
- Good kernel design is important for every single data analysis task
- String kernels perform computations in very high dimensional feature space
- Kernels on strings can be:
  - Substring kernels (e.g. Spectrum & WD kernel)
  - Based on probabilistic methods (e.g. Fisher Kernel)
  - Derived from similarity measures (e.g. Alignment kernels)
- Not mentioned: Kernels on graphs, images, structures
- Application goes far beyond computational biology

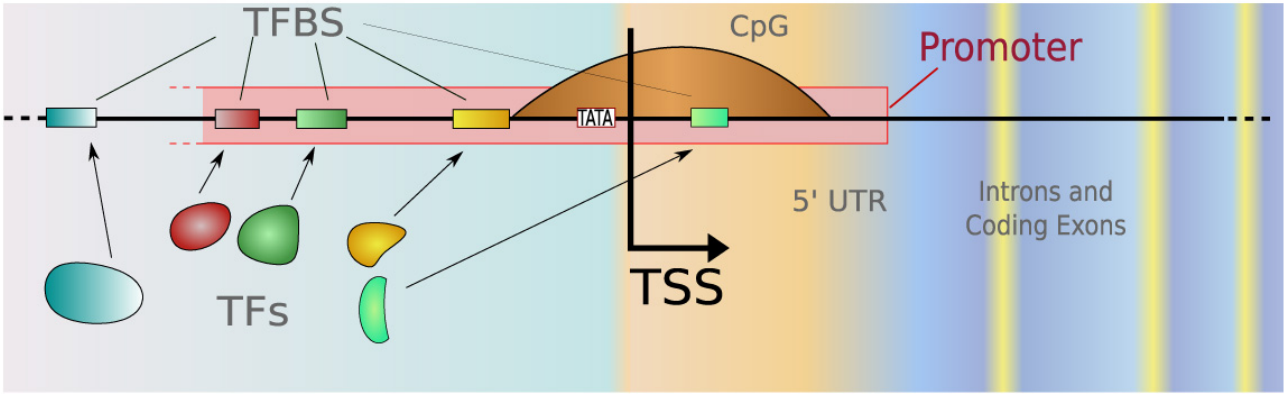
Tutorial Outline

- Basic concepts in Machine Learning
  - Classification, regression & co.
  - Generalization performance & model selection
- Support vector machines
  - “Large Margins” & the “Optimization Problem”
  - Learning nonlinearities
- Learning with Kernels
  - Kernels for sequences
  - Efficient data structures
- Applications in Computational Biology
  - Prediction of splicing events
  - Whole genome tiling arrays

Applications

- Remote homology detection [Vert et al., 2004]
- Gene finding
  - Transcriptions start [Sonnenburg et al., 2006b]
  - Splice form predictions [Rätsch et al., 2005, 2007]
- Resequencing array analysis [Clark et al., 2007]
- Protein characterization
  - Protein-protein interaction [Ben-Hur and Noble, 2005]
  - Subcellular localization [Hoglund et al., 2006]
  - Inference of networks of proteins [Kato et al., 2005]
- Inverse alignment algorithms [Rätsch et al., 2006, Joachims et al., 2005]
- Secondary structure prediction [Do et al., 2006]

Transcription Start Sites - Properties



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

SVMs with 5 sub-kernels

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

Training – Data Generation

True TSS:

- from dbTSSv4 (based on hg16) extract putative TSS windows of size $[-1000, +1000]$

Decoy TSS:

- annotate dbTSSv4 with transcription-stop (via $BLAT$ alignment of mRNAs)
- from the interior of the gene ($+100bp$ to gene end) sample negatives for training (10 per positive), again windows $[-1000, +1000]$

Processing:

- 8508 positive, 85042 negative examples
- split into disjoint training and validation set ($50\% : 50\%$)

Training & Model Selection

16 kernel parameters + SVM regularization to be tuned!

- full grid search infeasible
- local axis-parallel searches instead

SVM training/evaluation on $> 10,000$ examples computationally too demanding

Speedup trick:

$$f(\mathbf{x}) = \sum_{i=1}^{N_s} \alpha_i \mathrm{k}(\mathbf{x}_i, \mathbf{x}) + b = \underbrace{\sum_{i=1}^{N_s} \alpha_i \Phi(\mathbf{x}_i)}_{\mathbf{w}} \cdot \Phi(\mathbf{x}) + b = \mathbf{w} \cdot \Phi(\mathbf{x}) + b$$

before: $O(N_s \ell L S)$
now: $= O(\ell L) \Rightarrow \mathbf{speedup\ factor}$ up to $N_s \cdot S$

$\Rightarrow$ **Large scale training and evaluation possible**

Experimental Comparison

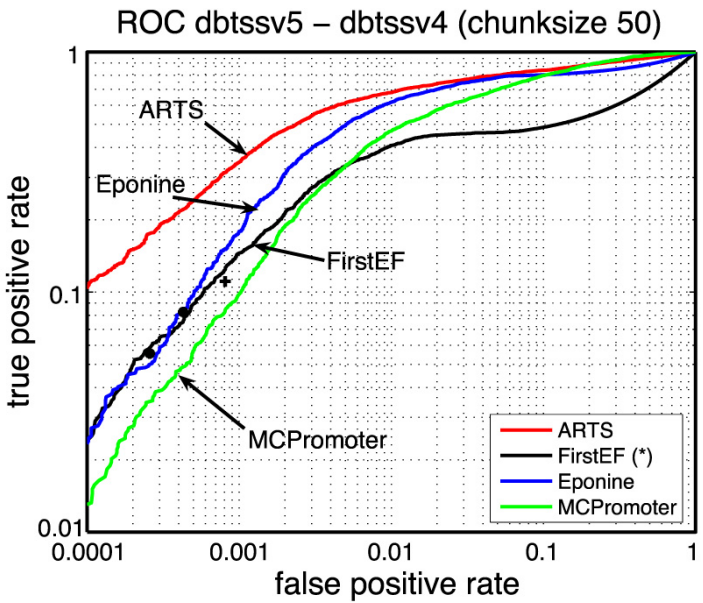
Current state-of-the-art methods:

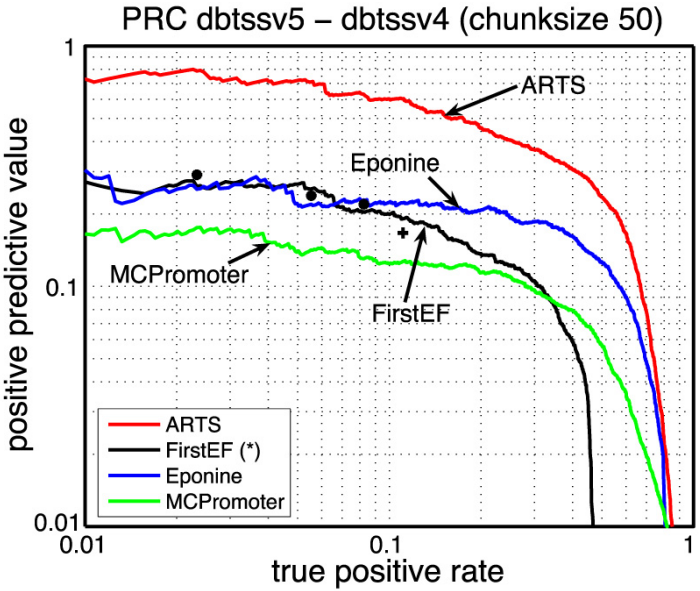
- **FirstEF** [Davuluri et al., 2001]
  DA: uses distance from CpG islands to first donor site
- **McPromotor** [Ohler et al., 2002]
  3-state HMM: upstream, TATA, downstream
- **Eponine** [Down and Hubbard, 2002]
  RVM: upstream CpG islands, window upstream of TATA, for TATA, downstream

$\Rightarrow$ **Do a genome wide evaluation!**
$\Rightarrow$ **How to do a fair comparison?**

Results

Receiver Operator Characteristic Curve *and* Precision Recall Curve

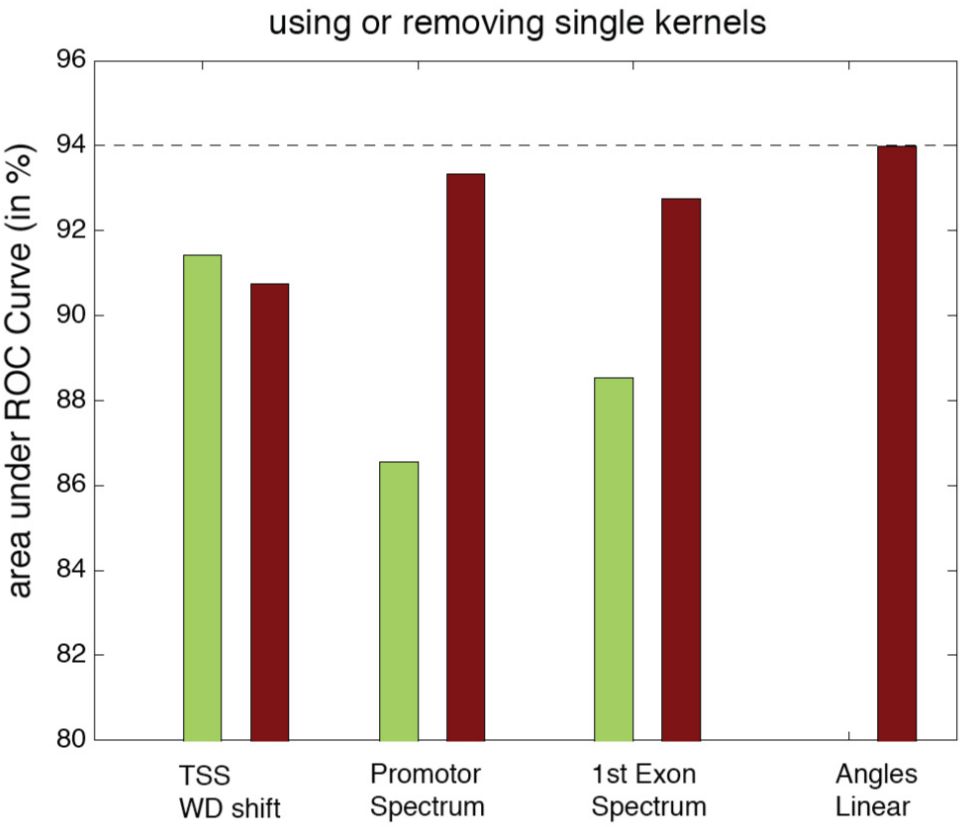




$\Rightarrow 35\%$ **true positives at a false positive rate of** $1/1000$
(best other method find about a half (18%**))**

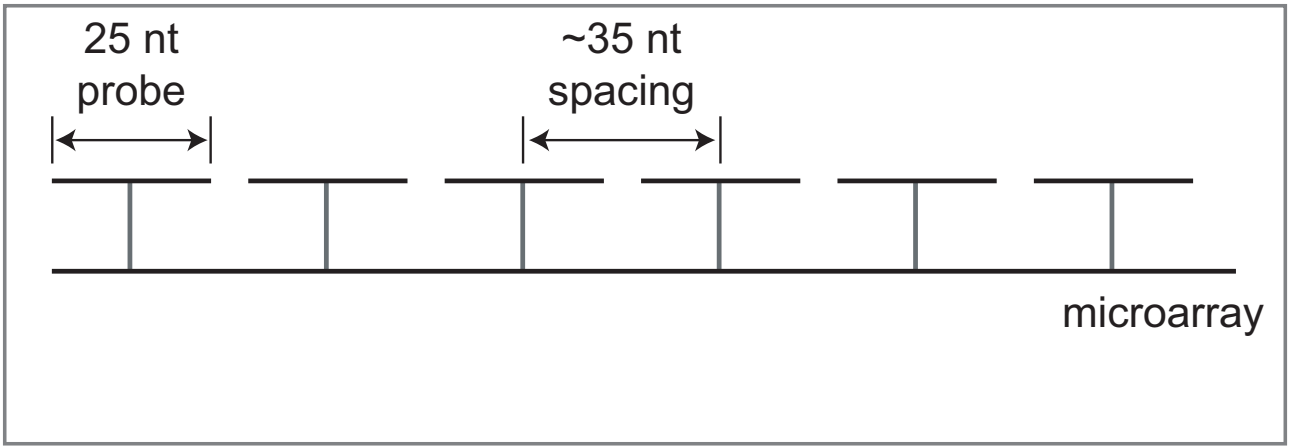
See Sonnenburg et al. [2006b] for more details.

Which kernel is most important?

