Kernel Methods for Predictive Sequence Analysis

Cheng Soon Ong[1,2] **and Gunnar Rätsch**[1]

[1] Friedrich Miescher Laboratory, Tübingen
[2] Max Planck Institute for Biological Cybernetics, Tübingen

Tutorial at the German Conference on Bioinformatics

September 19, 2006

http://www.fml.mpg.de/raetsch/projects/gcbtutorial

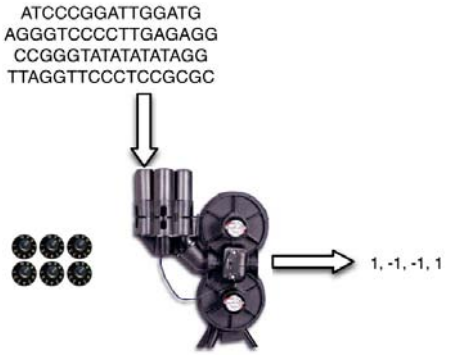
Tutorial Outline

- Machine learning & support vector machines
- Kernels
  - Basics
  - Substring kernels (Spectrum, WD, ...)
  - Efficient data structures
  - Other kernels (Fisher Kernel, ...)
- Some theoretical aspects
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

Classification of Sequences

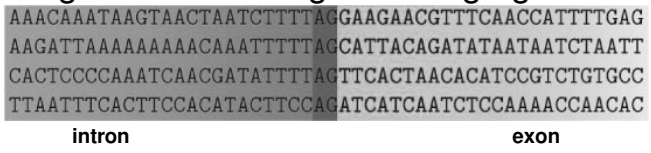
Example: Recognition of splice sites

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



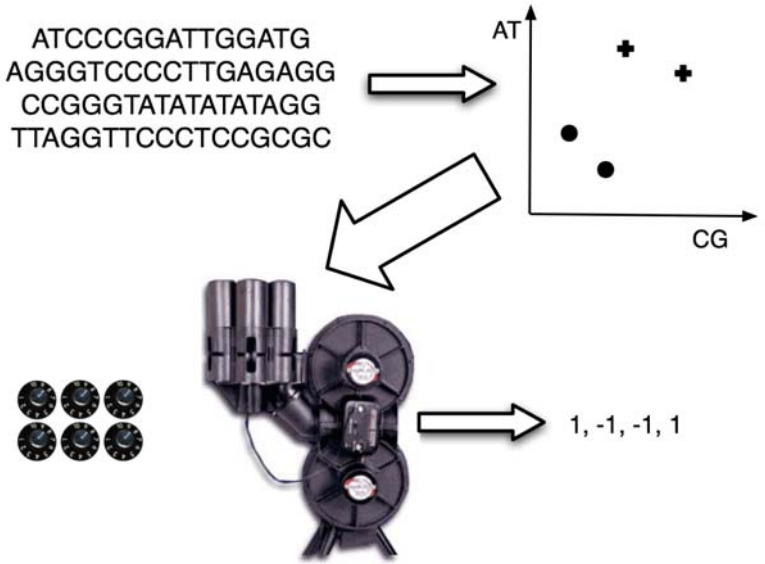
From Sequences to Features

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



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

Numerical Representation



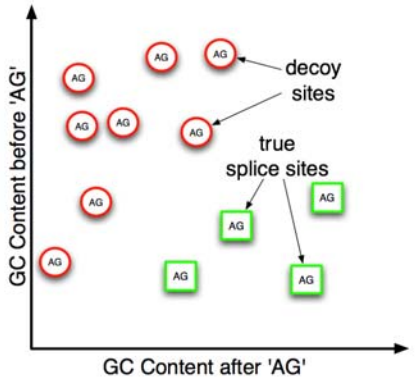
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron exon

- Goal: Rule that distinguishes true from false ones



e.g. exploit that exons have higher GC content

or

that certain motifs are located nearby

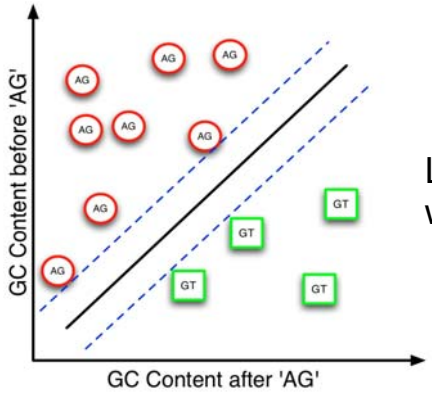
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

