Support Vector Machines and Kernels for Computational Biology

(SVMs, Kernels, and Beyond)

Gunnar Rätsch and Sören Sonnenburg

Friedrich Miescher Laboratory, Max Planck Society,
Tübingen, Germany



Friedrich Miescher Laboratory
of the Max Planck Society

Tutorial Outline

1. Introduction to Machine Learning
   - Classification, Regression, and Structure prediction
   - Complexity and Model Selection
2. Support Vector Machines and Kernels
   - Large Margin Separation
   - Non-linear Separation with Kernels
3. Kernels for Structured Data
   - Substring Kernels for Biological Sequences
   - Kernels for Graphs & Images
4. Useful Extensions of SVMs
   - Heterogeneous Data & Multiple Kernel Learning
   - Understanding the Learned SVM Classifier
5. Structured Output Learning
   - HMMs & Label Sequence Learning
   - Semi-Markov Extensions
6. Case Studies (Applications)
   - Transcription Start Site Prediction and Gene Finding
   - Tiling Array Analysis and Short Read Alignments

Material

Supporting Material is available online

- Slides
- Tutorial Example Scripts
- Software
- Toy Datasets
- Links

`http://www.fml.mpg.de/raetsch/lecture/ismb09tutorial`

Part I

Introduction to Machine Learning

Overview: Introduction to Machine Learning

Example: Sequence Classification
- Running Example

Empirical Inference
- Learning from Examples
- Loss Functions
- Measuring Complexity

Digestion
- Putting Things together
- Measuring the Performance
- Examples of Inference Problems

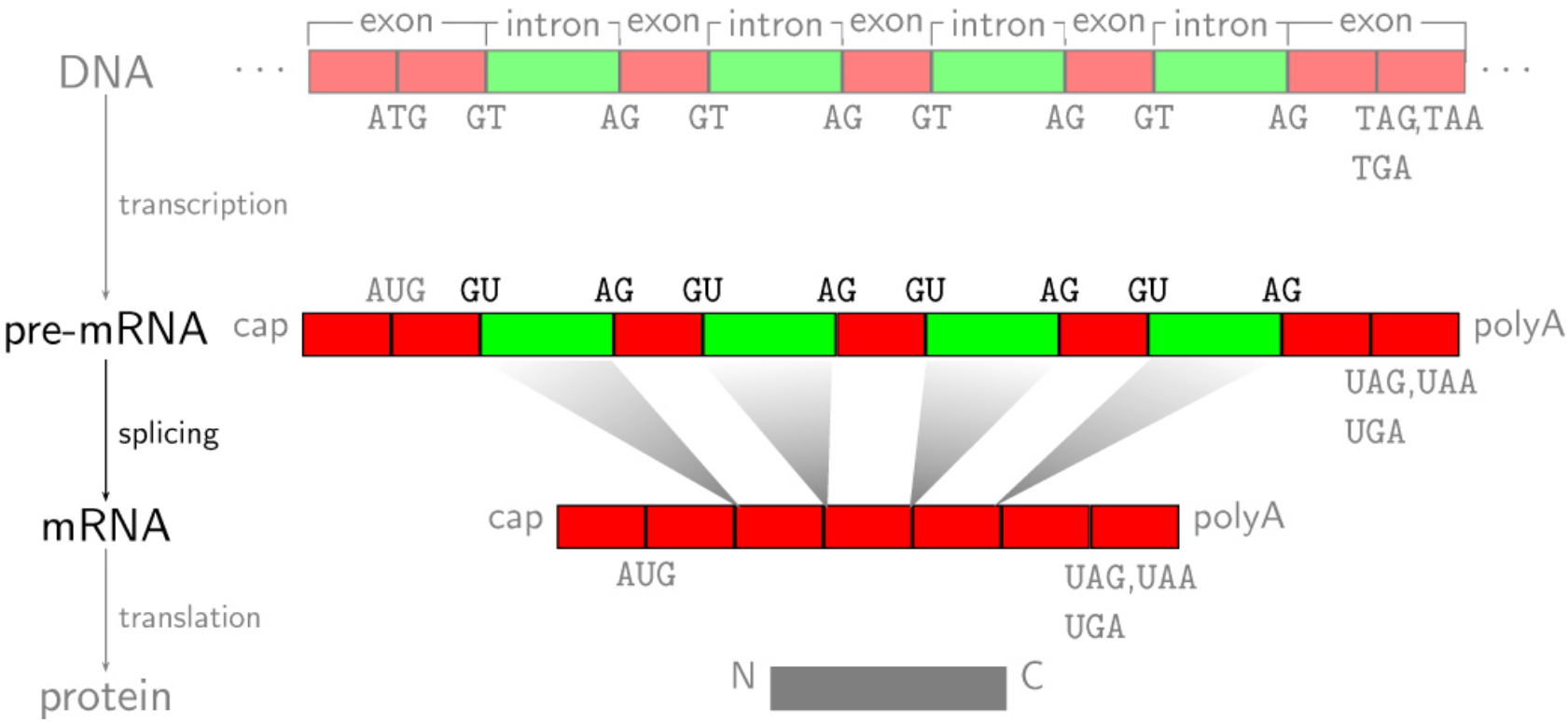
Why machine learning?



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

Let the data do the talking!

Running Example: Splicing

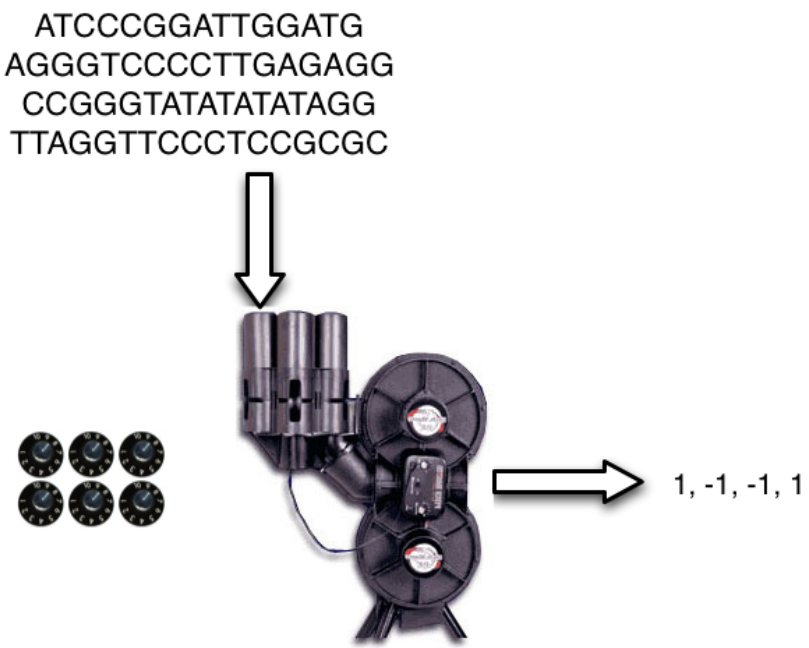


- Almost all *donor splice sites* exhibit `GU`
- Almost all *acceptor splice site* exhibit `AG`
- Not all `GU`s and `AG`s are used as splice site

Classification of Sequences

Example: Recognition of splice sites

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



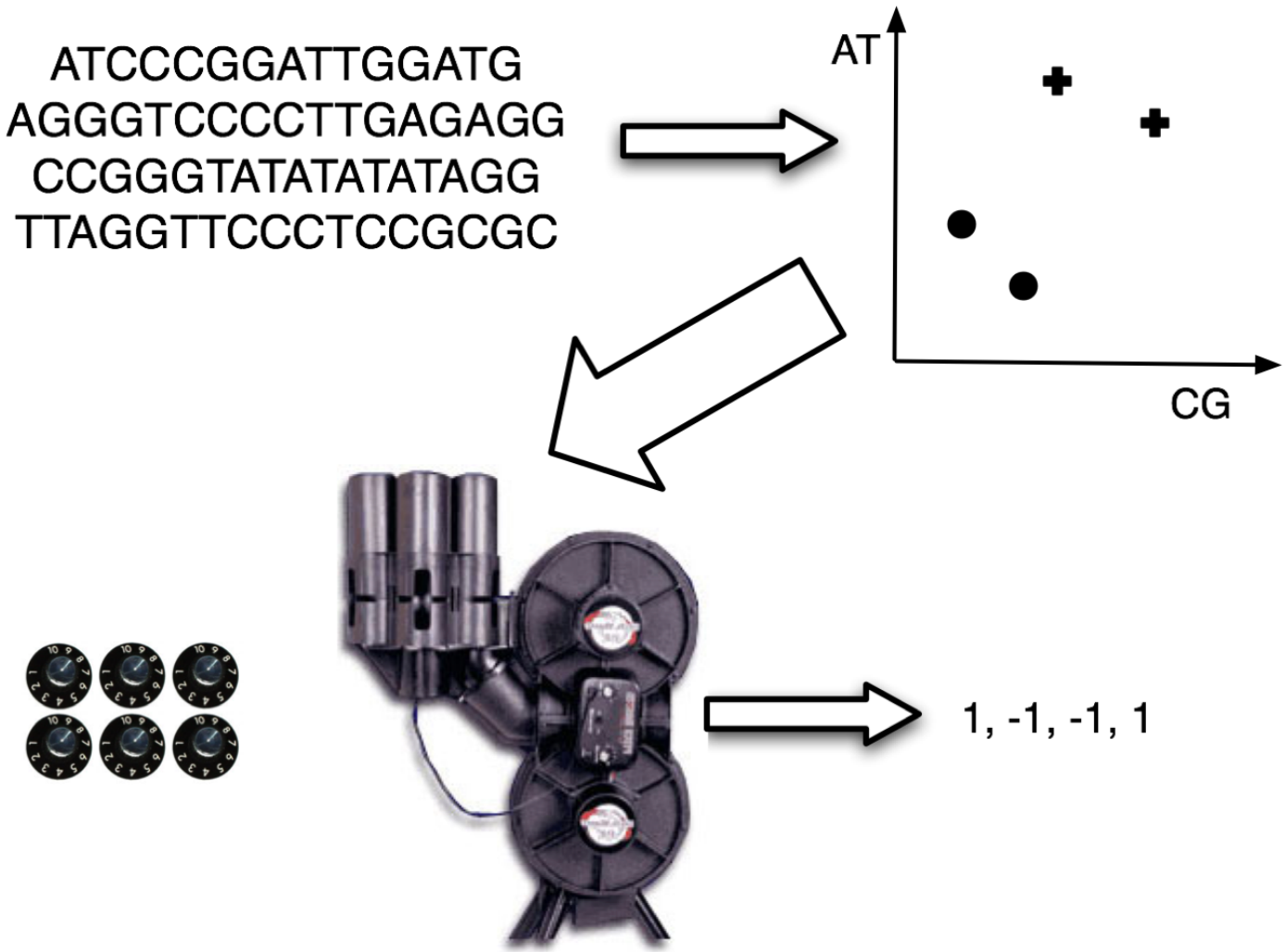
From Sequences to Features

- 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



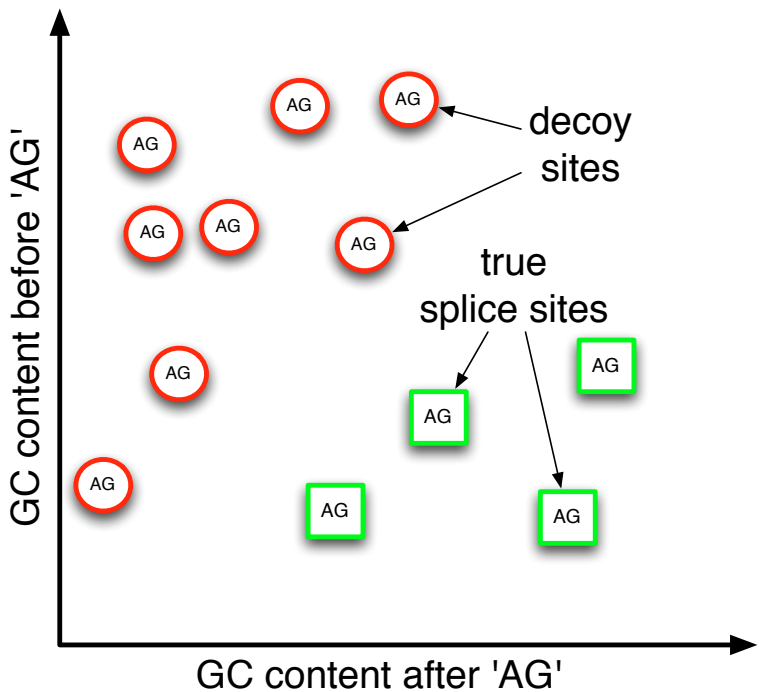
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

- Goal: Rule that distinguishes true from false ones

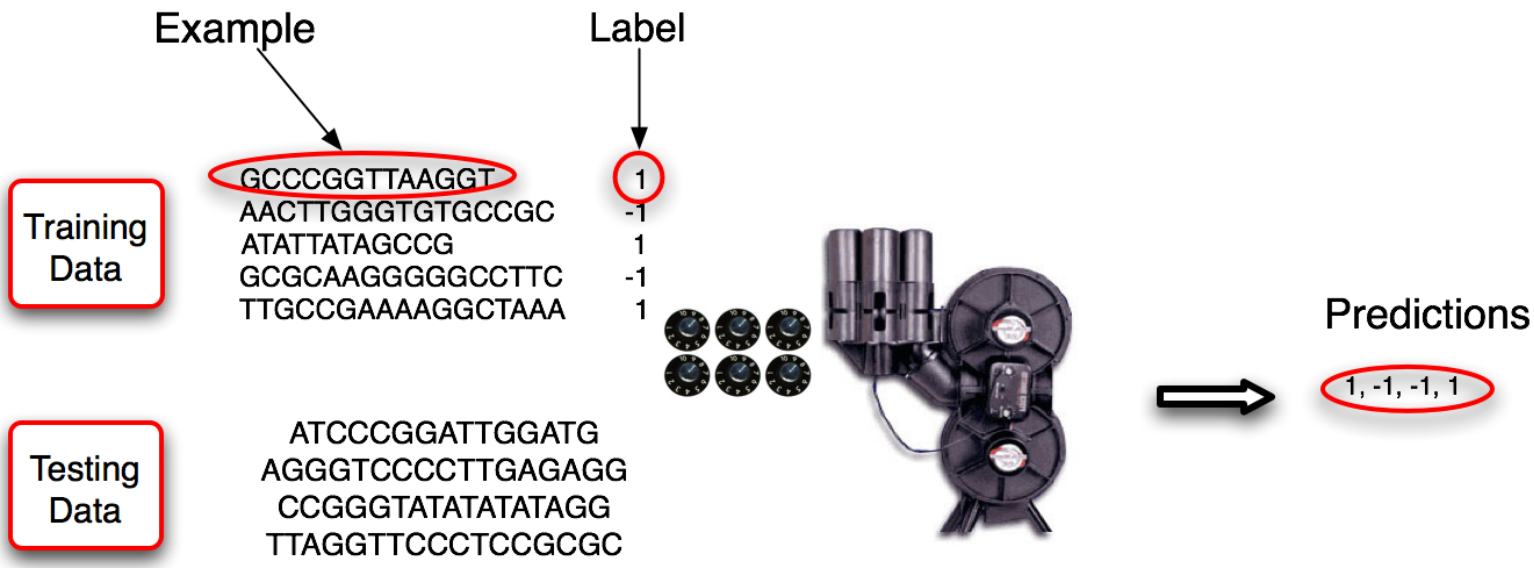


exploit that exons have higher GC content

or

that certain motifs are located nearby

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 predictions 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, input and labels, for each example. The aim is to learn about the pattern between input and labels. (The input is sometimes also called example.)

Unsupervised Learning

We do not have labels for the examples, but 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

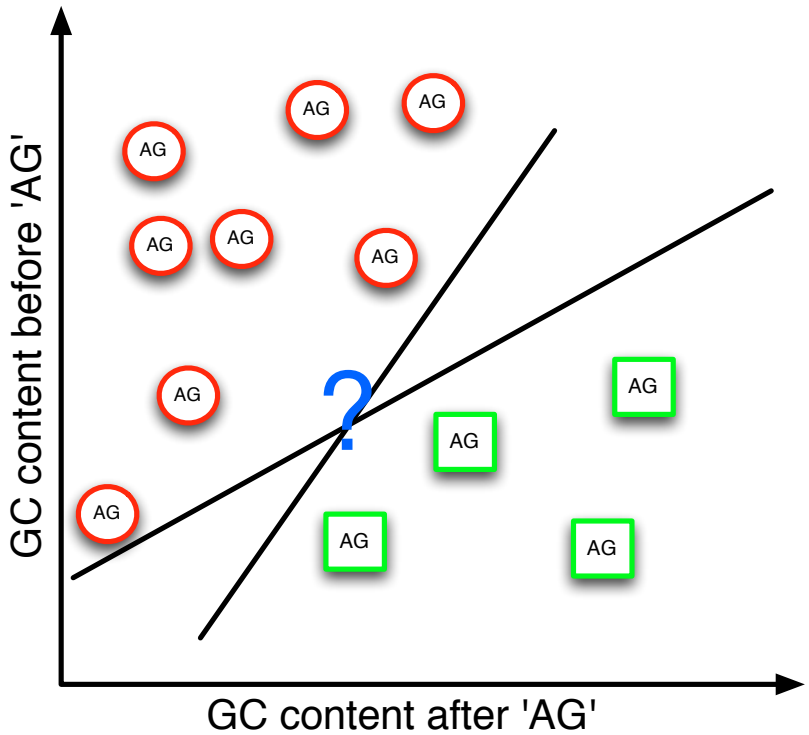
We define a loss function as

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

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 there is.

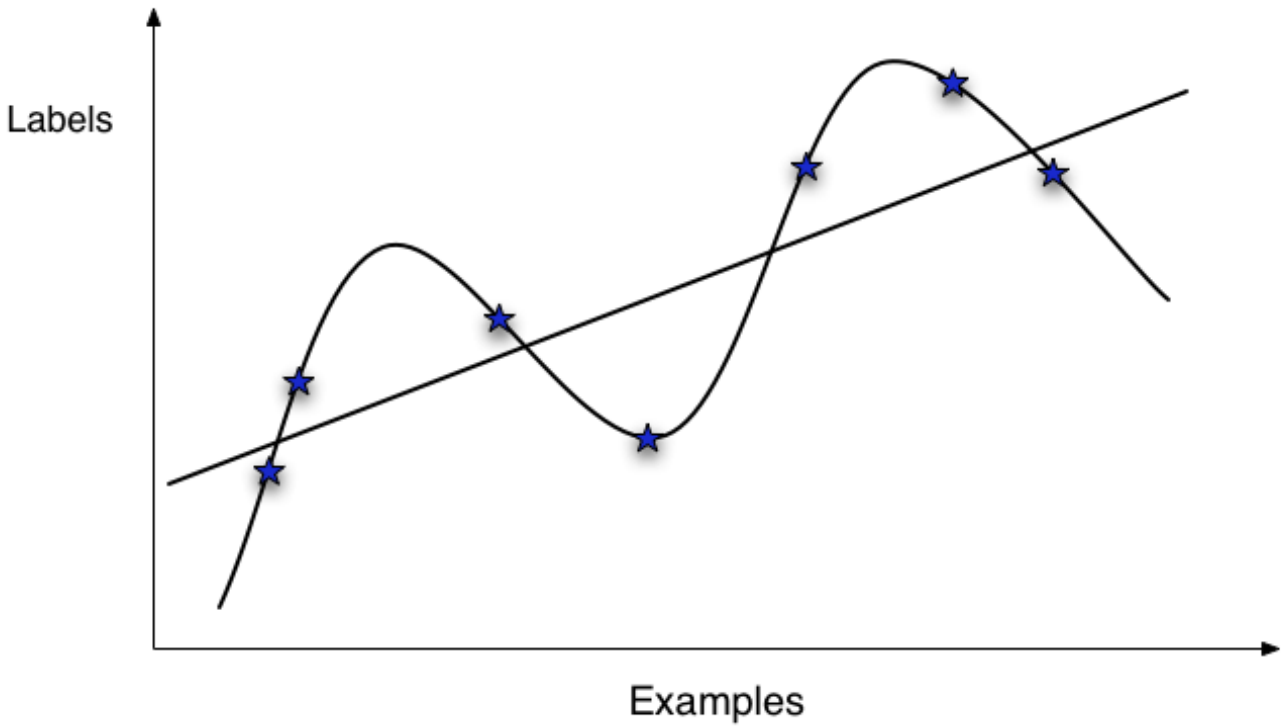
Classification



In binary classification ($\mathcal{Y} = \{-1, +1\}$), we one may use the 0/1-*loss function*:

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

Regression



In regression ($\mathcal{Y} = \mathbb{R}$), one often uses the *square loss function*:

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

Expected vs. Empirical Risk

Expected Risk

This is the average loss on *unseen examples*. We would like to have it as small as possible, but it is hard to compute.

Empirical Risk

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

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

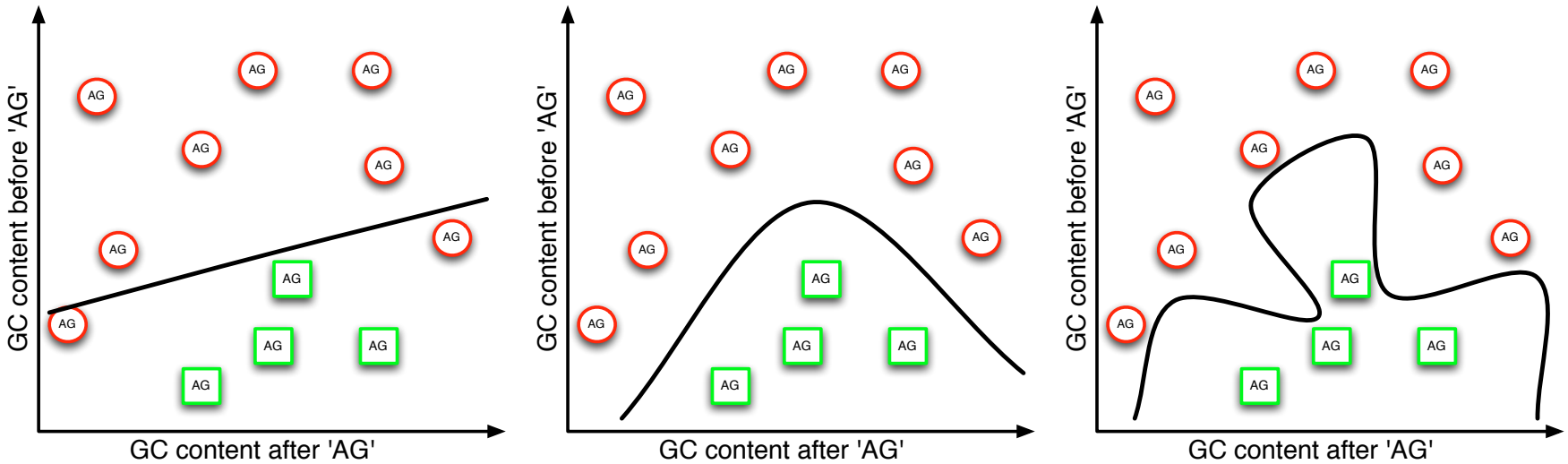
Basic Notion

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?



Occam's razor (a.k.a. Occam's Law of Parsimony):

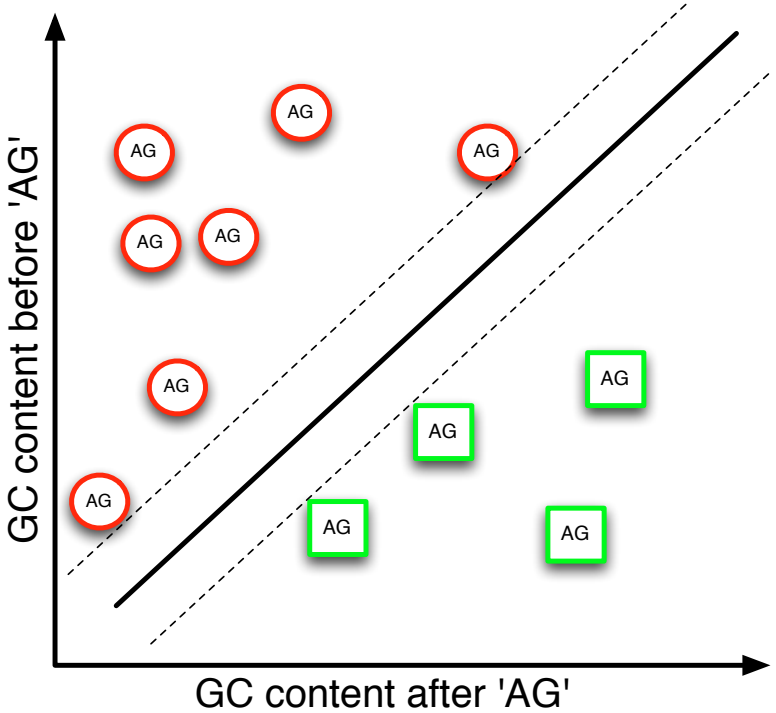
(William of Occam, 14th century)

"Entities should not be multiplied beyond necessity"

("Do not make the hypothesis more complex than necessary")

[http://www.franciscans.org]

Special Case: Complexity of Hyperplanes



What is the complexity of a hyperplane classifier?





Vladimir Vapnik and Alexey Chervonenkis: Vapnik-Chervonenkis (VC) dimension

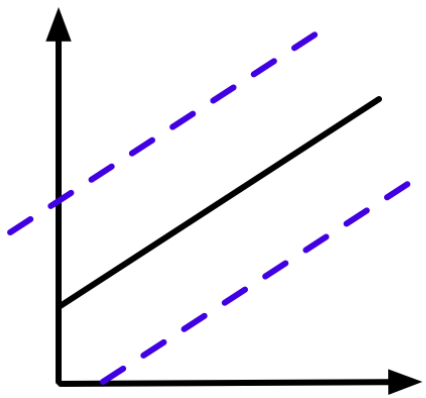
[Vapnik and Chervonenkis, 1971; Vapnik, 1995]

[http://tinyurl.com/cl8jo9,http://tinyurl.com/d7lmux]

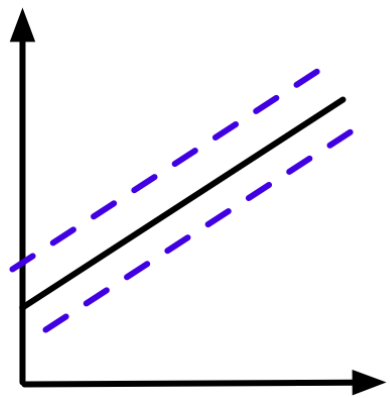
Larger Margin ⇒ Less Complex

Large Margin ⇒ Small VC dimension

Hyperplane classifiers with large margins have small VC dimension [Vapnik and Chervonenkis, 1971; Vapnik, 1995].



VC dim. small



VC dim. large

Maximum Margin ⇒ Minimum Complexity

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

Useful Idea:

Find the hyperplane that classifies all points correctly, while maximizing the margin (=SVMs).

Summary of Empirical Inference

Learn function $f : \mathcal{X} \to \mathcal{Y}$ given N labeled examples $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$.

Three important ingredients:

- Model $f_{\boldsymbol{\theta}}$ parametrized with some parameters $\boldsymbol{\theta} \in \Theta$
- Loss function $\ell(f(\mathbf{x}), y)$ measuring the "deviation" between predictions $f(\mathbf{x})$ and the label y
- Complexity term $P[f]$ defining model classes with limited complexity (via nested subsets $\{f \mid P[f] \leq p\} \subseteq \{f \mid P[f] \leq p'\}$ for $p \leq p'$)

Most algorithms find $\boldsymbol{\theta}$ in $f_{\boldsymbol{\theta}}$ by minimizing:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta} \in \Theta} \Bigg(\underbrace{\sum_{i=1}^{N} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)}_{\text{Empricial error}} + \overbrace{C}^{\text{Regularization parameter}} \underbrace{P[f_{\boldsymbol{\theta}}]}_{\text{Complexity term}} \Bigg) \quad \text{for given } C$$

Special case ($C \to 0$): Empirical error $\to 0$ and Complexity term $\to$ min

Measuring Performance in Practice

What to do in practice

Split the data into training and validation sets; use error on validation set as estimate of the expected error

A. Cross-validation

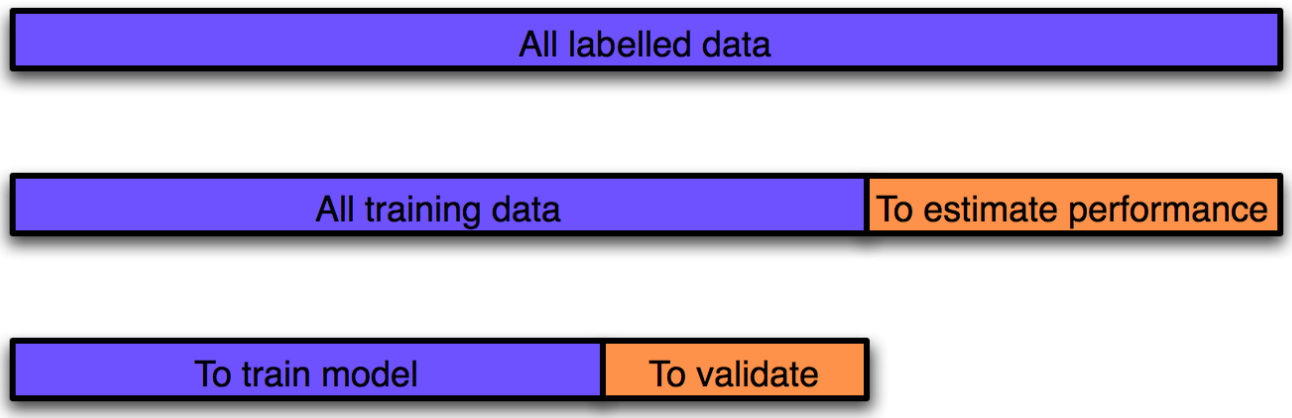
Split data into c disjoint parts; use each subset as validation set and rest as training set

B. Random splits

Randomly split data set into two parts, for example, 80% of data for training and 20% for validation; Repeat this many times

See, for instance, Duda et al. [2001] for more details.

Model Selection



Do not train on the "test set"!

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

Model selection = Find best parameters

- Regularization parameter C.
- Other parameters (introduced later)

Examples of Inference Problems

Binary classification

Separation into two classes: $\mathcal{Y} = \{-1, +1\}$,
$\mathcal{X}$ arbitrary (for instance $\mathbb{R}^d$, i.e. vectors; Σ^*, i.e. sequences of arbitrary length; etc.)

Multi-class classification

Separation into K classes: $\mathcal{Y} = \{1, \ldots, K\}$,
$\mathcal{X}$ arbitrary. (Typical approach: $f(\mathbf{x}) = \operatorname{argmax}_{k=1,\ldots,K} f^{(k)}(\mathbf{x})$.)

Regression

Prediction of a real value: $\mathcal{Y} = \mathbb{R}$, $\mathcal{X}$ arbitrary.

Label sequence learning

Prediction of a sequence of "classes" from a sequence of inputs, e.g. input is string of Σ-letters of length s, output is string of Σ'-letters.

Part II

Support Vector Machines and Kernels

Overview: Support Vector Machines and Kernels

Margin Maximization

- Support Vector Machines for Binary Classification
- Convex Optimization

Kernels & the "Trick"

- Inflating the Feature Space
- Kernel "Trick"
- Common Kernels
- Results for Running Example

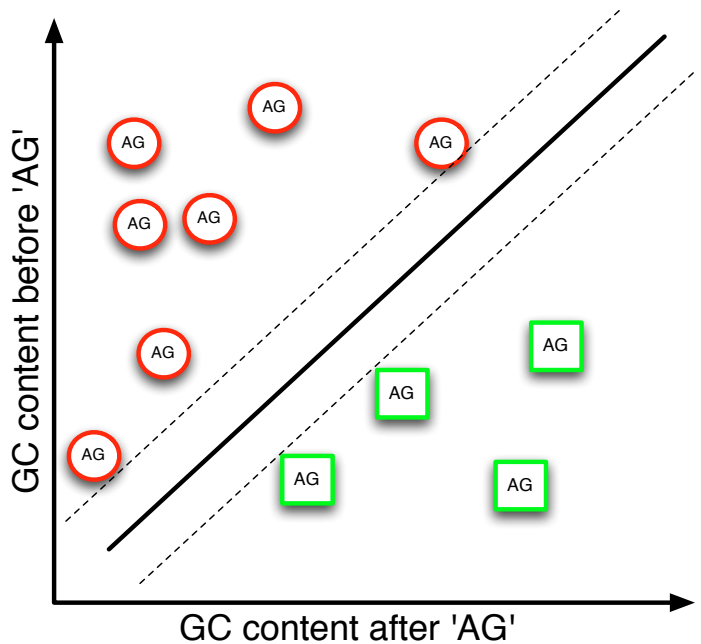
Beyond 2-Class Classification

- Multiple Kernel Learning
- Multi-Class Classification
- Support Vector Regression
- Semi-Supervised Learning & Transfer Learning

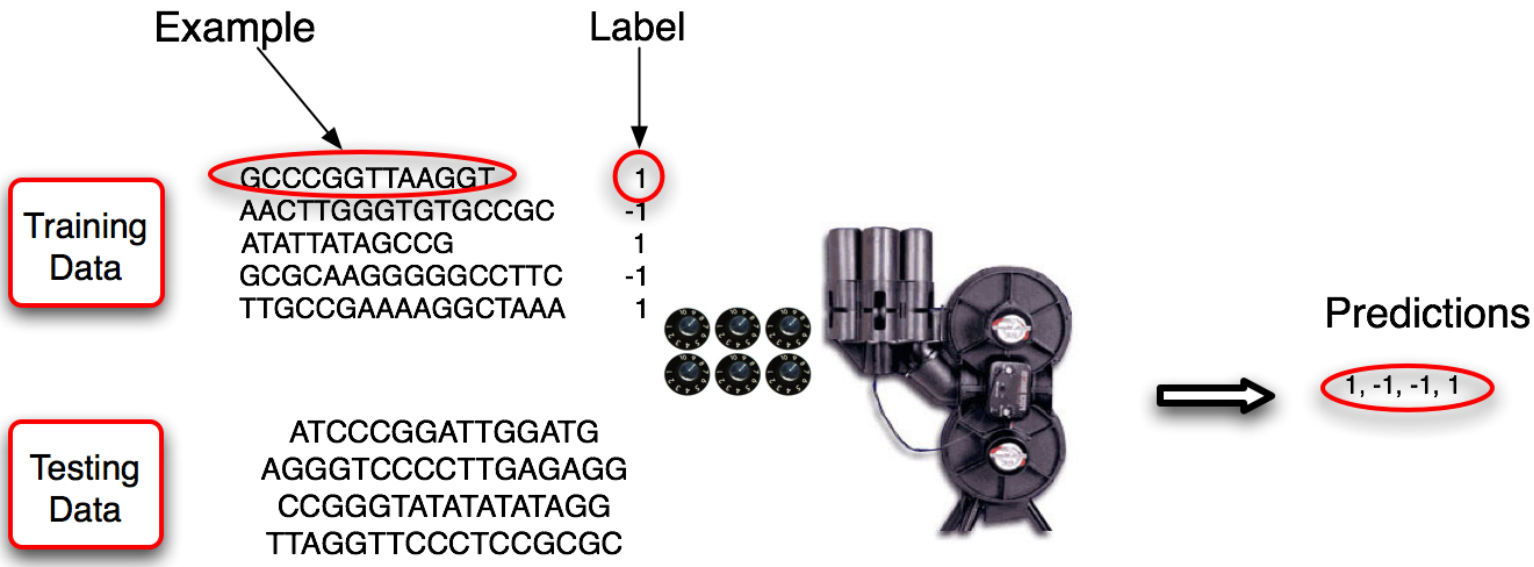
Software & Demonstration

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.

