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
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
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

```
ATCCCGGATTGGATG
AGGGTCCCCTTGAGAGG
CCGGGTATATATATAGG
TtaggTCCCTCCGCGC
```

1, -1, -1, 1
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.

```
|     | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GC before</td>
<td>0.6</td>
<td>0.2</td>
<td>0.4</td>
<td>0.3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.5</td>
<td>0.5</td>
<td>...</td>
</tr>
<tr>
<td>GC after</td>
<td>0.7</td>
<td>0.7</td>
<td>0.3</td>
<td>0.6</td>
<td>0.3</td>
<td>0.4</td>
<td>0.7</td>
<td>0.6</td>
<td>...</td>
</tr>
<tr>
<td>AGAAG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>TTAG</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>-1</td>
<td>...</td>
</tr>
</tbody>
</table>
```
Numerical Representation

ATCCCGGATTGGATG
AGGGTCCCCTTGAAGAGG
CCGGGTATATATATATAGG
TTAGGTTCCCTCCGCGC

AT
CG

1, -1, -1, 1
Recognition of Splice Sites

- Given: Potential acceptor splice sites
  
<table>
<thead>
<tr>
<th>intron</th>
<th>exon</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAACAAAATAAGTAACTAATCTTTTAGGAAGAAACGTGTTCAACCATTTTGAG</td>
<td></td>
</tr>
<tr>
<td>AAGATTTAAAAAAACAAAATTTTAGCATACAGATAATAATCTAATT</td>
<td></td>
</tr>
<tr>
<td>CACTCCCCAATCAACGATATTATTAGTTCACTAACACATCCGTCTGTGCC</td>
<td></td>
</tr>
<tr>
<td>TTAATTCACCTCCACATACTCCAGATCATCAATCCTCAAAAACCAACAC</td>
<td></td>
</tr>
</tbody>
</table>

- 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.
Example $x_i \in \mathcal{X}$, 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 $x_i$ and the associated label $y_i$.

Formally

We want to choose an estimator

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

Intuition

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

Question

How do we measure how well we are doing?
Basic Notion

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

Formally

We define a loss function as

\[ \ell(f(x_i), y_i) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+. \]

Intuition

For a given label \( y_i \) and a given prediction \( f(x_i) \), we want a positive value telling us how much error there is.
In binary classification ($\mathcal{Y} = \{-1, +1\}$), we one may use the 0/1-loss function:

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

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

$$\ell(f(x_i), y_i) = (f(x_i) - y_i)^2.$$
Expected vs. Empirical Risk

Expected Risk
This is the average loss on \textit{unseen examples}. We would like to have it as small as possible, but it is hard to compute.

Empirical Risk
We can compute the \textit{average on training data}. We define the \textbf{empirical risk} to be:

\[
R_{\text{emp}}(f, X, Y) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i).
\]

Basic Notion
Instead of minimizing the expected risk, we minimize the empirical risk. This is called \textbf{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]
What is the complexity of a hyperplane classifier?

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

[Vapnik and Chervonenkis, 1971; Vapnik, 1995]

Larger Margin ⇒ Less Complex

Large Margin ⇒ Small VC dimension

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

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 \( f : \mathcal{X} \rightarrow \mathcal{Y} \) given \( N \) labeled examples \((x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\).

Three important ingredients:

- **Model** \( f_\theta \) parametrized with some parameters \( \theta \in \Theta \)

- **Loss function** \( \ell(f(x), y) \) measuring the “deviation” between predictions \( f(x) \) and the label \( y \)

- **Complexity term** \( P[f] \) defining model classes with limited complexity (via nested subsets \( \{f | P[f] \leq p\} \subseteq \{f | P[f] \leq p'\} \) for \( p \leq p' \))

Most algorithms find \( \theta \) in \( f_\theta \) by minimizing:

\[
\theta^* = \arg\min_{\theta \in \Theta} \left( \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i) + C \underbrace{P[f_\theta]}_{\text{Complexity term}} \right)
\]

for given \( C \)

Special case \((C \rightarrow 0)\): Empirical error \( \rightarrow 0 \) and Complexity term \( \rightarrow \min \)
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; \( \Sigma^* \), i.e. sequences of arbitrary length; etc.)

Multi-class classification

Separation into \( K \) classes: \( \mathcal{Y} = \{1, \ldots, K\} \), \( \mathcal{X} \) arbitrary. (Typical approach: \( f(x) = \arg\max_{k=1,\ldots,K} f^{(k)}(x) \).)

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 \( \Sigma \)-letters of length \( s \), output is string of \( \Sigma' \)-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.
Consider linear hyperplanes with parameters $w, b$:

$$f(x) = \sum_{j=1}^{d} w_j x_j + b = \langle w, x \rangle + b$$
How to Maximize the Margin? II

Margin maximization is equivalent to minimizing $\|w\|$.

[Schölkopf and Smola, 2002]
How to Maximize the Margin? III

Minimize

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

Subject to

\[
y_i (\langle w, x_i \rangle + b) \geq 1 - \xi_i \\
\xi_i \geq 0 \\
\text{for all } i = 1, \ldots, n.
\]

▶ Examples on the margin are called support vectors [Vapnik, 1995]

▶ Soft margin SVMs [Cortes and Vapnik, 1995]
We have to solve an “Optimization Problem”

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| w \|_2^2 + C \sum_{i=1}^{n} \xi_i \\
\text{subject to} & \quad y_i (\langle w, x_i \rangle + b) \geq 1 - \xi_i \text{ for all } i = 1, \ldots, n. \\
& \quad \xi_i \geq 0 \text{ for all } i = 1, \ldots, n
\end{align*}
\]

Quadratic objective function, linear constraints in \( w, b, \) and \( \xi \):

- “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, \ldots)
- Much faster specialized solvers (liblinear, SVM OCAS, Nieme, SGD, \ldots)
An Important Detail

minimize \[ \alpha, b, \xi \]
\[ \frac{1}{2} \left\| \sum_{i=1}^{N} \alpha_i x_i \right\|^2 + C \sum_{i=1}^{n} \xi_i \]

subject to \[ y_i \left( \sum_{j=1}^{N} \alpha_j \langle x_j, x_i \rangle + b \right) \geq 1 - \xi_i \] for all \( i = 1, \ldots, n \).
\[ \xi_i \geq 0 \] for all \( i = 1, \ldots, n \)

Theorem: The optimal \( w \) can be written as a linear combination of the examples (for appropriate \( \alpha \)'s):
\[ w = \sum_{i=1}^{n} \alpha_i x_i \quad \Rightarrow \text{Plug in!} \]

Now optimize for the variables \( \alpha, b, \) and \( \xi \)!