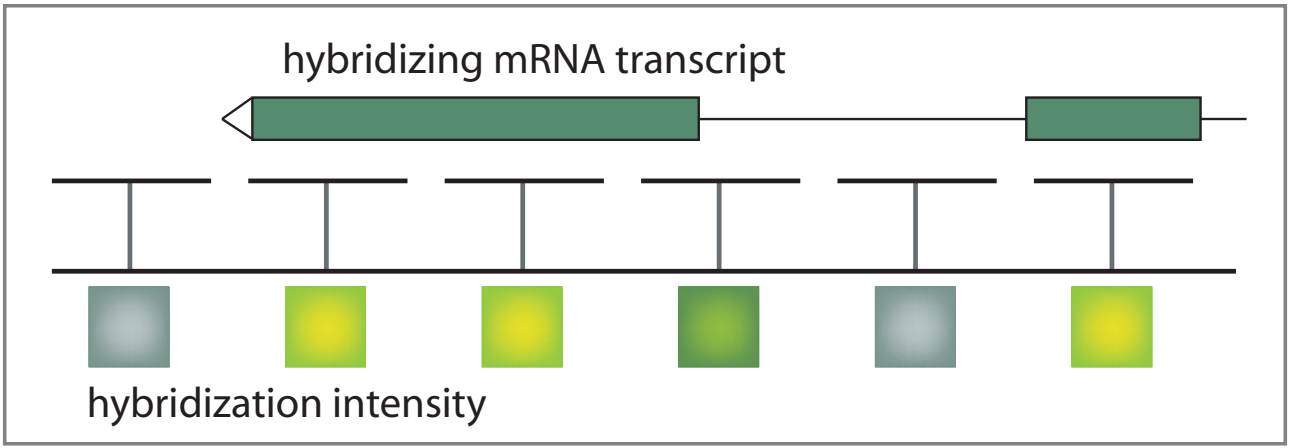


$\Rightarrow$ **Weighted Degree Shift kernel modeling TSS signal**

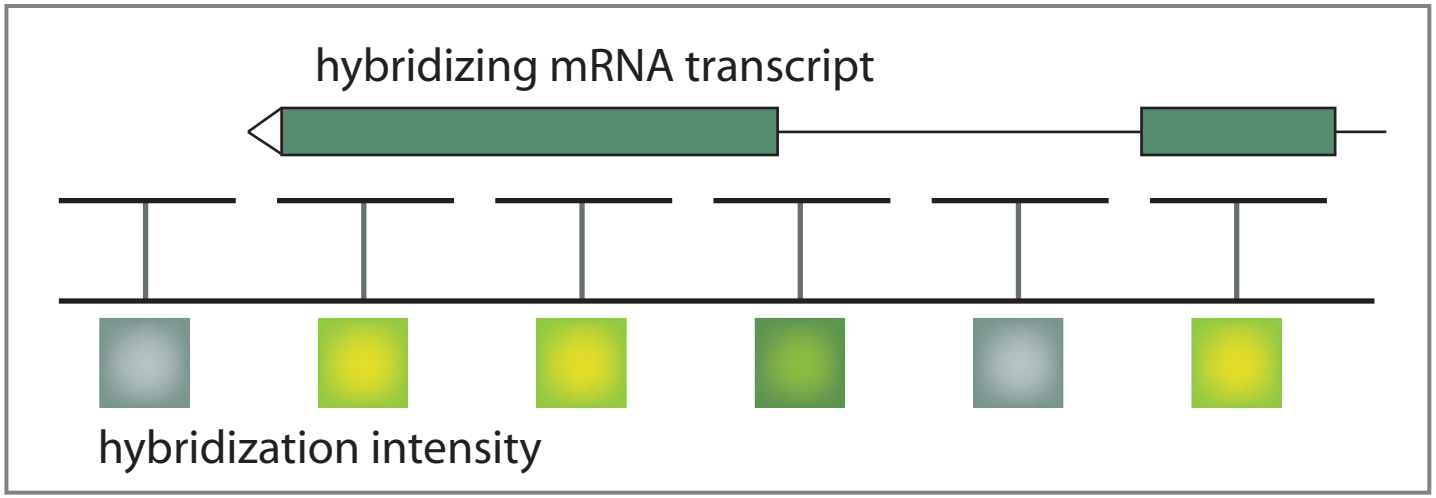
Tiling arrays for transcriptome analysis



Tiling arrays for transcriptome analysis

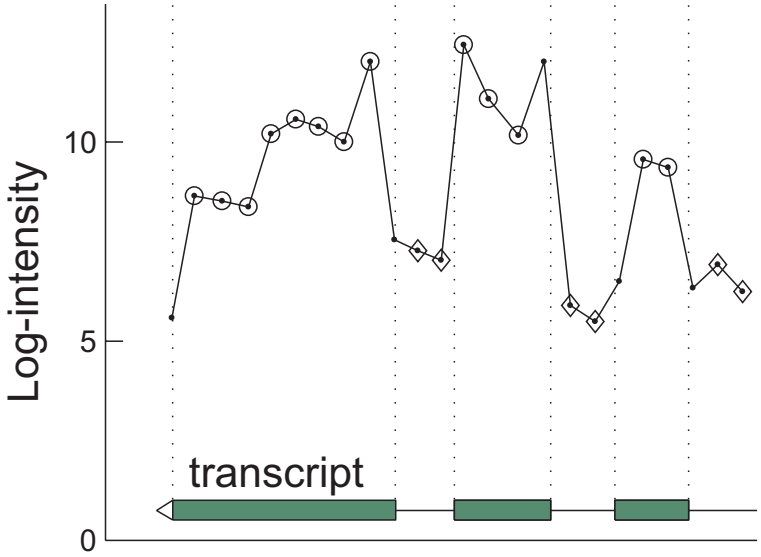


Tiling arrays for transcriptome analysis



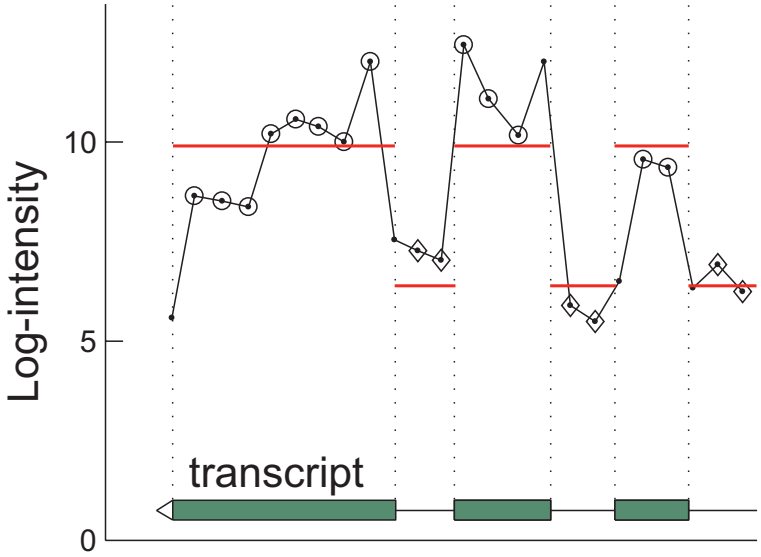
- whole-genome quantitative measurements
- cost-effective
  $\Rightarrow$ replicates affordable, many tissues / conditions
- unbiased
  $\Rightarrow$ does not rely on annotations or known cDNAs