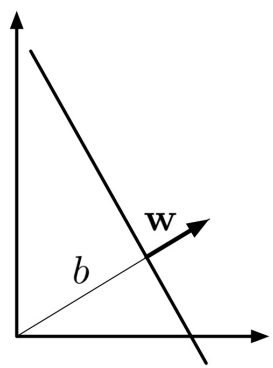


How to Maximize the Margin? I

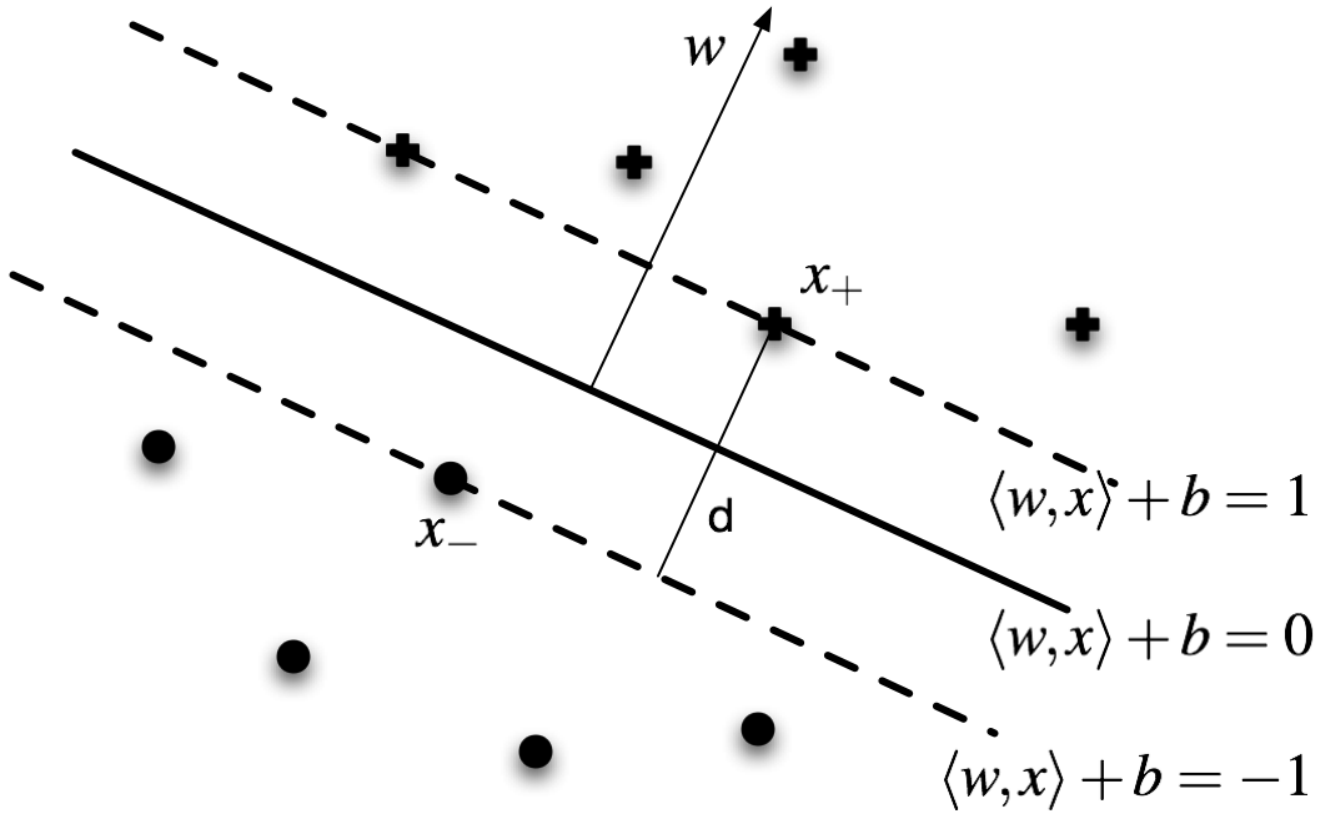


Consider linear hyperplanes with parameters $\mathbf{w}$, b:

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



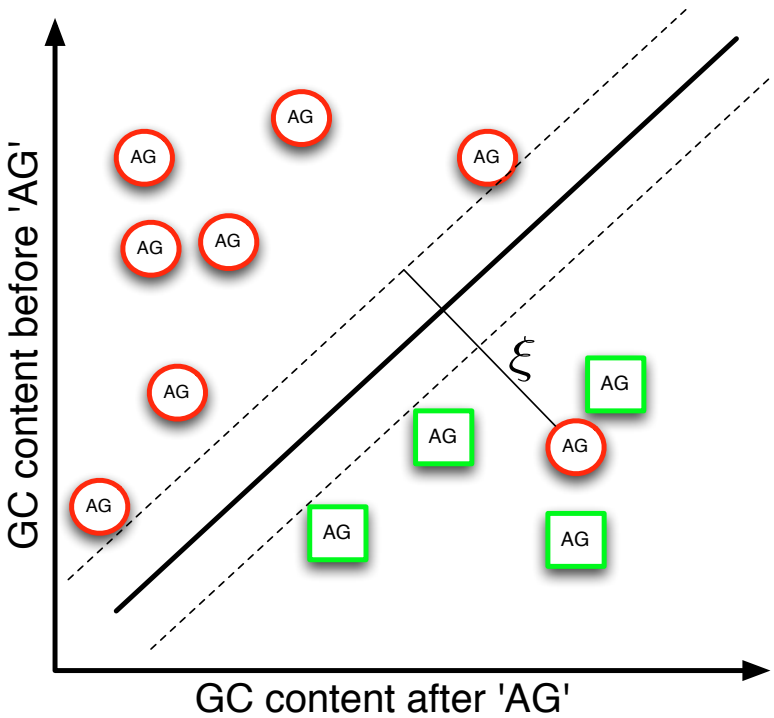
How to Maximize the Margin? II



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

[Schölkopf and Smola, 2002]

How to Maximize the Margin? III



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 $i = 1, \ldots, n$.

- Examples on the margin are called support vectors [Vapnik, 1995]
- Soft margin SVMs [Cortes and Vapnik, 1995]

We have to solve an "Optimization Problem"

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

Quadratic objective function, linear constraints in $\mathbf{w}$, b, and $\boldsymbol{\xi}$:

- "Quadratic Optimization Problem" (QP)
- "Convex Optimization Problem" (efficient solution possible, every local minimum is a global minimum)

How to solve it?

- General purpose optimization packages (GNU Linear Programming Kit, CPLEX, Mosek, ...)
- Much faster specialized solvers (liblinear, SVM OCAS, Nieme, SGD, ...)

An Important Detail

$$\underset{\boldsymbol{\alpha},b,\boldsymbol{\xi}}{\text{minimize}} \quad \frac{1}{2}\left\|\sum_{i=1}^{N}\alpha_i\mathbf{x}_i\right\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to} \quad y_i\left(\sum_{j=1}^{N}\alpha_j\langle\mathbf{x}_j,\mathbf{x}_i\rangle + b\right) \geqslant 1-\xi_i \text{ for all } i=1,\ldots,n.$$

$$\xi_i \geqslant 0 \text{ for all } i=1,\ldots,n$$

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

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i\mathbf{x}_i \qquad \Rightarrow \text{Plug in!}$$

Now optimize for the variables $\boldsymbol{\alpha}$, b, and $\boldsymbol{\xi}$!

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

$$\langle\mathbf{x},\hat{\mathbf{x}}\rangle = \sum_{d=1}^{D}x_d\hat{x}_d \qquad \text{Remember this!}$$

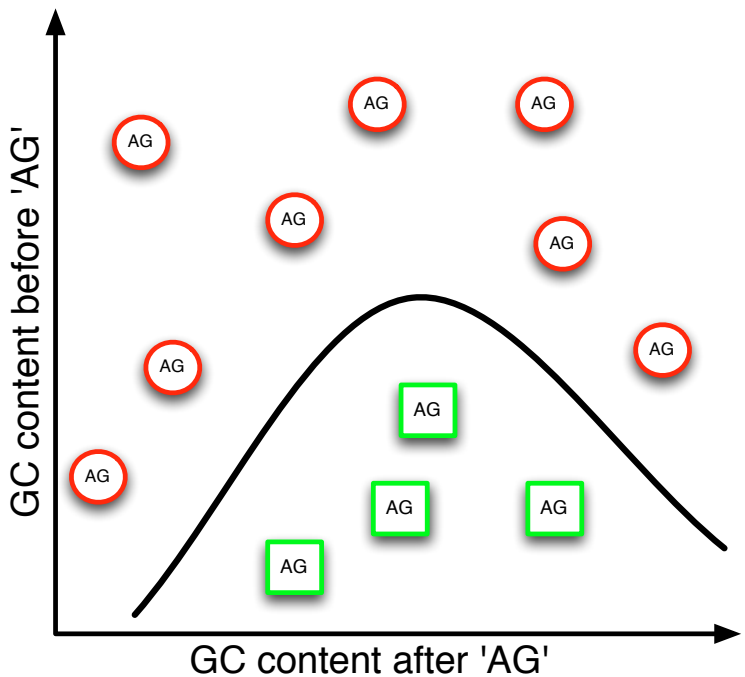
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

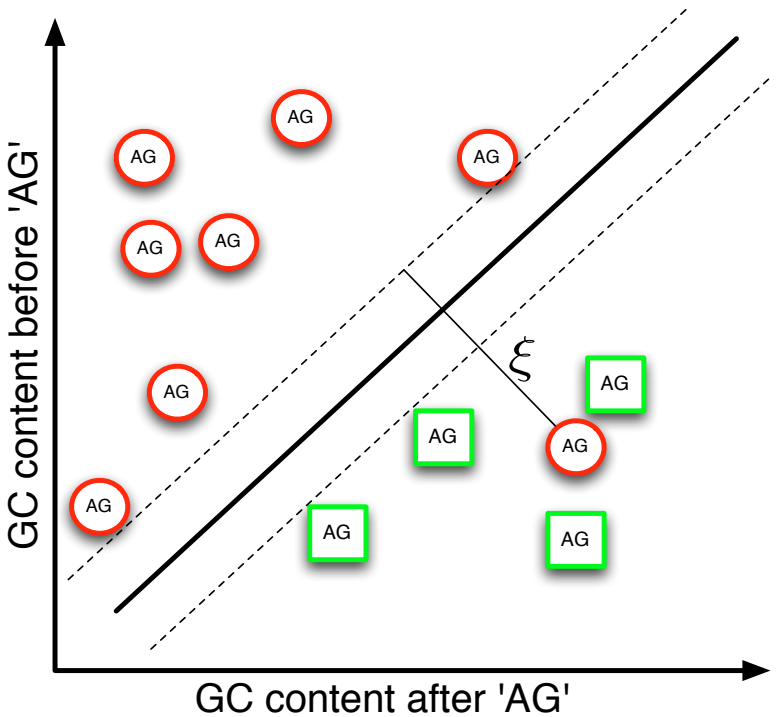
- Goal: Rule that distinguishes true from false ones



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 $i = 1, \ldots, n$.

- Examples on the margin are called **support vectors** [Vapnik, 1995]
- Soft margin SVMs [Cortes and Vapnik, 1995]

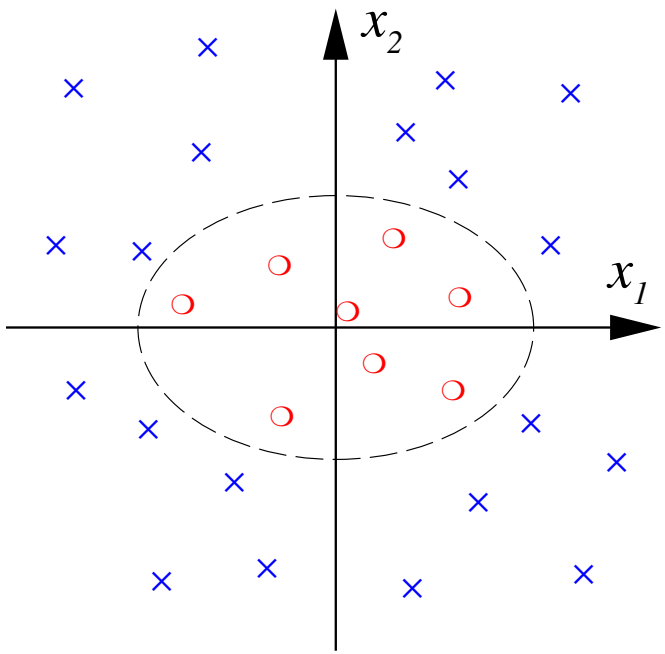
Nonlinear Separations

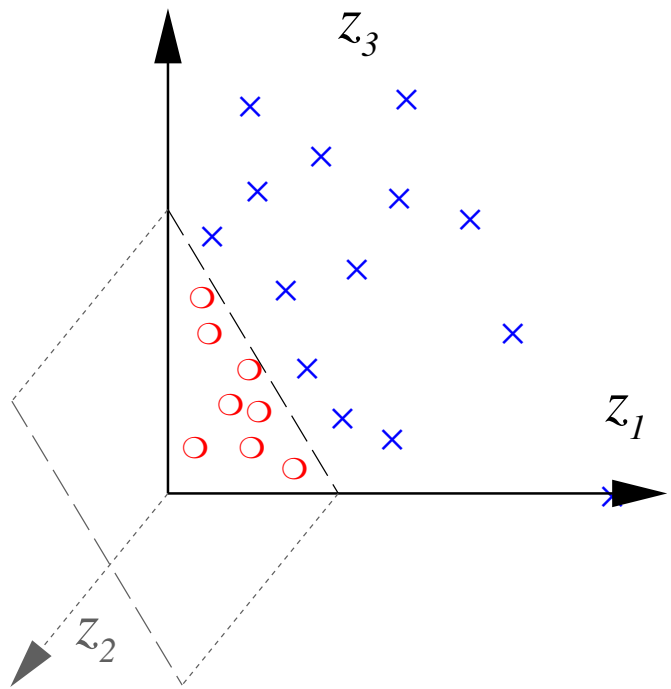
Linear separation might not be sufficient!

$\Rightarrow$ Map into a higher dimensional feature space

Example: all second order monomials

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





Kernel "Trick"

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

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

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

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

$$\text{scalar product} = 2 - \text{distance}$$

- 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's condition).

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's 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

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

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

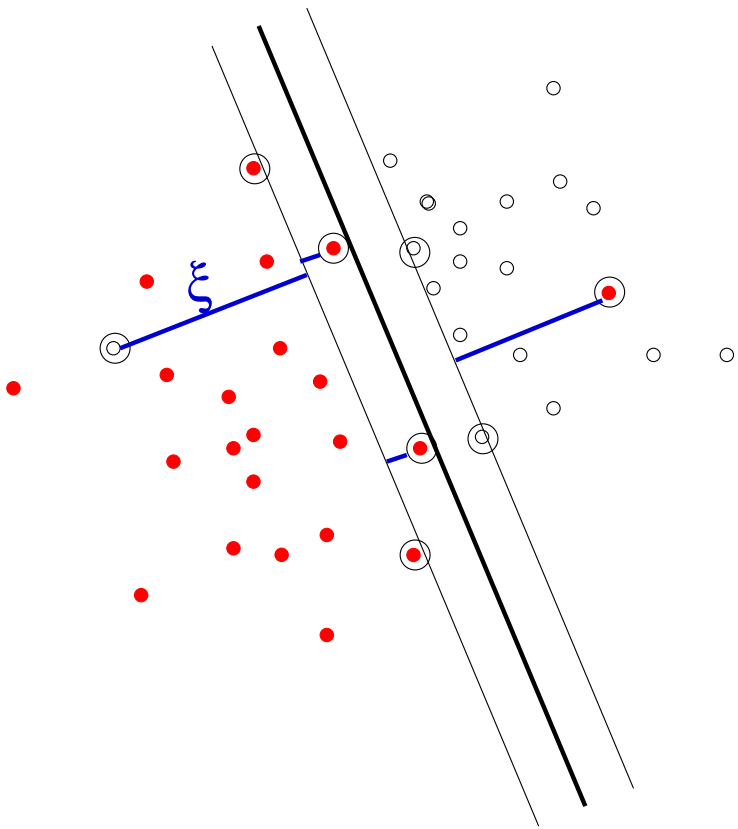
Notes:

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

Toy Examples

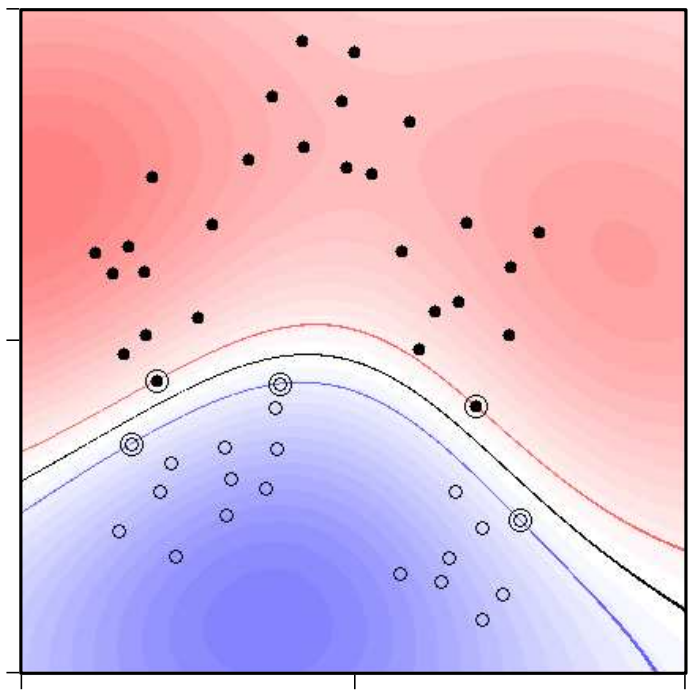
Linear kernel

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



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

Evaluation Measures for Classification

The Contingency Table/Confusion Matrix

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

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

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

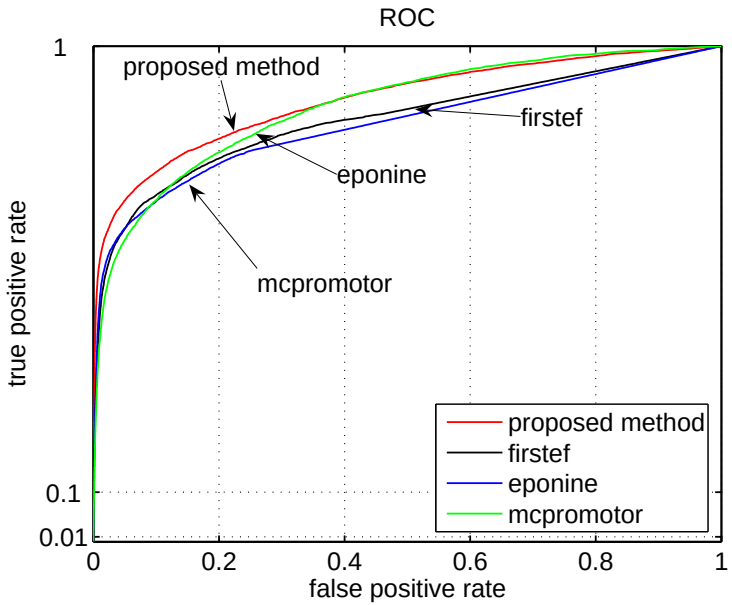
Evaluation Measures for Classification II

Several commonly used performance measures

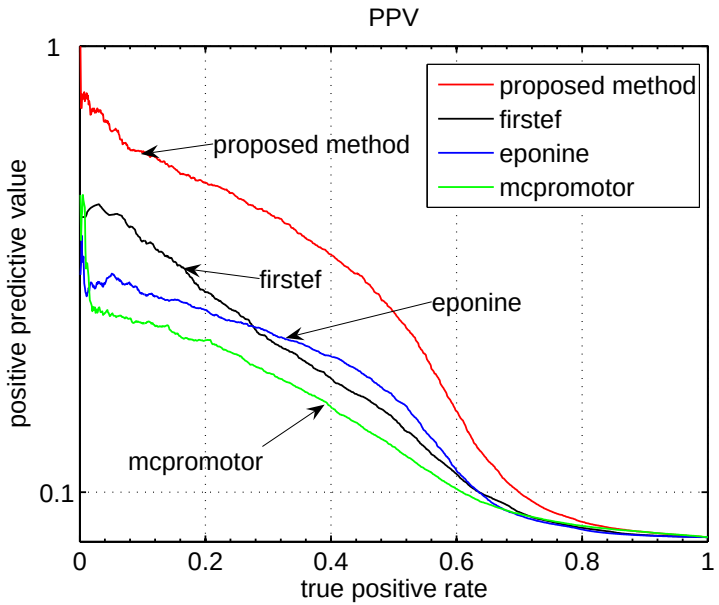
Name	Computation
Accuracy	$\mathrm{ACC} = \frac{TP+TN}{N}$
Error rate (1-accuracy)	$\mathrm{ERR} = \frac{FP+FN}{N}$
Balanced error rate	$\mathrm{BER} = \frac{1}{2}\left(\frac{FN}{FN+TP} + \frac{FP}{FP+TN}\right)$
Weighted relative accuracy	$\mathrm{WRACC} = \frac{TP}{TP+FN} - \frac{FP}{FP+TN}$
F1 score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

Evaluation Measures for Classification III

[left] Receiver Operating Characteristic (ROC) Curve



[right] Precision Recall Curve



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

Use bias independent scalar evaluation measure

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

GC-Content-based Splice Site Recognition

Kernel	auROC
Linear	88.2%
Polynomial $d = 3$	91.4%
Polynomial $d = 7$	90.4%
Gaussian $\sigma = 100$	87.9%
Gaussian $\sigma = 1$	88.6%
Gaussian $\sigma = 0.01$	77.3%

SVM accuracy of acceptor site recognition using polynomial and Gaussian kernels with different degrees d and widths σ. Accuracy is measured using the area under the ROC curve (auROC) and is computed using five-fold cross-validation

Multiple Kernel Learning (MKL)

Data may consist of sequence and structure information

Possible solution: We can add the two kernels,

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

Better solution: We can mix the two kernels,

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

where $t$is estimated from the training data

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

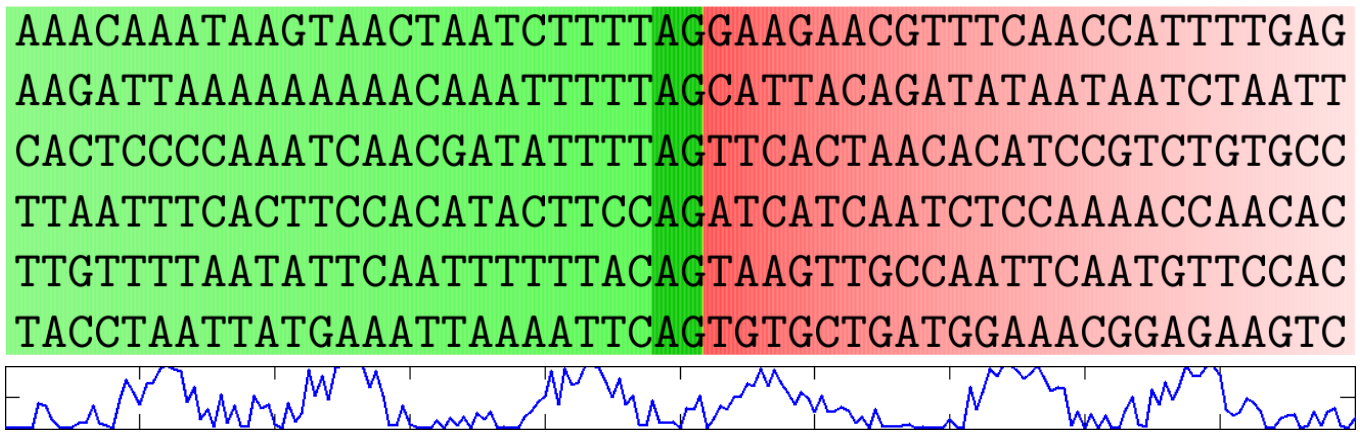
$$k(\mathbf{x}, \mathbf{x}') = \sum_{p=1}^{K} \beta_p k_p(\mathbf{x}, \mathbf{x}').$$

Applications

- Heterogeneous data
- Improving interpretability (more on this later)

Example: Combining Heterogeneous Data

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



$$k(\mathbf{x}, \mathbf{x}') = \beta_1 \, k_{dna}(\mathbf{x}_{dna}, \mathbf{x}'_{dna}) + \beta_2 \, k_{nrg}(\mathbf{x}_{nrg}, \mathbf{x}'_{nrg}) + \beta_3 \, k_{3d}(\mathbf{x}_{3d}, \mathbf{x}'_{3d}) + \cdots$$

MKL Primal Formulation

$$\begin{aligned}
\text{min} \quad & \frac{1}{2}\left(\sum_{j=1}^{M}\beta_j \left\|\mathbf{w}_j\right\|_2\right)^2 + C\sum_{i=1}^{N}\xi_n \\
\text{w.r.t.} \quad & \mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_M),\ \mathbf{w}_j \in \mathbb{R}^{D_j}, \quad \forall j = 1 \ldots M \\
& \boldsymbol{\beta} \in \mathbb{R}_+^M,\ \boldsymbol{\xi} \in \mathbb{R}_+^N,\ b \in \mathbb{R} \\
\text{s.t.} \quad & y_i\left(\sum_{j=1}^{M}\beta_j \mathbf{w}_j{}^{\mathsf{T}}\Phi_j(\mathbf{x}_i) + b\right) \geq 1 - \xi_i,\ \forall i = 1, \ldots, N \\
& \sum_{j=1}^{M}\beta_j = 1
\end{aligned}$$

Properties: equivalent to SVM for $M = 1$; solution sparse in "blocks"; each block j corresponds to one kernel

Solving MKL

- SDP Lanckriet et al. [2004], QCQP Bach et al. [2004]
- SILP Sonnenburg et al. [2006a]
- SimpleMKL Rakotomamonjy et al. [2008]
- Extended Level Set Method Xu et al. [2009]

SILP implemented in shogun-toolbox; examples available.

Multi-Class Classification

Real problems often have more than 2 classes

Generalize the SVM to multi-class classification, for $K > 2$.

Three approaches [Schölkopf and Smola, 2002]

One-vs-rest

For each class, label all other classes as "negative" (K binary problems).
$\Rightarrow$ Simple and hard to beat!

One-vs-one

Compare all classes pairwise ($\frac{1}{2}K(K-1)$ binary problems).

Multi-class loss

Define a new empirical risk term.

Multi-Class Loss for SVMs

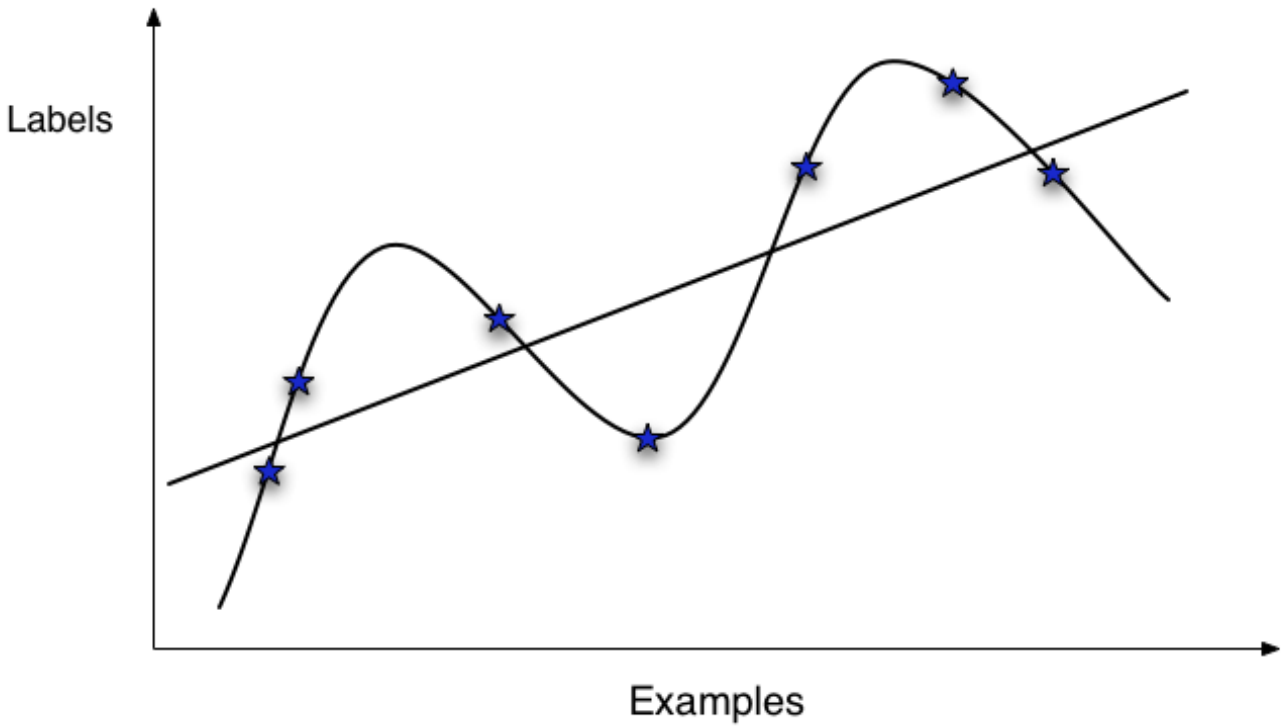
Two-Class SVM

$$\underset{\mathbf{w},b}{\text{minimize}}\ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \ell(f_{\mathbf{w},b}(\mathbf{x}), y_i),$$

Multi-Class SVM

$$\underset{\mathbf{w},b}{\text{minimize}}\quad \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \max_{u \neq y_i} \ell\left(f_{\mathbf{w},b}(\mathbf{x}_i, y_i) - f_{\mathbf{w},b}(\mathbf{x}_i, u), y_i\right)$$

Regression



Examples $x \in \mathcal{X}$

Labels $y \in \mathbb{R}$

Regression

Squared loss Simplest approach

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

Problem: All α's are non-zero $\Rightarrow$ Inefficient!

ε-insensitive loss function

Extend "margin" to regression. Establish a "tube" around the line where we can make mistakes.

$$\ell(f(\mathbf{x}_i), y_i) = \begin{cases} 0 & |f(\mathbf{x}_i) - y_i| < \varepsilon \\ |f(\mathbf{x}_i) - y_i| - \varepsilon & \text{otherwise} \end{cases}$$

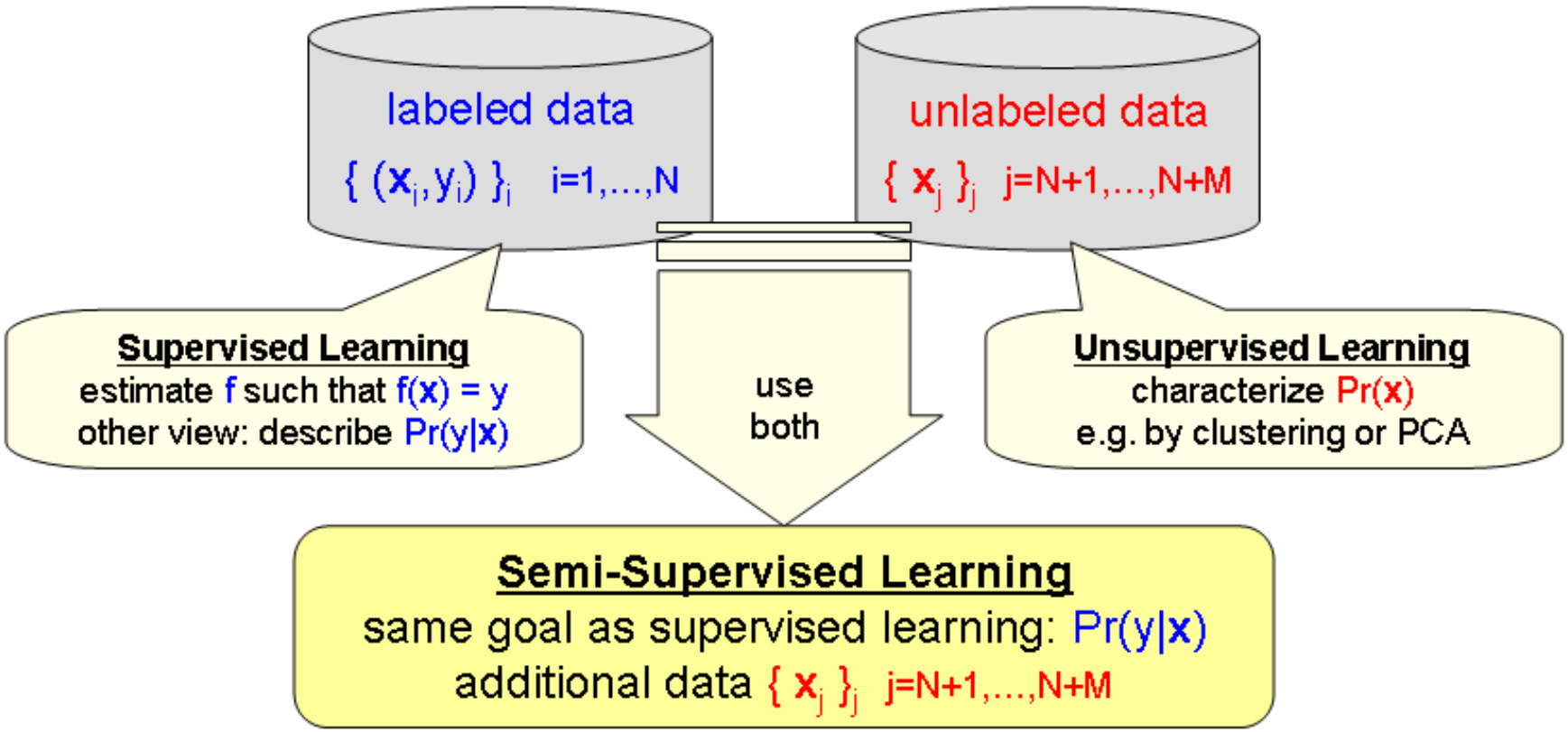
Idea: Examples $(\mathbf{x}_i, y_i)$ inside tube have $\alpha_i = 0$.

Huber's loss Combination of benefits

$$\ell(f(\mathbf{x}_i), y_i) := \begin{cases} \frac{1}{2}(y_i - f(\mathbf{x}_i))^2 & |y_i - f(\mathbf{x}_i)| < \gamma \\ \gamma|y_i - f(\mathbf{x}_i)| - \frac{1}{2}\gamma^2 & (y_i - f(\mathbf{x}_i)) \geqslant \gamma \end{cases}$$

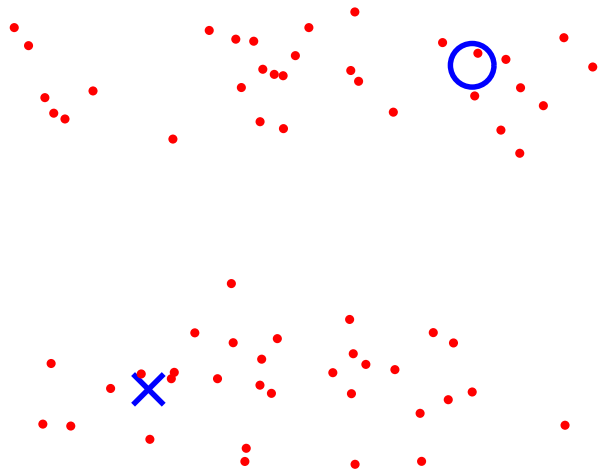
See e.g. Smola and Schölkopf [2001] for other loss functions and more details.

Semi-Supervised Learning: What Is It?



For most researchers: SSL = semi-supervised *classification*.

Semi-Supervised Learning: How Does It Work?



Cluster Assumption

Points in the **same cluster** are likely to be of the **same class**.