Corollary: Hyperplane only depends on the scalar products of the examples
\[ \langle x, \hat{x} \rangle = \sum_{d=1}^{D} x_d \hat{x}_d \quad \text{Remember this!} \]
Recognition of Splice Sites

- Given: Potential acceptor splice sites

- 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?

Examples on the margin are called **support vectors** [Vapnik, 1995]

Soft margin SVMs [Cortes and Vapnik, 1995]

Minimize

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

Subject to

\[
y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \]

\[
\xi_i \geq 0
\]

for all \( i = 1, \ldots, n \).
Nonlinear Separations

Linear separation might not be sufficient!
⇒ Map into a higher dimensional feature space

Example: all second order monomials

\[ \Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \]

\[ (x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \]
Kernel “Trick”

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

\[
\langle \Phi(x), \Phi(\hat{x}) \rangle = \left\langle (x_1^2, \sqrt{2} x_1 x_2, x_2^2), (\hat{x}_1^2, \sqrt{2} \hat{x}_1 \hat{x}_2, \hat{x}_2^2) \right\rangle \\
= \left\langle (x_1, x_2), (\hat{x}_1, \hat{x}_2) \right\rangle^2 \\
= \langle x, \hat{x} \rangle^2 \\
=: k(x, \hat{x})
\]

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

$$f(x) = \langle w, \Phi(x) \rangle + b$$

$$= \sum_{i=1}^{n} \alpha_i \langle \Phi(x_i), \Phi(x) \rangle + b$$

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

See e.g. Müller et al. [2001]; Schölkopf and Smola [2002]; Vapnik [1995] for details.
Kernel $\approx$ Similarity Measure

Distance:

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

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

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

- Angle between vectors, i.e.,

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

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 $\Phi$ and think about efficient ways to compute scalar product $\langle \Phi(x), \Phi(\hat{x}) \rangle$

▶ 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]

Polynomial \( k(x, \hat{x}) = (\langle x, \hat{x} \rangle + c)^d \)

Sigmoid \( k(x, \hat{x}) = \tanh(\kappa \langle x, \hat{x} \rangle) + \theta \)

RBF \( k(x, \hat{x}) = \exp\left(-\|x - \hat{x}\|^2/(2\sigma^2)\right) \)

Convex combinations \( k(x, \hat{x}) = \beta_1 k_1(x, \hat{x}) + \beta_2 k_2(x, \hat{x}) \)

Normalization \( k(x, \hat{x}) = \frac{k'(x, \hat{x})}{\sqrt{k'(x, x)k'(\hat{x}, \hat{x})}} \)

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
\[ k(x, \hat{x}) = \langle x, \hat{x} \rangle \]

RBF kernel
\[ k(x, \hat{x}) = \exp(-\|x - \hat{x}\|^2 / 2\sigma) \]
Kernel Summary

- Nonlinear separation $\Leftrightarrow$ linear separation of nonlinearly mapped examples
- Mapping $\Phi$ defines a kernel by
  \[ k(x, \hat{x}) := \langle \Phi(x), \Phi(\hat{x}) \rangle \]
- (Mercer) Kernel defines a mapping $\Phi$ (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
- \( N^+ = FN + TP \) number of positive examples
- \( N^- = FP + TN \) number of negative examples
- \( O^+ = TP + FP \) number of positive predictions
- \( O^- = FN + TN \) number of negative predictions

<table>
<thead>
<tr>
<th>outputs\ labeling</th>
<th>( y = +1 )</th>
<th>( y = -1 )</th>
<th>( \Sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(x) = +1 )</td>
<td>TP</td>
<td>FP</td>
<td>( O^+ )</td>
</tr>
<tr>
<td>( f(x) = -1 )</td>
<td>FN</td>
<td>TN</td>
<td>( O^- )</td>
</tr>
<tr>
<td>( \Sigma )</td>
<td>( N^+ )</td>
<td>( N^- )</td>
<td>( N )</td>
</tr>
</tbody>
</table>
Evaluation Measures for Classification II

Several commonly used performance measures

<table>
<thead>
<tr>
<th>Name</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>$\text{ACC} = \frac{TP + TN}{N}$</td>
</tr>
<tr>
<td>Error rate (1-accuracy)</td>
<td>$\text{ERR} = \frac{FP + FN}{N}$</td>
</tr>
<tr>
<td>Balanced error rate</td>
<td>$\text{BER} = \frac{1}{2} \left( \frac{FN}{FN+TP} + \frac{FP}{FP+TN} \right)$</td>
</tr>
<tr>
<td>Weighted relative accuracy</td>
<td>$\text{WRACC} = \frac{TP}{TP+FN} - \frac{FP}{FP+TN}$</td>
</tr>
<tr>
<td>F1 score</td>
<td>$\text{F1} = \frac{2 \times TP}{2 \times TP + FP + FN}$</td>
</tr>
<tr>
<td>Cross-correlation coefficient</td>
<td>$\text{CC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$</td>
</tr>
<tr>
<td>Sensitivity/recall</td>
<td>$\text{TPR} = \frac{TP}{N^+} = \frac{TP}{TP+FN}$</td>
</tr>
<tr>
<td>Specificity</td>
<td>$\text{TNR} = \frac{TN}{N^-} = \frac{TN}{TN+FP}$</td>
</tr>
<tr>
<td>1-sensitivity</td>
<td>$\text{FNR} = \frac{FN}{N^+} = \frac{FN}{FN+TP}$</td>
</tr>
<tr>
<td>1-specificity</td>
<td>$\text{FPR} = \frac{FP}{N^-} = \frac{FP}{FP+TN}$</td>
</tr>
<tr>
<td>P.p.v. / precision</td>
<td>$\text{PPV} = \frac{TP}{O^+} = \frac{TP}{TP+FP}$</td>
</tr>
<tr>
<td>False discovery rate</td>
<td>$\text{FDR} = \frac{FP}{O^+} = \frac{FP}{FP+TP}$</td>
</tr>
</tbody>
</table>
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)
### SVM accuracy of acceptor site recognition using polynomial and Gaussian kernels with different degrees $d$ and widths $\sigma$. Accuracy is measured using the area under the ROC curve (auROC) and is computed using five-fold cross-validation.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>auROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>88.2%</td>
</tr>
<tr>
<td>Polynomial $d = 3$</td>
<td>91.4%</td>
</tr>
<tr>
<td>Polynomial $d = 7$</td>
<td>90.4%</td>
</tr>
<tr>
<td>Gaussian $\sigma = 100$</td>
<td>87.9%</td>
</tr>
<tr>
<td>Gaussian $\sigma = 1$</td>
<td>88.6%</td>
</tr>
<tr>
<td>Gaussian $\sigma = 0.01$</td>
<td>77.3%</td>
</tr>
</tbody>
</table>
Multiple Kernel Learning (MKL)