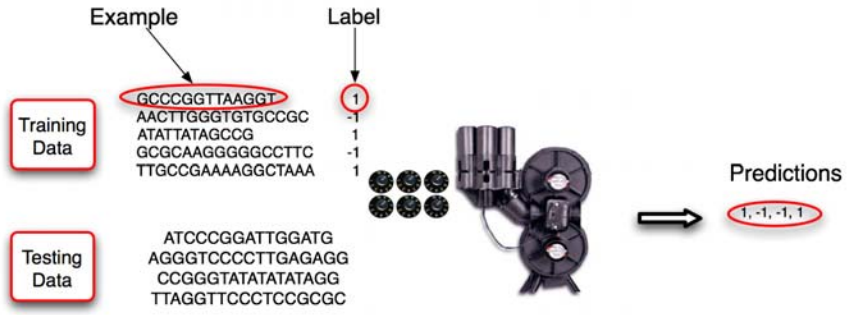
intron exon

- Goal: Rule that distinguishes true from false ones



Linear classifiers with large **margin**

Empirical Inference



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

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

Machine Learning: Main Tasks

Supervised Learning

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

Unsupervised Learning

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

Reinforcement Learning

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

How to measure performance?

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

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	...
GC before	0.6	0.2	0.4	0.3	0.2	0.4	0.5	0.5	...
GC after	0.7	0.7	0.3	0.6	0.3	0.4	0.7	0.6	...
AG**AG**AAG	0	0	0	1	1	0	0	1	...
TTT**AG**	1	1	1	0	0	1	0	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱
Label	+1	+1	+1	−1	−1	+1	−1	−1	...
	Training						Testing		

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

Measuring performance

What to do in practice

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

A. Cross validation

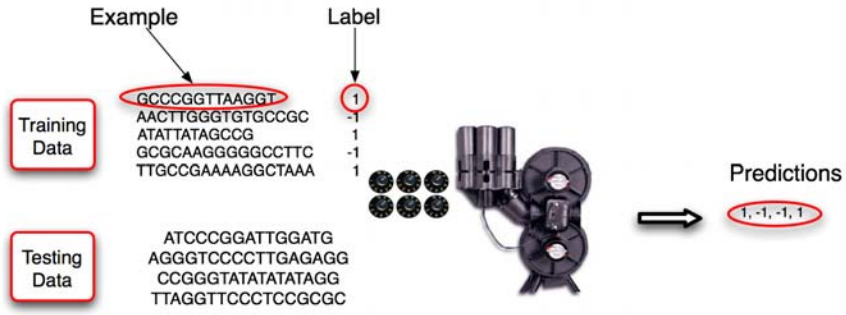
Split data into c disjoint parts, and use each subset as the validation set, while using the rest as the training set.

B. Random splits

Randomly split the data set into two parts, for example 80% of the data for training and 20% for validation. This is usually repeated many times.

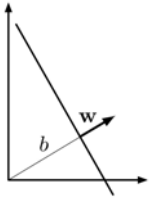
Report mean and standard deviation of performance on the va

Classifier: depends on training data

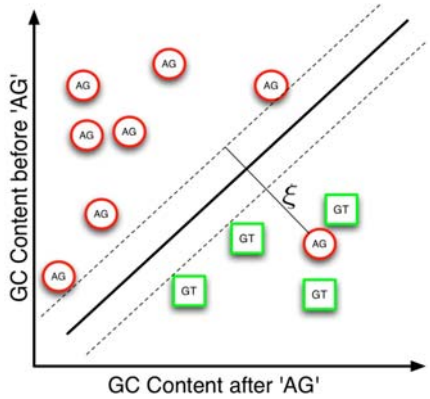


Consider linear classifiers with parameters $\mathbf{w}, b$:

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



Classifier: SVM



Minimize

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

Subject to

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

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

SVM is dependent on training data

Minimize

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

Subject to

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

Minimize

$$\frac{1}{2}\sum_{i,j}^{N}\alpha_i\alpha_j\langle \mathbf{x}_i, \mathbf{x}_j\rangle + C\sum_{i=1}^{N}\xi_i$$

Subject to

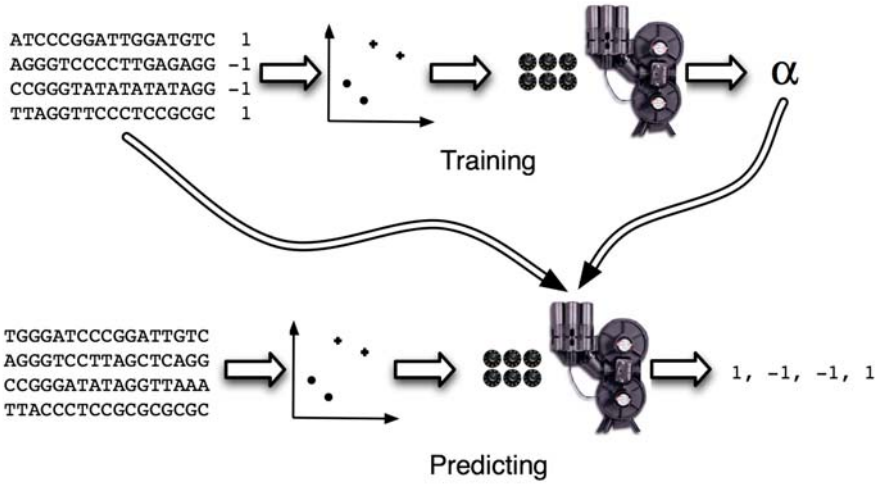
$$y_i(\langle \sum_{j=1}^{N}\alpha_j\mathbf{x}_j, \mathbf{x}_i\rangle + b) \geqslant 1 - \xi_i$$
$$\xi_i \geqslant 0$$
for all $i = 1, \ldots, N$.