Equivalent assumption:

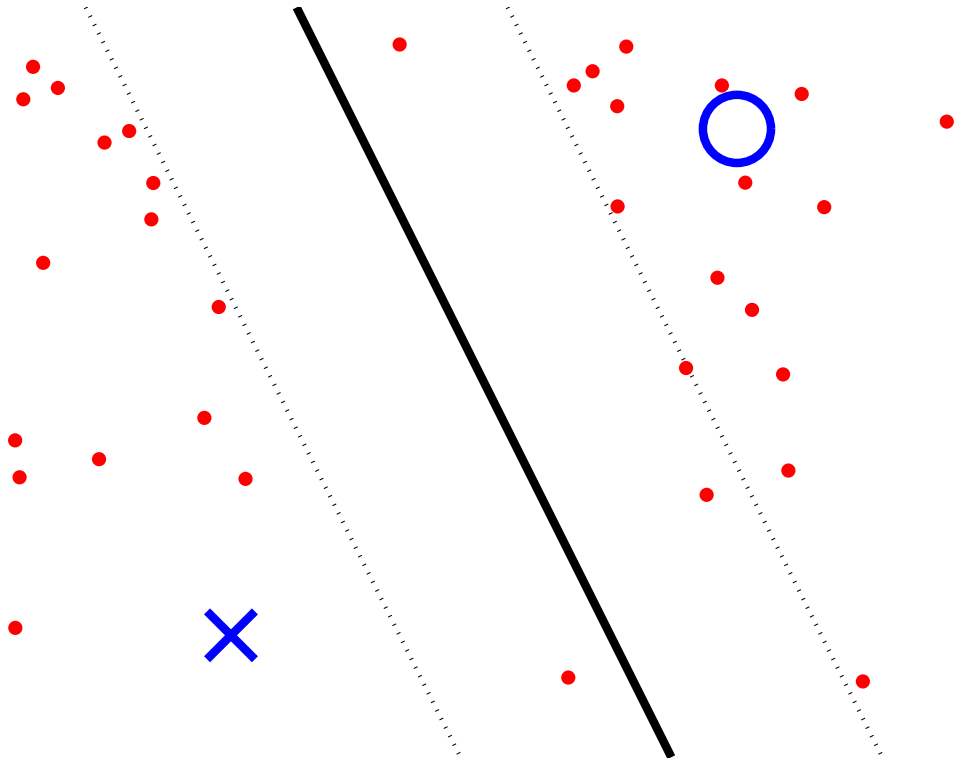
Low Density Separation Assumption

The decision boundary lies in a low density region.

$\Rightarrow$ Algorithmic idea: **Low Density Separation**

Semi-Supervised SVM

Soft margin
S^3VM

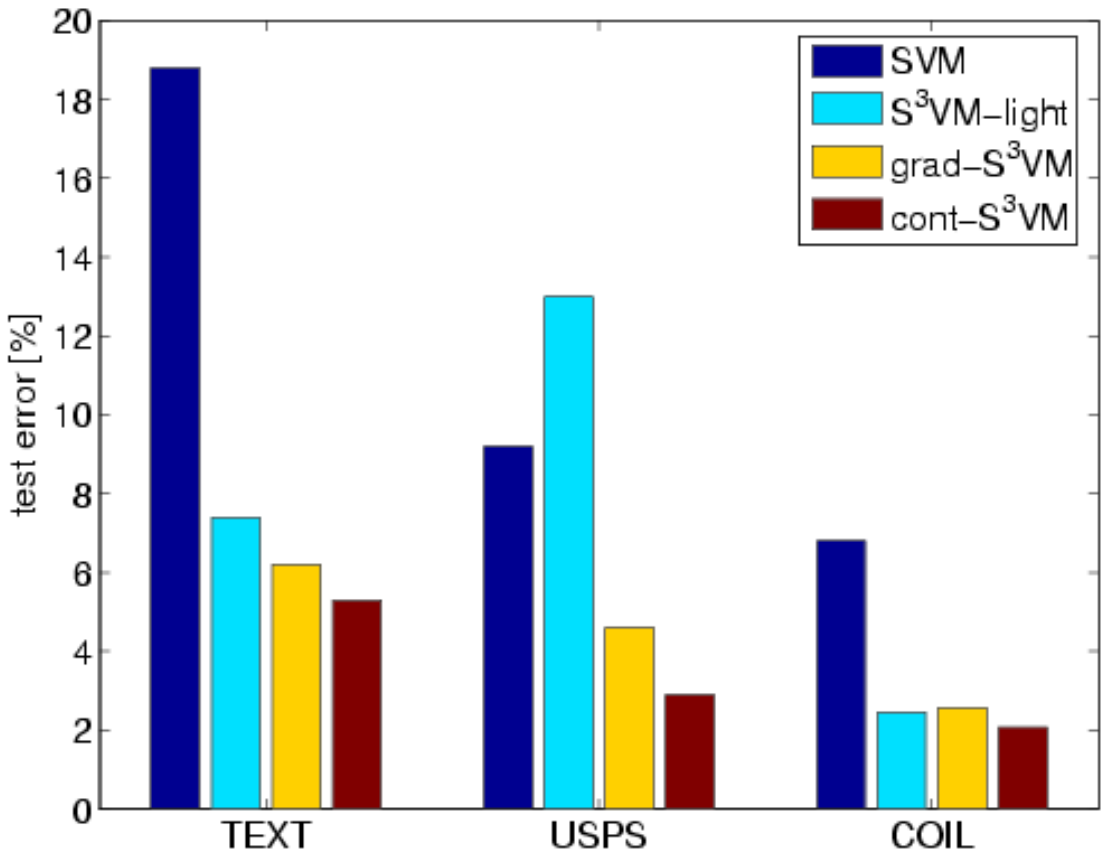


$$\min_{\mathbf{w},b,(y_j),(\xi_k)} \quad \begin{array}{l} \frac{1}{2}\mathbf{w}^\top\mathbf{w} \\ +C\sum_i \xi_i \\ +C^*\sum_j \xi_j \end{array} \quad s.t. \quad \begin{array}{c} \xi_i \geq 0 \quad \xi_j \geq 0 \\ y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1-\xi_i \\ y_j(\mathbf{w}^\top\mathbf{x}_j + b) \geq 1-\xi_j \end{array}$$

Semi-Supervised SVM: Optimization

$\Rightarrow$ **Optimization matters**

Comparison of S^3VM Optimization Methods



- Averaged over splits (and pairs of classes)
- Fixed hyperparams (close to hard margin)
- Similar results for other hyper-parameter settings

[Chapelle et al., 2006]

Covariate Shift & Domain Adaptation

The Idea of Domain Adaptation:

- Insufficient labeled training data for some problems
- Idea: Turn to related domains for which more data is available
- So-called *Source* and *Target* Domains can be different, but should be related enough to gain something

Distributional point of view:

- Supervised Learning: Example-label pairs drawn from $P(X, Y)$
- $P_{Source}(X, Y)$ might differ from $P_{Target}(X, Y)$

- Factorization: $P(X, Y) = P(Y|X) \cdot P(X)$
  - Covariate Shift: $P_{Source}(X) \neq P_{Target}(X)$
  - Differing Conditionals: $P_{Source}(Y|X) \neq P_{Target}(Y|X)$

$\rightarrow$ **There are numerous ways to approach this problem!**

[Ben-david et al., 2007; Evgeniou and Pontil, 2004; Schweikert et al., 2008]

Easy-to-use Software

Easysvm	an easy-to-use SVM toolbox based on Python and the Shogun toolbox, usable from command line or within Python
PyML	an easy-to-use Python-based SVM toolbox, usable from command line or within Python
Shogun toolbox	a powerful toolbox for large-scale data analysis, including many SVM implementations with support for Python, R, Matlab, and Octave
LibSVM	an SVM library with a graphic interface
SVM-Light	an efficient implementation of SVMs in C, usable from command line
Galaxy Web Service	a web service for using SVMs, using predefined kernels for real-valued data and string classification (based on Easysvm): `http://galaxy.fml.tuebingen.mpg.de`

Illustration Using Galaxy Web Service

Task 1: Learn to classify acceptor splice sites with GC features

1. Train classifier and predict using 5-fold cross-validation (`SVM Toolbox` → `Train and Test SVM`)
2. Evaluate classifier (`SVM Toolbox` → `Evaluate Predictions`)

Steps:

1. Use "`Upload file`" with URL `http://svmcompbio.tuebingen.mpg.de/data/C_elegans_acc_gc.arff`; set file format to `ARFF` and upload; file appears in history on right
2. Use "`Train and Test SVM`" on uploaded data set (choose `ARFF` data format) tool; set the kernel to `linear`, execute and look at the result
3. Use "`Evaluate Predictions`" on predictions and the labeled data (choose `ARFF` format), select `ROC Curve` and execute; check out the evaluation summary and the ROC curves

Illustration Using Galaxy Web Service

Task 2: Determine the best combination of polynomial degree $d = 1, \ldots, 5$ and SVMs $C = \{0.1, 1, 10\}$ using 5-fold cross-validation (`SVM Toolbox` → `SVM Model Selection`)

Steps:

1. Reuse the uploaded file from Task 1.
2. Use "`SVM Model Selection`" with uploaded data (choose `ARFF` format), set the number of cross-validation rounds to 5, set C's as $0.1, 1, 10$, select the polynomial kernel and choose the degrees as $1, 2, 3, 4, 5$. Execute and check the results.

Do-it-yourself with Easysvm (Preparations)

Install Shogun toolbox:

```
wget http://shogun-toolbox.org/archives/shogun/releases/0.7/sources/shogun-0.7.3.tar.bz2
tar xjf shogun-0.7.3.tar.bz2
cd shogun-0.7.3/src
./configure --interfaces=python_modular,libshogun,libshogunui --prefix=~/mylibs
make && make install && cd ../..

export PYTHONPATH=~/mylibs/lib/python2.5/site-packages/
export LD_LIBRARY_PATH=~/mylibs/lib
```

Install Easysvm and its dependencies and get data:

```
wget http://www.antlr.org/download/Python/antlr_python_runtime-3.0.1.tar.gz
tar xzf antlr_python_runtime-3.0.1.tar.gz
cd antlr_python* && python setup.py install --prefix=~/mylibs && cd ..

wget http://www.mit.edu/~sav/arff/dist/arff-1.0c.tar.gz
tar xzf arff-1.0c.tar.gz
cd arff-1.0c && python setup.py install --prefix=~/mylibs && cd ..

wget http://www.fml.tuebingen.mpg.de/raetsch/projects/easysvm/easysvm-0.3.tar.gz
tar xzf easysvm-0.3.tar.gz
cd easysvm-0.3 && python setup.py install --prefix=~/mylibs && cd ..

wget http://svmcompbio.tuebingen.mpg.de/data/C_elegans_acc_gc.arff
```

Do-it-yourself with Easysvm

Task 1: Learn to classify acceptor splice sites with GC features

1. Train classifier and predict using 5-fold cross-validation (cv)
2. Evaluate classifier (eval)

```
                                SVM C  kernel    data format and file    predictions
~/mylibs/bin/easysvm.py cv 5      1    linear  arff C_elegans_acc_gc.arff  lin_gc.out
  2 features, 2200 examples
  Using 5-fold crossvalidation
head -4 lin_gc.out
  #example output split
  0 -0.8740213 0
  1 -0.9755172 2
  2 -0.9060478 1
                                predictions    data format and file    output file
~/mylibs/bin/easysvm.py eval    lin_gc.out  arff C_elegans_acc_gc.arff  lin_gc.perf
tail -6 lin_gc.perf
  Averages
  Number of positive examples = 40
  Number of negative examples = 400
  Area under ROC curve = 91.3 %
  Area under PRC curve = 55.8 %
  Accuracy (at threshold 0) = 90.9 %
```

Do-it-yourself with Easysvm

Task 2: Determine the best combination of polynomial degree $d = 1, \ldots, 5$ and SVMs $C = \{0.1, 1, 10\}$ using 5-fold cross-validation (modelsel)

SVM C's: 0.1,1,10 — kernel & parameters: poly 1,2,3,4,5 true false — data format and file: arff C_elegans_acc_gc.arff — output file: poly_gc.modelsel

```
~/mylibs/bin/easysvm.py modelsel 5 0.1,1,10 poly 1,2,3,4,5 true false \
                                   arff C_elegans_acc_gc.arff poly_gc.modelsel
  2 features, 2200 examples
  Using 5-fold crossvalidation
  ...
head -8 poly_gc.modelsel
  Best model(s) according to ROC measure:
    C=10.0 degree=1
  Best model(s) according to PRC measure:
    C=1.0 degree=1
  Best model(s) according to accuracy measure:
    C=10.0 degree=1
  ...
```

Part III

Kernels for Sequences and Graphs

Overview: Kernels for Sequences and Graphs

- String Kernels
  - Example Sequence Classification
  - Position-Independent Kernels
  - Position-Dependent Kernels
  - Advanced Kernels
- Kernels on Graphs
  - Basics
  - Random Walks
  - Subtrees
- Kernels on Images
  - Basics for Classifying Images
  - Codebook & Spatial Kernels
- Extracting Insights from the Learned SVM Classifier
  - Why Are SVMs Hard to Interpret?
  - Understanding String Kernel based SVMs
  - Understanding SVMs Based on General Kernels

The String Kernel Recipe

General idea

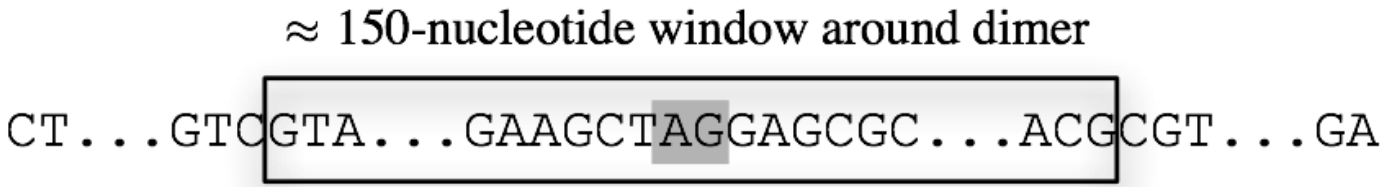
- Count substrings shared by two strings
- The greater the number of common substrings, the more two sequences are deemed similar

Variations

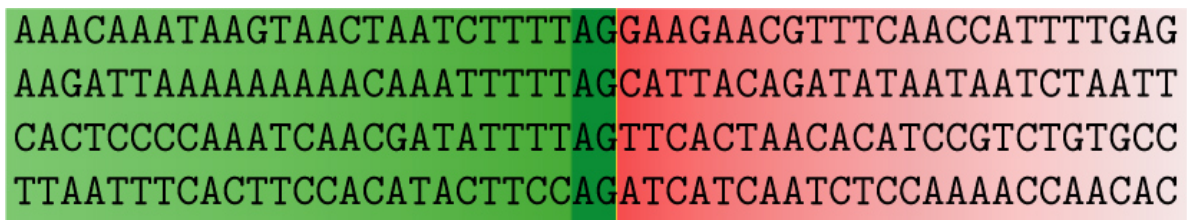
- Allow gaps
- Include wildcards
- Allow mismatches
- Include substitutions
- Motif kernels
- Assign weights to substrings

Recognizing Genomic Signals

Discriminate true signal positions from all other positions



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



Examples: Transcription start site finding, splice site prediction, alternative splicing prediction, trans-splicing, polyA signal detection, translation initiation site detection

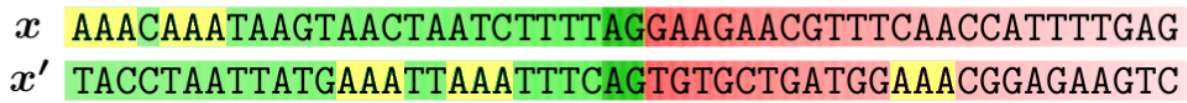
Types of Signal Detection Problems

Problem categorization

(based on positional variability of motifs)

Position-Independent

→ Motifs may occur anywhere,



for instance, tissue classification using promoter region

Types of Signal Detection Problems

Problem categorization

(based on positional variability of motifs)

Position-Dependent

→ Motifs very stiff, almost always at same position,



for instance, splice site identification

Types of Signal Detection Problems

Problem categorization

(based on positional variability of motifs)

Mixture of Position-Dependent/-Independent

$\longrightarrow$ variable but still positional information



for instance, promoter identification

Spectrum Kernel

To make use of position-independent motifs:

- **Idea:** like the bag-of-words-kernel (cf. text classification) but for biological sequences (words are now strings of length k, called k-mers)
  - Count k-mers in sequence A and sequence B.
  - Spectrum Kernel is sum of product of counts (for same k-mer)

Example $k = 3$:

x AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG

x' TACCTAATTATGAAATTAAATTTCAGTGTGCTGATGGAAACGGAGAAGTC

3-mer	AAA	AAC	...	CCA	CCC	...	TTT
# in **x**	2	4	...	1	0	...	3
# in **x**′	3	1	...	0	0	...	1

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

Spectrum Kernel with Mismatches

General idea [Leslie et al., 2003]

- Do not enforce strictly exact matches
- Define mismatch neighborhood of ℓ-mer s with up to m mismatches:

$$\phi^{\text{Mismatch}}_{(l,m)}(s) = (\phi_\beta(s))_{\beta \in \Sigma^\ell}$$

- For sequence x of any length, the map is then extended as:

$$\phi^{\text{Mismatch}}_{(l,m)}(\mathbf{x}) = \sum_{\ell\text{-mers } s \text{ in } \mathbf{x}} (\phi^{\text{Mismatch}}_{(l,m)}(s))$$

- The mismatch kernel is the inner product in feature space defined by:

$$k^{\text{Mismatch}}_{(l,m)}(\mathbf{x}, \mathbf{x}') = \left\langle \Phi^{\text{Mismatch}}_{(l,m)}(\mathbf{x}), \Phi^{\text{Mismatch}}_{(l,m)}(\mathbf{x}') \right\rangle$$

Spectrum Kernel with Gaps

General idea [Leslie and Kuang, 2004; Lodhi et al., 2002]

- Allows gaps in common substrings
  $\longrightarrow$ "subsequences"
- A g-mer then contributes to all its ℓ-mer subsequences:

$$\phi^{\text{Gap}}_{(g,\ell)}(s) = (\phi_\beta(s))_{\beta\in\Sigma^\ell}$$

- For sequence x of any length, the map is then extended as:

$$\phi^{\text{Gap}}_{(g,\ell)}(\mathbf{x}) = \sum_{g\text{-mers } s \text{ in } \mathbf{x}} (\phi^{\text{Gap}}_{(g,\ell)}(s))$$

- The gappy kernel is the inner product in feature space defined by:

$$k^{\text{Gap}}_{(g,\ell)}(\mathbf{x},\mathbf{x}') = \left\langle \Phi^{\text{Gap}}_{(g,\ell)}(\mathbf{x}), \Phi^{\text{Gap}}_{(g,\ell)}(\mathbf{x}') \right\rangle$$

Wildcard Kernels

General idea [Leslie and Kuang, 2004]

- Augment alphabet Σ by a wildcard character $*$: $\Sigma \cup \{*\}$
- Given s from Σ^ℓ and β from $(\Sigma \cup \{*\})^\ell$ with maximum m occurrences of $*$
- ℓ-mer s contributes to ℓ-mer β if their non-wildcard characters match
- For sequence $\mathbf{x}$ of any length, the map is then given by:

$$\phi^{\text{Wildcard}}_{(l,m,\lambda)}(\mathbf{x}) = \sum_{\ell-\text{mers } s \text{ in } \mathbf{x}} (\phi_\beta(s))_{\beta \in W}$$

where $\phi_\beta(s) = \lambda^j$ if s matches pattern β containing j wildcards, and $\phi_\beta(s) = 0$ if s does not match β, and $0 \leq \lambda \leq 1$.

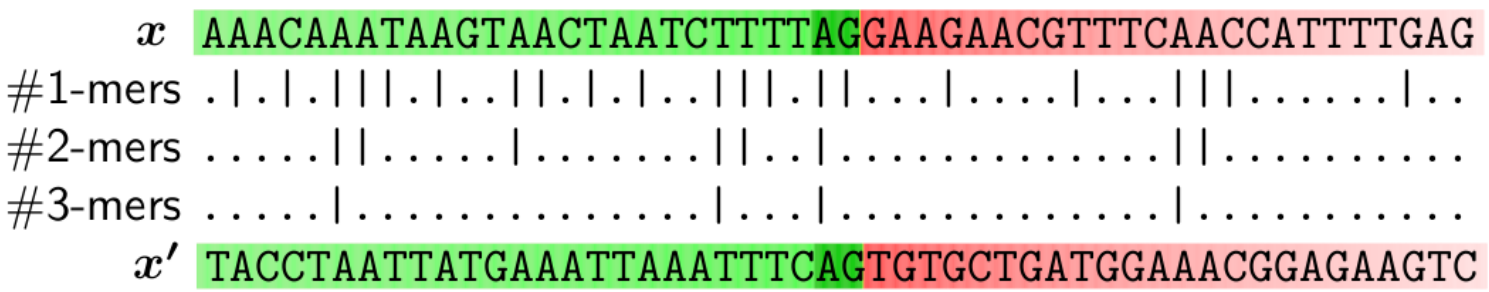
Weighted Degree Kernel = Spectrum kernels for each position

To make use of position-dependent motifs:

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

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

Example degree $d = 3$:



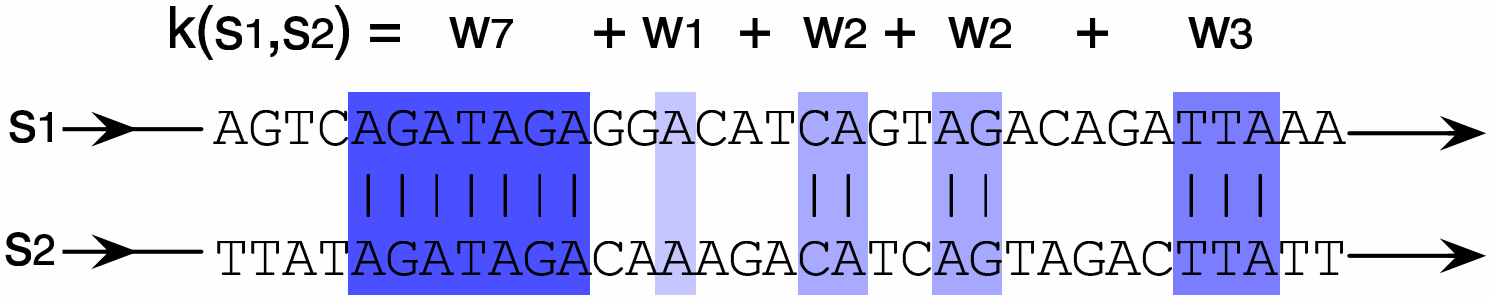
$$\mathrm{k}(\mathbf{x}, \mathbf{x}') = \beta_1 \cdot 21 + \beta_2 \cdot 8 + \beta_3 \cdot 4$$

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

Weighted Degree Kernel

- As weighting we use $\beta_k = 2\frac{d-k+1}{d(d+1)}$:
  - Longer matches are weighted less, but they imply many shorter matches
- Computational effort is $O(L \cdot d)$

Speed-up Idea: Reduce effort to $O(L)$ by finding matching "blocks" (computational effort $\mathcal{O}(L)$)



Exercise: Show that WD kernel and its "block" formulation are equivalent

Sequence-based Splice Site Recognition

Kernel	auROC
Spectrum $\ell = 1$	94.0%
Spectrum $\ell = 3$	96.4%
Spectrum $\ell = 5$	94.5%
Mixed spectrum $\ell = 1$	94.0%
Mixed spectrum $\ell = 3$	96.9%
Mixed spectrum $\ell = 5$	97.2%
WD $\ell = 1$	98.2%
WD $\ell = 3$	98.7%
WD $\ell = 5$	98.9%

The area under the ROC curve (auROC) of SVMs with the spectrum, mixed spectrum, and weighted degree kernels for the acceptor splice site recognition task for different substring lengths ℓ.

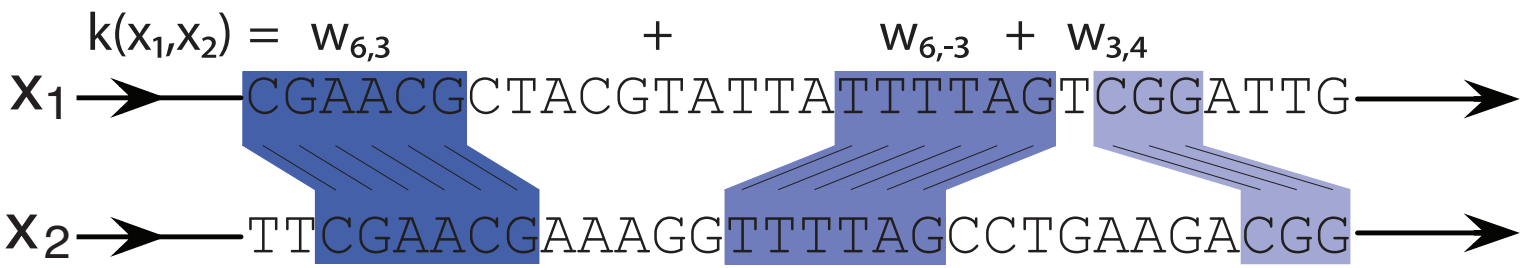
Weighted Degree Kernel with *Shifts*

To make use of partially position-dependent motifs:

- If sequence is slightly mutated (e.g. indels), WD kernel fails
- Extension: Allow some positional variance (shifts $S(l)$)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{K} \beta_k \sum_{l=1}^{L-k+1} \gamma_l \sum_{\substack{s=0 \\ s+l \le L}}^{S(l)} \delta_s \; \mu_{k,l,s,\mathbf{x}_i,\mathbf{x}_j},$$

$$\mu_{k,l,s,\mathbf{x}_i,\mathbf{x}_j} = \mathbf{I}(\mathbf{u}_{k,l+s}(\mathbf{x}_i) = \mathbf{u}_{k,l}(\mathbf{x}_j)) + \mathbf{I}(\mathbf{u}_{k,l}(\mathbf{x}_i) = \mathbf{u}_{k,l+s}(\mathbf{x}_j)),$$



[Rätsch et al., 2005]

Oligo Kernel

Oligo kernel

$$k(\mathbf{x}, \mathbf{x}') = \sqrt{\pi}\sigma \sum_{\mathbf{u}\in\Sigma^k} \sum_{p\in S_{\mathbf{u}}^{\mathbf{x}}} \sum_{q\in S_{\mathbf{u}}^{\mathbf{x}'}} e^{-\frac{1}{4\sigma^2}(p-q)^2},$$

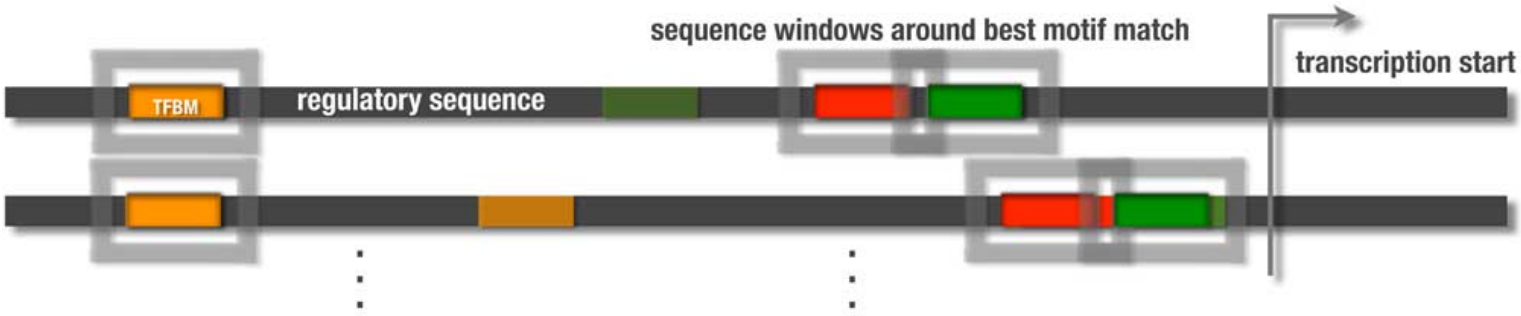
where

- $0 \leq \sigma$ is a smoothing parameter
- $\mathbf{u}$ is a k-mer and
- $S_{\mathbf{u}}^{\mathbf{x}}$ is the set of positions within sequence $\mathbf{x}$ at which $\mathbf{u}$ occurs as a substring

Similar to WD kernel with shifts.

[Meinicke et al., 2004]

Regulatory Modules Kernel [Schultheiss et al., 2008]



- Search for overrepresented motifs $m_1, \ldots, m_M$ (colored bars)
- Find best match of motif m_i in example x_j; extract windows $s_{i,j}$ at position $p_{i,j}$ around matches (boxed)
- Use a string kernel, e.g. k_{WDS}, on all extracted sequence windows, and define a combined kernel for the sequences:

$$k_{seq}(\mathbf{x}_j, \mathbf{x}_k) = \sum_{i=1}^{M} k_{WDS}(s_{i,j}, s_{i,k})$$

- Use a second kernel k_{pos}, e.g. based on RBF kernel, on vector of pairwise distances between the motif matches:

$$\mathbf{f}_j = (p_{1,j} - p_{2,j}, p_{1,j} - p_{3,j}, \ldots, p_{M-1,j} - p_{M,j})$$

- Regulatory Modules kernel: $k_{RM}(\mathbf{x}, \mathbf{x}') := k_{seq}(\mathbf{x}, \mathbf{x}') + k_{pos}(\mathbf{x}, \mathbf{x}')$

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]

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

Locality-Improved Kernel

Polynomial Kernel of degree d:

$$k_{\mathrm{POLY}}(\mathbf{x}, \mathbf{x}') = \left(\sum_{p=1}^{l} \mathsf{I}_p(\mathbf{x}, \mathbf{x}') \right)^d$$

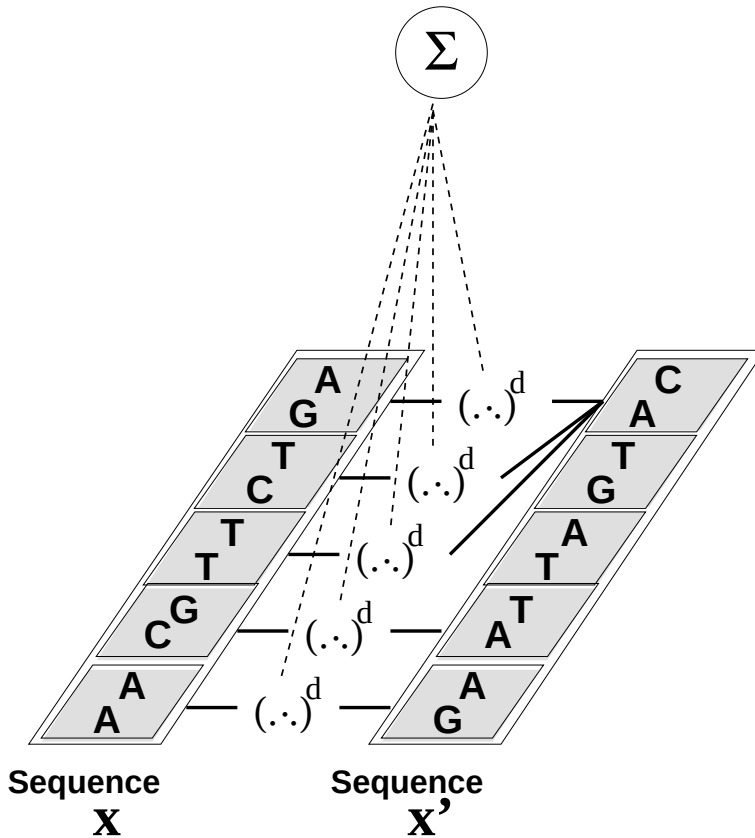
$\Rightarrow$ Computes all d-th order monomials: global information

Locality-Improved Kernel [Zien et al., 2000]

$$k_{\mathrm{LI}}(\mathbf{x}, \mathbf{y}) = \sum_{p=1}^{N} \mathrm{win}_p(\mathbf{x}, \mathbf{y})$$

$$\mathrm{win}_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=-l}^{+l} p_j \mathsf{I}_{p+j}(\mathbf{x}, \mathbf{y}) \right)^d$$

$$\mathsf{I}_i(x, x') = \begin{cases} 1, & x_i = x'_i \\ 0, & \text{otherwise} \end{cases}$$



local/global information

Fisher & TOP Kernel

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

- Combine probabilistic models and SVMs

Sequence representation

- Sequences s of arbitrary length
- Probabilistic model $p(s|\boldsymbol{\theta})$ (e.g. HMM, PSSMs)
- Maximum likelihood estimate $\boldsymbol{\theta}^* \in \mathbb{R}^d$
- Transformation into *Fisher score* features $\Phi(s) \in \mathbb{R}^d$
  - $\Phi(s) = \dfrac{\partial \log(p(s|\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}$
  - 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

- Sequences $s \in \Sigma^N$ of fixed length
- PSSMs: $p(s|\boldsymbol{\theta}) = \prod_{i=1}^{N} \theta_{i,s_i}$
- Fisher score 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.

See e.g. [Sonnenburg, 2002]

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

Summary of String Kernels

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

Demonstration Using Galaxy Webservice

Task 1: Learn to classify acceptor splice sites with sequences

1. Train classifier and predict, using 5-fold cross-validation (`SVM Toolbox` → `Train and Test SVM`)
2. Evaluate classifier (`SVM Toolbox` → `Evaluate Predictions`)

Steps:

1. Use "`Upload file`" with URL `http://svmcompbio.tuebingen.mpg.de/data/C_elegans_acc_seq.arff`. Set file format to `ARFF` and upload.
2. Use "`Train and Test SVM`" on uploaded dataset (choose `ARFF` data format) tool. Set the kernel to a) `Spectrum` with degree=6 and b) `Weight Degree` with degree=6 and shift=0. Execute and look at the result.
3. Use "`Evaluate Predictions`" on predictions and the labeled data (choose `ARFF` format). Select `ROC Curve` and execute. Check out the evaluation summary and the ROC curves.

Demonstration with Easysvm

Task 1: Learn to classify acceptor splice sites with sequences

1. Train classifier and predict using 5-fold cross-validation (cv)
2. Evaluate classifier (eval)

```
                                SVM C  kernel     data format and file      predictions
~/mylibs/bin/easysvm.py cv 5      1    spec 6  arff C_elegans_acc_seq.arff  spec_seq.out
                                predictions      data format and file       output file
~/mylibs/bin/easysvm.py eval  spec_seq.out  arff C_elegans_acc_seq.arff  spec_seq.perf
tail -3 spec_seq.perf
  Area under ROC curve = 80.4 %
  Area under PRC curve = 33.7 %
  accuracy (at threshold 0) = 90.8 %
                                SVM C  kernel     data format and file      predictions
~/mylibs/bin/easysvm.py cv 5      1    WD 6 0  arff C_elegans_acc_seq.arff  wd_seq.out
                                predictions      data format and file       output file
~/mylibs/bin/easysvm.py eval  wd_seq.out  arff C_elegans_acc_gc.arff  wd_seq.perf
tail -6 lin_gc.perf
  Area under ROC curve = 98.8 %
  Area under PRC curve = 87.5 %
  Accuracy (at threshold 0) = 97.0 %
```

Graphs are everywhere . . .

Graphs in Reality

- Graphs model objects and their relationships.
- Also referred to as *networks*.
- All common data structures can be modelled as graphs.

Graphs in Bioinformatics

- Molecular biology studies relationships between molecular components.
- Graphs are ideal to model:
  - Molecules
  - Protein-protein interaction networks
  - Metabolic networks

Central Questions

How similar are two graphs?

- Graph similarity is the central problem for all learning tasks such as clustering and classification on graphs.

Applications

- Function prediction for molecules, in particular, proteins
- Comparison of protein-protein interaction networks

Challenges

- Subgraph isomorphism is NP-complete.
- Comparing graphs via isomorphism checking is thus prohibitively expensive!
- Graph kernels offer a faster, yet one based on sound principles.

From the beginning ...

Definition of a Graph

- A *graph* G is a set of nodes (or vertices) V and edges E, where $E \subset V^2$.
- An *attributed* graph is a graph with labels on nodes and/or edges; we refer to labels as *attributes*.
- The *adjacency matrix* A of G is defined as

$$[A]_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise} \end{cases},$$

  where v_i and v_j are nodes in G.
- A *walk* w of length $k-1$ in a graph is a sequence of nodes $w = (v_1, v_2, \cdots, v_k)$ where $(v_{i-1}, v_i) \in E$ for $1 \leq i \leq k$.
- w is a *path* if $v_i \neq v_j$ for $i \neq j$.