Data may consist of sequence and structure information

Possible solution: We can add the two kernels,

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

Better solution: We can mix the two kernels,

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

where \( t \) is estimated from the training data

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

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

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, . . .

\[
k(x, x') = \beta_1 k_{dna}(x_{dna}, x'_{dna}) + \beta_2 k_{nrg}(x_{nrg}, x'_{nrg}) + \beta_3 k_3d(x_{3d}, x'_{3d}) + \cdots
\]
MKL Primal Formulation

\[
\begin{align*}
\min & \quad \frac{1}{2} \left( \sum_{j=1}^{M} \beta_j \|w_j\|_2 \right)^2 + C \sum_{i=1}^{N} \xi_n \\
\text{w.r.t.} & \quad w = (w_1, \ldots, w_M), \ w_j \in \mathbb{R}^{D_j}, \ \forall j = 1 \ldots M \\
& \quad \beta \in \mathbb{R}^+_M, \ \xi \in \mathbb{R}^+_N, \ b \in \mathbb{R} \\
\text{s.t.} & \quad y_i \left( \sum_{j=1}^{M} \beta_j w_j^T \Phi_j(x_i) + b \right) \geq 1 - \xi_i, \ \forall i = 1, \ldots, N \\
& \quad \sum_{j=1}^{M} \beta_j = 1
\end{align*}
\]

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).
⇒ 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

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

Multi-Class SVM

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

Examples $x \in \mathcal{X}$

Labels $y \in \mathbb{R}$
Regression

Squared loss  Simplest approach

\[ \ell(f(x_i), y_i) := (y_i - f(x_i))^2 \]

Problem: All \( \alpha \)'s are non-zero \( \Rightarrow \) Inefficient!

\( \varepsilon \)-insensitive loss function

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

\[ \ell(f(x_i), y_i) = \begin{cases} 
0 & |f(x_i) - y_i| < \varepsilon \\
|f(x_i) - y_i| - \varepsilon & \text{otherwise} 
\end{cases} \]

Idea: Examples \((x_i, y_i)\) inside tube have \( \alpha_i = 0 \).

Huber’s loss  Combination of benefits

\[ \ell(f(x_i), y_i) := \begin{cases} 
\frac{1}{2}(y_i - f(x_i))^2 & |y_i - f(x_i)| < \gamma \\
|y_i - f(x_i)| - \frac{1}{2}\gamma^2 & (y_i - f(x_i)) \geq \gamma 
\end{cases} \]

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.

⇒ Algorithmic idea: Low Density Separation
Semi-Supervised SVM

\[ \min_{w, b, (y_j), (\xi_k)} \frac{1}{2} w^\top w + C \sum_i \xi_i + C^* \sum_j \xi_j \]

s.t. \[
\begin{align*}
\xi_i &\geq 0 \\
\xi_j &\geq 0 \\
y_i(w^\top x_i + b) &\geq 1 - \xi_i \\
y_j(w^\top x_j + b) &\geq 1 - \xi_j
\end{align*}
\]
Semi-Supervised SVM: Optimization

⇒ Optimization matters

Comparison of $S^3\text{VM}$ 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: $P_{Source}(Y|X) \neq P_{Target}(Y|X)$

→ 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 $\rightarrow$ 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.
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)

```
~/mylibs/bin/easysvm.py cv 5  # SVM C kernel data format and file predictions
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
```

```
~/mylibs/bin/easysvm.py eval lin_gc.out arff C_elegans_acc_gc.arff lin_gcperf
```

```
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**) using:

```
~/mylibs/bin/easysvm.py modelsel 5 SVM C's 0.1, 1, 10
poly 1, 2, 3, 4, 5 true false
data format and file arff C_elegans_acc_gc.arff
output file 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*

\[ \approx 150\text{-nucleotide window around dimer} \]

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

- **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,

\[
x = \text{AAACAAAATAAGTAACTAATCTTTTAGAAGAACGTTTCAACCATTTTTGAG}
\]

\[
x' = \text{TACCTAATTATGAAATTAAATTTCAGTGCTGATGGAAACGGAAAGTC}
\]

for instance, tissue classification using promoter region
Types of Signal Detection Problems

Problem categorization

(based on positional variability of motifs)

Position-Dependent

→ Motifs very stiff, almost always at same position,

```
AAACAAATAAGTAACTAATCTTTTAAGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAAACAAATTTTAAACATTACAGATAATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAATTCACATAAACACATCCGTCTGTGCC
```

for instance, splice site identification
Types of Signal Detection Problems

Problem categorization

(based on *positional variability* of motifs)

Mixture of Position-Dependent/-Independent

→ 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 \(k = 3\):

\[
\begin{align*}
\textbf{x} & \quad \text{AAAACAAATAAGTAAC\ldots}& \text{G} & \quad \text{GGAAGAACGTTTCAACCATTTTGAG} \\
\textbf{x}' & \quad \text{TACCTAATTATGAAAT\ldots} & \text{T} & \quad \text{AAGGAAACGGAAGTTC} \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>3-mer</th>
<th>AAA</th>
<th>AAC</th>
<th>\ldots</th>
<th>CCA</th>
<th>CCC</th>
<th>\ldots</th>
<th>TTT</th>
</tr>
</thead>
<tbody>
<tr>
<td># in (x)</td>
<td>2</td>
<td>4</td>
<td>\ldots</td>
<td>1</td>
<td>0</td>
<td>\ldots</td>
<td>3</td>
</tr>
<tr>
<td># in (x')</td>
<td>3</td>
<td>1</td>
<td>\ldots</td>
<td>0</td>
<td>0</td>
<td>\ldots</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
k(\mathbf{x}, \mathbf{x}') = 2 \cdot 3 + 4 \cdot 1 + \ldots 1 \cdot 0 + 0 \cdot 0 \ldots 3 \cdot 1
\]
Spectrum Kernel with Mismatches

**General idea** [Leslie et al., 2003]

- Do not enforce strictly exact matches
- Define mismatch neighborhood of $\ell$-mer $s$ with up to $m$ mismatches:
  \[
  \phi_{(l,m)}(s) = (\phi_\beta(s))_{\beta \in \Sigma^l}
  \]
- For sequence $x$ of any length, the map is then extended as:
  \[
  \phi_{(l,m)}(x) = \sum_{\ell\text{-mers } s \text{ in } x} (\phi_{(l,m)}(s))
  \]
- The mismatch kernel is the inner product in feature space defined by:
  \[
  k_{(l,m)}(x, x') = \langle \Phi_{(l,m)}(x), \Phi_{(l,m)}(x') \rangle
  \]
General idea [Leslie and Kuang, 2004; Lodhi et al., 2002]

- Allows gaps in common substrings
  → “subsequences”
- A \(g\)-mer then contributes to all its \(\ell\)-mer subsequences:

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

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

\[
\phi_{(g, \ell)}^{\text{Gap}}(x) = \sum_{\text{\(g\)-mers } s \text{ in } x} (\phi_{(g, \ell)}^{\text{Gap}}(s))
\]

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

\[
k_{(g, \ell)}^{\text{Gap}}(x, x') = \left\langle \Phi_{(g, \ell)}^{\text{Gap}}(x), \Phi_{(g, \ell)}^{\text{Gap}}(x') \right\rangle
\]
Wildcard Kernels

General idea [Leslie and Kuang, 2004]

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

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

where \( \phi^\beta(s) = \lambda^j \) if \( s \) matches pattern \( \beta \) containing \( j \) wildcards, and \( \phi^\beta(s) = 0 \) if \( s \) does not match \( \beta \), and \( 0 \leq \lambda \leq 1 \).
Weighted Degree Kernel = Spectrum kernels for each position

To make use of position-dependent motifs:

\[
k(x, x') = \sum_{k=1}^{d} \beta_k \sum_{l=1}^{L-k} I(u_{k,l}(x) = u_{k,l}(x'))
\]

- \( L \) := length of the sequence \( x \)
- \( d \) := maximal “match length” taken into account
- \( u_{k,l}(x) := \) subsequence of length \( k \) at position \( l \) of sequence \( x \)

Example degree \( d = 3 \):


\( x' \)  T A C C T A A T T T A T T T G A A A T T A T T A T T T C A T T G T GT G T G T G A T G A A A C G G A A G A A G T C

\[
k(x, x') = \beta_1 \cdot 21 + \beta_2 \cdot 8 + \beta_3 \cdot 4
\]

[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 \( O(L) \))

\[
k(S_1,S_2) = W_7 + W_1 + W_2 + W_2 + W_3
\]

**Exercise:** Show that WD kernel and its “block” formulation are equivalent
Sequence-based Splice Site Recognition

<table>
<thead>
<tr>
<th>Kernel</th>
<th>auROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spectrum $\ell = 1$</td>
<td>94.0%</td>
</tr>
<tr>
<td>Spectrum $\ell = 3$</td>
<td>96.4%</td>
</tr>
<tr>
<td>Spectrum $\ell = 5$</td>
<td>94.5%</td>
</tr>
<tr>
<td>Mixed spectrum $\ell = 1$</td>
<td>94.0%</td>
</tr>
<tr>
<td>Mixed spectrum $\ell = 3$</td>
<td>96.9%</td>
</tr>
<tr>
<td>Mixed spectrum $\ell = 5$</td>
<td>97.2%</td>
</tr>
<tr>
<td>WD $\ell = 1$</td>
<td>98.2%</td>
</tr>
<tr>
<td>WD $\ell = 3$</td>
<td>98.7%</td>
</tr>
<tr>
<td>WD $\ell = 5$</td>
<td>98.9%</td>
</tr>
</tbody>
</table>

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 $\ell$. 
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 $S(l)$)

\[
k(x_i, x_j) = \sum_{k=1}^{K} \beta_k \sum_{l=1}^{L-k+1} \gamma_l \sum_{s=0}^{S(l)} \delta_s \mu_{k,l,s,x_i,x_j},
\]

\[
\mu_{k,l,s,x_i,x_j} = I(u_{k,l+s}(x_i) = u_{k,l}(x_j)) + I(u_{k,l}(x_i) = u_{k,l+s}(x_j)),
\]

\[
k(x_1, x_2) = w_{6,3} + w_{6,3} + w_{3,4}
\]

[Rätsch et al., 2005]
Oligo Kernel

Oligo kernel

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

where

- \( 0 \leq \sigma \) is a smoothing parameter
- \( u \) is a \( k \)-mer and
- \( S_u^x \) is the set of positions within sequence \( x \) at which \( u \) 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:

$$k_{seq}(x_j, x_k) = \sum_{i=1}^{M} k_{WDS}(s_{i,j}, s_{i,k})$$

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

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

- Regulatory Modules kernel: $k_{RM}(x, x') := k_{seq}(x, x') + k_{pos}(x, x')$
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 $\pi$ is then scored as follows:

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

Smith-Waterman score (not positive definite)

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

Local Alignment kernel [Vert et al., 2004]

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

Polynomial Kernel of degree $d$:

$$k_{\text{POLY}}(x, x') = \left( \sum_{p=1}^{l} l_p(x, x') \right)^d$$

⇒ Computes all $d$-th order monomials: global information

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

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

$$\text{win}_p(x, y) = \left( \sum_{j=-l}^{+l} p_j l_{p+j}(x, y) \right)^d$$

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

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 $p(s|\theta)$ (e.g. HMM, PSSMs)
- Maximum likelihood estimate $\theta^* \in \mathbb{R}^d$
- Transformation into Fisher score features $\Phi(s) \in \mathbb{R}^d$
  - $\Phi(s) = \frac{\partial \log(p(s|\theta))}{\partial \theta}$
  - Describes contribution of every parameter to $p(s|\theta)$
- $k(s, s') = \langle \Phi(s), \Phi(s') \rangle$
Example: Fisher Kernel on PSSMs

- Sequences $s \in \Sigma^N$ of fixed length
- PSSMs: $p(s|\theta) = \prod_{i=1}^{N} \theta_{i,s_i}$
- Fisher score features: $(\Phi(s))_{i,\sigma} = \frac{dp(s|\theta)}{d\theta_{i,\sigma}} = \text{Id}(s_i = \sigma)$
- Kernel: $k(s,s') = \langle \Phi(s), \Phi(s') \rangle = \sum_{i=1}^{N} \text{Id}(s_i = s_i')$
- 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                  | $l_x \neq l_x'$ | $Pr(x|\theta)$ | Positional? | Scope            | Complexity     |
|-------------------------|-----------------|-----------------|-------------|------------------|----------------|
| linear                  | no              | no              | yes         | local            | $O(l_x)$       |
| polynomial              | no              | no              | yes         | global           | $O(l_x)$       |
| locality-improved        | no              | no              | yes         | local/global     | $O(l \cdot l_x)$ |
| sub-sequence            | yes             | no              | yes         | global           | $O(nl_xl_x')$  |
| n-gram/Spectrum         | yes             | no              | no          | global           | $O(l_x)$       |
| WD                      | no              | no              | yes         | local            | $O(l_x)$       |
| WD with shifts          | no              | no              | yes         | local/global     | $O(s \cdot l_x)$ |
| Oligo                   | yes             | no              | yes         | local/global     | $O(l_xl_x')$   |
| 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**)

```
~/mylibs/bin/easysvm.py cv 5 svm C kernel data format and file predictions
   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 %