Representer Theorem

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

SVM solution only depends on scalar products between examples ($\rightsquigarrow$ kernel trick)

Summary: Empirical Inference

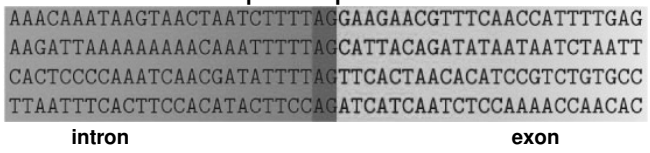


Tutorial Outline

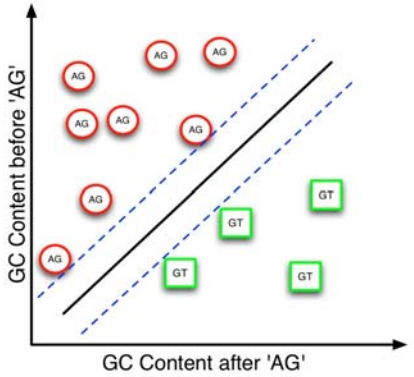
- Machine learning & support vector machines
- Kernels
  - Basics
  - Substring kernels (Spectrum, WD, . . .)
  - Efficient data structures
  - Other kernels (Fisher Kernel, . . .)
- Some theoretical aspects
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

Recognition of Splice Sites

- Given: Potential acceptor splice sites



- Goal: Rule that distinguishes true from false ones



Linear Classifiers with large **margin**

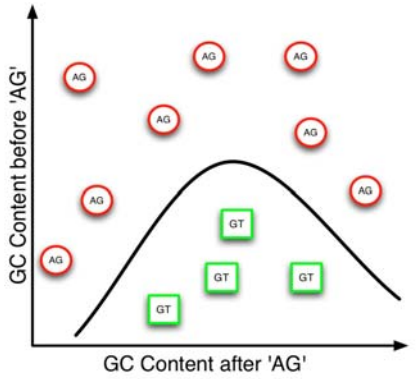
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron exon

- Goal: Rule that distinguishes true from false ones



More realistic problem!?

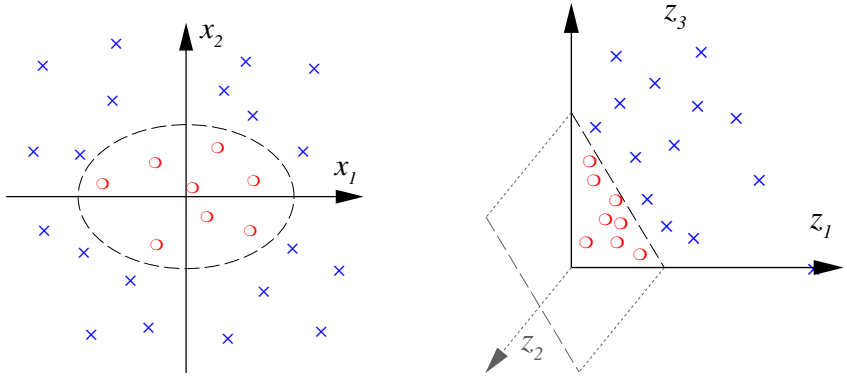
- Not linearly separable!
- **Need nonlinear separation!?**
- Need more features!?

Nonlinear Algorithms in Feature Space

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

Example: all second order monomials

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



Kernel "Trick"

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

$$\begin{aligned}\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle &= \left\langle (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2), (y_1^2, \sqrt{2}\, y_1 y_2, y_2^2) \right\rangle \\ &= \langle (x_1, x_2), (y_1, y_2) \rangle^2 \\ &= \langle \mathbf{x}, \mathbf{y} \rangle^2 \\ &=: \mathrm{k}(\mathbf{x}, \mathbf{y})\end{aligned}$$

- Scalar product in feature space (here $\mathbb{R}^3$) can be computed in input space (here $\mathbb{R}^2$)!
- Also works for higher orders and dimensions
  ⇒ relatively low dimensional input spaces
  ⇒ very high dimensional feature spaces
- works only for Mercer Kernels $\mathrm{k}(\mathbf{x}, \mathbf{y})$