Graph Isomorphism

Graph isomorphism (cf. Skiena, 1998)

- Find a mapping f of the vertices of G to the vertices of H such that G and H are identical; i.e. (x, y) is an edge of G iff $(f(x), f(y))$ is an edge of H. Then f is an isomorphism, and G and F are called isomorphic.
- No polynomial-time algorithm is known for graph isomorphism
- Neither is it known to be NP-complete

Subgraph isomorphism

- Subgraph isomorpism asks if there is a subset of edges and vertices of G that is isomorphic to a smaller graph H.
- Subgraph isomorphism is NP-complete

Subgraph Isomorphism

NP-completeness A decision problem C is NP-complete, iff

- C is in NP
- C is NP-hard, i.e. every other problem in NP is reducible to it

Problems for the practitioner

- Excessive runtime in worst case: Runtime may grow exponentially with number of nodes
- For large graphs with many nodes, and for large datasets of graphs, this is an enormous problem

Wanted Polynomial-time similarity measure for graphs

[Gärtner et al., 2003]

Polynomial Alternatives

Graph kernels

- Compare substructures of graphs that are computable in polynomial time
- Examples: walks, paths, cyclic patterns, trees

Criteria for a good graph kernel

- Expressive
- Efficient to compute
- Positive definite
- Applicable to wide range of graphs

Random Walks

Principle

- Compare walks in two input graphs
- Walks are sequences of nodes that allow repetitions of nodes

Important trick

- Walks of length k can be computed by taking the adjacency matrix A to the power of k
- $A^k(i,j) = c$ means that c walks of length k exist between vertex i and vertex j

Product Graph

How to find common walks in **two** graphs?

- Another trick: Use the product graph of G_1 and G_2

Definition

- $G_\times = (V_\times, E_\times)$, defined via

$$V_\times(G_1 \times G_2) = \{(v_1, w_1) \in V_1 \times V_2 : label(v_1) = label(w_1)\}$$

$$E_\times(G_1 \times G_2) = \{((v_1, w_1), (v_2, w_2)) \in V^2(G_1 \times G_2) (v_1, v_2) \in E_1 \wedge (w_1, w_2) \in E_2 \wedge (label(v_1, v_2) = label(w_1, w_2))\}$$

Meaning

- Product graph consists of pairs of identically labeled nodes and edges from G_1 and G_2

Random Walk Kernel

The trick

- Common walks can now be computed from $A_\times^k$

Definition of random walk kernel

- $$k_\times(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} [\sum_{n=0}^{\infty} \lambda^n A_\times^n]_{ij},$$

Meaning

- Random walk kernel counts all pairs of matching walks
- λ is decaying factor for the sum to converge

Runtime of Random Walk Kernels

Notation

- given two graphs G_1 and G_2
- n is the number of nodes in G_1 and G_2

Computing product graph

- requires comparison of all pairs of edges in G_1 and G_2
- runtime $O(n^4)$

Powers of adjacency matrix

- matrix multiplication or inversion for $n^2 * n^2$ matrix
- runtime $O(n^6)$

Total runtime

- $O(n^6)$

Tottering

Artificially high similarity scores

- Walk kernels allow walks to visit same edges and nodes multiple times $\longrightarrow$ artificially high similarity scores by repeated visits to same two nodes

Additional node labels

- Mahé et al. [2004] add additional node labels to reduce number of matching nodes $\longrightarrow$ improved classification accuracy

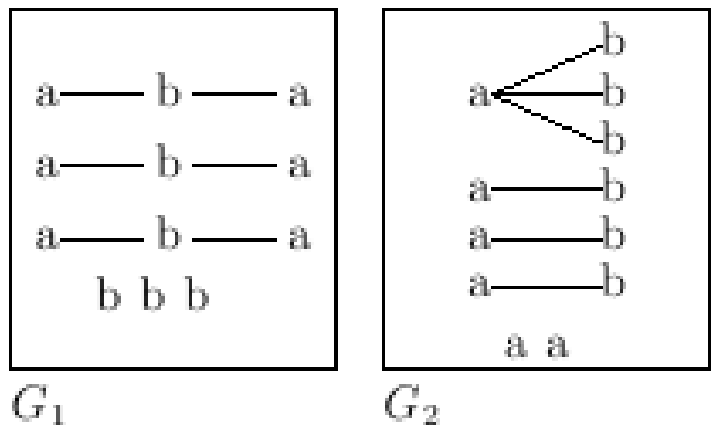
Forbidding cycles with 2 nodes

- Mahé et al. [2004] redefine walk kernel to forbid subcycles consisting of two nodes $\longrightarrow$ no practical improvement

Limitations of Walks

Different graphs mapped to identical points in walks feature space [Ramon and Gärtner, 2003]





Subtree Kernel

Motivation

- Compare tree-like substructures of graphs
- May distinguish between substructures that the walk kernel deems identical

Algorithmic principle For all pairs of nodes r from $\mathcal{V}_1(G_1)$ and s from $\mathcal{V}_2(G_2)$ and a predefined height h of subtrees:

- recursively compare neighbors (of neighbors) of r and s
- subtree kernel on graphs is sum of subtree kernels on nodes

Subtree Kernel

Matching of neighborhoods

- $\delta^+(r)$ is the set of nodes adjacent to node r
- $M(r,s)$ is the set of all matchings from $\delta^+(r)$ to $\delta^+(s)$

$$M(r,s) = \{R \subseteq \delta^+(r) \times \delta^+(s) |$$
$$(\forall (a,b),(c,d) \in R : a = c \iff b = d) \wedge$$
$$(\forall (a,b) \in R : label(a) = label(b))\}$$

Kernel computation on pairs of trees

- then, $k_h(r,s)$ can be computed as

$$k_h(r,s) = \lambda_r \lambda_s \sum_{R \in M(r,s)} \prod_{(r',s') \in R} k_{h-1}(r',s'),$$

- where λ_r and λ_s are positive scalars

Subtree Kernel

Subtree graph kernel

- The subtree graph kernel for fixed height h is

$$k_{tree,h}(G_1, G_2) = \sum_{r \in \mathcal{V}_1} \sum_{s \in \mathcal{V}_2} k_h(r, s).$$

- The subtree graph kernel for h approaching infinity is

$$k_{tree}(G_1, G_2) = \lim_{h \to \infty} k_{tree,h}(G_1, G_2),$$

  which will converge for suitable choice of λ_r and λ_s.
- Both versions are positive definite
- Large choice of h provides good approximation of k_{tree}.

Cycles instead of walks?

Idea

- Computing kernels based on cyclic and tree patterns [Horváth et al., 2004]
- Intersection kernel instead of kernel based on counts

Problems

- Computation of all general cycles is NP-hard
- Remedy: Consider graphs with up to k simple cycles only
- Problem: Cyclic pattern kernel can only be used on datasets fulfilling this constraint

Depth-first search paths?

Idea

- Computing kernels based on paths of length up to d starting from a node r [Ralaivola et al., 2005]
- These are determined by Depth-First Search (DFS)
- Once diverged, paths may not visit the same node
- Path counts are then combined into a kernel on graphs

Problems

- Only measures local, not global, similarity in structure
- DFS paths exclude edges that are not on these paths from graph comparison

All-paths Kernel?

Idea

- Determine all paths from two graphs
- Compare paths pairwise to yield kernel

Advantage

- No tottering

Problem

- All-paths kernel is NP-hard to compute.

Proof

- If determining all paths were not NP-hard, then one could check to see whether a Hamilton path of length n-1 exists
- However, since finding a Hamilton path is known to be NP-hard; determining all paths is NP-hard as well

Alternatives?

Longest paths?

- Also NP-hard – same reason as for all paths

Shortest Paths!

- computable in $O(n^3)$ by the classic Floyd-Warshall algorithm 'all-pairs shortest paths'

Shortest-path Kernels

Kernel computation

- Determine all shortest paths in two input graphs
- Compare all shortest distances in G_1 to all shortest distances in G_2
- Sum over kernels on all pairs of shortest distances gives shortest-path kernel

Runtime

- Given two graphs G_1 and G_2
- n is the number of nodes in G_1 and G_2
- Determine shortest paths in G_1 and G_2 separately: $O(n^3)$
- Compare these pairwise: $O(n^4)$
- Hence: Total runtime complexity $O(n^4)$

[Borgwardt and Kriegel, 2005]

Discussion

Advantages

- Compares meaningful features of graphs, namely shortest paths
- Positive definite
- No tottering
- Works on all graphs (using artificial edge length)
- Computable in $O(n^4) \rightarrow$ two magnitudes faster than the random walk kernel

Disadvantages

- Does not exploit sparsity of graphs
- Leads to full matrix representations of graphs
- Ignores information represented by longer paths
- Most meaningful if edge labels represent some type of distance

Applications in Bioinformatics

Current

- Comparing structures of proteins
- Comparing structures of RNA
- Measuring similarity between metabolic networks
- Measuring similarity between protein interaction networks
- Measuring similarity between gene regulatory networks

Future

- Detecting conserved paths in interspecies networks
- Finding differences in individual or interspecies networks
- Finding common motifs in biological networks

[Borgwardt et al., 2005; Ralaivola et al., 2005]

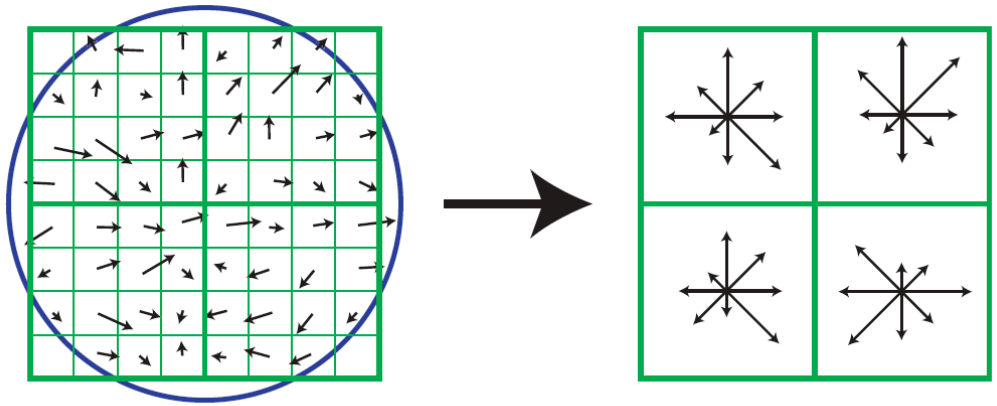
Image Classification (Caltech 101 dataset, [Fei-Fei et al., 2004])



Bag-of-visual-words representation is standard practice for object classification systems [Nowak et al., 2006]

Image Basics [Nowak et al., 2006]

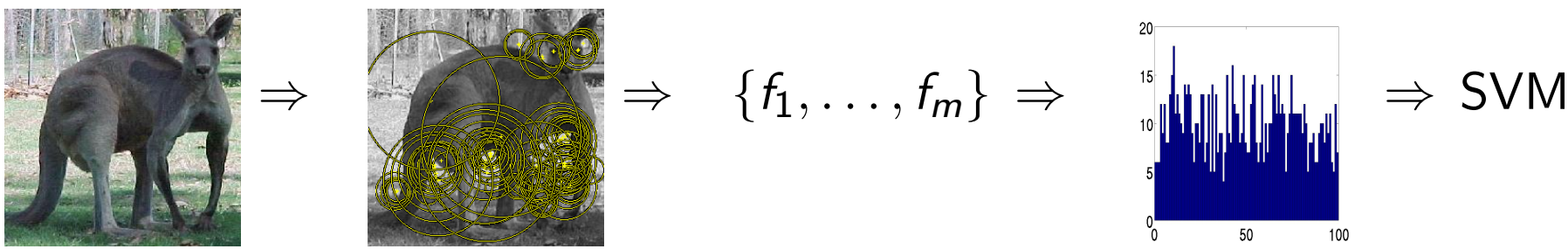
Describing key points in images, e.g. using SIFT features [Lowe, 2004]:



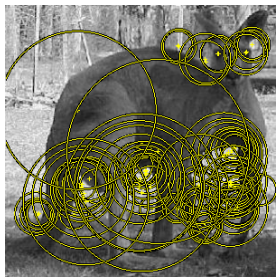
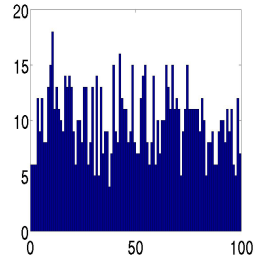
8x8 field leads to four 8-dimensional vectors

$\Rightarrow$ 32-dimensional SIFT feature vector describing the point in the image

1. Generate a set of key-points and corresponding vectors
2. Generate a set of representative "code vectors"
3. Record which code vector is *closest* to key-point vectors
4. Quantize image into histograms **h**



χ^2-Kernel for Histograms


$\Rightarrow$

$\Rightarrow \quad \{f_1, \ldots, f_m\} \Rightarrow$


$\Rightarrow$ SVM

Image described by histogram $h_{\mathcal{C}}$ implied by code book $\mathcal{C}$ of size d

Kernel for comparing two histograms:

$$k_{\gamma,\mathcal{C}}(\mathbf{h}_{\mathcal{C}}, \mathbf{h}'_{\mathcal{C}}) = \exp\left(-\gamma\chi^2(\mathbf{h}_{\mathcal{C}}, \mathbf{h}'_{\mathcal{C}})\right),$$

where γ is a hyper-parameter,

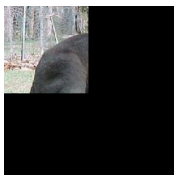
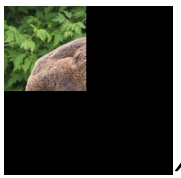
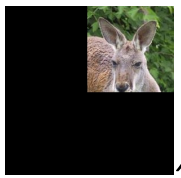
$$\chi^2(\mathbf{h}, \mathbf{h}') := \sum_{i=1}^{d} \frac{(\mathbf{h}_i - \mathbf{h}'_i)^2}{\mathbf{h}_i + \mathbf{h}'_i},$$

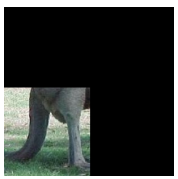
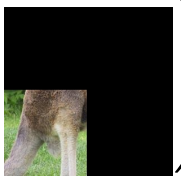
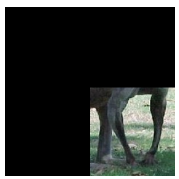
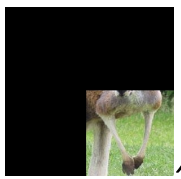
and we use the convention $x/0 := 0$

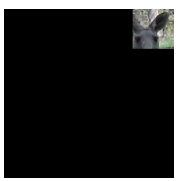
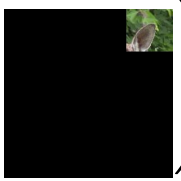
Spatial Pyramid Kernels

Decompose image into a pyramid of L levels

$$k_{\text{pyr}}\left(\text{}, \text{}\right) = \frac{1}{8}k\left(\text{}, \text{}\right)$$

$$+ \frac{1}{4}k\left(\text{}, \text{}\right) + \frac{1}{4}k\left(\text{}, \text{}\right)$$

$$+ \frac{1}{4}k\left(\text{}, \text{}\right) + \frac{1}{4}k\left(\text{}, \text{}\right)$$

$$+ \frac{1}{2}k\left(\text{}, \text{}\right) + \ldots$$

[Lazebnik et al., 2006]

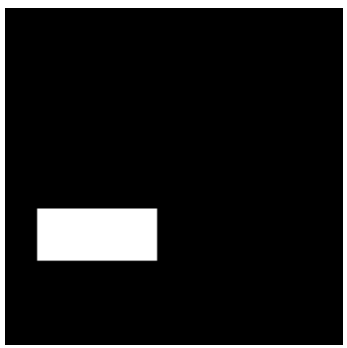
General Spatial Kernels

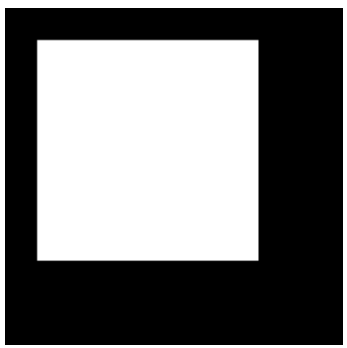
Use general spatial kernel with subwindow B

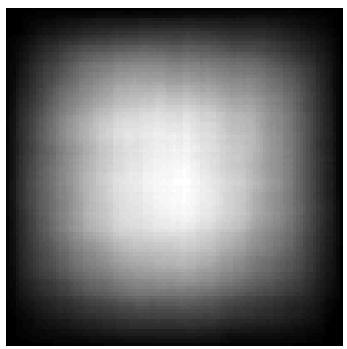
$$k_{\gamma,B}(\mathbf{h},\mathbf{h}';\{\gamma,B\}) = \exp\left(-\gamma^2\chi^2_B(\mathbf{h},\mathbf{h}')\right).$$

where $\chi^2_B(\mathbf{h},\mathbf{h}')$ only considers the key-points within region B

Example regions:







Application of Multiple Kernel Learning

- Consider set of code books $\mathcal{C}_1, \ldots, \mathcal{C}_K$ or regions $B_1, \ldots, B_K$
- Each code book $\mathcal{C}_p$ or region B_p leads to a kernel $k_p(\mathbf{x}, \mathbf{x}')$.
- Which kernel is best suited for classification?
- Define kernel as linear combination

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

- Use multiple kernel learning to determine the optimal β's.

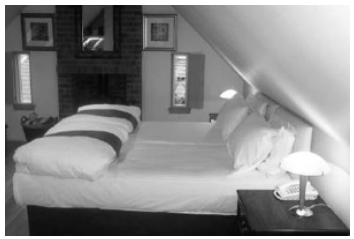
[Gehler and Nowozin, 2009]

Example: Scene 13 Datasets

Classify images into the following categories:


CALsuburb


kitchen


bedroom


livingroom


MITcoast


MITinsidecity

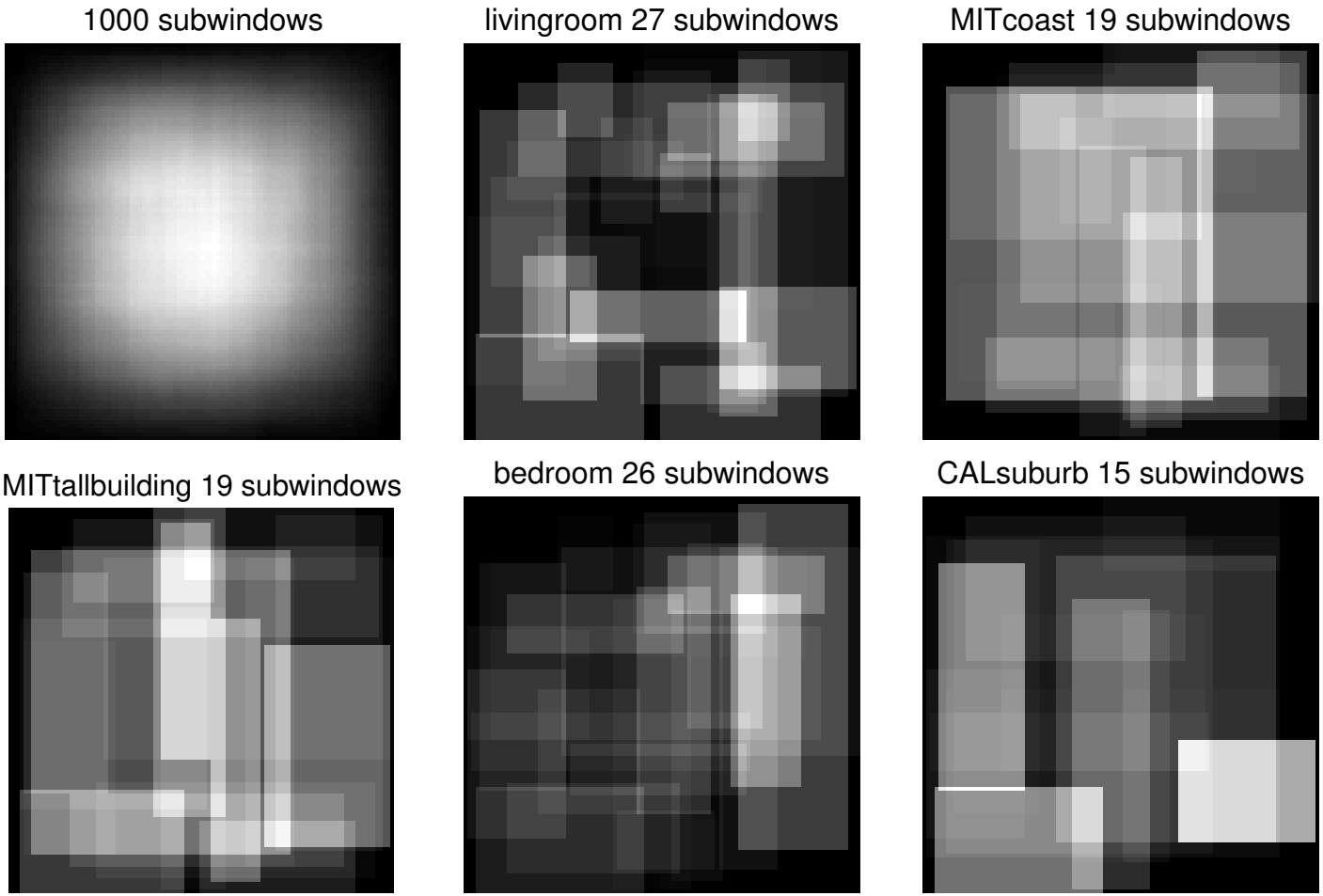

MITopencountry


MITtallbuilding

Each class has between 210-410 example images

[Fei-Fei and Perona, 2005]

Example: Optimal Spatial Kernel of Scene 13



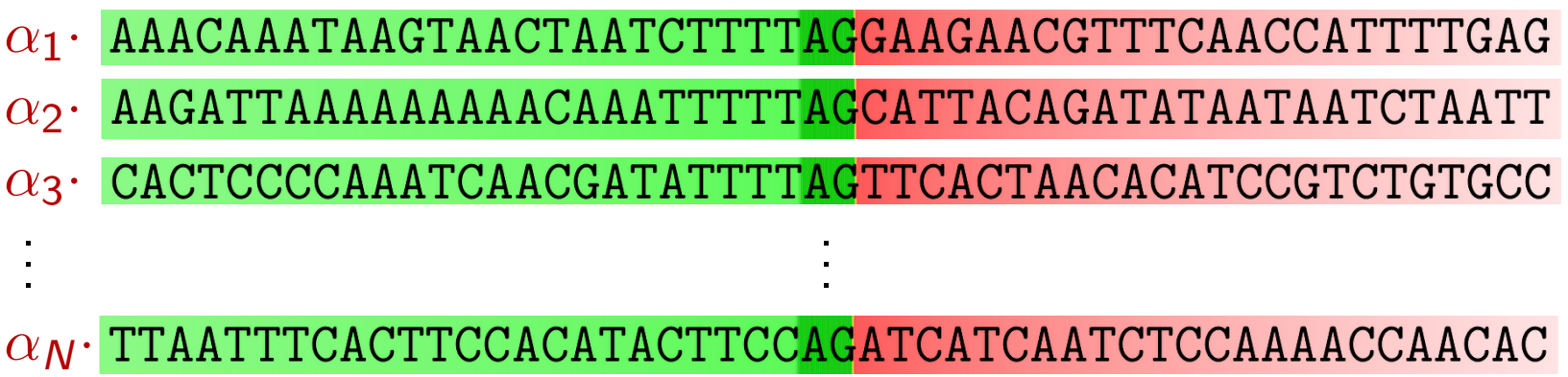
For each class differently shaped regions are optimal

[Gehler and Nowozin, 2009]

Why Are SVMs Hard to Interpret?

SVM decision function is α-weighting of training points

$$s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \, \mathrm{k}(\mathbf{x}_i, \mathbf{x}) + b$$



But we are interested in weights of features

Understanding Linear SVMs

Support Vector Machine

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

Use SVM w from feature space

- Recall SVM decision function in kernel feature space:

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

- Explicitly compute $\mathbf{w} = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i)$

Understanding Linear SVMs

Explicitly compute

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i)$$

Use $\mathbf{w}$ to rank importance

dim	$\lvert\mathbf{w}_{dim}\rvert$
17	+27.21
30	+13.1
5	-10.5
...	...

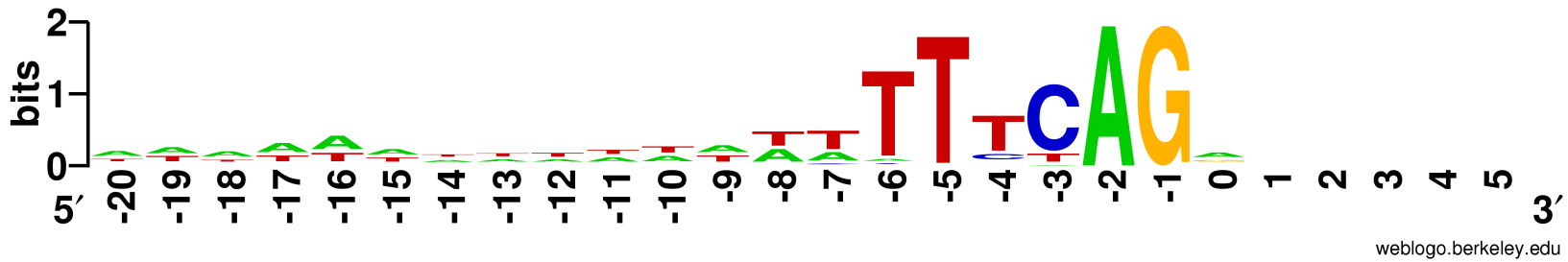
- For linear SVMs $\Phi(\mathbf{x}) = \mathbf{x}$
- For polynomial SVMs, e.g. degree 2:

$$\Phi(\mathbf{x}) = (x_1x_1, \sqrt{\frac{1}{2}}x_1x_2, \ldots \sqrt{\frac{1}{2}}x_1x_d, \sqrt{\frac{1}{2}}x_2x_3 \ldots x_dx_d)$$

Understanding String Kernel based SVMs

Understanding SVMs with sequence kernels is considerably more difficult

For PWMs we have sequence logos:



Goal: We would like to have similar means to understand Support Vector Machines

SVM Scoring Function

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i) \qquad s(\mathbf{x}) := \sum_{k=1}^{K} \sum_{i=1}^{L-k+1} w\left(\mathbf{x}[i]^k, i\right) + b$$

k-mer	pos. 1	pos. 2	pos. 3	pos. 4	$\cdots$
A	+0.1	-0.3	-0.2	+0.2	$\cdots$
C	0.0	-0.1	+2.4	-0.2	$\cdots$
G	+0.1	-0.7	0.0	-0.5	$\cdots$
T	-0.2	-0.2	0.1	+0.5	$\cdots$
AA	+0.1	-0.3	+0.1	0.0	$\cdots$
AC	+0.2	0.0	-0.2	+0.2	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
TT	0.0	-0.1	+1.7	-0.2	$\cdots$
AAA	+0.1	0.0	0.0	+0.1	$\cdots$
AAC	0.0	-0.1	+1.2	-0.2	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
TTT	+0.2	-0.7	0.0	0.0	$\cdots$

SVM Scoring Function - Examples

$$s(\mathbf{x}) := \sum_{k=1}^{K} \sum_{i=1}^{L-k+1} w\Big(\mathbf{x}[i]^k, i\Big) + b$$

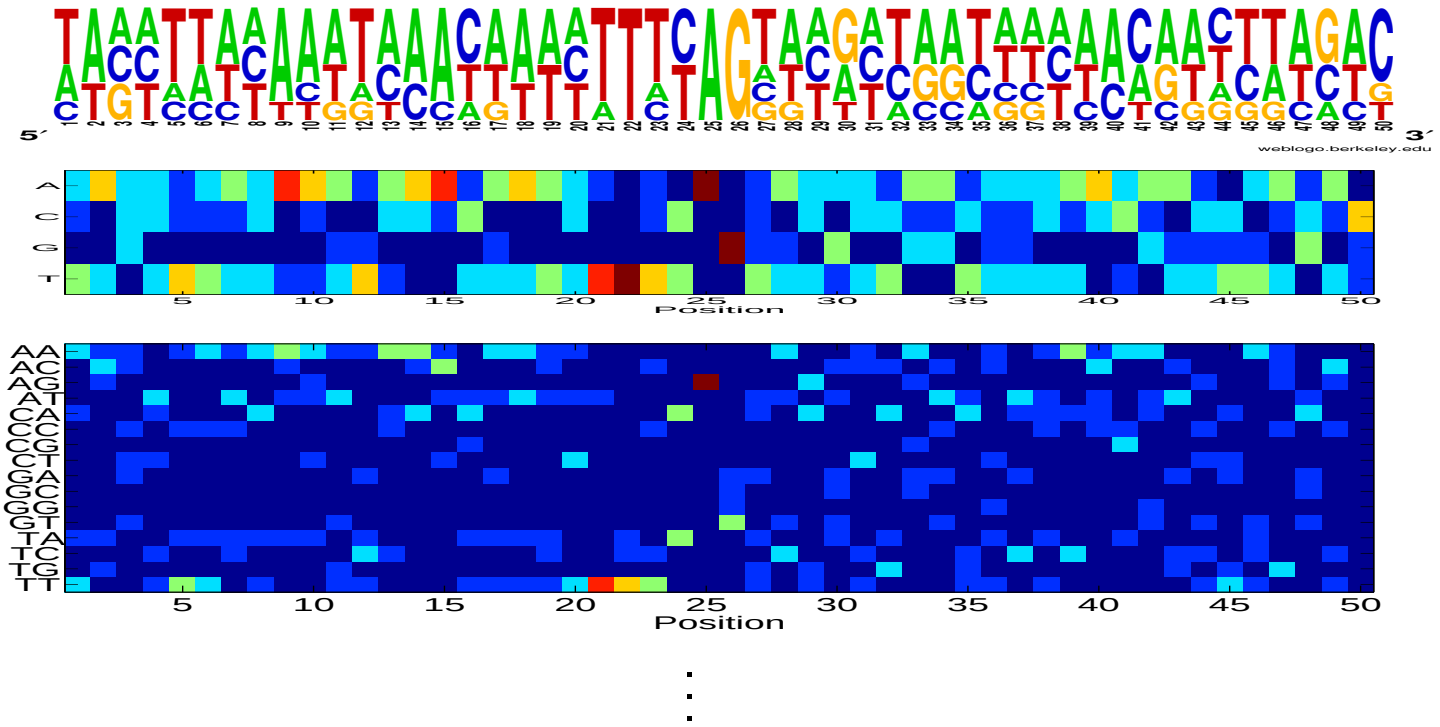
Examples:

- WD kernel (Rätsch, Sonnenburg, 2005)
- WD kernel with shifts (Rätsch, Sonnenburg, 2005)
- Spectrum kernel (Leslie, Eskin, Noble, 2002)
- Oligo kernel (Meinicke et al., 2004)

Not limited to SVM's:

- Markov chains (higher order/heterogeneous/mixed order)

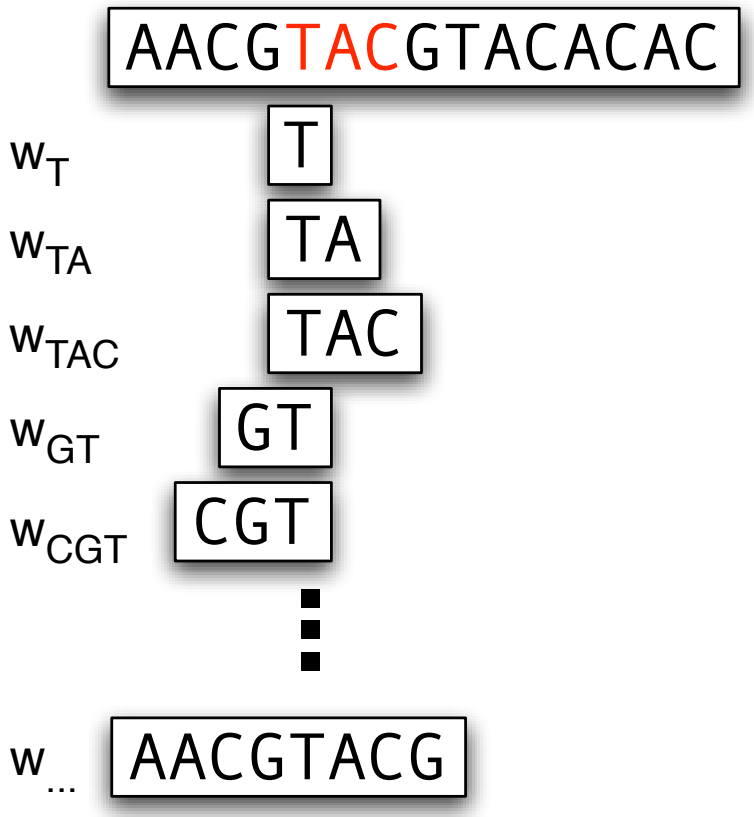
The SVM Weight Vector **w**



- Explicit representation of **w** allows (some) interpretation!
- String kernel SVMs capable of efficiently dealing with large k-mers $k > 10$

But: Weights for substrings not independent

Interdependence of $k-$mer Weights



What is the score for TAC?

- Take w_{TAC}?
- But substrings and overlapping strings contribute, too!

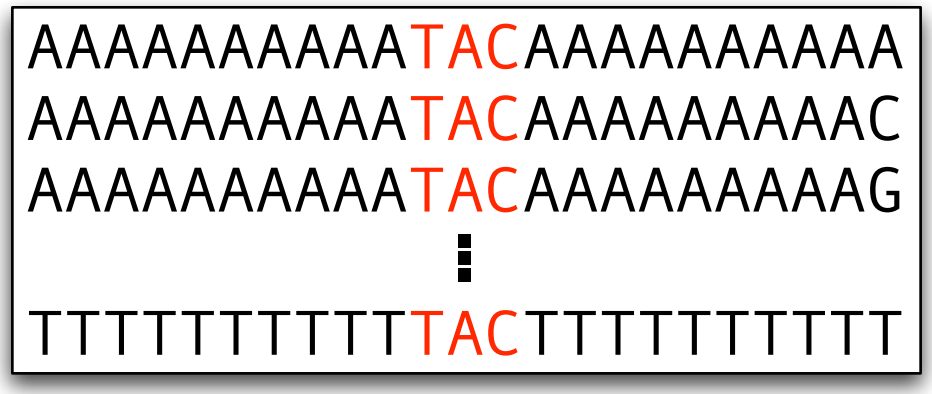
Problem

The SVM-w does not reflect the score for a motif

Positional Oligomer Importance Matrices (POIMs)

Idea:

- Given $k-$mer $\mathbf{z}$ at position j in the sequence, compute expected score $\mathbb{E}\left[\, s(\mathbf{x}) \,|\, \mathbf{x}[j] = \mathbf{z} \,\right]$ (**for small k**)



- Normalize with *expected score* over **all sequences**

POIMs

$$Q(\mathbf{z}, j) \;:=\; \mathbb{E}\left[\, s(\mathbf{x}) \,|\, \mathbf{x}[j] = \mathbf{z} \,\right] - \mathbb{E}\left[\, s(\mathbf{x}) \,\right]$$

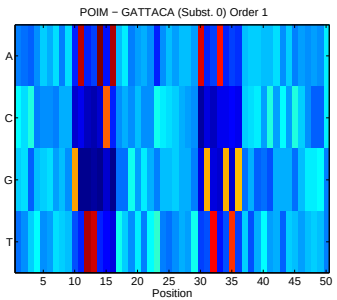
$\Rightarrow$ Needs efficient algorithm for computation [Sonnenburg et al., 2008]

Ranking Features and Condensing Information

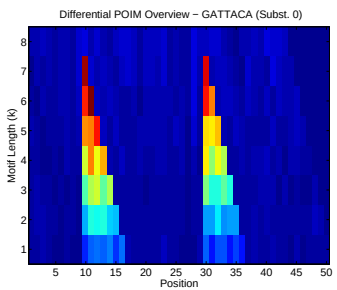
- Obtain highest scoring $\mathbf{z}$ from $Q(\mathbf{z}, i)$ (Enhancer or Silencer)

$\mathbf{z}$	i	$Q(\mathbf{z}, i)$
GATTACA	10	+30
AGTAGTG	30	+20
AAAAAAA	10	-10
...	...	...