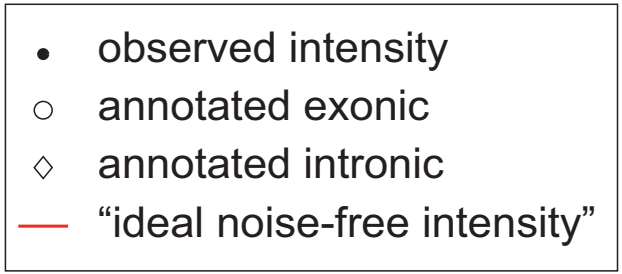
Intensities are noisy



- observed intensity
- annotated exonic
- annotated intronic

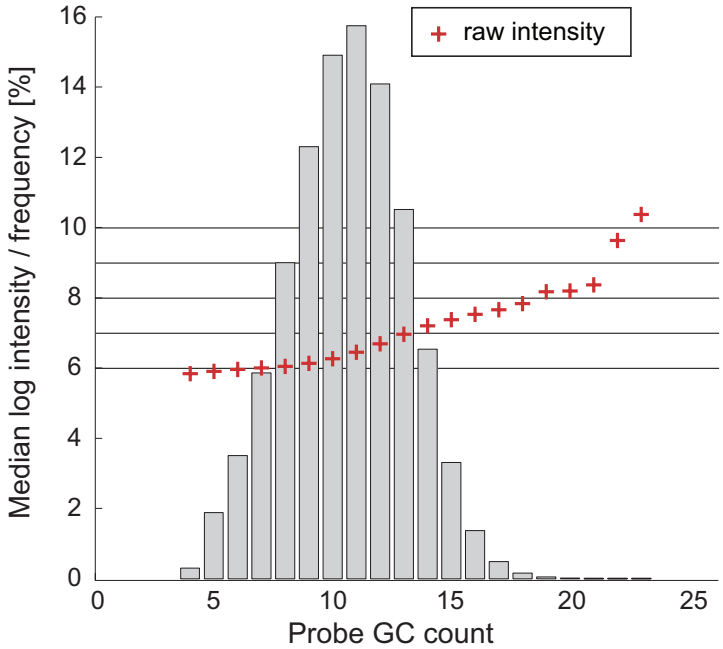
Intensities are noisy





Systematic bias induced by probe sequence effects
$\Rightarrow$ model effect for normalization

Intensity depends on probe seq.

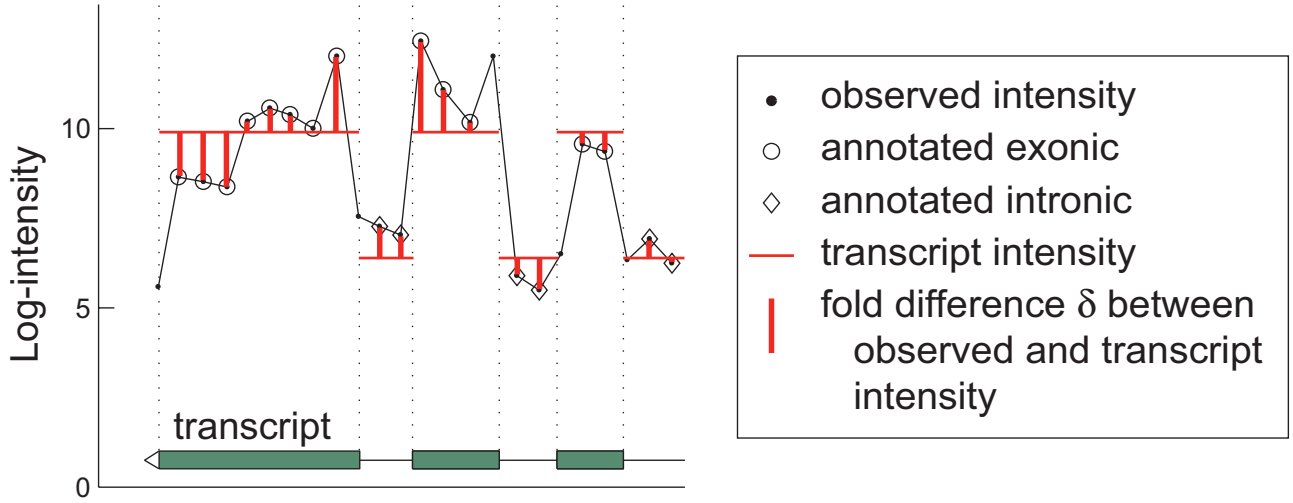


Results for the hybridization of polyadenylated RNA samples from *Arabidopsis thaliana*.

Previously proposed: Sequence Quantile Normalization [Royce et al., 2007]

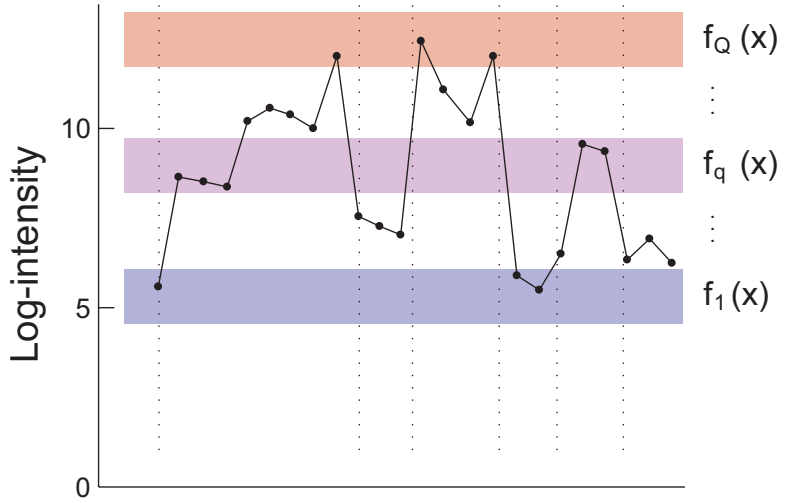
Transcript normalization

- Assume constant transcript intensities $\overline{y}_i$
- Learn deviation from transcript intensity $\delta_i := y_i - \overline{y}_i$
- Take probe sequence $\boldsymbol{x}_i$ as input for regression.
  Model probe seq. effect depending on y_i: $f(\boldsymbol{x}_i, y_i) \approx \delta_i$



Transcript normalization

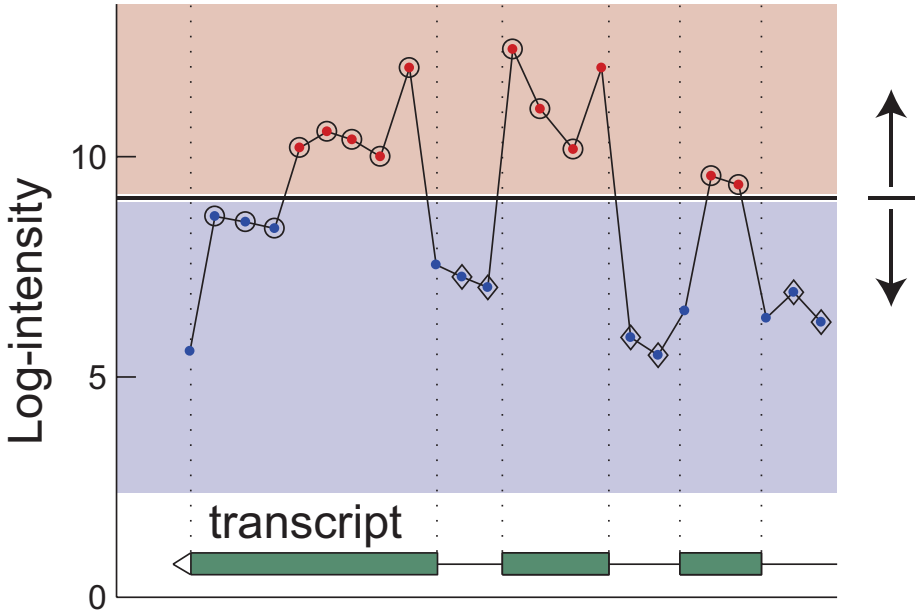
- Assume constant transcript intensities $\overline{y}_i$
- Learn deviation from transcript intensity $\delta_i := y_i - \overline{y}_i$
- Take probe sequence $\boldsymbol{x}_i$ as input for regression.
  Model probe seq. effect depending on y_i: $f(\boldsymbol{x}_i, y_i) \approx \delta_i$