Kernology I

If k is a continuous kernel of a positive integral operator on $L_2(\mathcal{D})$ (where $\mathcal{D}$ is some compact space),

$$\int f(\mathbf{x}) \mathrm{k}(\mathbf{x}, \mathbf{y}) f(\mathbf{y})\, d\mathbf{x}\, d\mathbf{y} \geq 0, \quad \text{for} \quad f \neq 0$$

it can be expanded as

$$\mathrm{k}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N_F} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{y})$$

with $\lambda_i > 0$, and $N_F \in \mathbb{N}$ or $N_F = \infty$. In that case

$$\Phi(\mathbf{x}) := \begin{pmatrix} \sqrt{\lambda_1}\psi_1(\mathbf{x}) \\ \sqrt{\lambda_2}\psi_2(\mathbf{x}) \\ \vdots \end{pmatrix}$$

satisfies $\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle = \mathrm{k}(\mathbf{x}, \mathbf{y})$ [Mercer, 1909].

Kernology II

Common kernels: [Vapnik, 1995, Müller et al., 2001, Schölkopf and Smola, 2002]

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

Notes:

- A kernel implies a mapping Φ to a feature space. In this potentially infinite dimensional space one finds a linear separation hyperplane.
- Every kernel corresponds to a regularization operator implying different smoothness properties in input space.

SVMs with kernels (Primal)

Minimize

$$\frac{1}{2}\sum_{i,j}^{N}\alpha_i\alpha_j\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)\rangle + C\sum_{i=1}^{N}\xi_i$$

Subject to

$$y_i\left(\sum_{j=1}^{N}\alpha_j\langle\Phi(\mathbf{x}_j),\Phi(\mathbf{x}_i)\rangle + b\right) \geqslant 1-\xi_i$$
$$\xi_i \geqslant 0$$
for all $i = 1,\ldots,N$.

Minimize

$$\frac{1}{2}\sum_{i,j}^{N}\alpha_i\alpha_j\mathrm{k}(\mathbf{x}_i,\mathbf{x}_j) + C\sum_{i=1}^{N}\xi_i$$

Subject to

$$y_i\left(\sum_{j=1}^{N}\alpha_j\mathrm{k}(\mathbf{x}_j,\mathbf{x}_i) + b\right) \geqslant 1-\xi_i$$
$$\xi_i \geqslant 0$$
for all $i = 1,\ldots,N$.

°Hyperplane $y = \mathrm{sign}\left(\langle\mathbf{w},\Phi(\mathbf{x})\rangle + b\right)$ in $\mathcal{F}$

$$\begin{aligned}
\text{minimize} \quad & \|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\
\text{w.r.t.} \quad & \mathbf{w}\in\mathcal{F}, b\in\mathbb{R}, \quad \xi_i \geq 0 \quad (i=1,\ldots,N) \\
\text{subject to} \quad & y_i(\langle\mathbf{w},\Phi(\mathbf{x}_i)\rangle + b) \geq 1-\xi_i \quad (i=1\ldots N)
\end{aligned}$$

Lagrangian with multipliers $\alpha_i \geq 0$ $(i = 1,\ldots,N)$:

$$L(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N}\alpha_i\left(y_i(\langle\mathbf{w},\Phi(\mathbf{x}_i)\rangle) + b\right) - 1\right).$$

Obtain unique α_i by QP: dual problem

$$\frac{\partial}{\partial b}L(\mathbf{w},b,\alpha) = 0, \quad \frac{\partial}{\partial\mathbf{w}}L(\mathbf{w},b,\alpha) = 0,$$

$$\Rightarrow \quad \sum_{i=1}^{N}\alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i\Phi(\mathbf{x}_i).$$

Substitute both into L to get the *dual problem*

°Dual problem

$$\begin{aligned}
\text{maximize} \quad & W(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \underbrace{\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)\rangle}_{=k(\mathbf{x}_i,\mathbf{x}_j)} \\
\text{subject to} \quad & 0 \leq \alpha_i \leq C \quad (i=1,\ldots,N) \quad \text{and} \quad \sum_{i=1}^{N}\alpha_i y_i = 0.
\end{aligned}$$

Note: solution is determined by training examples (SVs) on the edge or in the margin area:

$$\begin{aligned}
y_i[\langle\mathbf{w},\Phi(\mathbf{x}_i)\rangle + b] > 1 \quad & \Rightarrow \alpha_i = 0 \Rightarrow \mathbf{x}_i \text{ irrelevant} \\
y_i[\langle\mathbf{w},\Phi(\mathbf{x}_i)\rangle + b] \leq 1 \quad & \text{(in margin area)} \Rightarrow \mathbf{x}_i \text{ Support Vector}
\end{aligned}$$

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

SVMs with kernels (Primal & Dual)

Minimize

$$\frac{1}{2}\sum_{i,j}^{N}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) + C\sum_{i=1}^{N}\xi_i$$