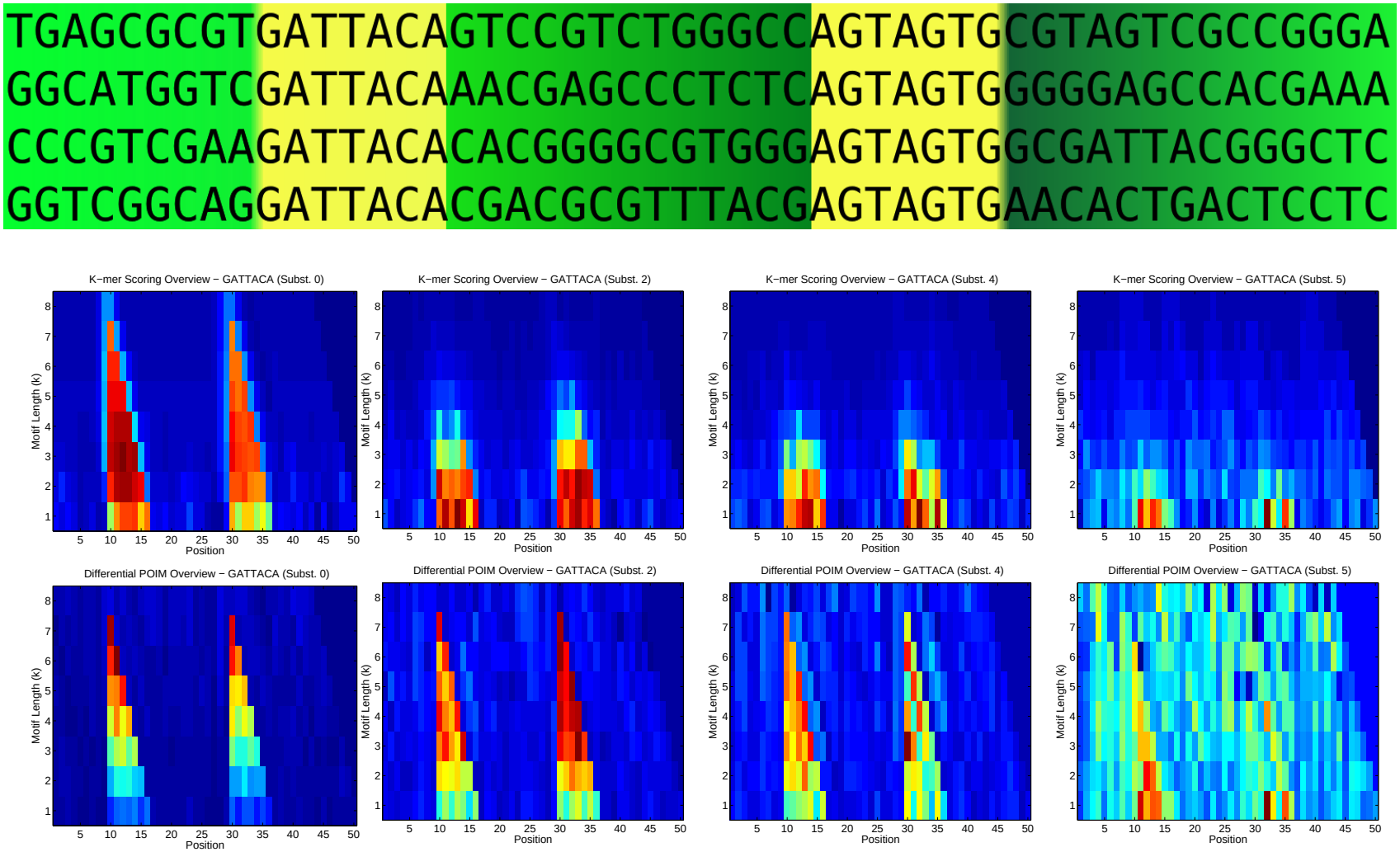
- Visualize POIM as heat map;
  x-axis: position
  y-axis: k-mer
  color: importance



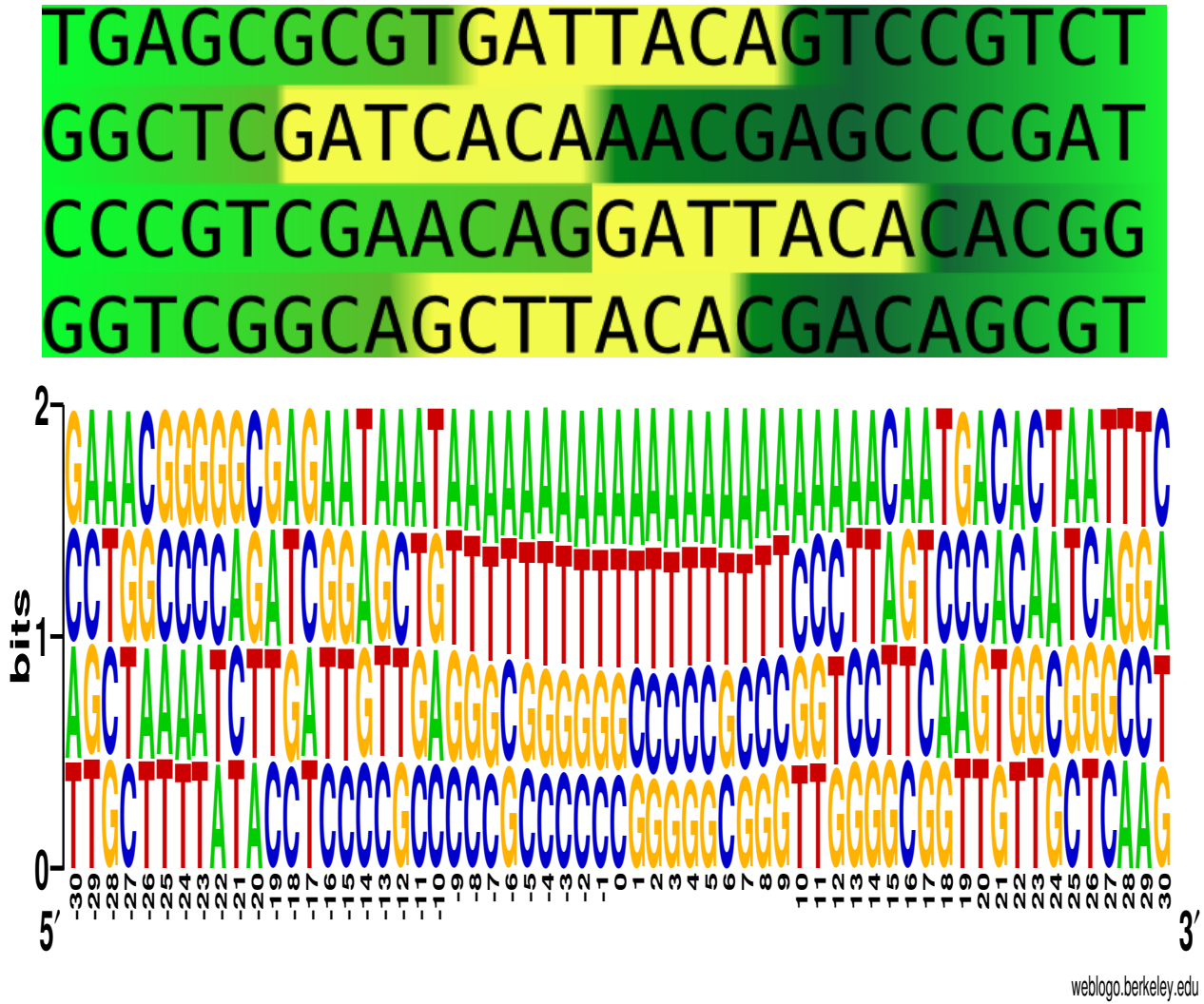
- For large k: differential POIMs;
  x-axis: position
  y-axis: k-mer length
  color: importance



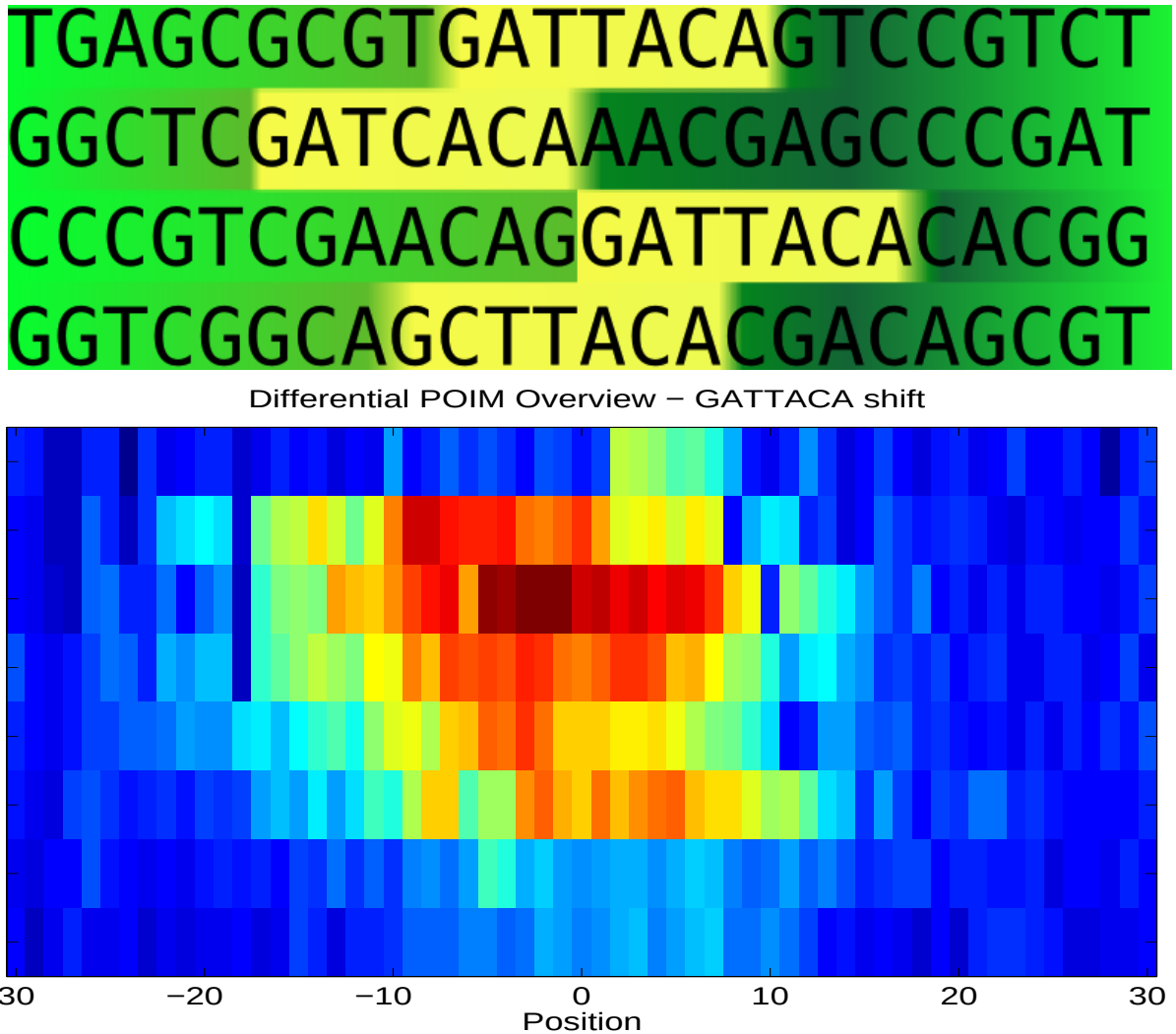
GATTACA and AGTAGTG at Fixed Positions 10 and 30



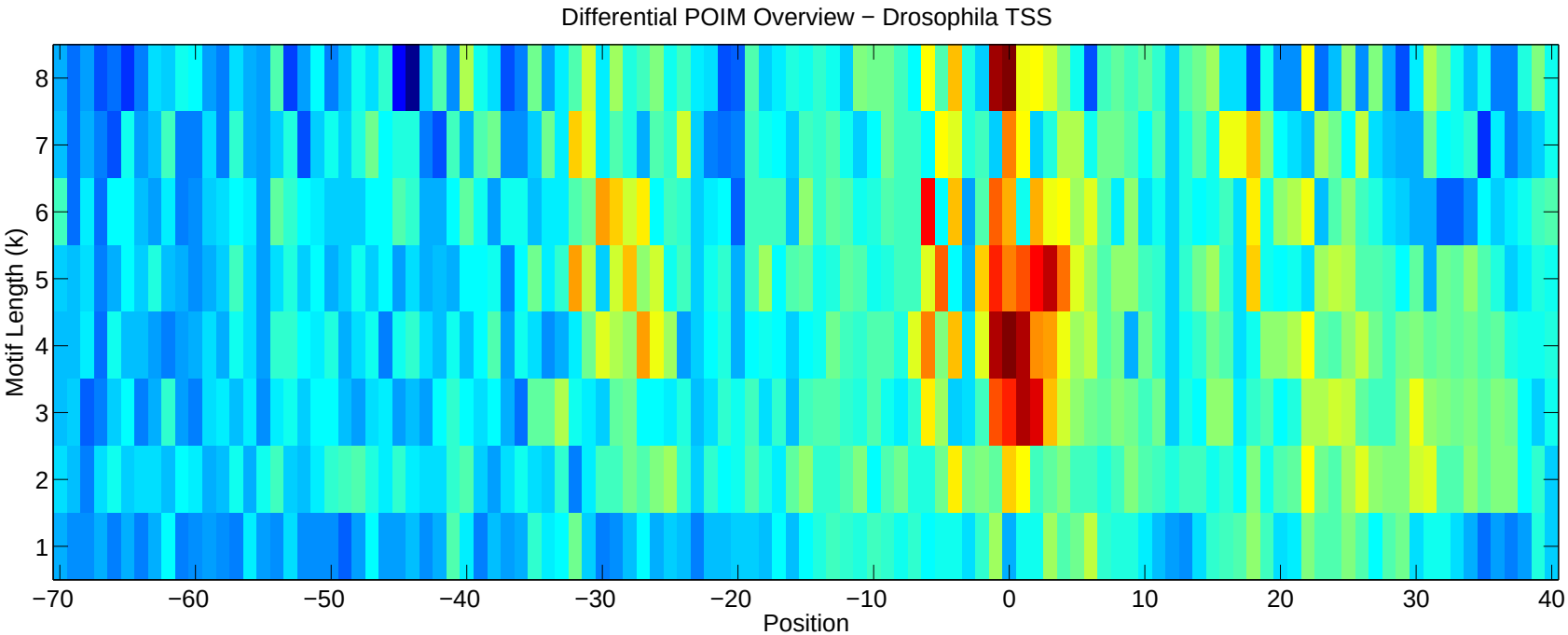
GATTACA at Variable Positions



GATTACA at Variable Positions



Drosophila Transcription Start Sites



TATAAAA	-29/++	**CAGT**CAGT	-01/++	CGTCGCG	+18/++
GTATAAA	-30/++	T**CAGT**TGT	-01/++	GCGCGCG	+23/++
ATATAAA	-28/++	CGT**CAGT**T	-03/++	CGCGCGC	+22/++
TATA-box		**Inr** **TCA**$\frac{G}{T}$**T**$\frac{T}{C}$		**CpG**	

Understanding General SVMs

A few possibilities:

- Perform feature selection using wrapper methods

  [Kohavi and John, 1997]
- Define kernels on suitable subsets of features
  - Determine which kernels contribute most to improve the performance
  - Multiple Kernel Learning to find a weighting over the kernel giving an indication which kernels are important

    [Gehler and Nowozin, 2009; Rätsch et al., 2006]
- Extend the POIM concept to general kernels (e.g. Feature Importance Ranking Measure [Zien et al., 2009])

Approach: Optimize Combination of Kernels

- Define kernel as convex combination of subkernels:

$$\mathsf{k}(\mathbf{x},\mathbf{y}) = \sum_{l=1}^{L} \beta_l \, \mathsf{k}_l(\mathbf{x},\mathbf{y})$$

  for instance, Weighted Degree kernel

$$\mathsf{k}(\mathbf{x},\mathbf{x}') = \sum_{l=1}^{L} \beta_l \sum_{k=1}^{K} \mathbf{I}(\mathbf{u}_{k,l}(\mathbf{x}) = \mathbf{u}_{k,l}(\mathbf{x}'))$$

- Optimize weights $\boldsymbol{\beta}$ such that margin is maximized

  $\Rightarrow$ determine $(\boldsymbol{\beta}, \boldsymbol{\alpha}, b)$ simultaneously

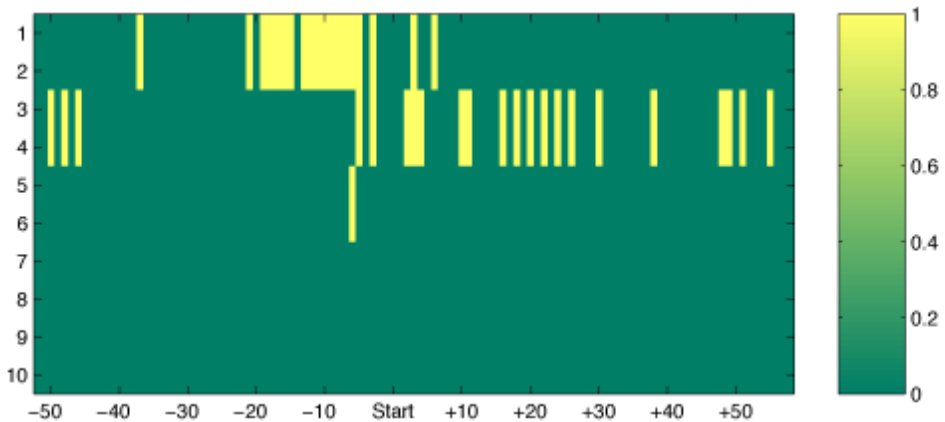
  $\Rightarrow$ **Multiple Kernel Learning** [Bach et al., 2004]

Method for Interpreting SVMs

- Weighted Degree kernel: linear comb. of LD kernels

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

- Example: Classifying splice sites



See Rätsch et al. [2006] for more details

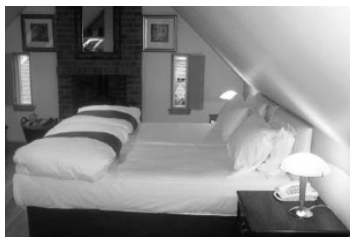
Scene 13: Datasets



CALsuburb



kitchen



bedroom



livingroom



MITcoast



MITinsidecity

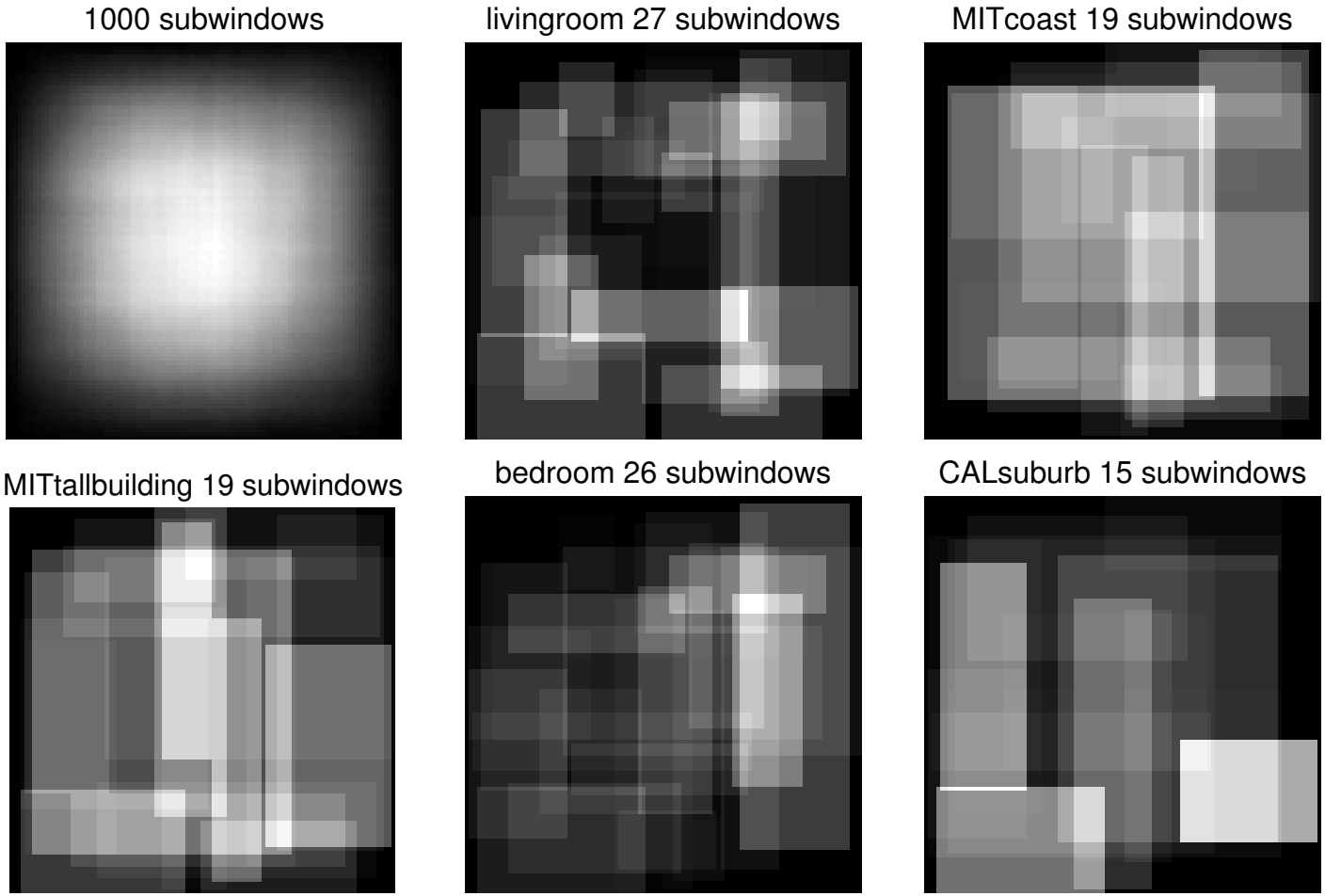


MITopencountry



MITtallbuilding

Scene 13: Optimal Spatial Kernel



Part IV

Structured Output Learning

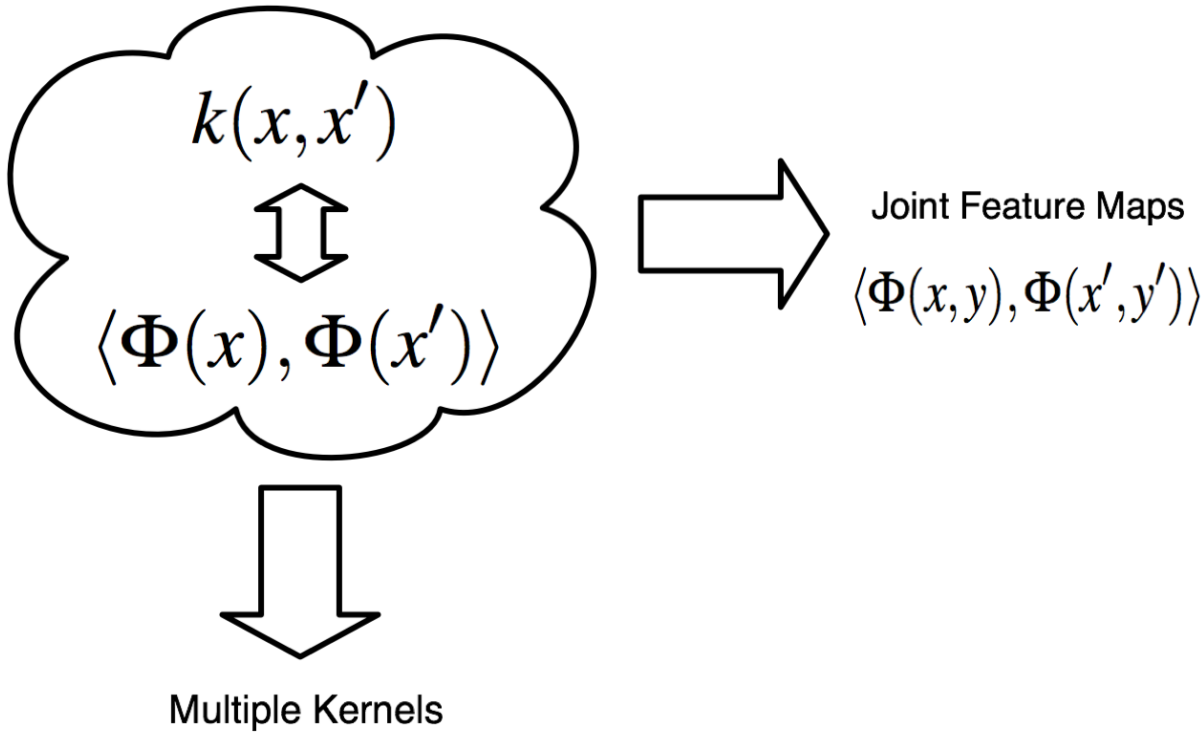
Overview: Structured Output Learning

Introduction

Generative Models
- Hidden Markov Models
- Dynamic Programming

Discriminative Methods
- Conditional Random Fields
- Hidden Markov SVMs
- Structure Learning with Kernels
- Algorithm
- Using Loss Function for Segmentations

Generalizing Kernels



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

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 $\mathcal{Y}$ are the identity, that is,

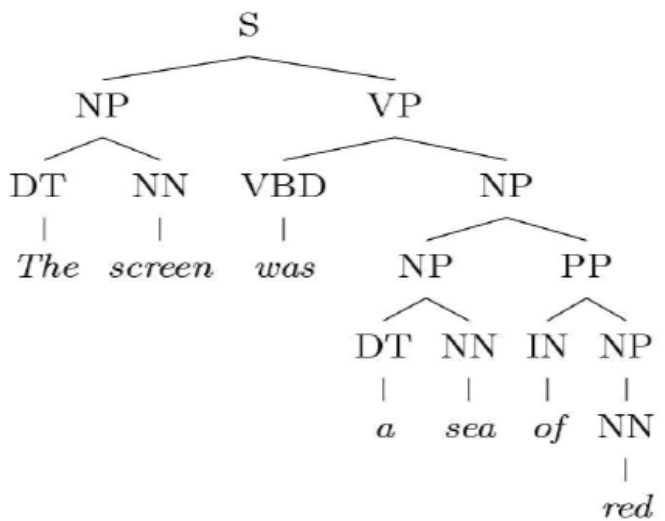
$$k((\mathbf{x}, y), (\mathbf{x}', y')) := [[y = y']] k(\mathbf{x}, \mathbf{x}')$$

Example: Context-free Grammar Parsing

x

The screen was
a sea of red

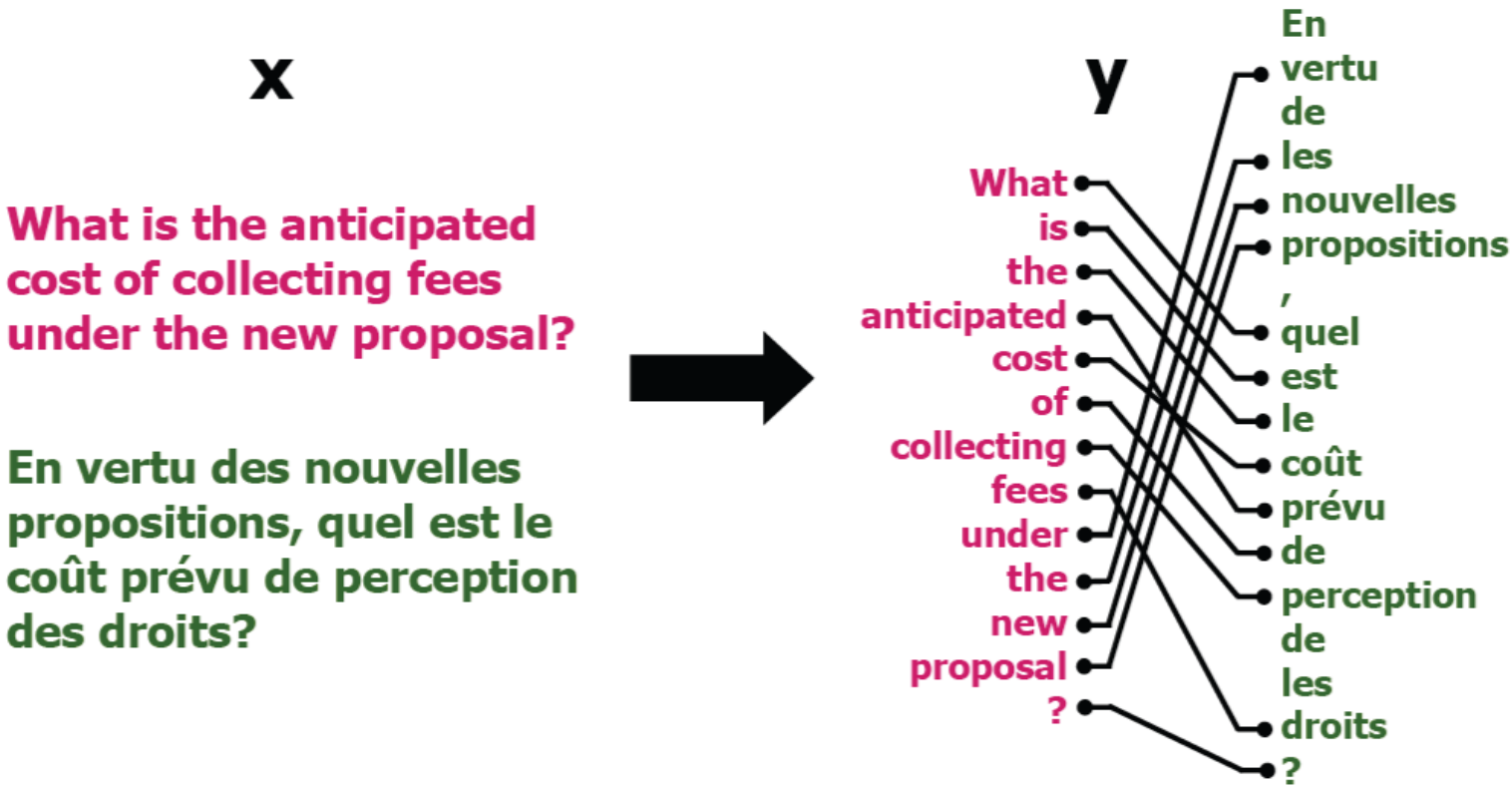
y



Recursive Structure

[Klein & Taskar, ACL'05 Tutorial]

Example: Bilingual Word Alignment



Combinatorial Structure

[Klein & Taskar, ACL'05 Tutorial]

Example: Handwritten Letter Sequences

x y



Sequential Structure

[Klein & Taskar, ACL'05 Tutorial]

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: Secondary Structure Prediction of Proteins

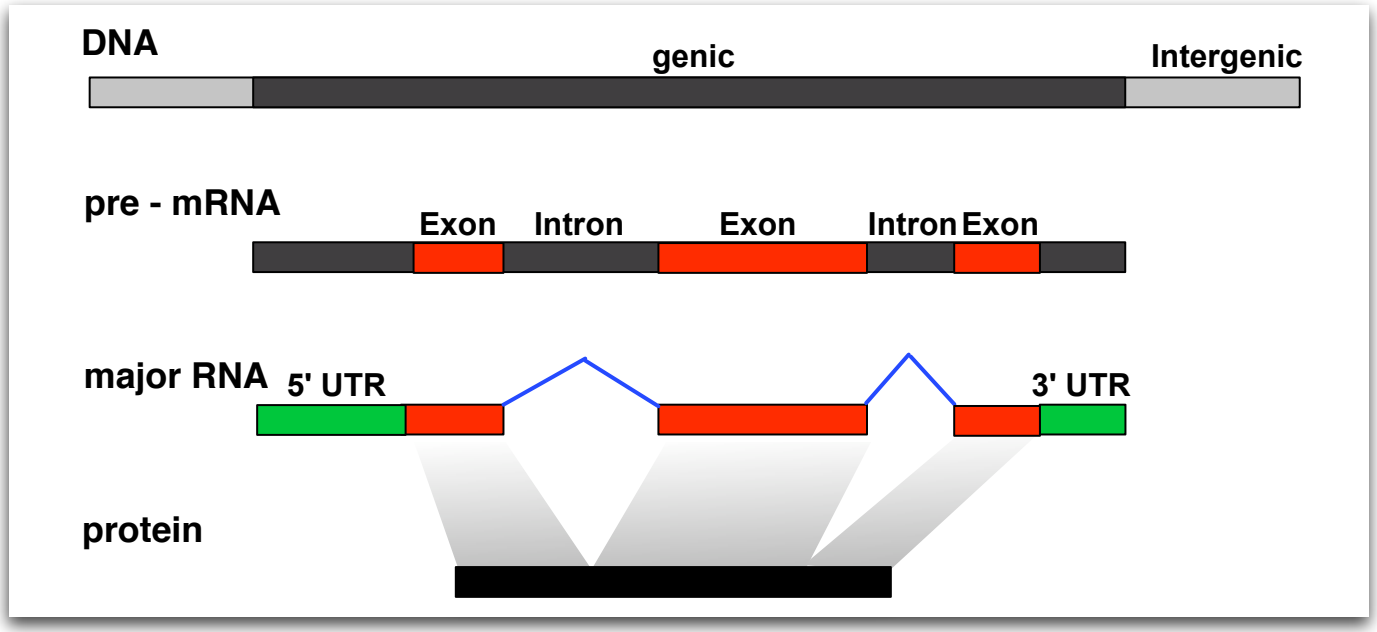


Residue Sequence: NWVLSTAADMQGVVTDGMASFL...

Secondary Structure: LLEEEELLLLHHHHHHHHHHLH...