```
~/mylibs/bin/easysvm.py cv 5 svm C kernel data format and file predictions
   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 isomorphism 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

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

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

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 $A^k$

Definition of random walk kernel

$k_X(G_1, G_2) = \sum_{i,j=1}^{\left|V_X\right|} \left[ \sum_{n=0}^{\infty} \lambda^n A^n_X \right]_{ij}$

Meaning

- Random walk kernel counts all pairs of matching walks
- $\lambda$ 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 $n^2 \ast n^2$ 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 → 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 → improved classification accuracy

Forbidding cycles with 2 nodes

- Mahé et al. [2004] redefine walk kernel to forbid subcycles consisting of two nodes → no practical improvement
Limitations of Walks

Different graphs mapped to identical points in walks feature space [Ramon and Gärtner, 2003]
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 \( V_1(G_1) \) and \( s \) from \( V_2(G_2) \) 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 $\delta^+(s)$

$$M(r, s) = \{ R \subseteq \delta^+(r) \times \delta^+(s) |$$

$$(\forall (a, b), (c, d) \in R : a = c \iff b = d) \land$$

$$(\forall (a, b) \in R : \text{label}(a) = \text{label}(b)) \}$$

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 $\lambda_r$ and $\lambda_s$ are positive scalars
Subtree Kernel

Subtree graph kernel

- The subtree graph kernel for fixed height $h$ is
  \[
  k_{\text{tree},h}(G_1, G_2) = \sum_{r \in V_1} \sum_{s \in V_2} k_h(r, s).
  \]

- The subtree graph kernel for $h$ approaching infinity is
  \[
  k_{\text{tree}}(G_1, G_2) = \lim_{h \to \infty} k_{\text{tree},h}(G_1, G_2),
  \]
  which will converge for suitable choice of $\lambda_r$ and $\lambda_s$.

- Both versions are positive definite
- Large choice of $h$ provides good approximation of $k_{\text{tree}}$. 
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 $O(n^4)$ → 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]:

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 $h$ 

\[ \Rightarrow \quad \Rightarrow \quad \{ f_1, \ldots, f_m \} \Rightarrow \quad \Rightarrow \text{SVM} \]
\( \chi^2 \)-Kernel for Histograms

Image described by histogram \( h_C \) implied by code book \( C \) of size \( d \)

Kernel for comparing two histograms:

\[
k_{\gamma, C}(h_C, h'_C) = \exp \left( -\gamma \chi^2(h_C, h'_C) \right),
\]

where \( \gamma \) is a hyper-parameter,

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

and we use the convention \( x/0 := 0 \)
Spatial Pyramid Kernels

Decompose image into a pyramid of $L$ levels

$$k_{\text{pyr}}(\alpha, \beta) = \frac{1}{8} k(\alpha, \beta) + \frac{1}{4} k(\alpha, \beta) + \frac{1}{4} k(\alpha, \beta) + \frac{1}{2} k(\alpha, \beta) + \ldots$$

[Lazebnik et al., 2006]
General Spatial Kernels

Use general spatial kernel with subwindow $B$

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

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

Example regions:

1000 subwindows
Consider set of code books $C_1, \ldots, C_K$ or regions $B_1, \ldots, B_K$.

Each code book $C_p$ or region $B_p$ leads to a kernel $k_p(x, x')$.

Which kernel is best suited for classification?

Define kernel as linear combination

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

Use multiple kernel learning to determine the optimal $\beta$’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 \( \alpha \)-weighting of training points

\[
s(x) = \sum_{i=1}^{N} \alpha_i y_i \cdot k(x_i, x) + b
\]

\( \alpha_1 \cdot \text{AAACAAATAGTAACCTACTTCTTTTAGGAAGAAGCCTTTCACACATTTTGAG} \)

\( \alpha_2 \cdot \text{AAGATTAACACAAATTTTTTAGCATTACAGATATAATTAATCTAAATT} \)

\( \alpha_3 \cdot \text{CACTCCCGAAAATCAACGATATTTTTAGTTCACTAACACATCCGTCTGTGCC} \)

\[ \vdots \]

\( \alpha_N \cdot \text{TTAATTTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC} \)

But we are interested in weights of features
Understanding Linear SVMs

Support Vector Machine

\[
f(x) = \text{sign} \left( \sum_{i=1}^{N} y_i \alpha_i k(x, x_i) + b \right),
\]

Use SVM w from feature space

- Recall SVM decision function in kernel feature space:

\[
f(x) = \sum_{i=1}^{N} y_i \alpha_i \Phi(x) \cdot \Phi(x_i) + b
= k(x, x_i)
\]

- Explicitly compute \( w = \sum_{i=1}^{N} \alpha_i \Phi(x_i) \)
Understanding Linear SVMs

Explicitly compute

\[ w = \sum_{i=1}^{N} \alpha_i \Phi(x_i) \]

Use \( w \) to rank importance

| \( \text{dim} \) | \( |w_{\text{dim}}| \) |
|----------------|-----------------|
| 17             | +27.21          |
| 30             | +13.1           |
| 5              | -10.5           |
| ...            | ...             |

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

\[ \Phi(x) = (x_1 x_1, \sqrt{\frac{1}{2}} x_1 x_2, \ldots \sqrt{\frac{1}{2}} x_1 x_d, \sqrt{\frac{1}{2}} x_2 x_3 \ldots x_d x_d) \]
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

\[ w = \sum_{i=1}^{N} \alpha_i y_i \Phi(x_i) \]

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

<table>
<thead>
<tr>
<th>k-mer</th>
<th>pos. 1</th>
<th>pos. 2</th>
<th>pos. 3</th>
<th>pos. 4</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>+0.1</td>
<td>-0.3</td>
<td>-0.2</td>
<td>+0.2</td>
<td>…</td>
</tr>
<tr>
<td>C</td>
<td>0.0</td>
<td>-0.1</td>
<td>+2.4</td>
<td>-0.2</td>
<td>…</td>
</tr>
<tr>
<td>G</td>
<td>+0.1</td>
<td>-0.7</td>
<td>0.0</td>
<td>-0.5</td>
<td>…</td>
</tr>
<tr>
<td>T</td>
<td>-0.2</td>
<td>-0.2</td>
<td>0.1</td>
<td>+0.5</td>
<td>…</td>
</tr>
<tr>
<td>AA</td>
<td>+0.1</td>
<td>-0.3</td>
<td>+0.1</td>
<td>0.0</td>
<td>…</td>
</tr>
<tr>
<td>AC</td>
<td>+0.2</td>
<td>0.0</td>
<td>-0.2</td>
<td>+0.2</td>
<td>…</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>TT</td>
<td>0.0</td>
<td>-0.1</td>
<td>+1.7</td>
<td>-0.2</td>
<td>…</td>
</tr>
<tr>
<td>AAA</td>
<td>+0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>+0.1</td>
<td>…</td>
</tr>
<tr>
<td>AAC</td>
<td>0.0</td>
<td>-0.1</td>
<td>+1.2</td>
<td>-0.2</td>
<td>…</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>TTT</td>
<td>+0.2</td>
<td>-0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>…</td>
</tr>
</tbody>
</table>