Discretize y into $Q = 20$ quantiles and estimate Q independent functions $f_1(\boldsymbol{x}), \ldots, f_Q(\boldsymbol{x})$

Linear regression $f_q(\boldsymbol{x}) = \boldsymbol{w}_q^T \boldsymbol{x}$

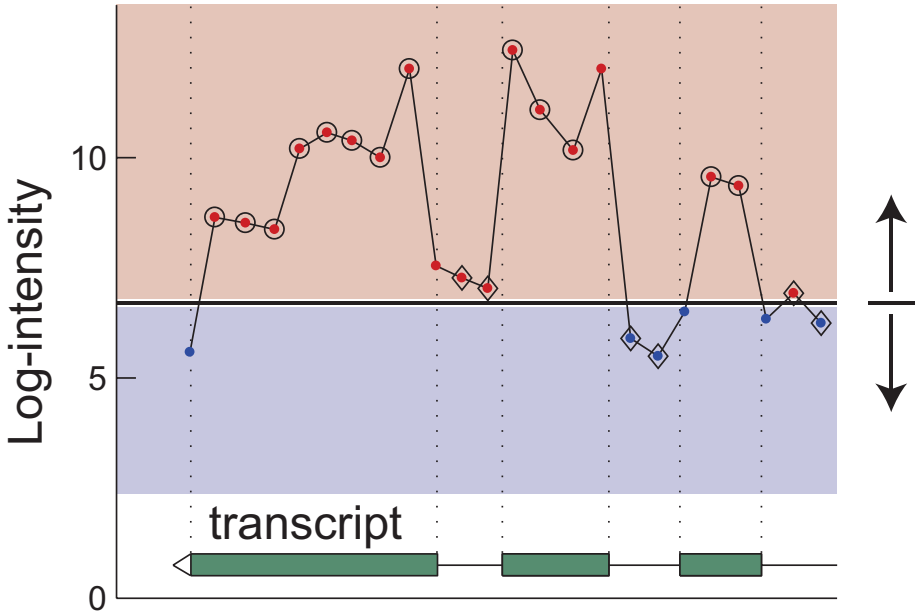
Exon-background separation



Global thresholding of probe intensities

$\Rightarrow$ bi-partition into active (exonic) and inactive (intronic/intergenic) probes.

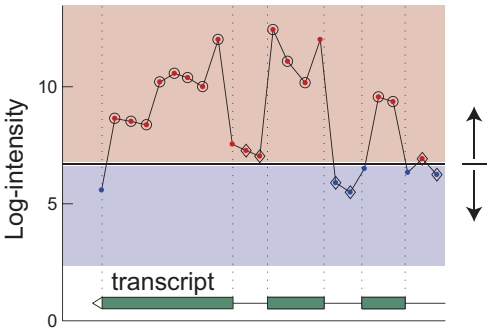
Exon-background separation

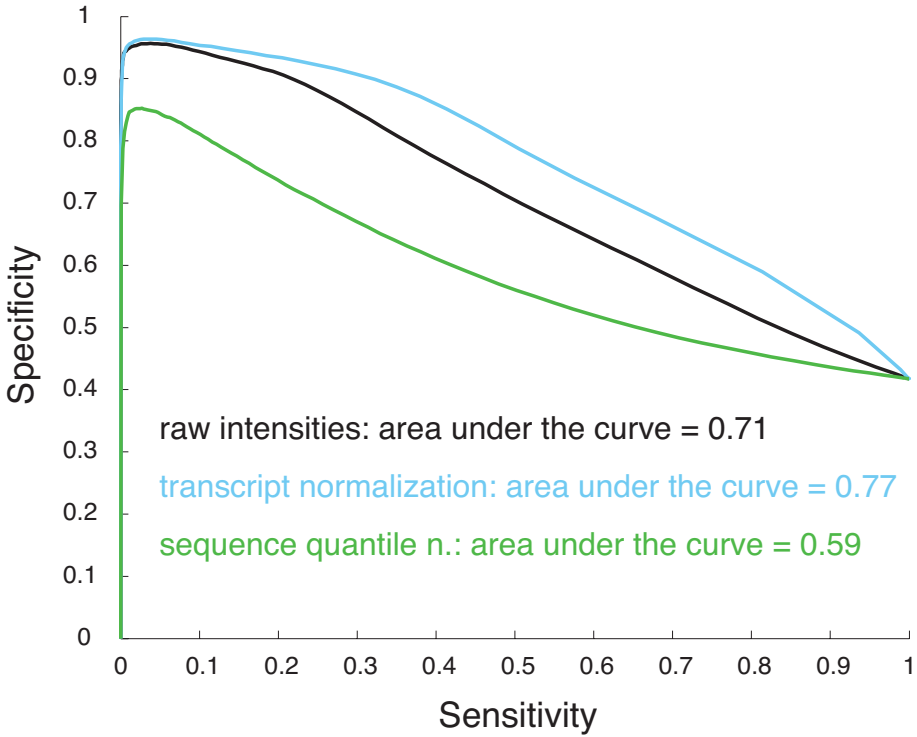


Global thresholding of probe intensities

$\Rightarrow$ bi-partition into active (exonic) and inactive (intronic/intergenic) probes.