Label Sequence Learning

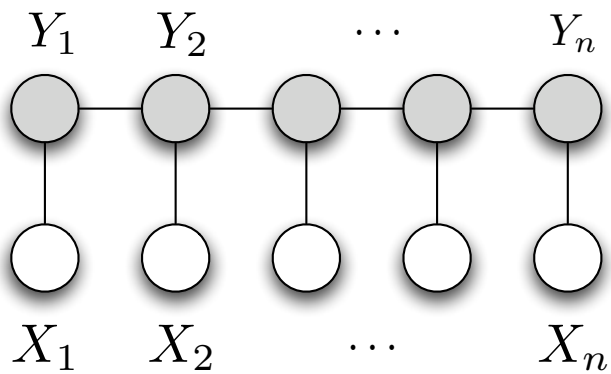
- Given: observation sequence
- Problem: predict corresponding state sequence
- Often: several subsequent positions have the same state
  $\Rightarrow$ state sequence defines a "segmentation"
- Example 2: Gene Finding



Generative Models

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

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

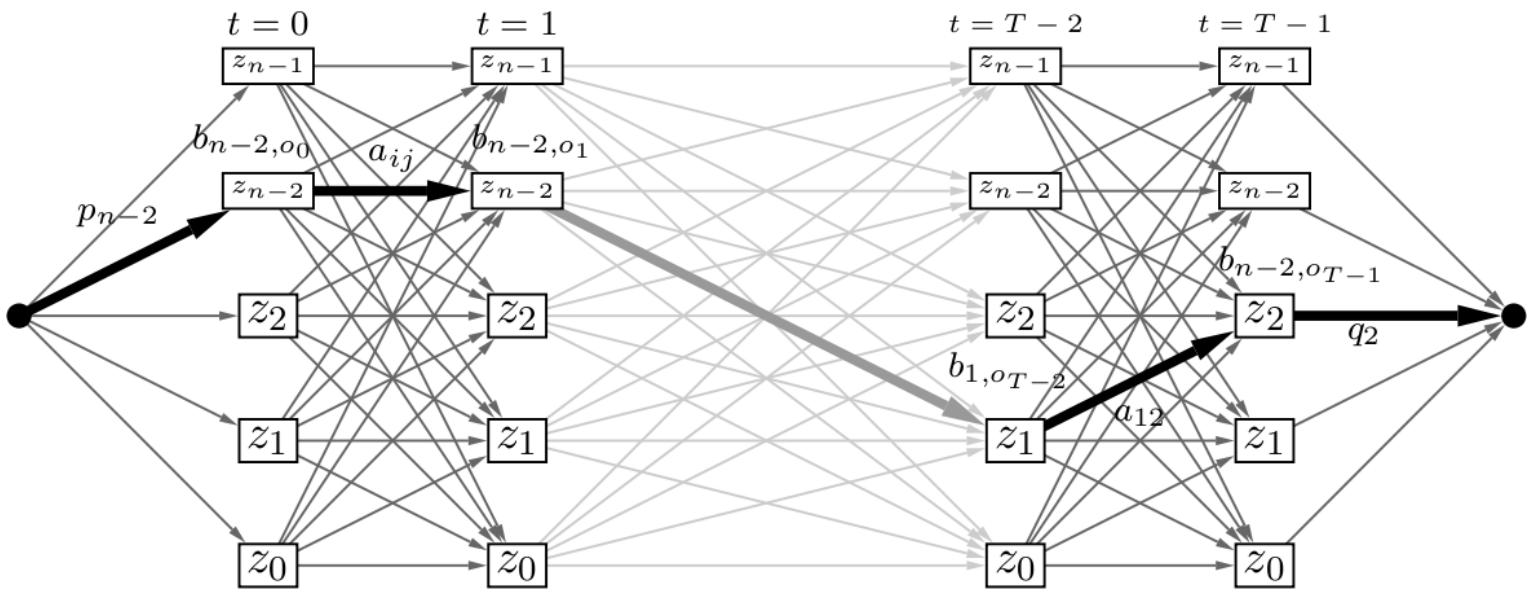


  - Efficient dynamic programming (DP) algorithms

Dynamic Programming

- Number of possible paths of length T for a (fully connected) model with n states is n^T
- Infeasible even for small T

Solution: Use dynamic programming (Viterbi decoding)



- Runtime complexity **before:** $\mathcal{O}(n^T) \Rightarrow$ **now:** $\mathcal{O}(n^2 \cdot T)$

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. $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}^n} \log p(\mathbf{x},\mathbf{y})$

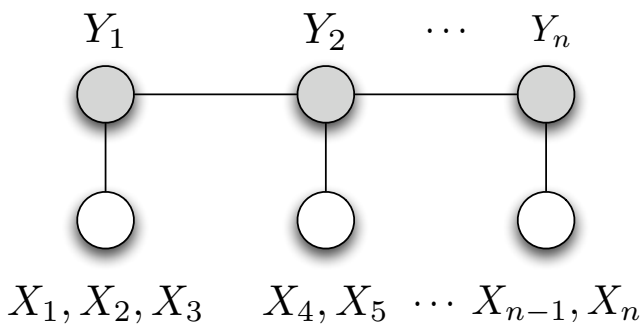
Dynamic Programming Approach:

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

Generative Models

- Generalized Hidden Markov Models
  $=$ Hidden Semi-Markov Models
  - Only one state variable per segment
  - Allow non-independence of positions within segment
  - Example: first-order Hidden Semi-Markov Model

$$p(x,y) = \prod_j p(\underbrace{(x_{i(j-1)+1}, \ldots, x_{i(j)})}_{\mathbf{x}_j} | y_j) p(y_j | y_{j-1})$$



(use with care)

- Use generalization of DP algorithms of HMMs

Decoding *via* Dynamic Programming

$$\begin{aligned}\log p(\mathbf{x},\mathbf{y}) &= \prod_j p((x_{i(j)},\ldots,x_{i(j+1)-1})|y_j)p(y_j|y_{j-1}) \\ &= \sum_j g(y_{i-1},y_i,\underbrace{(x_{i(j-1)+1},\ldots,x_{i(j)})}_{\mathbf{x}_j})\end{aligned}$$

with $g(y_{j-1},y_j,\mathbf{x}_j) = \log p(\mathbf{x}_j|y_j) + \log p(y_j|y_{j-1})$.

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

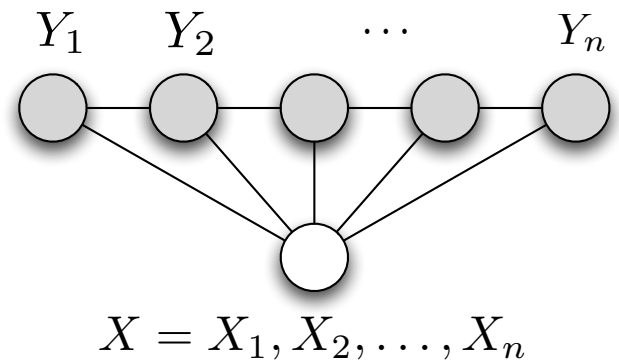
Dynamic Programming Approach:

$$V(i,y) := \begin{cases} \max\limits_{y'\in\mathcal{Y},d=1,\ldots,i-1} (V(i-d,y') + g(y',y,\mathbf{x}_{i-d+1,\ldots,i})) & i>1 \\ 0 & \text{otherwise} \end{cases}$$

Discriminative Models

- Conditional Random Fields [Lafferty et al., 2001]
  - Conditional probability $p(y|x)$ instead of joint probability $p(x, y)$

$$p_{\mathbf{w}}(y|x) = \frac{1}{Z(x, \mathbf{w})} \exp(f_{\mathbf{w}}(y|x))$$



  - Can handle non-independent input features
  - Parameter estimation: $\max_{\mathbf{w}} \sum_{n=1}^{N} \log p_{\mathbf{w}}(\mathbf{y}_n|\mathbf{x}_n)$
- Decoding: Viterbi or Maximum Expected Accuracy algorithms (cf. [Gross et al., 2007])

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:

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

- 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:

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

  - Exponentially many constraints!

Joint Feature Map

Recall the kernel trick

For each kernel, there exists a corresponding feature mapping $\Phi(\mathbf{x})$ on the inputs such that $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$

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 $\mathcal{Y}$ form the identity, that is,

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

Structured Output Learning with Kernels

- Assume $f(\mathbf{y}|\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$, where $\mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \in \mathcal{F}$
- Use ℓ_2 regularizer: $\mathbf{P}[f] = \|w\|^2$

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{F}, \boldsymbol{\xi} \in \mathbb{R}^N} \quad & C \sum_{n=1}^{N} \xi_n + \|w\|^2 \\ \text{w.r.t.} \quad & \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}_n) - \Phi(\mathbf{x}, \mathbf{y}) \rangle \geq 1 - \xi_n \\ & \text{for all } \mathbf{y}_n \neq \mathbf{y} \in \mathcal{Y}^*, n = 1, \ldots, N \end{aligned}$$

- Linear classifier that separates true from false labeling

Special Case: Only Two "Structures"

- Assume $f(\mathbf{y}|\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$, where $\mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \in \mathcal{F}$

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{F}, \boldsymbol{\xi} \in \mathbb{R}^N} \quad & C \sum_{n=1}^{N} \xi_n + \|w\|^2 \\ \text{w.r.t.} \quad & \langle \mathbf{w}, \Phi(\mathbf{x}, y_n) - \Phi(\mathbf{x}, 1 - y_n) \rangle \geq 1 - \xi_n \\ & \text{for all } n = 1, \ldots, N \end{aligned}$$

Exercise: Show that it is equivalent to standard 2-class SVM for appropriate values of Φ

Optimization

- Optimization problem too big (dual as well)

$$\begin{aligned} \min_{\mathbf{w}\in\mathcal{F},\boldsymbol{\xi}} \quad & C\sum_{n=1}^{N}\xi_n + \|w\|^2 \\ \text{w.r.t.} \quad & \langle \mathbf{w}, \Phi(\mathbf{x},\mathbf{y}_n) - \Phi(\mathbf{x},\mathbf{y})\rangle \geq 1-\xi_n \\ & \text{for all } \mathbf{y}_n \neq \mathbf{y} \in \mathcal{Y}^*, n = 1,\ldots,N \end{aligned}$$

- One constraint per example and wrong labeling
- Iterative solution
  - Begin with small set of wrong labelings
  - Solve reduced optimization problem
  - Find labelings that violate constraints
  - Add constraints, resolve
- Guaranteed Convergence

How to Find Violated Constraints?

- Constraint

$$\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}_n) - \Phi(\mathbf{x}, \mathbf{y}) \rangle \geq 1 - \xi_n$$

- Find labeling $\mathbf{y}$ that maximizes

$$\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$$

- Use dynamic programming decoding

$$\mathbf{y} = \underset{\mathbf{y} \in \mathcal{Y}^*}{\operatorname{argmax}} \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$$

  (DP only works if Φ has a certain decomposition structure)
- If $\mathbf{y} = \mathbf{y}_n$, then compute second best labeling as well
- If constraint is violated, then add to optimization problem

A Structured Output Algorithm

1. $\mathcal{Y}_n^1 = \emptyset$, for $n = 1, \ldots, N$
2. Solve

$$(\mathbf{w}^t, \boldsymbol{\xi}^t) = \underset{\mathbf{w}\in\mathcal{F},\boldsymbol{\xi}}{\operatorname{argmin}} \quad C\sum_{n=1}^{N}\xi_n + \|w\|^2$$
$$\text{w.r.t.} \quad \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}_n) - \Phi(\mathbf{x}, \mathbf{y})\rangle \geq 1 - \xi_n$$
$$\text{for all } \mathbf{y}_n \neq \mathbf{y} \in \mathcal{Y}_n^t, n = 1, \ldots, N$$

3. Find violated constraints ($n = 1, \ldots, N$)

$$\mathbf{y}_n^t = \underset{\mathbf{y}_n\neq\mathbf{y}\in\mathcal{Y}^*}{\operatorname{argmax}} \langle \mathbf{w}^t, \Phi(\mathbf{x}, \mathbf{y})\rangle$$

   If $\langle \mathbf{w}^t, \Phi(\mathbf{x}, \mathbf{y}_n) - \Phi(\mathbf{x}, \mathbf{y}_n^t)\rangle < 1 - \xi_n^t$, set $\mathcal{Y}_n^{t+1} = \mathcal{Y}_n^t \cup \{\mathbf{y}_n^t\}$
4. If violated constraint exists then go to 2
5. Otherwise terminate $\Rightarrow$ optimal solution

Loss Functions

- So far, 0/1-loss with slacks: If $\mathbf{y} \neq \mathbf{y}$, then prediction is wrong, but it does not matter how wrong
- Introduce loss function on labelings $\ell(\mathbf{y}, \mathbf{y}')$, e.g.
  - How many segments are wrong or missing
  - How different are the segments, etc.
- Extend optimization problem (Margin rescaling):

$$\begin{aligned} \min_{\mathbf{w}\in\mathcal{F},\boldsymbol{\xi}} \quad & C\sum_{n=1}^{N}\xi_n + \|w\|^2 \\ \text{w.r.t.} \quad & \langle \mathbf{w}, \Phi(\mathbf{x}_n, \mathbf{y}_n) - \Phi(\mathbf{x}_n, \mathbf{y})\rangle \geq \ell(\mathbf{y}_n, \mathbf{y}) - \xi_n \\ & \text{for all } \mathbf{y}_n \neq \mathbf{y} \in \mathcal{Y}^*, n = 1, \ldots, N \end{aligned}$$

- Find violated constraints ($n = 1, \ldots, N$)

$$\mathbf{y}_n^t = \operatorname*{argmax}_{\mathbf{y}_n\neq\mathbf{y}\in\mathcal{Y}^*} \left(\langle \mathbf{w}^t, \Phi(\mathbf{x}_n, \mathbf{y})\rangle + \ell(\mathbf{y}, \mathbf{y}_n)\right)$$

Problems

- Optimization may require many iterations
- Number of variables increases linearly
- When using kernels, solving optimization problems can become infeasible
- Evaluation of $\langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$ in dynamic programming can be very expensive
  - Optimization and decoding become too expensive
- Approximation algorithms useful
- Decompose problem
  - First part uses kernels, can be pre-computed
  - Second part without kernels and only combines ingredients

Part V

Case Studies (Applications)

Overview: Case Studies (Applications)

- Transcript Start Site Recognition
  - Prior Knowledge
  - Setting up the SVM
- Computational Gene Finding
  - Motivation
  - Method
  - Results
- Optimized Spliced Alignments
  - Motivation
  - Alignment Algorithms
  - Learning Algorithm
  - Experiments
- Array-based Resequencing
  - Motivation
  - Polymorphism Detection
  - SNP Prediction Results

Human Transcript Start Site Recognition

BIOINFORMATICS

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

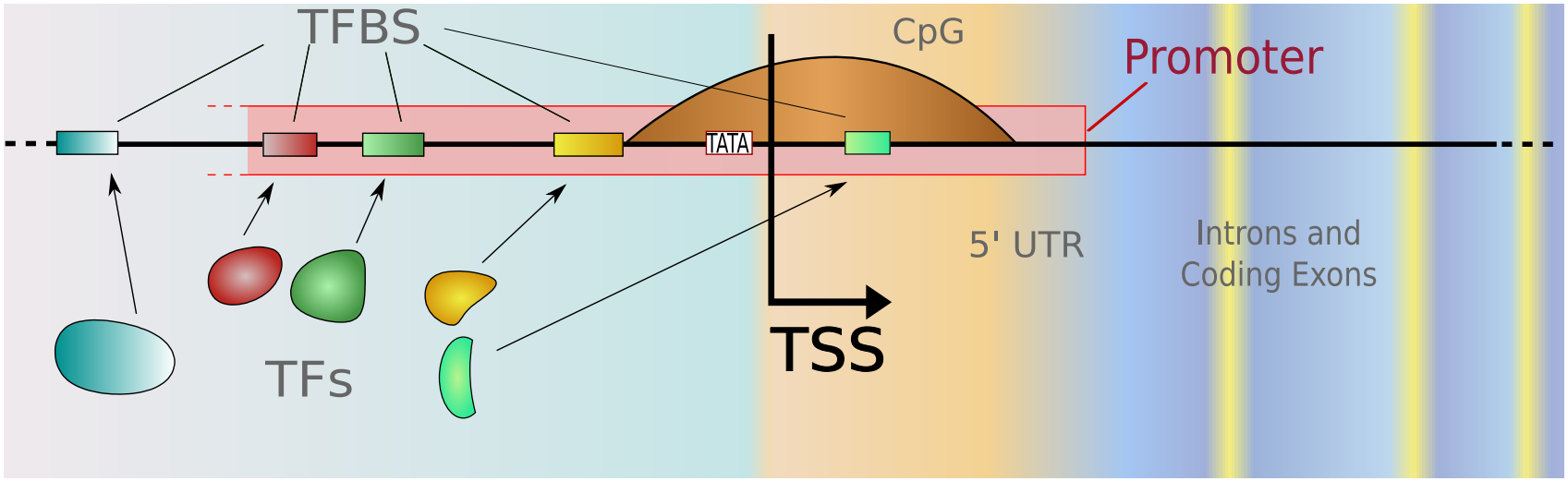
ARTS: accurate recognition of transcription starts in human

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

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

[Sonnenburg et al., 2006b]

Detecting Transcription Start Sites



- 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

$\Rightarrow$ **Promoter prediction is non-trivial**

Features to Describe a TSS

- TFBS in promoter region
- condition: DNA should not be too twisted
- CpG islands (often over TSS/first exon; in most, but not all promoters)
- TSS with TATA box (≈ -30 bp upstream)
- Exon content in UTR 5" region
- Distance to first donor splice site

Idea:

Combine weak features to build strong promoter predictor

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

The 5 Sub-kernels

1. TSS signal (including parts of core promoter with TATA box)

   – use **Weighted Degree Shift kernel**

2. CpG Islands, distant enhancers and TFBS upstream of TSS

   – use **Spectrum kernel** (large window upstream of TSS)

3. Model coding sequence TFBS downstream of TSS

   – use another **Spectrum kernel** (small window downstream of TSS)

4. Stacking energy of DNA

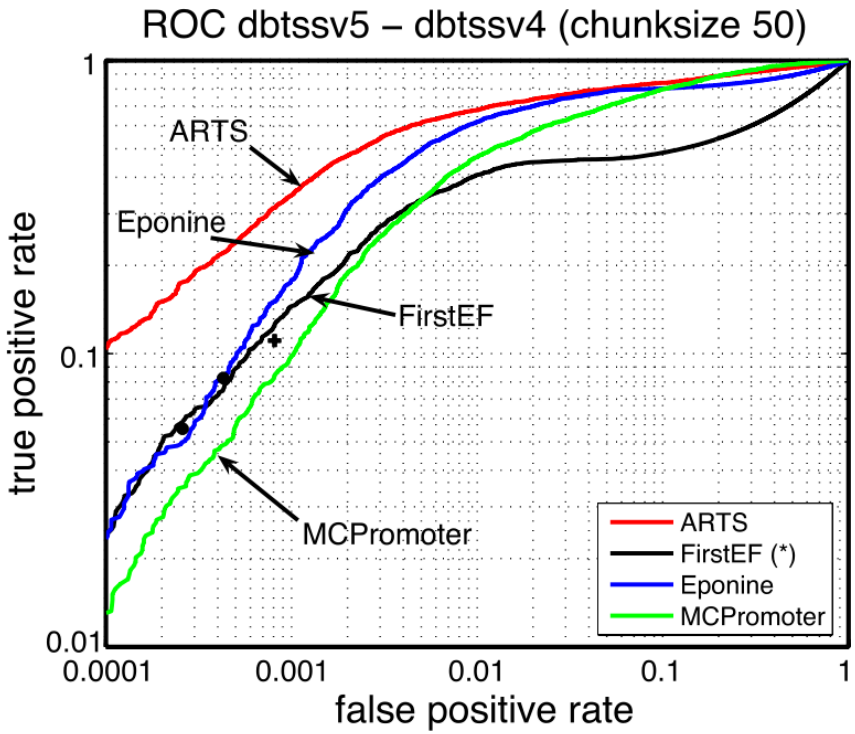
   – use *btwist* energy of dinucleotides with **Linear kernel**

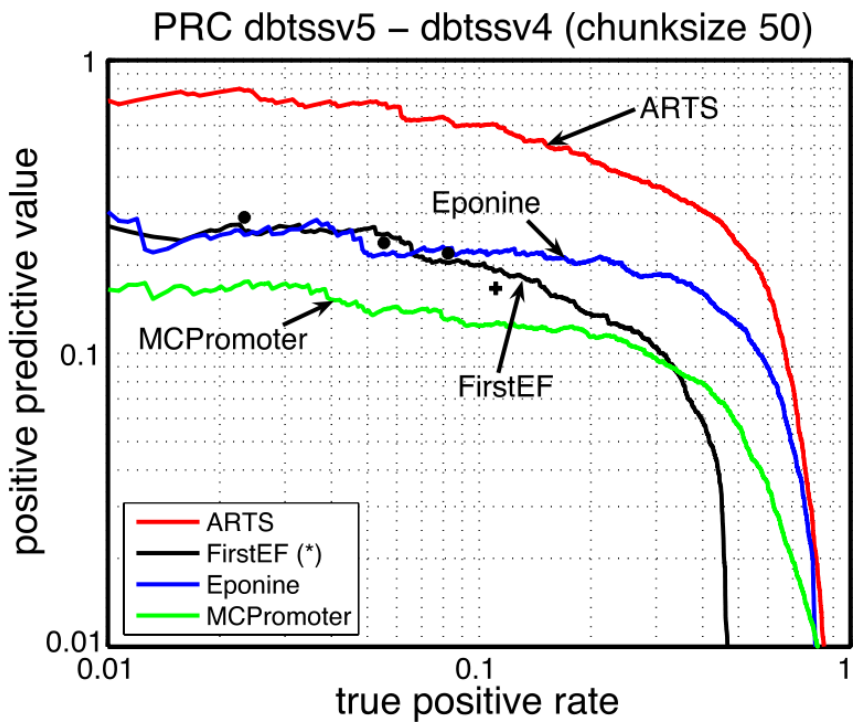
5. Twistedness of DNA

   – use *btwist* angle of dinucleotides with **Linear kernel**

State-of-the-art Performance

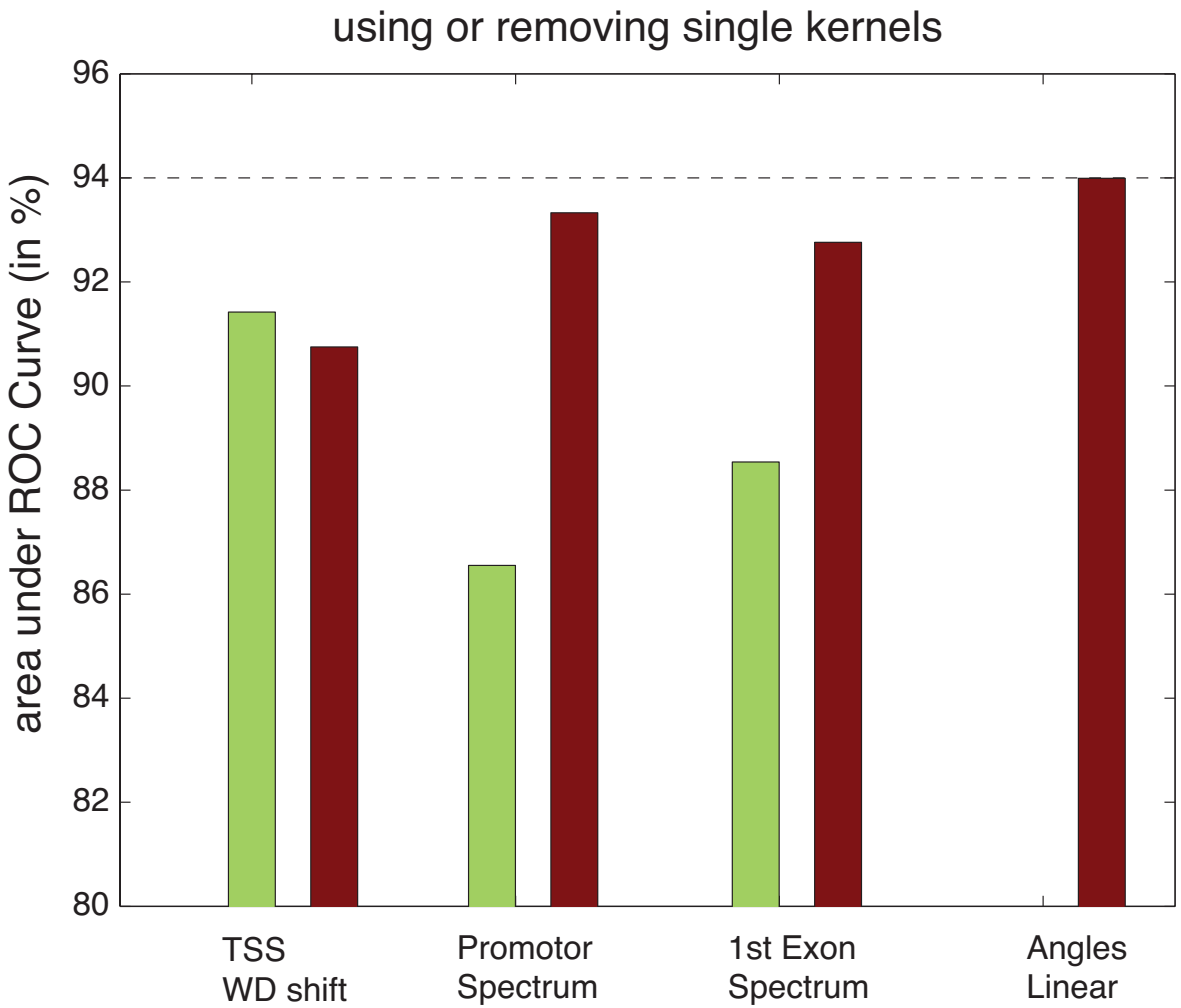
Receiver Operator Characteristic Curve and Precision Recall Curve





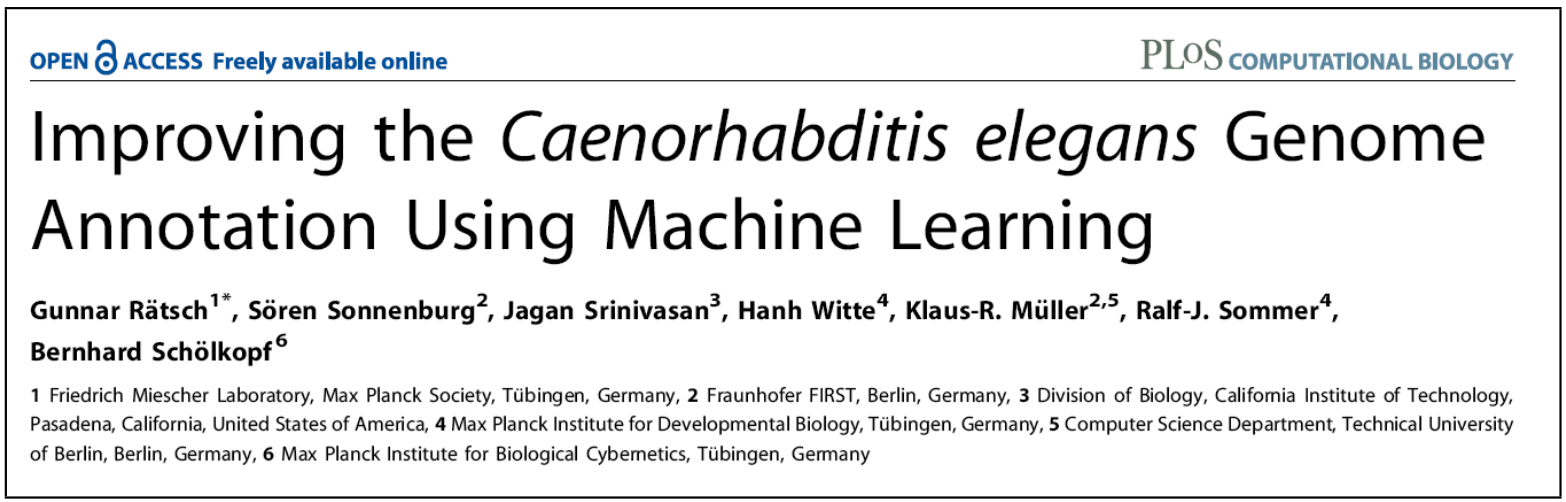
$\Rightarrow$ 35% **true positives at a false positive rate of** $1/1000$
(best other method finds about one half (18%**))**

Contributions of the Kernels



$\Rightarrow$ Most important: Weighted Degree Shift kernel modeling the TSS signal

Discovering Sequence Variations in *Arabidopsis thaliana*



OPEN ACCESS Freely available online

PLoS COMPUTATIONAL BIOLOGY

Improving the *Caenorhabditis elegans* Genome Annotation Using Machine Learning

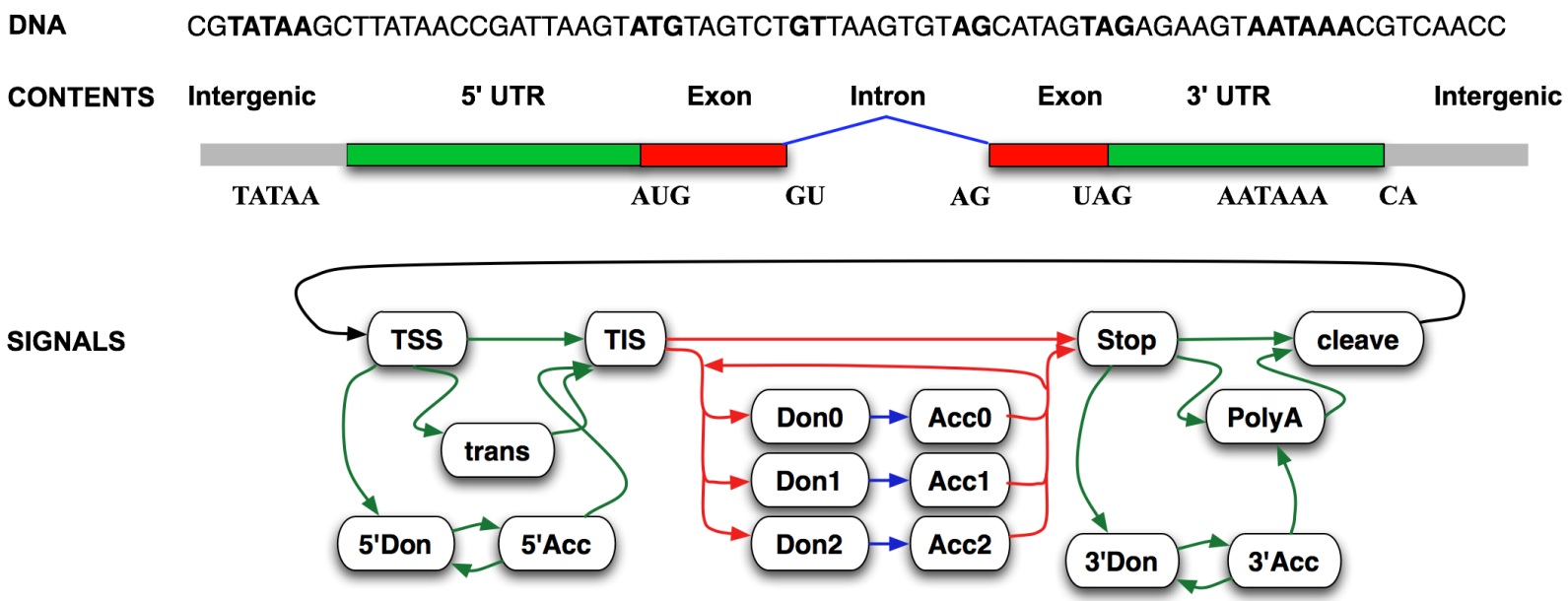
Gunnar Rätsch[1*], Sören Sonnenburg[2], Jagan Srinivasan[3], Hanh Witte[4], Klaus-R. Müller[2,5], Ralf-J. Sommer[4], Bernhard Schölkopf[6]

1 Friedrich Miescher Laboratory, Max Planck Society, Tübingen, Germany, **2** Fraunhofer FIRST, Berlin, Germany, **3** Division of Biology, California Institute of Technology, Pasadena, California, United States of America, **4** Max Planck Institute for Developmental Biology, Tübingen, Germany, **5** Computer Science Department, Technical University of Berlin, Berlin, Germany, **6** Max Planck Institute for Biological Cybernetics, Tübingen, Germany

[Rätsch et al., 2007]

Approach: Carefully Model Signals & Content

- States correspond to sequence signals
  - Depends on recognition of signals on the DNA
- Transitions correspond to segments
  - Model length and content of segment
- Markovian on segment level, non-Markovian within segments



Recognition of Signals and Content

Sensors to recognize signals:

- Transcription start and cleavage sites, polyA site
- Translation initiation site and stop codon
- Donor and acceptor splice sites

Distinguish true signal positions against all other positions

Sensors to recognize contents:

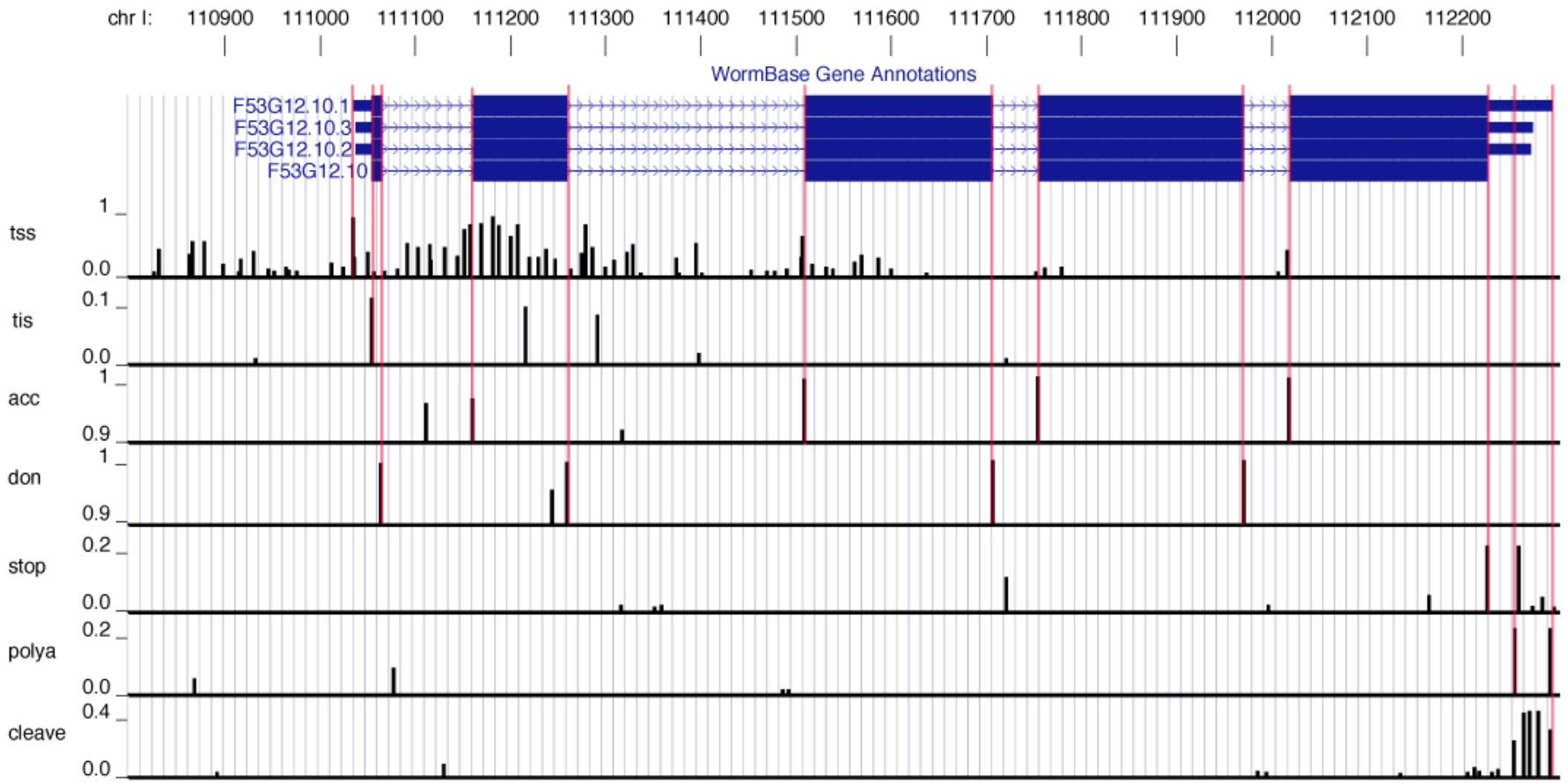
- Exons
- Introns
- Intergenic

Distinguish one content type from all others

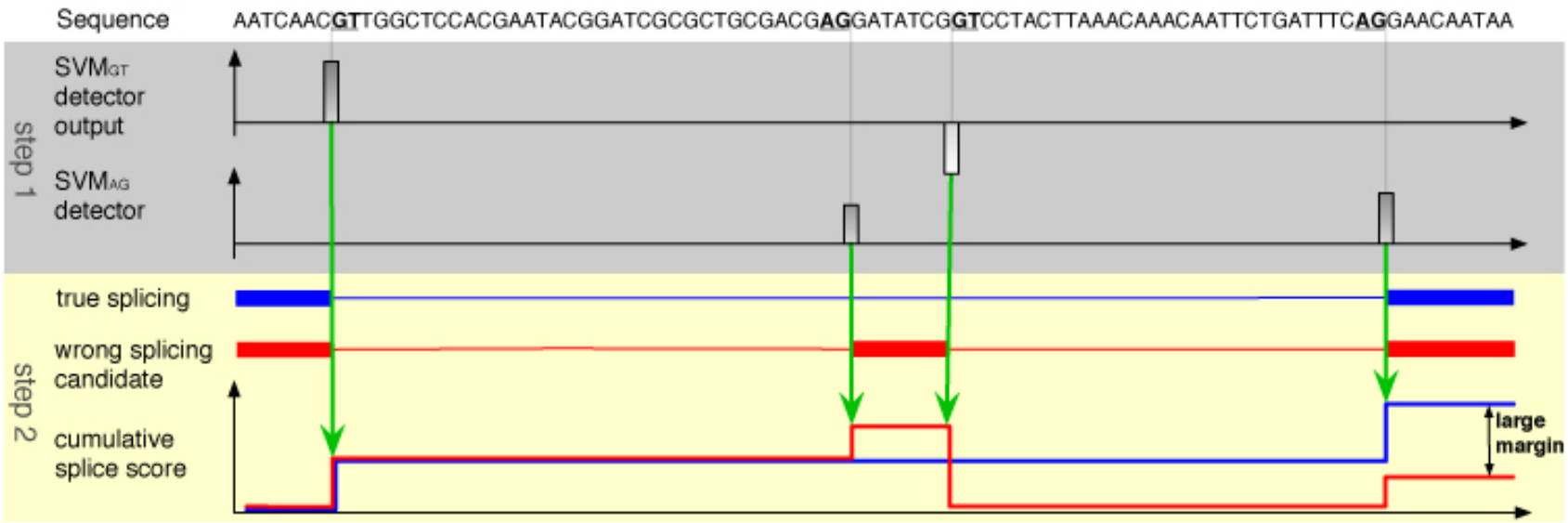
Typical approach: PWMs/PSSMs or higher-order Markov chains