Subject to

$$y_i\left(\sum_{j=1}^{N}\alpha_j k(\mathbf{x}_j,\mathbf{x}_i)+b\right) \geqslant 1-\xi_i$$
$$\xi_i \geqslant 0 \quad \text{for all } i=1,\ldots,N.$$

Maximize

$$\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j}^{N}\alpha_i\alpha_j\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x}_j)\rangle$$

Subject to

$$\sum_{i=1}^{N}\alpha_i y_i = 0$$
$$0 \leq y_i\alpha_i \leq C \quad \text{for all } i=1,\ldots,N.$$

Summary "Kernel Trick"

- Representer Theorem: $\mathbf{w} = \sum_{i=1}^{N}\alpha_i\Phi(\mathbf{x}_i)$.
- Hyperplane in $\mathcal{F}$: $y = \text{sgn}\left(\langle\mathbf{w},\Phi(\mathbf{x})\rangle + b\right)$
- Putting things together

$$\begin{aligned} f(\mathbf{x}) &= \text{sgn}\left(\langle\mathbf{w},\Phi(\mathbf{x})\rangle + b\right) \\ &= \text{sgn}\left(\sum_{i=1}^{N}\alpha_i\langle\Phi(\mathbf{x}_i),\Phi(\mathbf{x})\rangle + b\right) \\ &= \text{sgn}\left(\sum_{i:\alpha_i\neq 0}\alpha_i k(\mathbf{x}_i,\mathbf{x}) + b\right) \end{aligned}$$

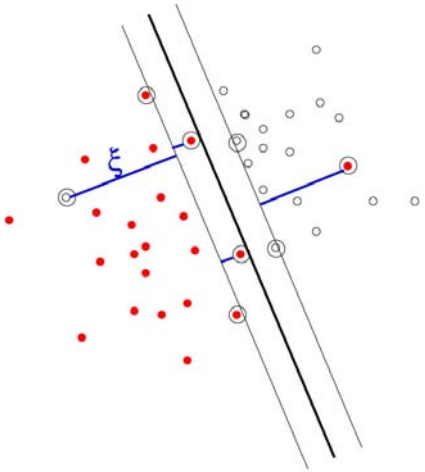
sparse!

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

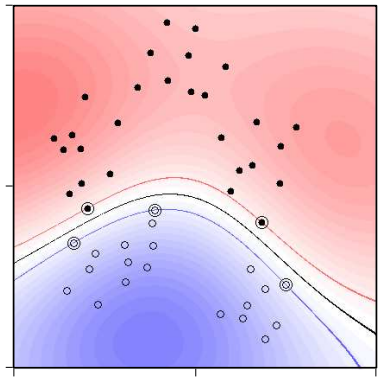
See e.g. Vapnik [1995], Müller et al. [2001], Schölkopf and Smola [2002] for details.

Toy Examples

Linear kernel
$k(\mathbf{x},\mathbf{y}) = \langle\mathbf{x},\mathbf{y}\rangle$



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



Tutorial Outline

- Machine learning & support vector machines
- Kernels
  - Basics
  - Substring kernels (Spectrum, WD, ...)
  - Efficient data structures
  - Other kernels (Fisher Kernel, ...)
- Some theoretical aspects
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

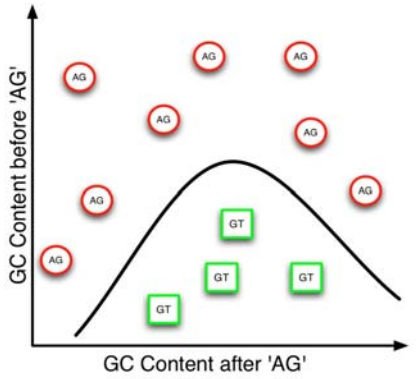
Recognition of Splice Sites

- Given: Potential acceptor splice sites

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron exon

- Goal: Rule that distinguishes true from false ones



More realistic problem!?

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

More Features?

Some ideas:

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

Approaches:

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

Best in practice: Combination of both

Spectrum Kernel

General idea [Leslie et al., 2002]

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

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

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

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

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

Spectrum Kernel

Principle

- Spectrum kernel: Count exactly common ℓ-mers

Protein A: ILVFMC

Protein B: WLVFQC

Common 1-mers
L, V, F, C

Common 2-mers
LV
VF

Common 3-mers
LVF

- $\Phi(\mathbf{x})$ has only very few non-zero dimensions
  $\Rightarrow$ efficient kernel computations possible ($\mathcal{O}(|\mathbf{x}| + |\mathbf{x}'|)$)

Substring Kernels



General idea

- Count common substrings in two strings
- Sequences are deemed the more similar, the more common substrings they contain

Variations

- Allow for gaps
- (Include wildcards)
- Allow for mismatches
- (Include substitutions)
- Motif Kernels
- (Assign weights to substrings)