© G. Rätsch & S. Sonnenburg (FML)
Kernels for Sequences and Graphs
SVM Scoring Function - Examples

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

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 \( z \) at position \( j \) in the sequence, compute expected score \( \mathbb{E}[s(x) \mid x[j] = z] \) (for small \( k \))

\[
\begin{align*}
\text{AAAAAAAAAA} & \text{TAC} \text{AAAAAAAAAA} \\
\text{AAAAAAAAAA} & \text{TAC} \text{AAAAAAAAAAAC} \\
\text{AAAAAAAAAA} & \text{TAC} \text{AAAAAAAAAAAG} \\
& \vdots \\
\text{TTTTTTTTTTTT} & \text{TAC} \text{TTTTTTTTTTTT}
\end{align*}
\]

▶ Normalize with expected score over all sequences

POIMs

\[
Q(z, j) := \mathbb{E}[s(x) \mid x[j] = z] - \mathbb{E}[s(x)]
\]

⇒ Needs efficient algorithm for computation [Sonnenburg et al., 2008]
Obtain highest scoring \( z \) from \( Q(z, i) \) (Enhancer or Silencer)

<table>
<thead>
<tr>
<th>( z )</th>
<th>( i )</th>
<th>( Q(z, i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATTACA</td>
<td>10</td>
<td>+30</td>
</tr>
<tr>
<td>AGTAGTG</td>
<td>30</td>
<td>+20</td>
</tr>
<tr>
<td>AAAAAAAA</td>
<td>10</td>
<td>-10</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

TGAGCGCGTGATTACA GTCCGTCTGGGCCAGTAGTG CGTAGTCGCCGGGA
GCATGGTCGATTACA AACGAGCCCTCTCAGTAGTG GGGGAGCCACGAAA
CCCGTCGAAGATTACACACGGGCGTGAGTAGTGGCATTACGGGCTC
GGTGGCAGAGATTACACGACGCGTGTACGAGTAGTGAACACTGACTCCTC
GATTACA at Variable Positions

TGAGCGCGTGATTACAGTCCGTCT
GGCTCGATCACAACAGGAGCCCGAT
CCCGTCGAACAGGATTACACACGG
GGTCGGCAGCTTACACGACACGCT

© G. Rätsch & S. Sonnenburg (FML)  Kernels for Sequences and Graphs
GATTACA at Variable Positions

Differential POIM Overview – GATTACA shift

Motif Length (k)
Position
−30 −20 −10 0 10 20 30
8
7
6
5
4
3
2
1
Drosophila Transcription Start Sites

TATAAAA  -29/++  CAGTCAGT  -01/++  CGTCGCG  +18/++
GTATAAA  -30/++  TCAGTTGT  -01/++  GCGCGCG  +23/++
ATATAAA  -28/++  CGTCAGTT  -03/++  CGCGCGC  +22/++

TATA-box  Inr  TCA $\frac{G}{T}$ $\frac{T}{C}$  CpG

© G. Rätsch & S. Sonnenburg (FML)  Kernels for Sequences and Graphs 134 / 224
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:

\[ k(x, y) = \sum_{l=1}^{L} \beta_l k_l(x, y) \]

for instance, Weighted Degree kernel

\[ k(x, x') = \sum_{l=1}^{L} \beta_l \sum_{k=1}^{K} I(u_{k,l}(x) = u_{k,l}(x')) \]

▶ Optimize weights \( \beta \) such that margin is maximized

⇒ determine \( (\beta, \alpha, b) \) simultaneously

⇒ **Multiple Kernel Learning** [Bach et al., 2004]
Method for Interpreting SVMs

- **Weighted Degree kernel:** linear comb. of $LD$ kernels

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

- **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

1000 subwindows

livingroom 27 subwindows

MITcoast 19 subwindows

MITtallbuilding 19 subwindows

bedroom 26 subwindows

CALsuburb 15 subwindows
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 $\Phi(x, y)$. Then the corresponding kernel function is

$$k((x, y), (x', y')) := \langle \Phi(x, y), \Phi(x', y') \rangle$$

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

$$k((x, y), (x', y')) := [[y = y']]k(x, x')$$
Example: Context-free Grammar Parsing

The screen was a sea of red

Recursive Structure

[Klein & Taskar, ACL’05 Tutorial]
Example: Bilingual Word Alignment

What is the anticipated cost of collecting fees under the new proposal?

En vertu des nouvelles propositions, quel est le coût prévu de perception des droits?

Combinatorial Structure

[Klein & Taskar, ACL’05 Tutorial]
Example: Handwritten Letter Sequences

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
  ⇒ state sequence defines a “segmentation”
- Example 1: Secondary Structure Prediction of Proteins

![Diagram of protein structure with residue sequence and secondary structure annotation]
Label Sequence Learning

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

![Gene Finding Diagram](image_url)
Generative Models

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

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

- 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

\[ \log p(x, y) = \sum_i (\log p(x_i | y_i) + \log p(y_i | y_{i-1})) \]

\[ = \sum_i g(y_{i-1}, y_i, x_i) \]

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 \( x \), find sequence \( y \) such that \( \log p(x, y) \) is maximized, i.e. \( y^* = \arg \max_{y \in \mathcal{Y}} \log p(x, y) \)

**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

\[
p(x, y) = \prod_j p((x_{i(j)-1} + 1, \ldots, x_{i(j)}) | y_j) p(y_j | y_{j-1})
\]

(use with care)

► Use generalization of DP algorithms of HMMs
Decoding via Dynamic Programming

\[
\log p(x, y) = \prod_j p((x_{i(j)}, \ldots, x_{i(j+1)-1})|y_j)p(y_j|y_{j-1}) \\
= \sum_j g(y_{i-1}, y_i, (x_{i(j-1)+1}, \ldots, x_{i(j)}))
\]

with \( g(y_{j-1}, y_j, x_j) = \log p(x_j|y_j) + \log p(y_j|y_{j-1}) \).