Here: Support Vector Machines (SVMs)

Example: Predictions in UCSC Browser



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

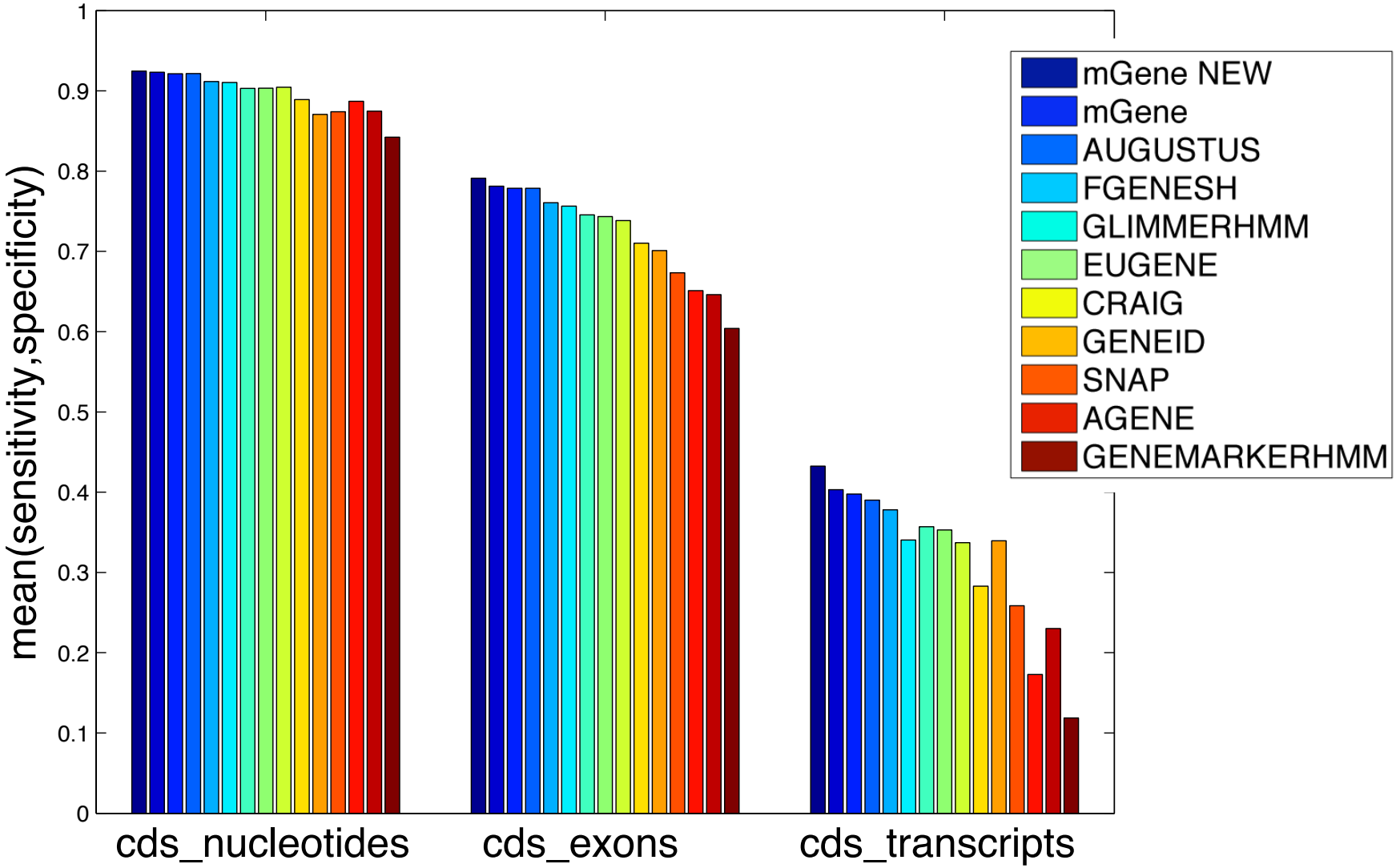
Results Summary

- Prediction of exon-intron structure only [Rätsch et al., PLoS Comp. Biol., 2007]
  - Considerable improvements compared to other methods
  - Analysis of 20 disagreeing cases:
    - 15 cases correctly predicted
    - annotation was never correct
  - Annotation available on `http://www.wormbase.org`
- Full gene predictions
  - Participation in nGASP competition
  - Preliminary evaluation

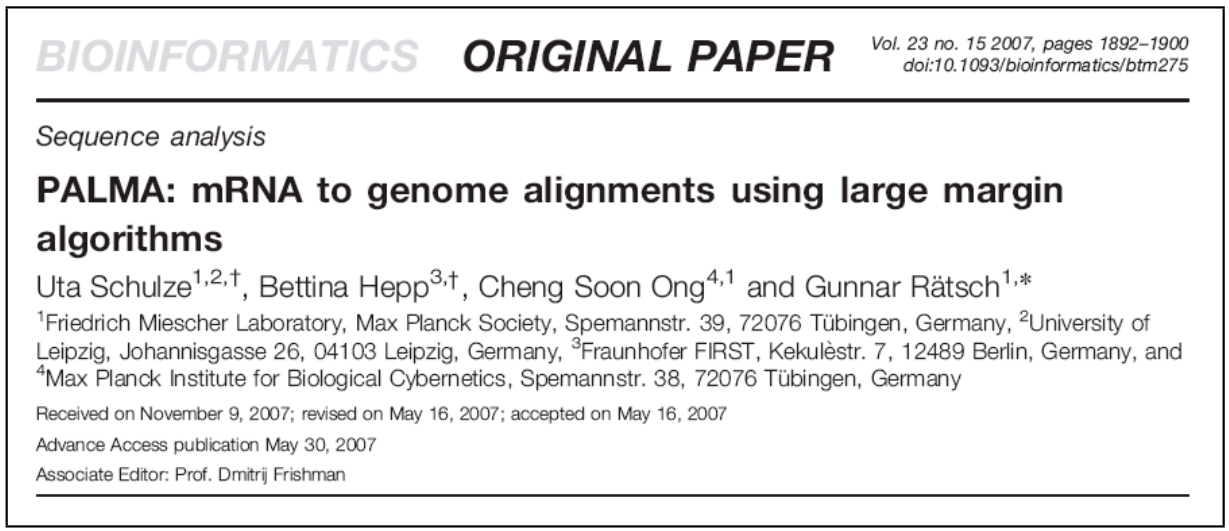
nGASP Competition

- Controlled competition conditions:
  - 10% for training methods
  - 10% for evaluation
- Phase I: single predictors
- Phase II: combining algorithms
- Four categories:
  - *Ab initio* gene finders
  - Dual-/multi-genome gene finders
  - Gene finders that use sequence alignments
  - Gene finders that use any of the above information
- Preliminary evaluation of *all* WS160 genes in test regions
  - Agrees with preliminary nGASP evaluation (see nGASP poster)

nGASP Category 1 Evaluation (prelim.)



Spliced Alignments Using Large Margins



BIOINFORMATICS ORIGINAL PAPER
Vol. 23 no. 15 2007, pages 1892–1900
doi:10.1093/bioinformatics/btm275

Sequence analysis

PALMA: mRNA to genome alignments using large margin algorithms

Uta Schulze[1,2,†], Bettina Hepp[3,†], Cheng Soon Ong[4,1] and Gunnar Rätsch[1,*]

[1]Friedrich Miescher Laboratory, Max Planck Society, Spemannstr. 39, 72076 Tübingen, Germany, [2]University of Leipzig, Johannisgasse 26, 04103 Leipzig, Germany, [3]Fraunhofer FIRST, Kekulèstr. 7, 12489 Berlin, Germany, and [4]Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany

Received on November 9, 2007; revised on May 16, 2007; accepted on May 16, 2007

Advance Access publication May 30, 2007

Associate Editor: Prof. Dmitrij Frishman

[Schulze et al., 2007]



BIOINFORMATICS
Vol. 24 ECCB 2008, pages i174–i180
doi:10.1093/bioinformatics/btn300

Optimal spliced alignments of short sequence reads

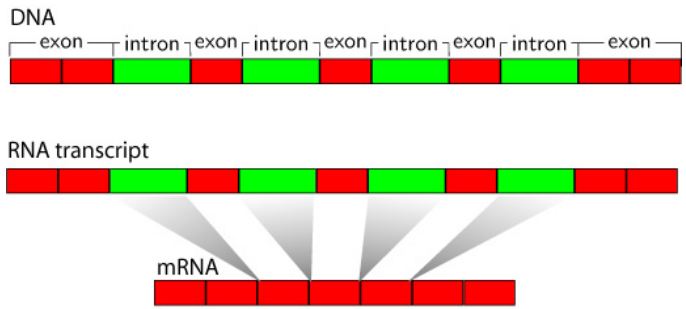
Fabio De Bona[1], Stephan Ossowski[2], Korbinian Schneeberger[2] and Gunnar Rätsch[1,*]

[1]Friedrich Miescher Laboratory, Max Planck Society, Spemannstr. 39, 72076 Tübingen and [2]Max Planck Institute for Developmental Biology, Spemannstr. 35, 72076 Tübingen, Germany

[De Bona et al., 2008]

Motivation & Background

- Abundant experimental data:
  - Expressed Sequence Tags (EST)
  - Full-length mRNAs
- Alignment to genomic sequences helps
  - Discovery of new genes
  - Delineation of exon/intron boundaries
  - Identification of alternative splice forms
  - Finding SNPs
  - ...
- Problems
  - Repetitive elements, paralogs, pseudo-genes
  - Sequencing errors, polymorphisms
  - Non-canonical splice sites
  - Microexons



Previous Work

More than 10 years of research on spliced alignments

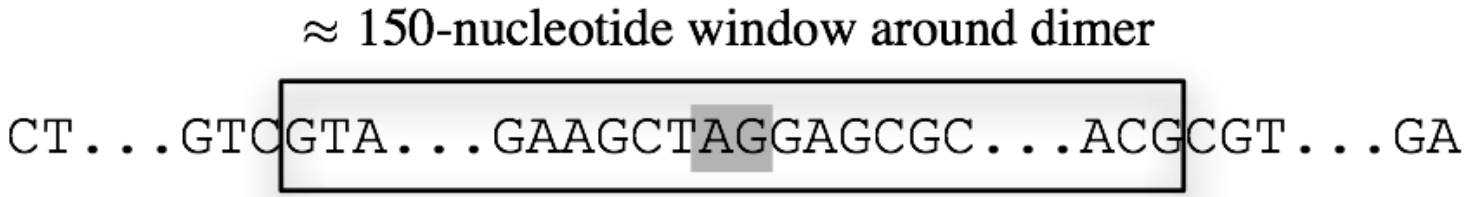
- Greedy algorithms (extend seed words or `BLAST`-based)
  - `Sim4` [Florea et al., 1998], `Spidey` [Wheelan et al., 2001]
  - `BLAT` (prefers `AG`/`GT`) [Kent, 2002]
- `EST_Genome` (DP-based, prefers `AG`/`GS`) [Mott, 1997]
- `Exalin` (DP-based, `AG`/`GS` only) [Zhang and Gish, 2006]
  - Fixed substitution and gap costs
  - Splice site model (PWMs)

  } Maximum likelihood combination

Why another tool?

- More accurate splice site models (SVM-based)
- Intron length model
- Combinations that are based on sounder principles (based on large margins)

2-Class Splice Site Detection



- True sites: fixed window around a true splice site
- Decoys sites: generated by shifting the window

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

⇒ Very unbalanced problem (1:200)
⇒ Millions of points from EST databases
⇒ Large-scale methods necessary

Alignment Algorithms

Input

- Two sequences over the alphabet $\{A, C, G, T, N\}$
  - EST sequence S_E of length m
  - DNA sequence S_D of length n
- Substitution matrix $M : \Sigma \times \Sigma \to \mathbb{R}$,
  where $\Sigma := \{A, C, G, T, N, -\}$

Output

- Sequence alignment $\mathcal{A}$
  - Sequence of pairs, i.e. $\mathcal{A} = (a_r, b_r)_{r=1,\ldots,R}$, $a_r, b_r \in \Sigma$
  - $R \leq m + n$ depends on the alignment
- Alignment that maximizes the alignment score

$$s(\mathcal{A}) = \sum_{r=1}^{R} M(a_r, b_r)$$

Maximizing the Alignment Score

Needleman-Wunsch Algorithm

- Maximizes alignment score by dynamic programming
- Fills $m \cdot n$ alignment matrix V:
  - $V(i,0) := 0$ and $V(0,j) := 0$ for all i, j
  - Recursion

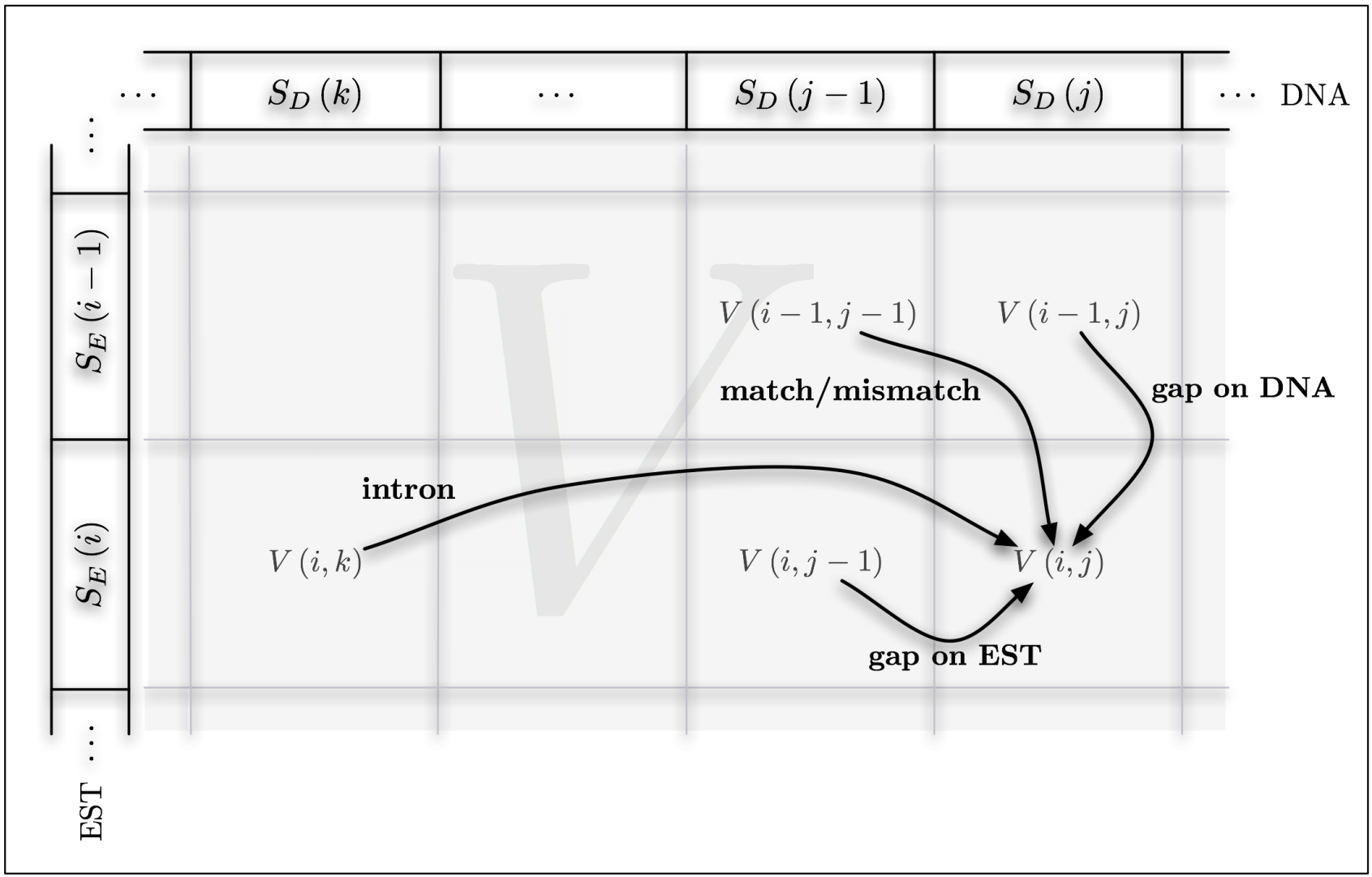
$$V(i,j) = \max \begin{cases} V(i-1,j-1) + M(S_E(i), S_D(j)) \\ V(i-1,j) + M(S_E(i), '-') \\ V(i,j-1) + M('-', S_D(j)) \end{cases}$$

- Runtime and space complexity: $\mathcal{O}(m \cdot n)$

Problems:

- Does not distinguish between gaps and introns
- How to choose M? No splice site model!
- Too expensive for alignments of whole genomes

Needleman-Wunsch Algorithm with Introns



Recursion with Intron Model

- Extended recursion formula ($\forall i = 1 \ldots m, j = 1 \ldots n$)

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + M(S_E(i), S_D(j)) \\ V(i-1,j) + M(S_E(i), '-') \\ V(i,j-1) + M('-', S_D(j)) \\ \max_{1 \le k \le j-1} (V(i,k) + f_I(k,j)) \end{cases}$$

- For intron score $f_I(k,j)$, consider:
  - Splice sites scores s_k^{Don} and s_k^{Acc} (SVM predictions)
    $\Rightarrow$ contribute $f^{\mathrm{Don}}(s_k^{\mathrm{Don}}) + f^{\mathrm{Acc}}(s_k^{\mathrm{Acc}})$
  - Length of intron
    $\Rightarrow$ contributes $f^{\mathrm{Len}}(j-k)$
- Unspecified functions $f^{\mathrm{Don}}, f^{\mathrm{Acc}}, f^{\mathrm{Len}}$ as well as M!

Idea: Learn functions on training set with known alignments

Parameterization

- Substitution matrix $M : \Sigma \times \Sigma \to \mathbb{R}$
- Functions f^{Len}, f^{Acc} and f^{Don}
  - Piecewise linear functions (support points $x_1, \ldots, x_s$):

$$f(x) = \begin{cases} \theta_1 & x \leq x_1 \\ \frac{\theta_i(x_{i+1}-x)+\theta_{i+1}(x-x_i)}{x_{i+1}-x_i} & x_i \leq x \leq x_{i+1} \\ \theta_s & x \geq x_s \end{cases} .$$

  - $\boldsymbol{\theta} := (\theta_1, \ldots, \theta_s)$ parametrizes function
- Let $\boldsymbol{\theta} := (\boldsymbol{\theta}^{\mathrm{Acc}}, \boldsymbol{\theta}^{\mathrm{Don}}, \boldsymbol{\theta}^{\mathrm{Len}}, \boldsymbol{\theta}^{M})$
- Given $\boldsymbol{\theta}$, alignment score $s_{\boldsymbol{\theta}}(\mathcal{A})$ is fully defined

Parameters: Optimization

Idea

Find $\boldsymbol{\theta}$ such that for a known alignment $\mathcal{A}^+$

$$s_{\boldsymbol{\theta}}(\mathcal{A}^+) \gg s_{\boldsymbol{\theta}}(\mathcal{A}^-)$$

where $\mathcal{A}^- \neq \mathcal{A}^+$ is any wrong alignment

Given N known alignments $\mathcal{A}_i^+, i = 1, \ldots, N$

solve quadratic optimization problem (QP)

$$\begin{aligned} \min_{\boldsymbol{\xi} \geq \mathbf{0}, \boldsymbol{\theta}} \quad & \frac{1}{N} \sum_{i=1}^{N} \xi_i + \mathbf{P}(\boldsymbol{\theta}) \\ \text{s.t.} \quad & s_{\boldsymbol{\theta}}(\mathcal{A}_i^+) - s_{\boldsymbol{\theta}}(\mathcal{A}_i^-) \geq 1 - \xi_i \quad \forall \mathcal{A}_i^- \neq \mathcal{A}_i^+, i = 1, \ldots, N \end{aligned}$$

- ξ_i: Slack variables to implement a soft-margin
- $\mathbf{P}(\boldsymbol{\theta})$: Regularizer leading to smooth functions

Iterative Algorithm

- Set $\boldsymbol{\theta} := (\boldsymbol{\theta}^{\mathrm{Acc}}, \boldsymbol{\theta}^{\mathrm{Don}}, \boldsymbol{\theta}^{\mathrm{Len}}, \boldsymbol{\theta}^{M})$ randomly, $A_i^- = \emptyset$
- For $t = 1, \ldots, T$
  - For $i = 1, \ldots, N$
    - Compute (wrong) alignments $\mathcal{A}^-$ based on $\boldsymbol{\theta}$
    - If $\mathcal{A}^- \neq \mathcal{A}_i^+$, then $A_i^- := A_i^- \cup \{\mathcal{A}^-\}$
  - Obtain new parameters $\boldsymbol{\theta}$ by solving the restricted QP

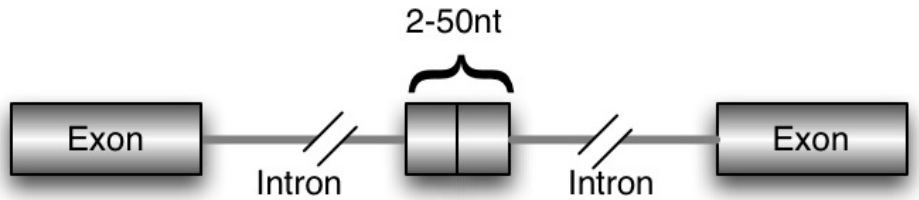
$$\begin{aligned} \min_{\boldsymbol{\xi} \geq \mathbf{0}, \boldsymbol{\theta}} \quad & \frac{1}{N} \sum_{i=1}^{N} \xi_i + \mathbf{P}(\boldsymbol{\theta}) \\ \text{s.t.} \quad & s_{\boldsymbol{\theta}}(\mathcal{A}_i^+) - s_{\boldsymbol{\theta}}(\mathcal{A}^-) \geq 1 - \xi_i \quad \forall \mathcal{A}^- \in A_i^-, i = 1, \ldots, N \end{aligned}$$

- Only need to solve small optimization problems!
- Guaranteed convergence!

Microexon Simulation Study

Artificial Data

- Consider EST-confirmed exon triples (*C. elegans*)
- Shorten middle exon in central region (EST and DNA)



(found by `BLAT` & strict post-proc.)

$\Rightarrow$ Microexon generated

$\Rightarrow$ Splice sites still intact

- Generate insertions/deletions/mutations in artificial EST ($\sigma = 0\%, 1\%, 2\%, 10\%, 20\%, 50\%$)
- Train `PALMA` on 4608 exon triples ($\approx 1h$)
- Test `BLAT`, `sim4`, `exalin` and `PALMA` on 4358 triples
- Correct only if all boundaries are correctly predicted

Conclusion: Alignment Algorithm

- Alignment algorithm for *accurate* alignments of mRNA and DNA
- Exploits very accurate SVM-based splice site predictions
- New idea of combining different sources of information:
  - Similarity, splice site scores and intron lengths
  - Large margin based iterative algorithm
  - Guaranteed convergence
- Significantly reduced error rates (short exons/much noise)
- Better detection of microexons & altern. spliced exons
- Current work: Reduce computational complexity
- Source code (Python/C++, GPL) and data available at
  `http://www.fml.mpg.de/raetsch/projects/palma`
  `http://www.fml.mpg.de/raetsch/projects/qpalma`

Discovering Sequence Variations in *Arabidopsis thaliana*

RESEARCH ARTICLE

Common Sequence Polymorphisms Shaping Genetic Diversity in *Arabidopsis thaliana*

Richard M. Clark,[1] Gabriele Schweikert,[1,2,3]* Christopher Toomajian,[4]* Stephan Ossowski,[1]* Georg Zeller,[1,2,5]* Paul Shinn,[6] Norman Warthmann,[1] Tina T. Hu,[4] Glenn Fu,[7] David A. Hinds,[7] Huaming Chen,[6] Kelly A. Frazer,[7] Daniel H. Huson,[5] Bernhard Schölkopf,[3] Magnus Nordborg,[4] Gunnar Rätsch,[2] Joseph R. Ecker,[6,8] Detlef Weigel[1,8]†

[Clark et al., 2007]

Methods

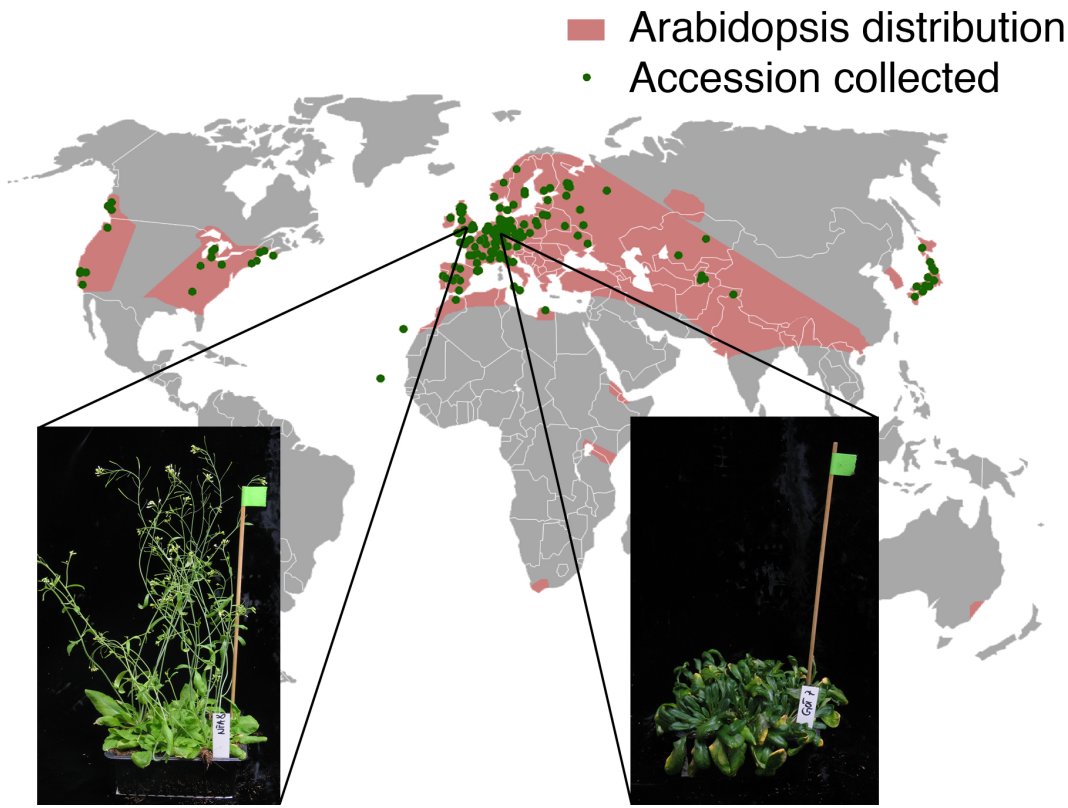
Detecting polymorphic regions in *Arabidopsis thaliana* with resequencing microarrays

Georg Zeller,[1,2] Richard M. Clark,[2,3] Korbinian Schneeberger,[2,3] Anja Bohlen,[1] Detlef Weigel,[2] and Gunnar Rätsch[1,4]

[1]*Friedrich Miescher Laboratory of the Max Planck Society, Tübingen 72070, Germany;* [2]*Max Planck Institute for Developmental Biology, Department of Molecular Biology, Tübingen 72070, Germany*

[Zeller et al., 2008]

Analysis of Polymorphisms: Introduction



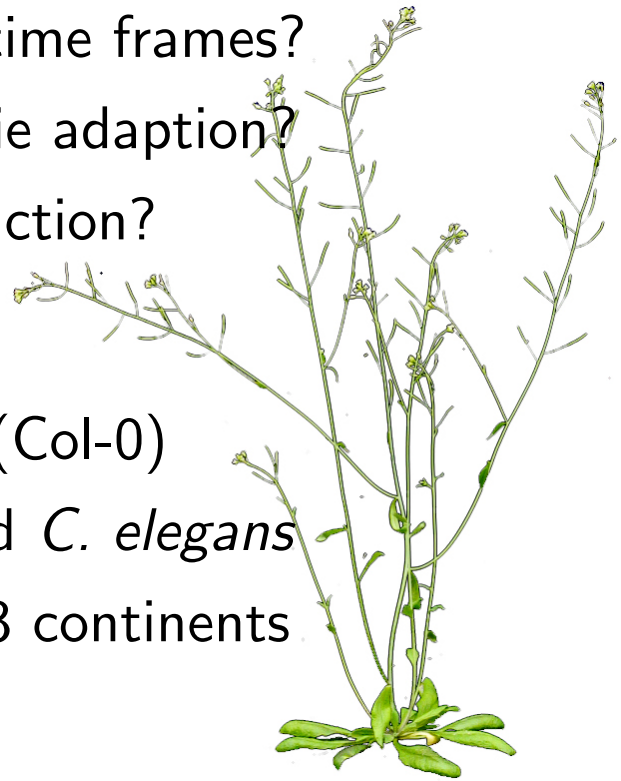
What is the genetic basis of variation?

Modified from Koornneef et al. 2004. *Ann. Rev. Plant Biology*, 55, 141-172.

Introduction

Questions:

- What sequence changes occur in short time frames?
- Which polymorphisms and genes underlie adaption?
- What are the consequences for gene function?



Arabidopsis thaliana:

- 119 Mb finished euchromatic sequence (Col-0)
- Resources comparable to *Drosophila* and *C. elegans*
- Collections of >1000 wild strains from 3 continents
- Strains are largely homozygous

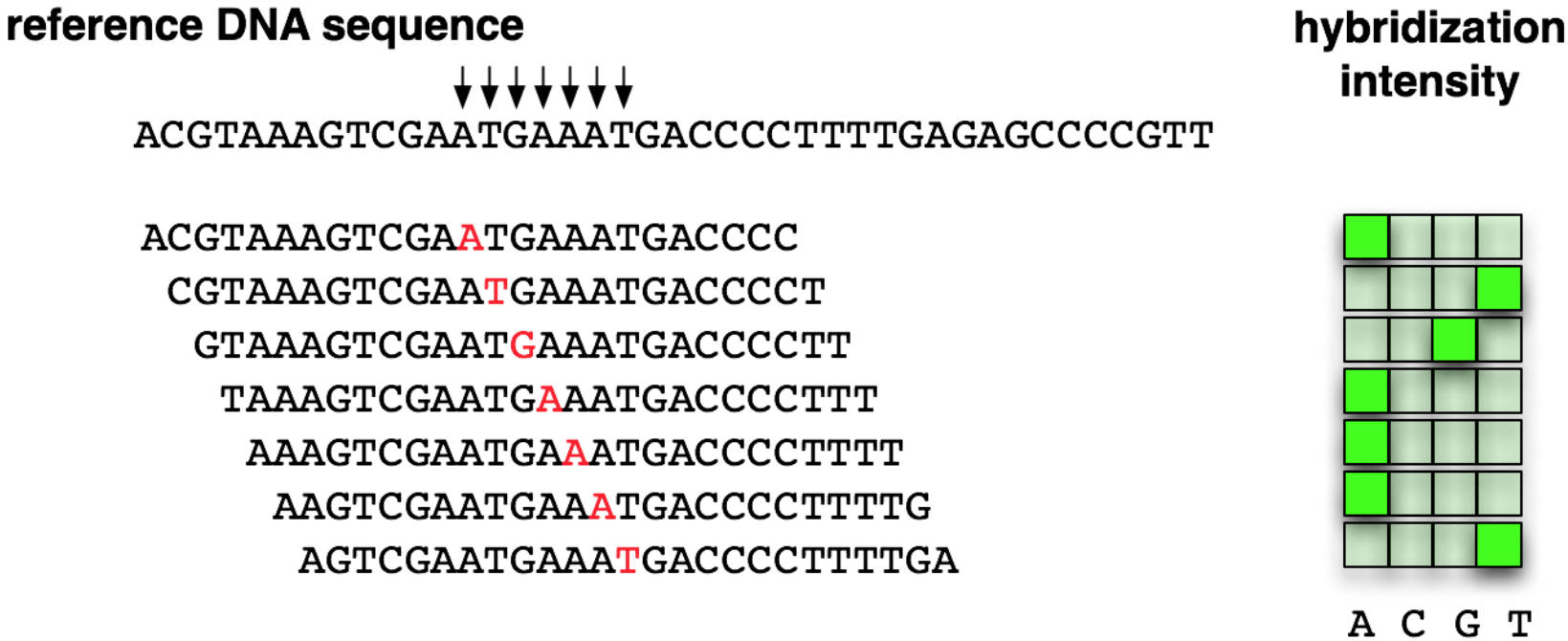
Resequencing of 20 wild strains

- Genome-wide identification of sequence polymorphisms
- High-density oligo-nucleotide arrays for high-throughput resequencing

Resequencing Array Basics I

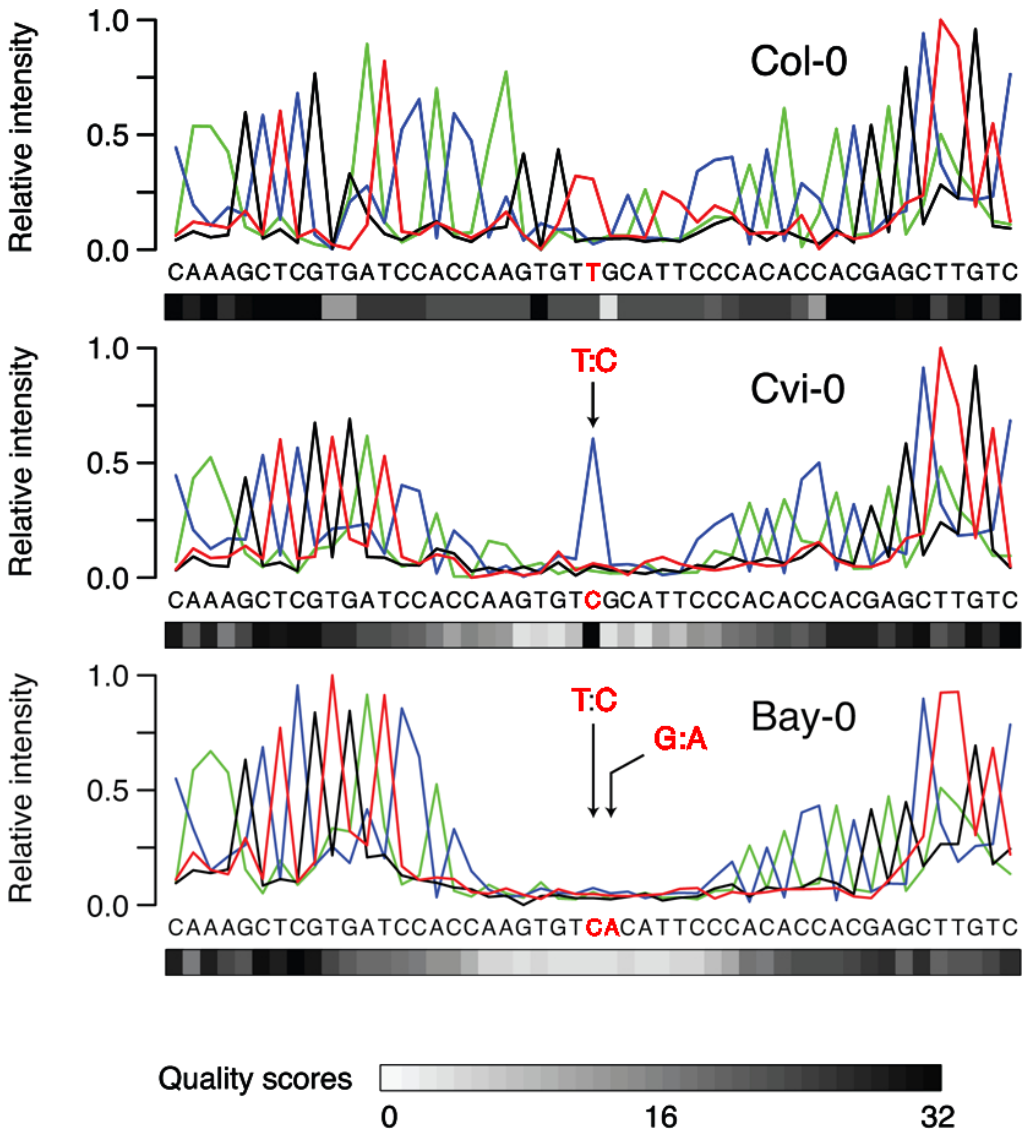


Resequencing Array Basics II



- >99.99% of bases represented
- Each base queried with forward and reverse strand probe quartets
- Nearly 1 billion oligos per accession
- 19+1 accessions surveyed representing worldwide distribution