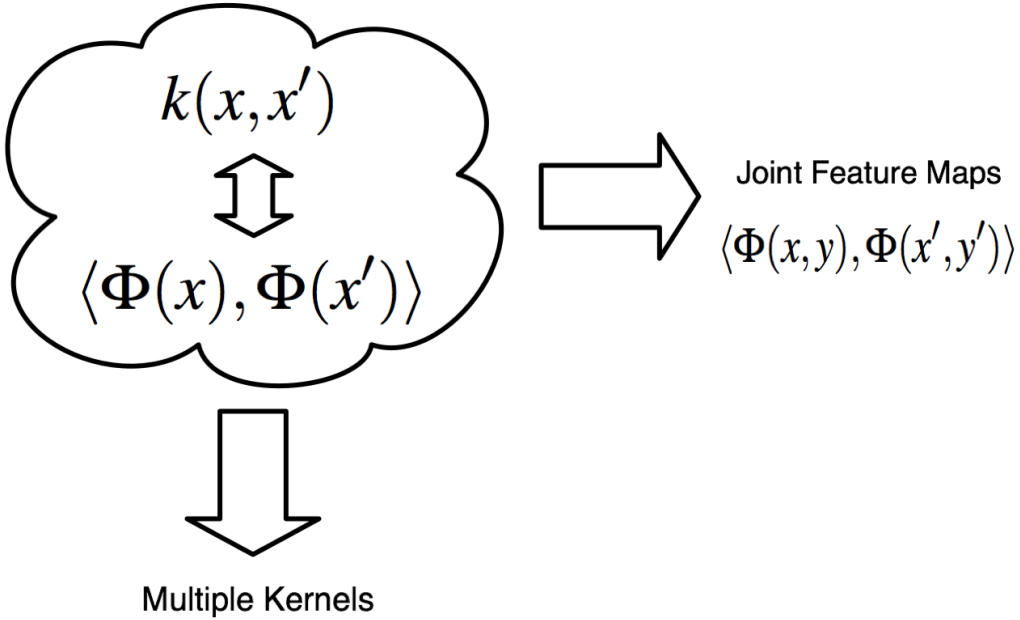
Exon-background separation





Results for the hybridization of polyadenylated RNA samples from *Arabidopsis thaliana*.

Last Topic: Generalizing kernels



- Finding the optimal combination of kernels
- Learning structured output spaces

Multiple Kernel Learning (MKL)

Possible solution We can add the two kernels, that is

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

Better solution We can mix the two kernels,

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

where t should be estimated from the training data.

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

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

Applications

- Heterogeneous data [Lanckriet et al., 2004]
- Improving interpretability [Rätsch et al., 2006]

Structured Output Spaces

Learning Task

For a set of labeled data, we predict the label.

Difference from multiclass

The set of possible labels $\mathcal{Y}$ may be very large or hierarchical.

Joint kernel on $\mathcal{X}$ and $\mathcal{Y}$

We define a joint feature map on $\mathcal{X} \times \mathcal{Y}$, denoted by $\Phi(\mathbf{x}, y)$. Then the corresponding kernel function is

$$k((\mathbf{x}, y), (\mathbf{x}', y')) := \langle \Phi(\mathbf{x}, y), \Phi(\mathbf{x}', y') \rangle.$$

For multiclass

For normal multiclass classification, the joint feature map decomposes and the kernels on $\mathcal{Y}$ is the identity, that is

$$k((\mathbf{x}, y), (\mathbf{x}', y')) := [[y = y']] k(\mathbf{x}, \mathbf{x}').$$

Label Sequence Learning

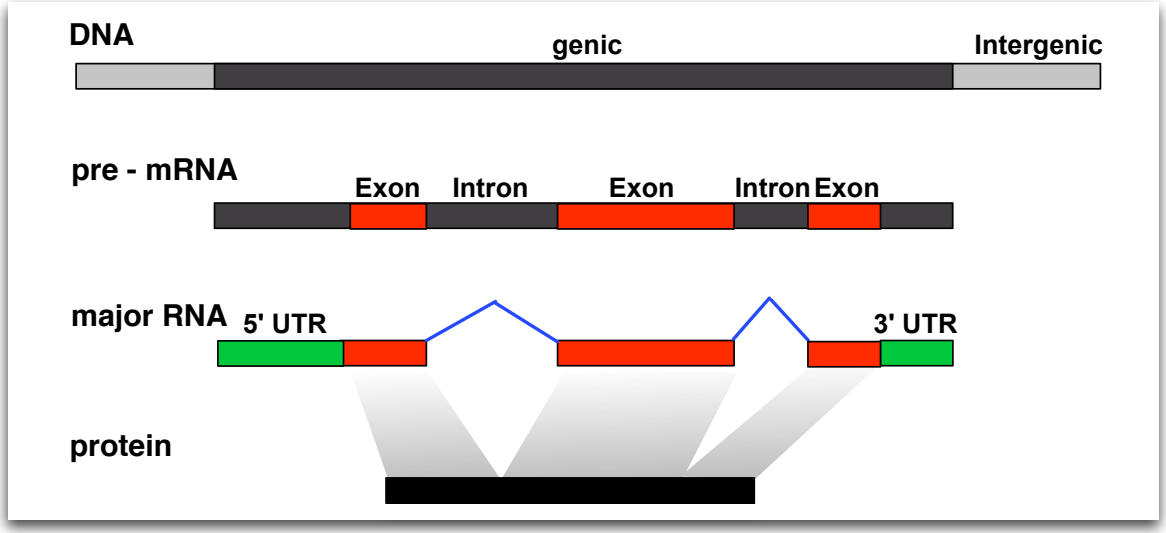
- Given: observation sequence
- Problem: predict corresponding state sequence
- Often: several subsequent positions have the same state
  $\Rightarrow$ state sequence defines a “segmentation”
- Example 1: Protein Secondary Structure Prediction



Residue Sequence:	NWVLSTAADMQGVVTDGMASFLDKD
Secondary Structure:	LLEEEELLLLHHHHHHHHHHLHHHL

Label Sequence Learning

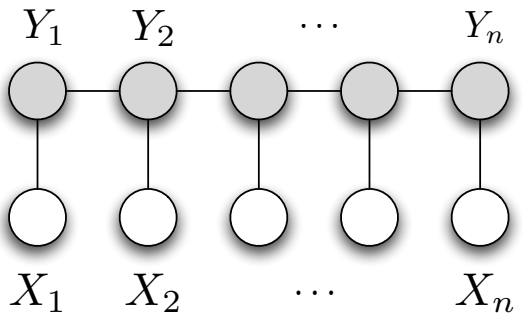
- Given: observation sequence
- Problem: predict corresponding state sequence
- Often: several subsequent positions have the same state
  $\Rightarrow$ state sequence defines a “segmentation”
- Example 2: Gene Finding



Generative Models

- Hidden Markov Models [Rabiner, 1989]
  - State sequence treated as Markov chain
  - No direct dependencies between observations
  - Example: first-order HMM (simplified)

$$p(\mathbf{x}, \mathbf{y}) = \prod_i p(x_i|y_i)p(y_i|y_{i-1})$$



  - Efficient dynamic programming (DP) algorithms

Decoding *via* Dynamic Programming

$$\begin{aligned}\log p(\mathbf{x},\mathbf{y}) &= \sum_i (\log p(x_i|y_i) + \log p(y_i|y_{i-1})) \\ &= \sum_i g(y_{i-1}, y_i, x_i)\end{aligned}$$

with $g(y_{i-1}, y_i, x_i) = \log p(x_i|y_i) + \log p(y_i|y_{i-1})$.