**Problem:** Given sequence \( x \), find sequence \( y \) such that \( \log p(x, y) \) is maximized, i.e., \( y^* = \arg\max_{y \in Y^*} \log p(x, y) \)

**Dynamic Programming Approach:**

\[
V(i, y) := \begin{cases} 
\max_{y' \in \mathcal{Y}, d=1, \ldots, i-1} (V(i-d, y') + g(y', y, x_{i-d+1}, \ldots, i)) & i > 1 \\
0 & \text{otherwise}
\end{cases}
\]
Discriminative Models

- Conditional Random Fields [Lafferty et al., 2001]
  - Conditional probability $p(y|x)$ instead of joint probability $p(x, y)$
    
    $$p_w(y|x) = \frac{1}{Z(x, w)} \exp(f_w(y|x))$$

  ![Graphical representation of Conditional Random Fields]

  - Can handle non-independent input features
  - Parameter estimation: $\max_w \sum_{n=1}^{N} \log p_w(y_n|x_n)$

- Decoding: Viterbi or Maximum Expected Accuracy algorithms (cf. [Gross et al., 2007])
Max-Margin Structured Output Learning

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

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

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

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

$$\min_f \left\{ C \sum_{n=1}^{N} \xi_n + P[f] \right\}$$

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

- Exponentially many constraints!
Joint Feature Map

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

Joint kernel on $\mathcal{X}$ and $\mathcal{Y}$
We define a joint feature map on $\mathcal{X} \times \mathcal{Y}$, denoted by $\Phi(x, y)$. Then the corresponding kernel function is
$$k(((x, y), (x', y'))) := \langle \Phi(x, y), \Phi(x', y') \rangle$$

For multiclass
For normal multiclass classification, the joint feature map decomposes and the kernels on $\mathcal{Y}$ form the identity, that is,
$$k(((x, y), (x', y'))) := [[y = y']]k(x, x')$$
Structured Output Learning with Kernels

- Assume $f(y|x) = \langle w, \Phi(x, y) \rangle$, where $w, \Phi(x, y) \in \mathcal{F}$
- Use $\ell_2$ regularizer: $P[f] = \|w\|^2$

$$\min_{w \in \mathcal{F}, \xi \in \mathbb{R}^N} \sum_{n=1}^{N} \xi_n + \|w\|^2$$

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

for all $y_n \neq y \in \mathcal{Y}^*, n = 1, \ldots, N$

- Linear classifier that separates true from false labeling
Special Case: Only Two “Structures”

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

\[
\min_{w \in \mathcal{F}, \xi \in \mathbb{R}^N} \sum_{n=1}^{N} \xi_n + \|w\|^2 \quad \text{w.r.t.} \quad \langle w, \Phi(x, y_n) - \Phi(x, 1 - y_n) \rangle \geq 1 - \xi_n
\]

for all \( n = 1, \ldots, N \)

**Exercise:** Show that it is equivalent to standard 2-class SVM for appropriate values of \( \Phi \)
Optimization

- Optimization problem too big (dual as well)

$$\min_{w \in \mathcal{F}, \xi} \quad C \sum_{n=1}^{N} \xi_n + \|w\|^2$$

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

for all $$y_n \neq y \in \mathcal{Y}^*, n = 1, \ldots, N$$

- 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

\[ \langle w, \Phi(x, y_n) - \Phi(x, y) \rangle \geq 1 - \xi_n \]

- Find labeling \( y \) that maximizes

\[ \langle w, \Phi(x, y) \rangle \]

- Use dynamic programming decoding

\[ y = \arg\max_{y \in Y^*} \langle w, \Phi(x, y) \rangle \]

(DP only works if \( \Phi \) has a certain decomposition structure)

- If \( y = y_n \), 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

\[
(w^t, \xi^t) = \arg\min_{w \in F, \xi} C \sum_{n=1}^{N} \xi_n + \|w\|^2 \\
\text{w.r.t.} \quad \langle w, \Phi(x, y_n) - \Phi(x, y) \rangle \geq 1 - \xi_n
\]

for all \( y_n \neq y \in \mathcal{Y}_n^t, n = 1, \ldots, N \)

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

\[
y^t_n = \arg\max_{y_n \neq y \in \mathcal{Y}^*} \langle w^t, \Phi(x, y) \rangle
\]

If \( \langle w^t, \Phi(x, y_n) - \Phi(x, y^t_n) \rangle < 1 - \xi^t_n \), set \( \mathcal{Y}_n^{t+1} = \mathcal{Y}_n^t \cup \{y^t_n\} \)

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 $y \neq y$, then prediction is wrong, but it does not matter how wrong
- Introduce loss function on labelings $\ell(y, y')$, e.g.
  - How many segments are wrong or missing
  - How different are the segments, etc.
- Extend optimization problem (Margin rescaling):

$$
\min_{w \in \mathcal{F}, \xi} \sum_{n=1}^{N} \xi_n + \|w\|^2 \quad \text{w.r.t.} \quad \langle w, \Phi(x_n, y_n) - \Phi(x_n, y) \rangle \geq \ell(y_n, y) - \xi_n
$$
for all $y_n \neq y \in \mathcal{Y}^*$, $n = 1, \ldots, N$

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

$$
y_n^t = \arg\max_{y_n \neq y \in \mathcal{Y}^*} \left( \langle w^t, \Phi(x_n, y) \rangle + \ell(y, y_n) \right)
$$
Problems

- Optimization may require many iterations
- Number of variables increases linearly
- When using kernels, solving optimization problems can become infeasible
- Evaluation of $\langle w, \Phi(x, y) \rangle$ 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
ARTS: accurate recognition of transcription starts in human
Sören Sonnenburg¹, Alexander Zien²,³ and Gunnar Rätsch³,*
¹Fraunhofer Institute FIRST, Kekuléstr. 7, Berlin, Germany, ²Max Planck Institute for Biological Cybernetics, Spemannstr. 38, Tübingen, Germany and ³Friedrich 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

$\implies$ 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 ($\approx -30$ bp upstream)
- Exon content in UTR 5’ region
- Distance to first donor splice site

Idea:
Combine weak features to build strong promoter predictor

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

⇒ 35% true positives at a false positive rate of 1/1000 (best other method finds about one half (18%))
Contributions of the Kernels

⇒ Most important: Weighted Degree Shift kernel modeling the TSS signal
Discovering Sequence Variations in *Arabidopsis thaliana*

© G. Rätsch & S. Sonnenburg (FML)  
Case Studies (Applications)

[© G. Rätsch & S. Sonnenburg (FML)  
Case Studies (Applications) 168 / 224]
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
Simplified Model: Score for splice form $y = \{(p_j, q_j)\}_{j=1}^J$:

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

- **splice signals**
- **segment lengths**

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

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

The image shows a bar chart comparing the mean sensitivity and specificity of several gene prediction tools across different categories.