Gappy Kernel



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

- Allow for gaps in common substrings
  $\rightarrow$ "subsequences"
- A g-mer then contributes to all its ℓ-mer subsequences

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

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

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

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

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

Gappy Kernel



Principle

- Gappy kernel: Count common ℓ-subsequences of g-mers

g=3, l=2

Protein A: ILVFMC

ILV: IL, IV, LV
LVF: LV, VF, LF
VFM: VF, FM, VM
FMC: FM, FC, MC

Protein B: WLVFQC

WLV: WL, LV, WV
LVF: LV, VF, LF
VFQ: VF, FQ, VQ
FQC: FQ, QC, FC

Wildcard Kernels



General idea [Leslie and Kuang, 2004]

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

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

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

Wildcard Kernels



Principle

- Wildcard kernel: Count ℓ-mers that match except for wildcards

l=3, m=1

Protein A: ILVFMC

ILV: IL*, *LV, W*L
LVF: LV*, *VF, L*F
VFM: VF*,*FM, V*M
FMC: FM*, *MC, F*C

Protein B: WLVFQC

WLV: WL*, *LV, W*V
LVF: LV*, *VF, L*F
VFQ: VF*, *FQ, V*Q
FQC: FQ*, *QC, F*C

Mismatch Kernel



General idea [Leslie et al., 2003]

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

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

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

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

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

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

Mismatch Kernel



Principle

- Mismatch kernel: Count common ℓ-mers with max. m mismatches

Protein A: ILVFMC

Common 3-mers with max. 1 mismatch:

Protein B: WLVFQC

ILV - WLV
LVF - LVF
VFM - VFQ
FMC - FQC

Substitution Kernel



General idea [Leslie and Kuang, 2004]

- mismatch neighborhood → substitution neighborhood
- An ℓ-mer then contributes to all ℓ-mers in its substitution neighborhood

$$M_{(l,\sigma)}(s) = \{\beta = b_1 b_2 \dots b_\ell \in \Sigma^\ell : -\sum_i^\ell \log P(a_i|b_i) < \sigma\}$$

- For a sequence $\mathbf{x}$ of any length, the map is then extended as

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

- The substitution kernel is now:

$$k^{\text{Sub}}_{(l,\sigma)}(\mathbf{x},\mathbf{x}') = \langle \Phi^{\text{Sub}}_{(l,\sigma)}(\mathbf{x}), \Phi^{\text{Sub}}_{(l,\sigma)}(\mathbf{x}') \rangle$$

Substitution Kernel

Principle

- Substitution kernel: Count common ℓ-subsequences in substitution neighborhood

Protein A: ILVFMC

Protein B: WLVFQC

Substitution probabilities:
W-I : 0.5
M-Q: 0.5
all others: 0

ILV: ILV, WLV
LVF: LVF
VFM: VFM, VFQ
FMC: FMC, FQC

WLV: WLV, ILV
LVF: LVF
VFQ: VFQ, VFM
FQC: FQC, FMC

Motif kernels

General idea

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

Motif databases

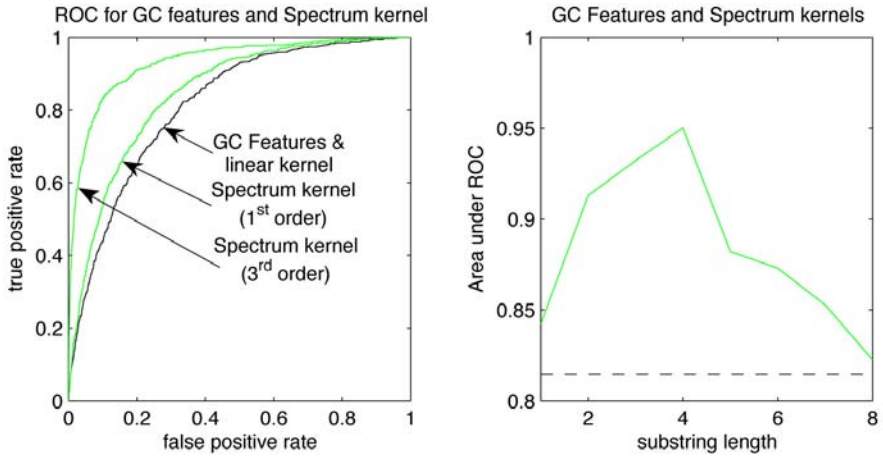
- Protein: Pfam, PROSITE, ...
- DNA: Transfac, Jaspar, ...
- RNA: Rfam, Structures, Regulatory sequences, ...