Problem: Given sequence $\mathbf{x}$, find sequence $\mathbf{y}$ such that $\log p(\mathbf{x},\mathbf{y})$ is maximized, i.e. $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}^n} \log p(\mathbf{x},\mathbf{y})$

Dynamic Programming Approach:

$$V(i,y) := \begin{cases} \max_{y'\in\mathcal{Y}} (V(i-1,y') + g(y',y,x_i)) & i > 1 \\ 0 & \text{otherwise} \end{cases}$$

Max-Margin Structured Output Learning

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

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

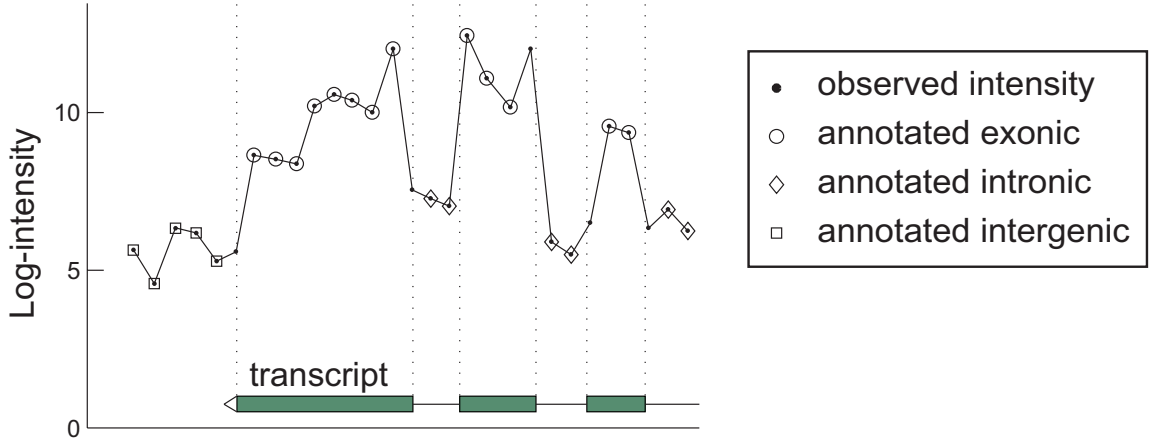
- Given N sequence pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$ for training
- Determine f such that there is a large margin between true and wrong segmentations

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

- Exponentially many constraints!

Segmentation task

Goal: characterize each probe as either intergenic, exonic or intronic

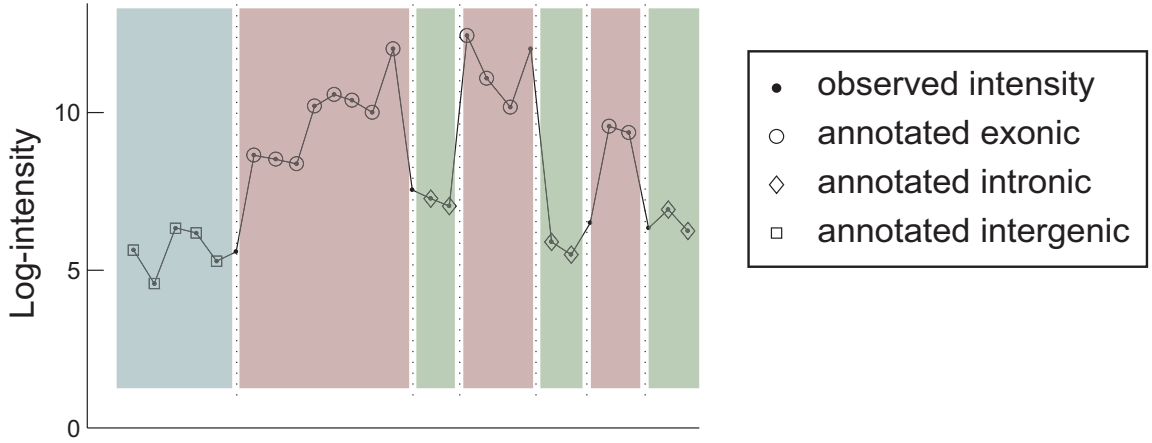


Extension of a previously proposed segmentation method [Huber et al., 2006]

- model non-Gaussian noise
- account for spliced transcripts
- learn to predict an appropriate number of segments

Segmentation task

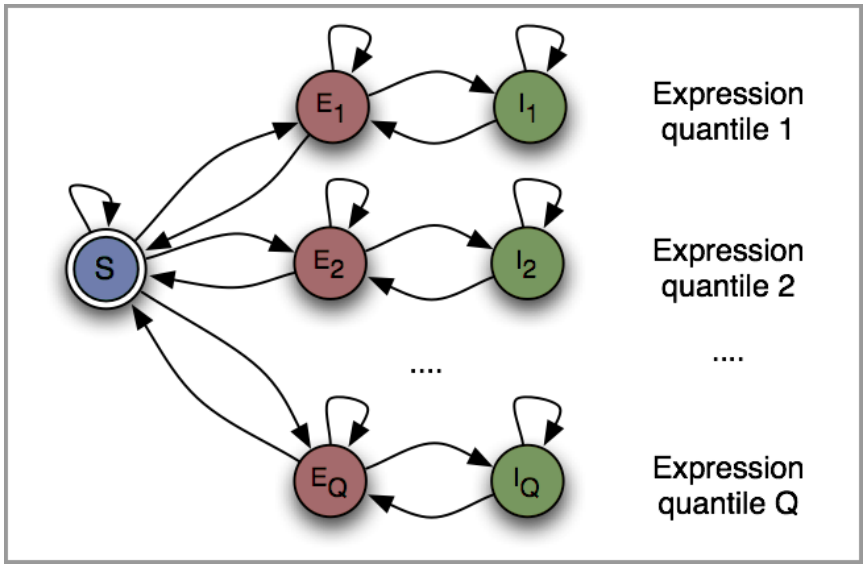
Goal: characterize each probe as either intergenic, exonic or intronic



Extension of a previously proposed segmentation method [Huber et al., 2006]

- model non-Gaussian noise
- account for spliced transcripts
- learn to predict an appropriate number of segments

Segmentation



$Q = 20$ **discrete expression levels**

Learn to associate a state with each probe given its hybridization signal and local context

Use regions around annotated genes (TAIR7) for training.

Discriminative training

$f : \mathbb{R}^\star \to \Sigma^\star$

given a sequence of hybridization measurements $\boldsymbol{\chi} \in \mathbb{R}^\star$ predicts a state sequence (path) $\boldsymbol{\sigma} \in \Sigma^\star$

Discriminant function $F_{\boldsymbol{\theta}} : \mathbb{R}^\star \times \Sigma^\star \to \mathbb{R}$ such that for decoding: $f(\boldsymbol{\chi}) = \underset{\boldsymbol{\sigma} \in \mathcal{S}^\star}{\operatorname{argmax}} F_{\boldsymbol{\theta}}(\boldsymbol{\chi}, \boldsymbol{\sigma})$.