- cds_nucleotides
- cds_exons
- cds_transcripts

The tools compared include:
- mGene NEW
- mGene
- AUGUSTUS
- FGENESH
- GLIMMERHMM
- EUGENE
- CRAIG
- GENEID
- SNAP
- AGENE
- GENEMARKERHMM

The chart illustrates the performance of these tools in predicting coding sequences, with each category showing a range of values for sensitivity and specificity.
Spliced Alignments Using Large Margins

**BIOINFORMATICS** ORIGINAL PAPER

Sequence analysis

**PALMA: mRNA to genome alignments using large margin algorithms**

Uta Schulze¹,2,†, Bettina Hepp³,†, Cheng Soon Ong⁴,1 and Gunnar Rätsch¹,≈

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

Received on November 9, 2007; revised on May 18, 2007; accepted on May 16, 2007


Associate Editor: Prof. Dmitrij Fridman

**BIOINFORMATICS**

Optimal spliced alignments of short sequence reads

Fabio De Bona¹, Stephan Ossowski², Korbinian Schneeberger² and Gunnar Rätsch¹,†

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

≈ 150-nucleotide window around dimer

CT...GTCGTA...GAAGCTAGGGAGCGC...ACGCGT...GA

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

⇒ 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 \( M : \Sigma \times \Sigma \rightarrow \mathbb{R} \), where \( \Sigma := \{A, C, G, T, N, -\} \)

**Output**
- Sequence alignment \( \mathcal{A} \)
  - Sequence of pairs, i.e. \( \mathcal{A} = (a_r, b_r)_{r=1,...,R}, a_r, b_r \in \Sigma \)
  - \( R \leq m + n \) 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 ($\forall i = 1 \ldots m, j = 1 \ldots n$)

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

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

**Idea:** Learn functions on training set with known alignments
Parameterization

- Substitution matrix $M : \Sigma \times \Sigma \rightarrow \mathbb{R}$
- Functions $f^{\text{Len}}$, $f^{\text{Acc}}$ and $f^{\text{Don}}$
  - Piecewise linear functions (support points $x_1, \ldots, x_s$):
    
    $f(x) = \begin{cases} 
    \theta_1 & x \leq x_1 \\
    \frac{\theta_i(x_{i+1}-x)+\theta_{i+1}(x-x_i)}{x_{i+1}-x_i} & x_i \leq x \leq x_{i+1} \\
    \theta_s & x \geq x_s
    \end{cases}$

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

Idea

Find $\theta$ such that for a known alignment $A^+$

$$s_\theta(A^+) \gg s_\theta(A^-)$$

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

Given $N$ known alignments $A_i^+, i = 1, \ldots, N$

solve quadratic optimization problem (QP)

$$\min_{\xi \geq 0, \theta} \frac{1}{N} \sum_{i=1}^{N} \xi_i + P(\theta)$$

s.t. $s_\theta(A_i^+) - s_\theta(A_i^-) \geq 1 - \xi_i \quad \forall A_i^- \neq A_i^+, i = 1, \ldots, N$

- $\xi_i$: Slack variables to implement a soft-margin
- $P(\theta)$: Regularizer leading to smooth functions
Iterative Algorithm

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

$$
\min_{\xi \geq 0, \theta} \frac{1}{N} \sum_{i=1}^{N} \xi_{i} + P(\theta)
$$

s.t. $s_{\theta}(A_{i}^{+}) - s_{\theta}(A^{-}) \geq 1 - \xi_{i} \quad \forall A^{-} \in A_{i}^{-}, i = 1, \ldots, N$

- 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)

⇒ Microexon generated
⇒ Splice sites still intact

▶ Generate insertions/deletions/mutations in artificial EST
  \( (\sigma = 0\%, 1\%, 2\%, 10\%, 20\%, 50\%) \)
▶ Train PALMA on 4608 exon triples (≈ 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 Garbrele 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 Weigel1,8*

**Methods**

Detecting polymorphic regions in *Arabidopsis thaliana* with resequencing microarrays

Georg Zeller,1,2 Richard M. Clark,2,3 Korbinian Schnellerberger,2,3 Anja Bohlen,1 Detlef Weigel,2 and Gunnar Rätsch1,4

1Friedrich Miescher Laboratory of the Max Planck Society, Tübingen 72070, Germany; 2Max Planck Institute for Developmental Biology, Department of Molecular Biology, Tübingen 72070, Germany
What is the genetic basis of variation?

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

reference DNA sequence

ACGTAAGTCGAATGAAATGACCCCTTTTGAGAGCCCGTT

reference call

ACGTAAGTCGAATTGAAATGACCC

hybridization intensity

A

ACGTAAGTCGACTGAAATGACCC

C

SNP

ACGTAAGTCGAGTGAAATGACCC

G

TGCATTTTCAGCTCACTTTACTGGGAAACTCTC

T

ACGTAAGTCGATTGAAATGACCC
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 $\approx 550$bp 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-accession 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

- **Polymorphic Region Prediction (PRP)**
- Use HMSVM for segmenting the sequence
Modeling polymorphic regions

Exon

Intron

Exon

Log max intensity

bp

Col-0

Cvi-0

D

A

B

C

Exon

Exon

(1)

(2)

(3)

(4)

(5)

Transition Labels
Not polymorphic

PR

Predictions

MBML2 SNP

Known polymorphisms

Insertion

SNP

Deletion

T

D1

D2

D3

P

TU1

TU2

TU3

C

TD1

TD2

TD3

C

TD2

TD1

TD3

P

TU1

TU2

TU3

G. Rätsch & S. Sonnenburg (FML)

Case Studies (Applications)
The Learning Problem

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

\[ f : X \rightarrow Y \]

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

Employ a function

\[ F_\Theta : X \times Y \rightarrow \mathbb{R} \quad \text{(path scoring)} \]

with which

\[ f(x) = \arg \max_{\pi \in Y} F(x, \pi) \quad \text{(Viterbi decoding)} \]

(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):

<table>
<thead>
<tr>
<th></th>
<th>SNP calling (MB+ML)</th>
<th>Region predictor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNPs</td>
<td>∼32%</td>
<td>∼65%</td>
</tr>
<tr>
<td>Deletions (per base)</td>
<td>∼53%</td>
<td></td>
</tr>
<tr>
<td>Insertions</td>
<td>∼39%</td>
<td></td>
</tr>
</tbody>
</table>

[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
  - ≈ US$50
Learning with Kernels [Schölkopf and Smola, 2002]


  ▶ MIT Press, Sept. 2002
  ▶ by B. Schölkopf and A. Smola
  ▶ ≈ 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
  - ≈ US$36
Semi-Supervised Learning [B. Schölkopf, 2006; Zhu, 2008]

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

Structured Output Learning [Bakir et al., 2007]

- **Predicting Structured Outputs.**

  - MIT Press, Sept. 2007
  - \( \approx \) US$36

- Structured Output Learning tutorial in preparation for PLoS Computational Biology
Final Remarks

[Sonnenberg 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 II


References V


References VIII