Generated by

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

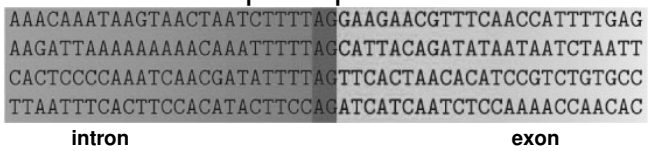
Simulation Example

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



Position Dependence

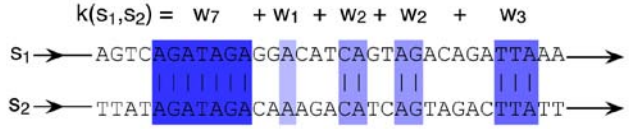
- Given: Potential acceptor splice sites



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

Weighted Degree Kernel

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

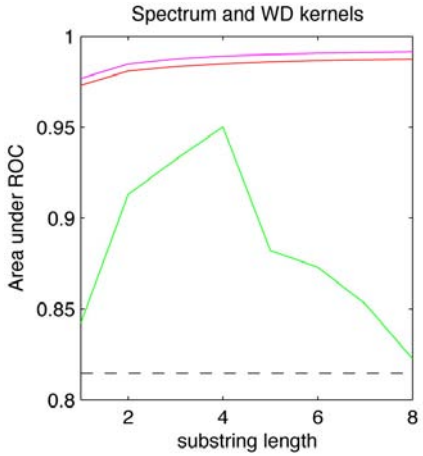


Equivalent to a mixture of spectrum kernels (up to order ℓ) at every position for appropriately chosen weights w (depending on ℓ).

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



Substring Kernel Comparison



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

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

Tutorial Outline

- Machine learning & support vector machines
- Kernels
  - Basics
  - Substring kernels (Spectrum, WD, . . .)
  - Efficient data structures
  - Other kernels (Fisher Kernel, . . .)
- Some theoretical aspects
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

Fast string kernels?

Direct approach is slow

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

Solution

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

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

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

Efficient data structures

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

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

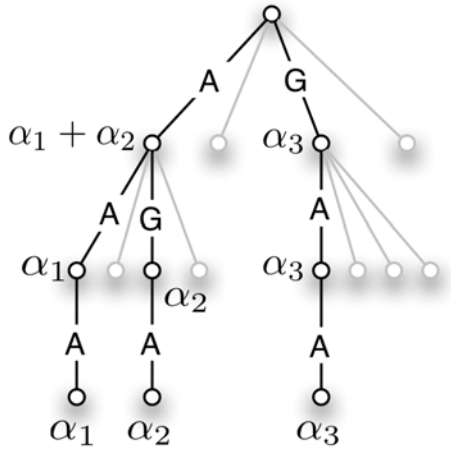
- Use trees or arrays to store only non-zero elements

$\Rightarrow$ Substring is the index into the tree or array

- Leads to more efficient optimization algorithms:
  - Precompute $v = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i)$
  - Compute $\sum_{i=1}^{N} \alpha_i \mathrm{k}(\mathbf{x}_i, \mathbf{x})$ by

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

Example: Trees & Tries



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

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

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

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

Results with WD Kernel (human acceptors)

N	Computing time (s) WD	WD $w/$ $tries$	ROC (%)
500	17	83	75.61
1,000	17	83	79.70
5,000	28	105	90.38
10,000	47	134	92.79
30,000	195	266	94.73
50,000	441	389	95.48
100,000	1,794	740	96.13
500,000	31,320	7,757	96.93
1,000,000	102,384	26,190	97.20
2,000,000	-	(115,944)	97.36
5,000,000	-	(764,144)	97.52
10,000,000	-	(2,825,816)	97.64
10,000,000	PWMs		96.03

Tutorial Outline

- Machine learning & support vector machines
- Kernels
  - Basics
  - Substring kernels (Spectrum, WD, . . .)
  - Efficient data structures
  - Other kernels (Fisher Kernel, . . .)
- Some theoretical aspects
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

Fisher Kernel



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

- Combine probabilistic models and SVMs
- Best-paper award at ISMB 1999

Sequence representation

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

Example: Fisher Kernel on PWMs



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

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

Pairwise comparison kernels



General idea [Liao and Noble, 2002]

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

Advantage

- Utilizes decades of practical experience with Blast

Disadvantage

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

Alleviation

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

Local Alignment Kernel



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

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

An alignment π is then scored as follows:

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

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

Smith-Waterman score (not positive definite)

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

Local Alignment Kernel [Vert et al., 2004]

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

Haussler's R-convolution kernel



Composite Objects

- Objects that consist of substructures, e.g.
- a graph consists of nodes and edges
- a string consists of substrings

Haussler's idea

- Build kernel for composite objects from kernels on *substructures*.