Resequencing Data



Data analysis challenge

- Hybridization intensities depend on
  - Oligomer
  - Accession
  - Repeats
- Measurement noise
- Identify SNPs

Problematic cases

- Highly polymorphic regions
- Deletions/insertions

Labeled data and training

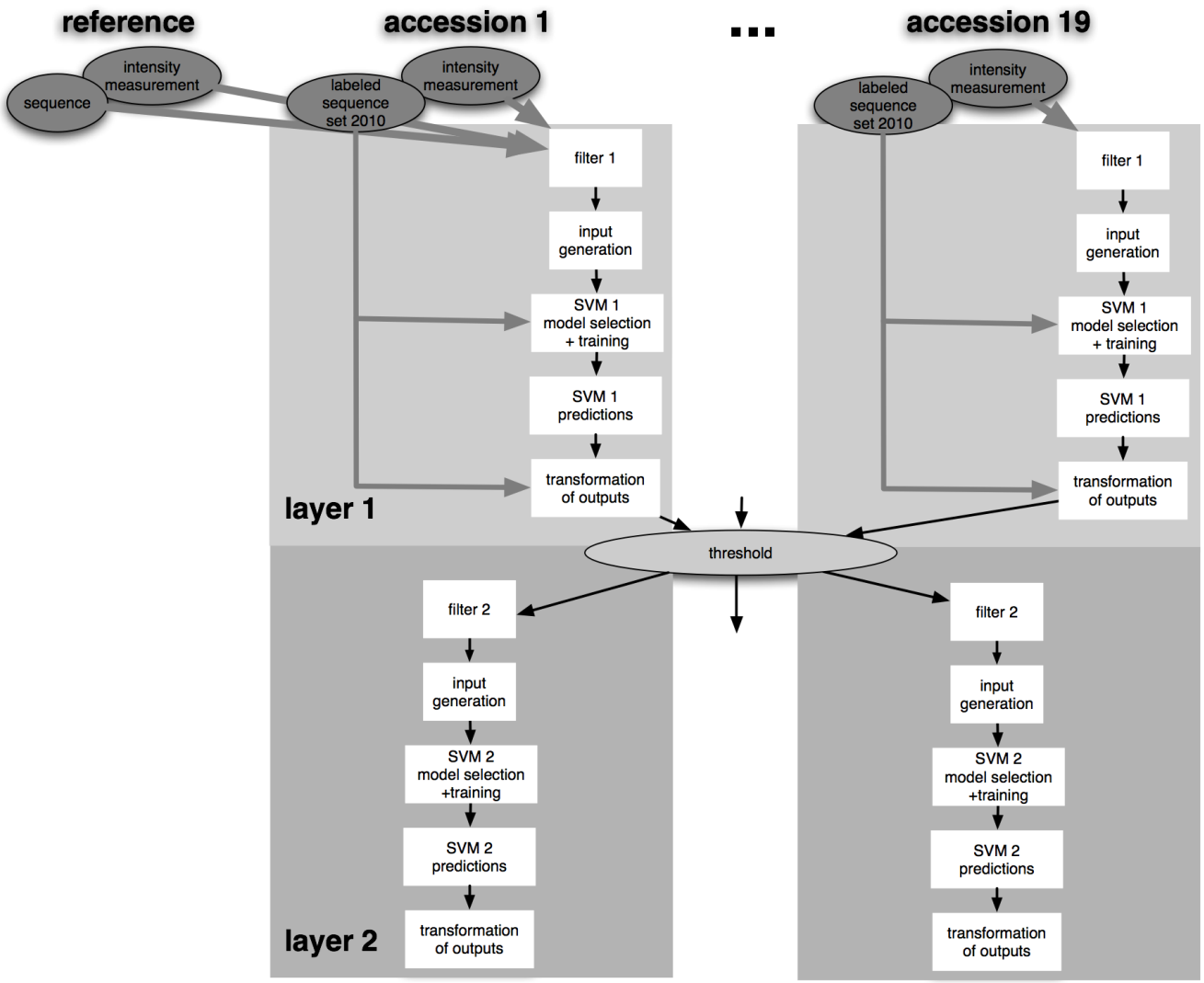
Labeled training set

- 1,213 fragments of length $\approx$ 550bp for 19 accessions
- sampled by PCR and dideoxy sequencing
- $\approx$2,700 known SNPs/accession (Nordborg et al., PLoS Biol., 2005)
- $\approx$400 indel polymorphisms/accession

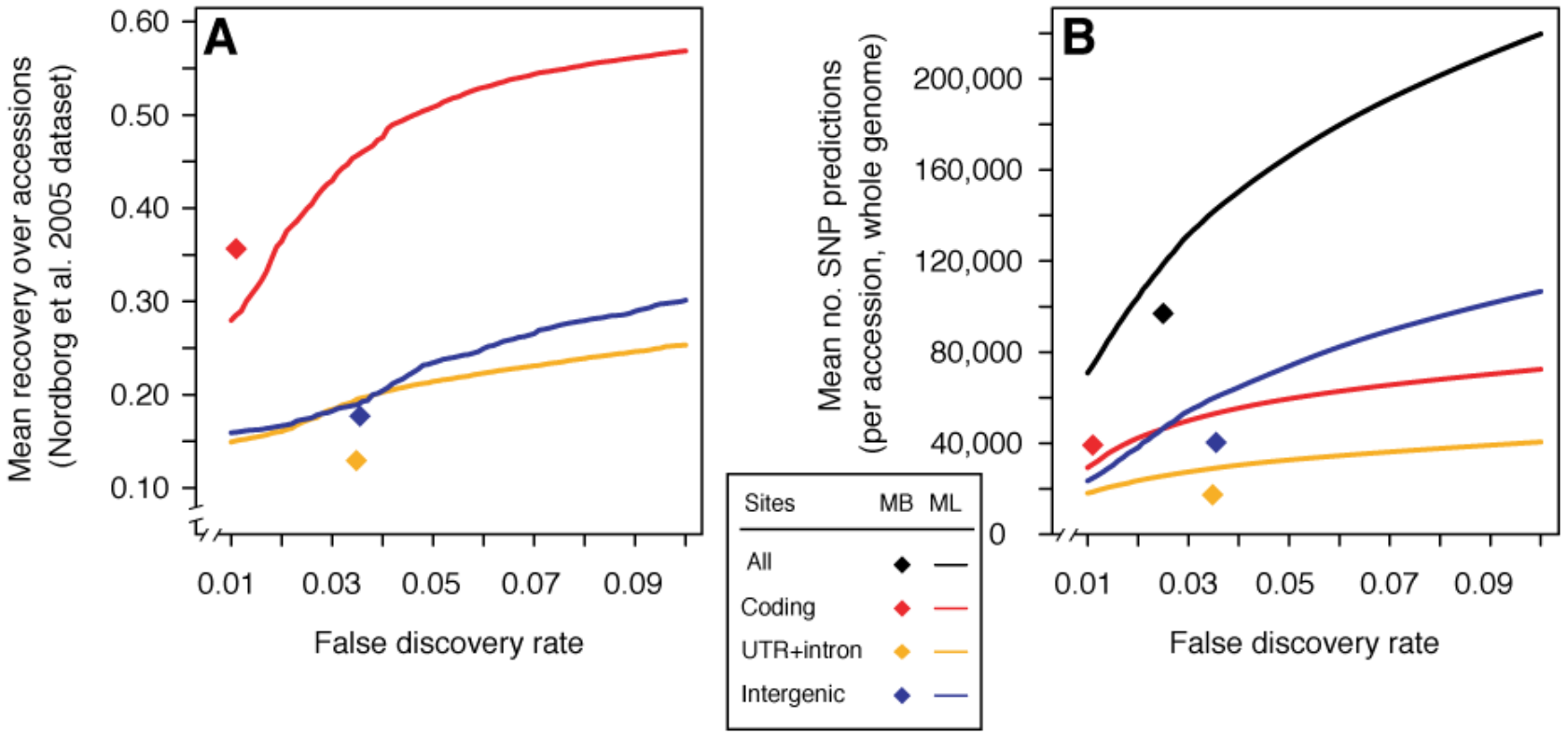
Training

- Classification using Support Vector Machines with 302 features
- two-layered approach; including cross-accesion features in second layer
- Out-of-sample evaluation and prediction on whole genome
- Comparison with Perlegen's model based method (Hinds et al., Science, 2005)

2-Layered Architecture for Inter-Strain Integration

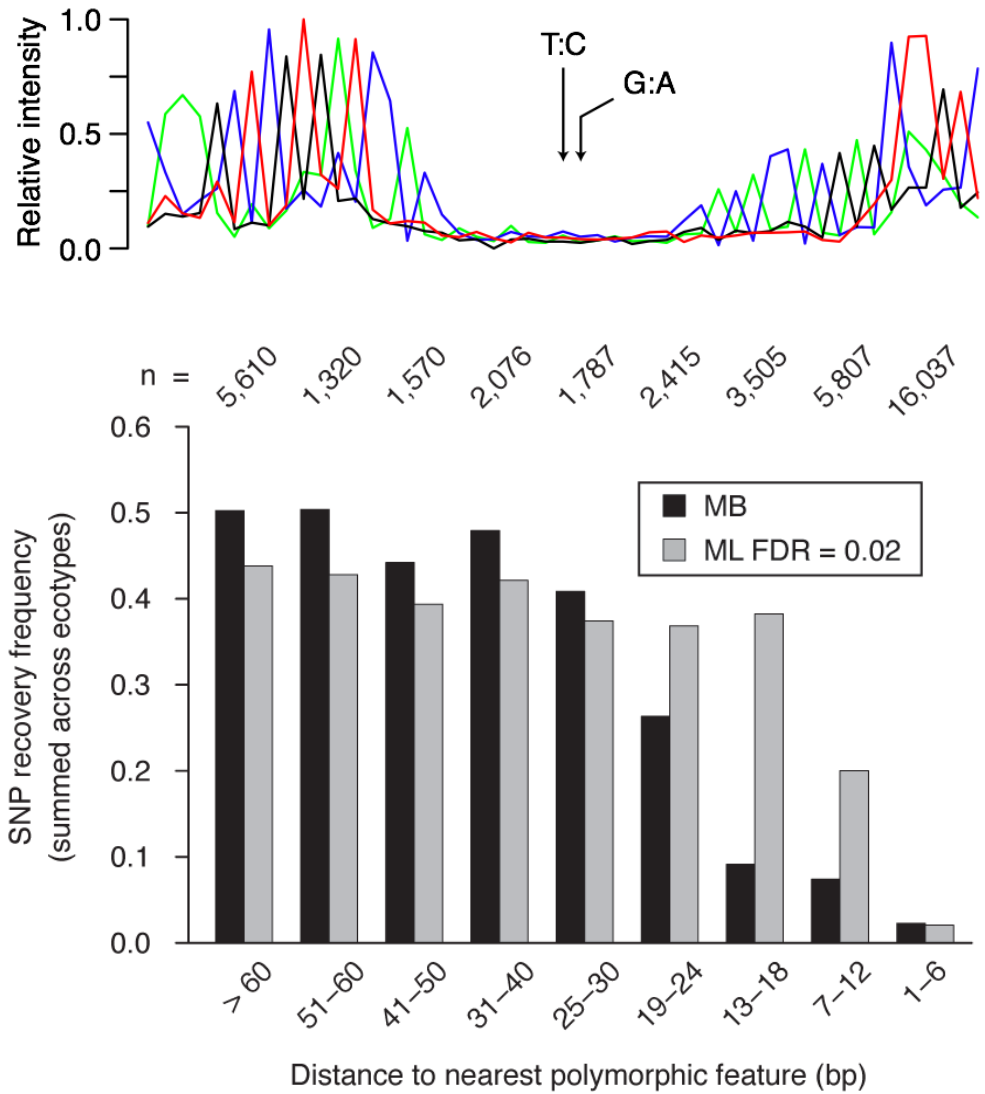


Application to SNP discovery



[Clark et al., 2007]

Identification of Highly Polymorphic Regions



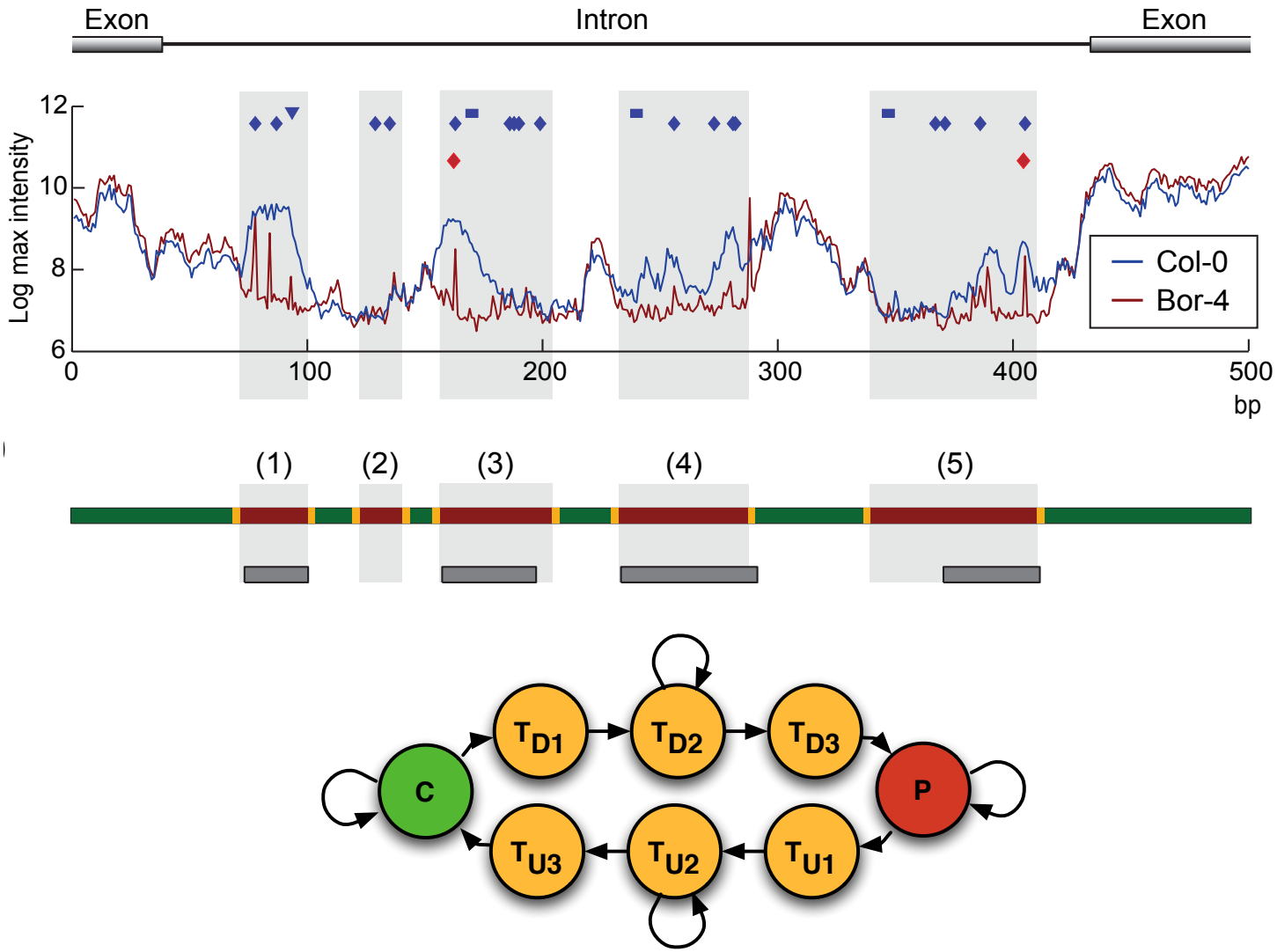
Results

- Performance drops, when other SNPs are in vicinity (1-20nt)
- Least predicted SNPs in highly polymorphic regions!
- ML more sensitive

New Approach

- **P**olymorphic **R**egion **P**rediction (PRP)
- Use HMSVM for segmenting the sequence

Modeling polymorphic regions



The Learning Problem

Given a sequence of observations (features) $x \in X$
We want to learn a function:

$$f : X \to Y$$

which yields a **label sequence (or path)** $\pi \in Y$
(of equal length: $|x| = |y|$).

Employ a function

$$F_\Theta : X \times Y \to \mathbb{R} \qquad \text{(path scoring)}$$

with which

$$f(x) = \arg\max_{\pi \in Y} F(x, \pi) \qquad \text{(Viterbi decoding)}$$

(Altun, ICML, 2003)

Evaluation

Count prediction as

- True positive (TP), if it overlaps by $\geq 75\%$
- False positive (FP), else.

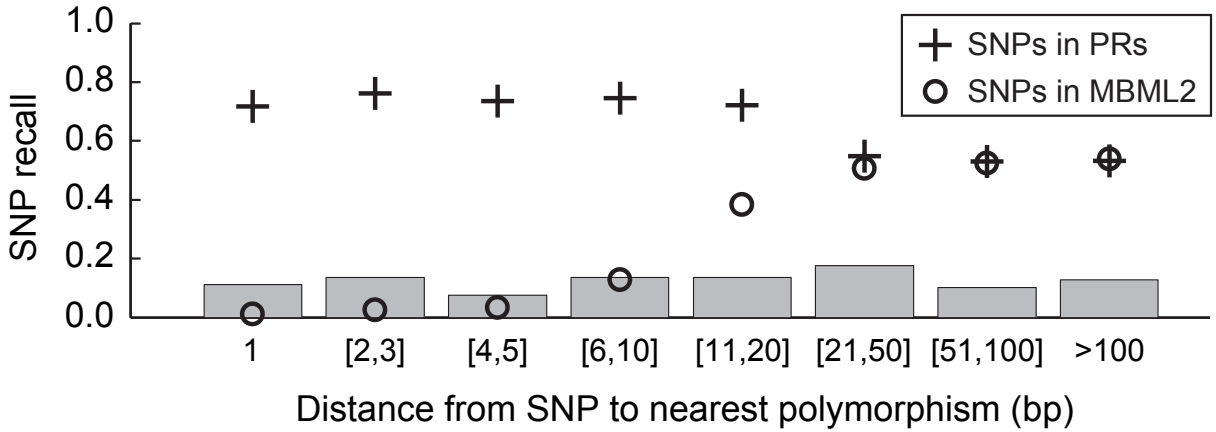
Count known polymorphic region as

- True discovery (TD), if polymorphisms covered, or $\geq 75\%$ included in prediction
- False negative (FN), else.

56% Sensitivity, 90% Specificity

[Zeller et al., 2008]

Complementing SNP Calls



Fraction of called/covered polymorphisms (test set):

	SNP calling (MB+ML)	Region predictor
SNPs	∼32%	∼65%
Deletions (per base)		∼53%
Insertions		∼39%

[Zeller et al., 2008]

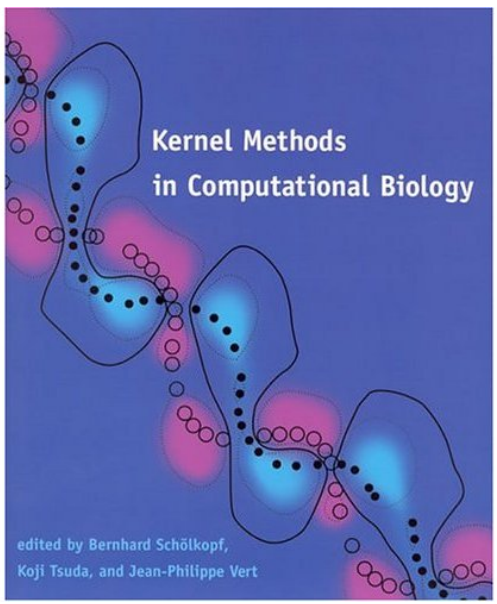
Part VI

Final Remarks

Kernel Methods for Computational Biology

[Schölkopf et al., 2004]

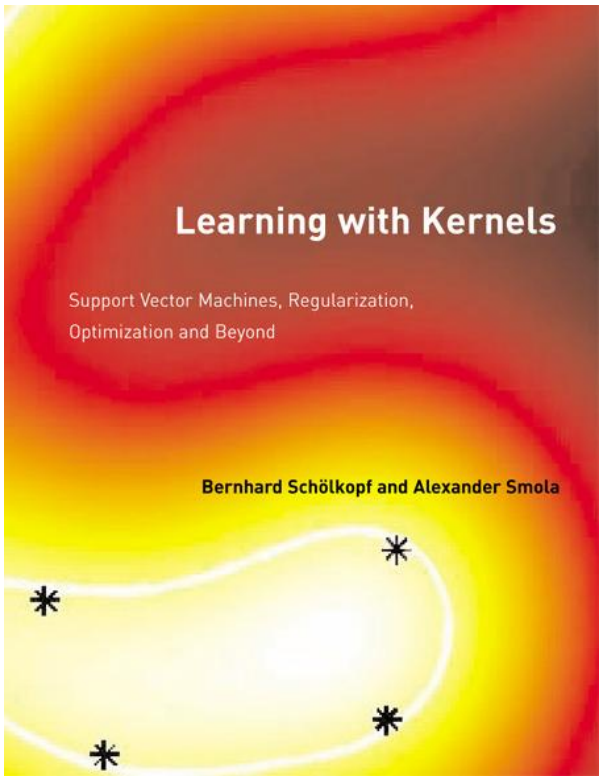
- **Kernel Methods in Computational Biology.**



- MIT Press, Aug. 2004
- by B. Schölkopf, K. Tsuda and J.P. Vert
- $\approx$ US$50

Learning with Kernels [Schölkopf and Smola, 2002]

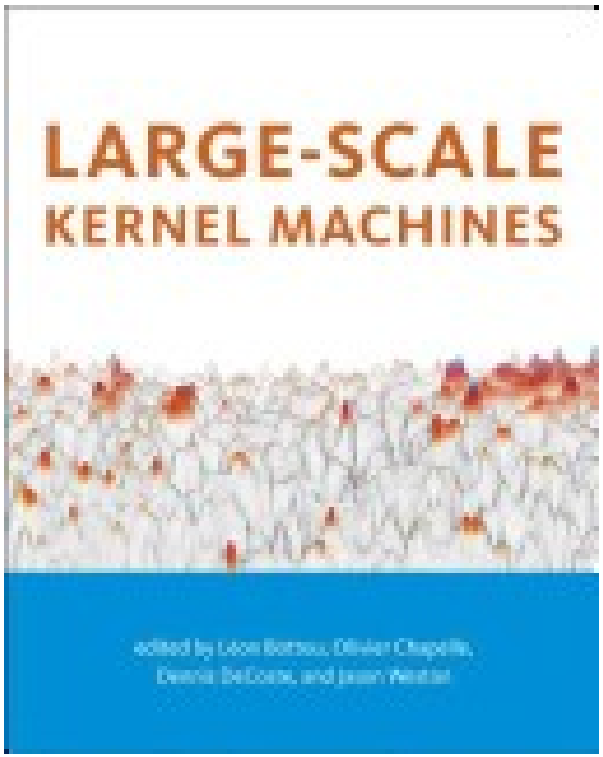
- **Kernel Learning Book.** `http://www.learning-with-kernels.org`



- MIT Press, Sept. 2002
- by B. Schölkopf and A. Smola
- $\approx$ US$54

Large-Scale Kernel Machines [Bottou et al., 2007]

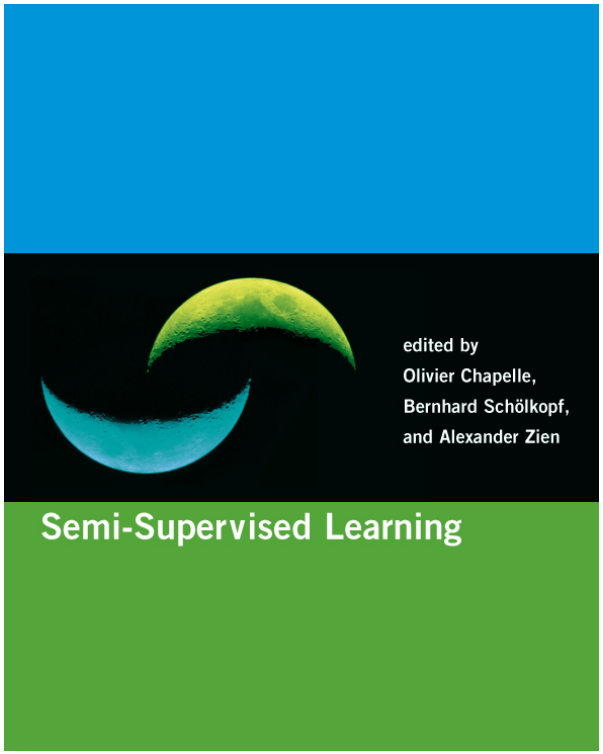
- **Large-Scale Learning Book** `http://mitpress.mit.edu/9780262026253`



- MIT Press, Sept. 2007
- by L. Bottou, O. Chapelle, D. Decoste, J. Weston
- $\approx$ US$36

Semi-Supervised Learning [B. Schölkopf, 2006; Zhu, 2008]

• **SSL Book.** `http://www.kyb.mpg.de/ssl-book`

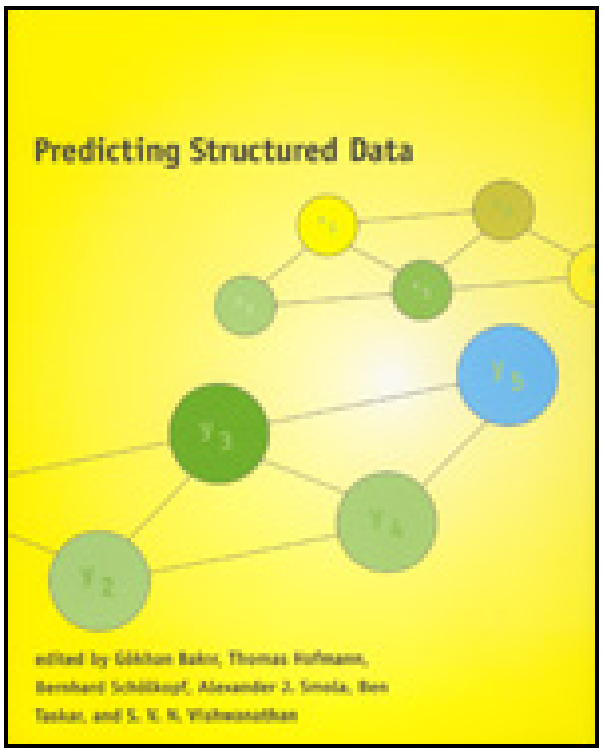


- MIT Press, Sept. 2006
- edited by B. Schölkopf, O. Chapelle, A. Zien
- $\approx$ US$37

• Xiaojin Zhu. Semi-Supervised Learning Literature Survey. TR 1530, U. Wisconsin.

Structured Output Learning [Bakir et al., 2007]

- **Predicting Structured Outputs.**



- MIT Press, Sept. 2007
- edited by G.H. Bakir, T. Hofmann, B. Schölkopf, A.J. Smola, B. Tasker, and S.V.N. Vishwanathan
- $\approx$ US$36

- Structured Output Learning tutorial in preparation for PLoS Computational Biology

Machine Learning Open Source Software



[Sonnenburg et al., 2007a]

Machine Learning Open Source Software

To support the open source movement JMLR is proud to announce a new track on machine learning open source software.

Contributions to http://jmlr.org/mloss/ should be related to

- implementations of machine learning algorithms,
- toolboxes,
- languages for scientific computing

and should include

- a 4 page description,
- the code,
- a recognised open source license.

Acknowledgements

We gratefully acknowledge the help in preparing the material by the following people:

Cheng Soon Ong (ETH Zürich), Petra Philips (European Patent Office), Karsten Borgwardt (Max Planck Campus, Tübingen), Peter Gehler (Max Planck Institute for Biological Cybernetics, Tübingen), Sebastian Schultheiss (Friedrich Miescher Laboratory, Tübingen), Marc Begin (Vancouver)

References I

A. Zien B. Schölkopf, O. Chapelle, editor. *Semi-Supervised Learning*. MIT Press, 2006.

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

G.H. Bakir, T. Hofmann, B. Schölkopf, A.J. Smola, B. Tasker, and S.V.N. Vishwanathan, editors. *Predicting Structured Outputs*. MIT Press, 2007.

S. Ben-david, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. *NIPS*, pages 137–144, 2007.

K.M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Proc. of International Conference on Data Mining (ICDM 2005)*, pages 74–81, Houston, USA, 2005.

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

References II

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.

L. Bottou, O. Chapelle, D. Decoste, and J. Weston, editors. *Large-Scale Kernel Machines*. MIT Press, 2007.

O. Chapelle, M. Chi, and A. Zien. A continuation method for semi-supervised svms. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 185–192, New York, NY, USA, 2006. ACM Press. URL `http://www.icml2006.org/icml2006/technical/accepted.html`.

R.M. Clark, G. Schweikert, C. Toomajian, S. Ossowski, G. Zeller, P. Shinn, N. Warthmann, T.T. Hu, G. Fu, D.A. Hinds, H. Chen, K.A. Frazer, D.H. Huson, B. Schölkopf, M. Nordborg, G. Rätsch, J.R. 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. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20: 273–297, 1995.

References III

F. De Bona, S. Ossowski, K. Schneeberger, and G. Rätsch. Optimal spliced alignments of short sequence reads. In *Bioinformatics/Proc. ECCB'08*. Oxford University Press, 2008. URL `http://www.fml.tuebingen.mpg.de/raetsch/projects/qpalma`.

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

T. Evgeniou and M. Pontil. Regularized multi-task learning. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117, 2004.

L. Fei-Fei and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 524–531. IEEE Computer Society, 2005. ISBN 0-7695-2372-2.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. In *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 12*, page 178, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2158-4.

References IV

L. Florea, G. Hartzell, Z. Zhang, G.M. Rubin, and W. Miller. A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Research*, 8:967–974, 1998.

T. Gärtner, P.A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proc. Annual Conf. Computational Learning Theory*. Springer, 2003.

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

S.S. Gross, O. Russakovsky, C.B. Do, and S. Batzoglou. Training conditional random fields for maximum labelwise accuracy. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, 2007.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, 1999.

T. Horváth, T. Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proc. KDD*, pages 158–167, 2004.

References V

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

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proc. Intl. Conf. Machine Learning*, Washington, DC, United States, 2003.

W. J. Kent. BLAT–the BLAST-like alignment tool. *Genome Res*, 12(4): 656–664, April 2002.

R. Kohavi and G.H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(97)00043-X.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proc. 18th Int. Conf. Mach. Learn.*, 2001.

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.

References VI

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2169–2178. IEEE Computer Society, 2006. ISBN 0-7695-2597-0. doi: http://dx.doi.org/10.1109/CVPR.2006.68.

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

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

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

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2: 419–444, 2002.

D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

References VII

P. Mahé, N. Ueda, T. Akutsu, Perret J.-L, and J.-P. Vert. Extensions of marginalized graph kernels. In *Proc. ICML*, 2004.

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

R. Mott. EST_GENOME: a program to align spliced DNA sequences to unspliced genomic DNA. *Comput. Appl. Biosci.*, 13:477–478, 1997.

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

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

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

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

References VIII

J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*, 2003.

G. Rätsch and S. Sonnenburg. Accurate splice site detection for *Caenorhabditis elegans*. In *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. Schölkopf. 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`.

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

B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, 2004.

References IX

S.J. Schultheiss, W. Busch, J.U. Lohmann, O. Kohlbacher, and G. Rätsch. Kirmes: Kernel-based identification of regulatory modules in euchromatic sequences. In *German Conference on Bioinformatics*, Lecture notes in Informatics, pages 158–167, Heidelberg, 2008. GI, Springer Verlag. URL `http://www.fml.tuebingen.mpg.de/raetsch/projects/kirmes`.

U. Schulze, B. Hepp, C.S. Ong, and G. Rätsch. Palma: mRNA to genome alignments using large margin algorithms. *Bioinformatics*, 23(15): 1892–1900, 2007. URL `http://bioinformatics.oxfordjournals.org/cgi/content/full/23/15/1892`. Project URL: http://www.fml.mpg.de/raetsch/projects/palma.

G. Schweikert, C. Widmer, B. Schölkopf, and G Rätsch. An Empirical Analysis of Domain Adaptation Algorithms for Genomic Sequence Analysis. *NIPS*, 2008.

N. Shervashidze, S.V.N. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proc. AISTATS*, 2009. Accepted.

A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 2001.

References X

S. Sonnenburg. New methods for splice site recognition. Master's thesis, Humbolt University Berlin, 2002.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large-Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, 7:1531–1565, July 2006a.

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

S. Sonnenburg, M.L Braun, C.S. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, Müller K.-R., F. Pereira, C.E. Rasmussen, G. Rätsch, B. Schölkopf, A. Smola, P. Vincent, J. Weston, and R.C. Williamson. The need for open source software in machine learning. *Journal of Machine Learning Research*, pages 2443–2466, 2007a.

S. Sonnenburg, G. Rätsch, and K. Rieck. Large-scale learning with string kernels. In *Large-Scale Kernel Machines*. MIT Press, 2007b.

S. Sonnenburg, A. Zien, P. Philips, and G. Rätsch. POIMs: positional oligomer importance matrices — understanding support vector machine-based signal detectors. *Bioinformatics*, 2008. (received the Outstanding Student Paper Award at ISMB 2008).

References XI

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. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

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 *Kernel Methods in Computational Biology*. MIT Press, 2004.

S.V.N. Vishwanathan, K.M. Borgwardt, and N. Schraudolph. Fast computation of graph kernels. In *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2006. MIT Press.

S.J. Wheelan, D.M. Church, and J.M. Ostell. Spidey: a tool for mRNA-to-genomic alignments. *Genome Research*, 11(11):1952–7, 2001.

References XII

Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. In *Advances in Neural Information Processing Systems*, 2009.

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

M. Zhang and W. Gish. Improved spliced alignment from an information theoretic approach. *Bioinformatics*, 22(1):13–20, January 2006.

X. Zhu. Semi-supervised learning literature survey. Technical Report TR1530, Computer Sciences, University of Wisconsin, 2008.

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.

A. Zien, N. Krämer, S. Sonnenburg, and G. Rätsch. The feature importance ranking measure. Technical report, Arxiv, 2009. Submitted to ECML 2009.