Training:

For each training example $(\boldsymbol{\chi}^{(i)}, \boldsymbol{\sigma}^{(i)})$, enforce a **large margin of separation**

$$F_\theta(\boldsymbol{\chi}^{(i)}, \boldsymbol{\sigma}^{(i)}) - F_\theta(\boldsymbol{\chi}^{(i)}, \overline{\boldsymbol{\sigma}}) \geq \rho$$

between correct path $\boldsymbol{\sigma}^{(i)}$ and *any* wrong path $\overline{\boldsymbol{\sigma}} \neq \boldsymbol{\sigma}^{(i)}$.

[Altun et al., 2003, Rätsch et al., 2007, Zeller et al., 2008]

Result: segmentation accuracy

Segmentation accuracy on probe level

	Global thresholding	HM-SVM
Raw intensities	70.4%	77.1%
Sequence quantile normalized	65.5%	70.9%
Transcript normalized	73.9%	**82.5%**

- HM-SVMs segment hybridization data more accurately than naive thresholding approach.
- Segmentation accuracy is further improved by transcript normalization.

Last slide!

“Machine Learning for Predictive Sequence Analysis”

- **Machine Learning and Kernels**
  - General purpose kernels
  - Substring kernels & data structures
  - Kernels from probabilistic methods
- **Prediction methods**
  - Classification & Regression
  - Generalization & Model selection
- **Sequence Analysis**
  - Automated genome annotation
  - Protein characterization
  - New alignment algorithms

Last slide!

“Machine Learning for Predictive Sequence Analysis”

- **Machine Learning and Kernels**
  - General purpose kernels
  - Substring kernels & data structures
  - Kernels from probabilistic methods
- **Prediction methods**
  - Classification & Regression
  - Generalization & Model selection
- **Sequence Analysis**
  - Automated genome annotation
  - Protein characterization
  - New alignment algorithms

Last slide!

“Machine Learning for Predictive Sequence Analysis”

- **Machine Learning and Kernels**
  - General purpose kernels
  - Substring kernels & data structures
  - Kernels from probabilistic methods
- **Prediction methods**
  - Classification & Regression
  - Generalization & Model selection
- **Sequence Analysis**
  - Automated genome annotation
  - Protein characterization
  - New alignment algorithms

Last slide!

“Machine Learning for Predictive Sequence Analysis”

- **Machine Learning and Kernels**
  - General purpose kernels
  - Substring kernels & data structures
  - Kernels from probabilistic methods
- **Prediction methods**
  - Classification & Regression
  - Generalization & Model selection
- **Sequence Analysis**
  - Automated genome annotation
  - Protein characterization
  - New alignment algorithms

Thank you!

Last slide!

“Machine Learning for Predictive Sequence Analysis”

- **Machine Learning and Kernels**
  - General purpose kernels
  - Substring kernels & data structures
  - Kernels from probabilistic methods
- **Prediction methods**
  - Classification & Regression
  - Generalization & Model selection
- **Sequence Analysis**
  - Automated genome annotation
  - Protein characterization
  - New alignment algorithms

Questions?

References

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov Support Vector Machines. In *Proc. 20th Int. Conf. Mach. Learn.*, pages 3–10, 2003.

A. Ben-Hur and W.S. Noble. Kernel methods for predicting proteinprotein interactions. *Bioinformatics*, 21(Suppl 1):i38–i46, 2005.

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

R.M. Clark, S. Ossowski, G. Schweikert, G. Zeller, P. Shinn, N. Warthmann, G. Fu, D. Hin ds, H. Chen, K. Frazer, D. Huson, B. Schölkopf, M. Nordborg, J. Ecker, G. Rätsch, and D. Weigel. An inventory of sequence polymorphisms for arabidopsis. submitted, 2007.

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

R.V. Davuluri, I. Grosse, and M.Q. Zhang. Computational identification of promoters and first exons in t he human genome. *Nat Genet*, 29(4):412–417, December 2001.

C.B. Do, D.A. Woods, and S. Batzoglou. Contrafold: Rna secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–e98, 2006.

T.A. Down and T.J.P. Hubbard. Computational detection and location of transcription start sites in mammalian genomic DNA. *Genome Res*, 12:458–461, 2002.

R.O. Duda, P.E.Hart, and D.G.Stork. *Pattern classification*. John Wiley & Sons, second edition, 2001.

A. Hoglund, P. Donnes, T. Blum, H.W. Adolph, and O. Kohlbacher. Multiloc: prediction of protein subcellular localization using n-terminal targeting sequences, sequence motifs and amino acid composition. *Bioinformatics*, 22(10):1158–65, 2006.

W. Huber, J. Toedling, and L. M. Steinmetz. Transcript mapping with high-density oligonucleotide tiling arrays. *Bioinformatics*, 22(6): 1963–1970, 2006.

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

T. Joachims, T. Galor, and R. Elber. Learning to align sequences: A maximum-margin approach. In B. Leimkuhler, C. Chipot, R. Elber, A. Laaksonen, and A. Mark, editors, *New Algorithms for Macromolecular Simulation*, number 49 in LNCS, pages 57–71. Springer, 2005.

T. Kato, K. Tsuda, and K. Asai. Selective integration of multiple biological data for supervised network inference. *Bioinformatics*, 21 (10):2488–95, 2005.

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

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

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

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

U. Ohler, G.C. Liao, H. Niemann, and G.M. Rubin. Computational analysis of core promoters in the drosophila genome. *Genome Biol*, 3(12):RESEARCH0087, 2002.

L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2): 257–285, February 1989.

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. Rätsch, S. Sonnenburg, and C. Schäfer. Learning interpretable svms for biological sequence classification. *BMC Bioinformatics*, 7 (Suppl 1):S9, February 2006.

G. Rätsch, S. Sonnenburg, J. Srinivasan, H. Witte, K.-R. Müller, R. Sommer, and B. Schl̈kopf. Improving the c. elegans genome annotation using machine learning. *PLoS Computational Biology*, 3(2):e20, 2007. URL http://dx.doi.org/10.1371/journal.pcbi.0030020.eor.

Gunnar Rätsch, Bettina Hepp, Uta Schulze, and Cheng Soon Ong. PALMA: Perfect alignments using large margin algorithms. In *German Conference on Bioinformatics*, 2006.

T.E. Royce, J.S. Rozowsky, and M.B. Gerstein. Assessing the need for sequence-based normalization in tiling microarray experiments. *Bioinformatics*, 23(8):988–997, 2007. doi: 10.1093/bioinformatics/btm052. URL http://bioinformatics.oxfordjournals.org/cgi/content/abstract/23/8/988.

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

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

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

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

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

K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18:268S–275S, 2002b.

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

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

G. Zeller, S. Henz, S. Laubinger, D. Weigel, and G. Rätsch. Transcript normalization and segmentation of tiling array data. In *Proc. PSB 2008*. World Scientific, 2008.