Mathematical prerequisites

- Object $\mathbf{x} \in \mathcal{X}$ is composed of parts $\mathbf{x}_d \in \mathcal{X}_d$, where $d = 1, \ldots, D$
- R is a relation such that $R(\mathbf{x}_1, \ldots, \mathbf{x}_D, \mathbf{x}) = 1$ iff $\mathbf{x}_1, \ldots, \mathbf{x}_D$ constitute the composite object $\mathbf{x}$; R is zero otherwise

Haussler's R-convolution kernel



R-convolution

- k_d is a kernel defined on $\mathcal{X}_d$.
- Then the R-convolution of $\mathrm{k}_1, \ldots, \mathrm{k}_D$ is

$$(\mathrm{k}_1 \star \ldots \star \mathrm{k}_D)(\mathbf{x}, \mathbf{x}') := \sum_R \prod_{d=1}^{D} \mathrm{k}_d(\mathbf{x}_d, \mathbf{x}'_d)$$

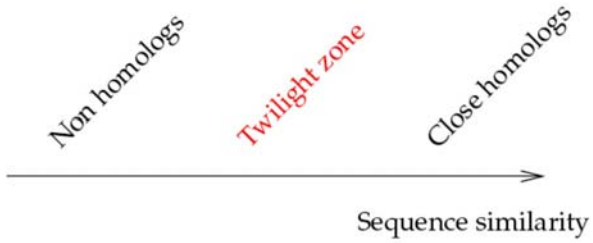
- For R finite, this is a valid kernel.

Meaning

- $\mathbf{x}$ and $\mathbf{x}'$ and compared by comparing all their decompositions into parts
- decompositions are compared via kernels on parts

Application: Remote Homology



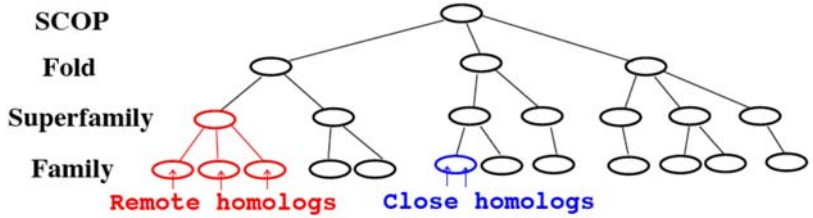


- Homologs have common ancestors
- Structures and functions are more conserved than sequences
- Remote homologs can not easily be detected by direct sequence comparison

(Thanks to J.-P. Vert for providing the slides on remote homology detection.)

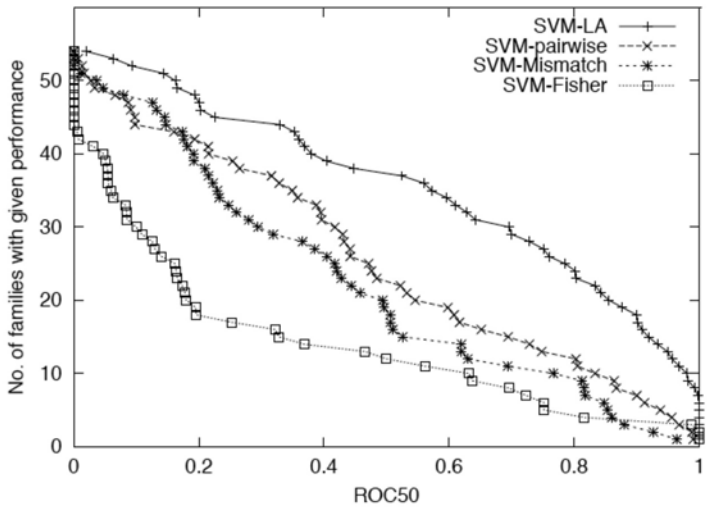
SCOP Database & Experiment





- Goal: Recognize the superfamily
- Training: for a sequence, positive examples come from the same superfamily, but different family. Negative examples come from other superfamilies
- Test: Predict the superfamily

Difference in Performance



- Performance on SCOP superfamily benchmark [Vert et al., 2004]
- ROC50 is the area under the ROC curve up to the first 50 FPs

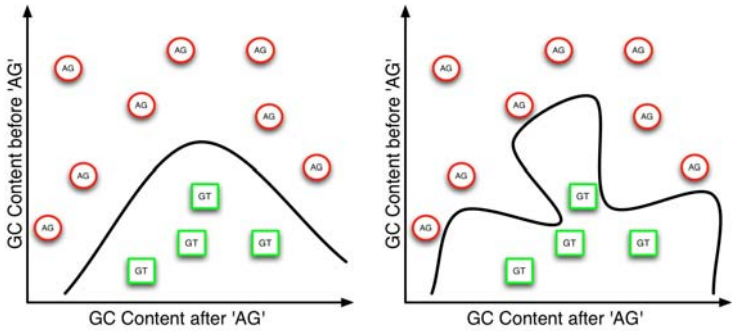
Kernel Summary

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

Tutorial Outline

- Machine learning & support vector machines
- Kernels
  - Substring kernels (Spectrum, WD, . . .)
  - Other kernels (Fisher Kernel, . . .)
- Some theoretical aspects
  - Margins & Complexity Control
  - Model Selection
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

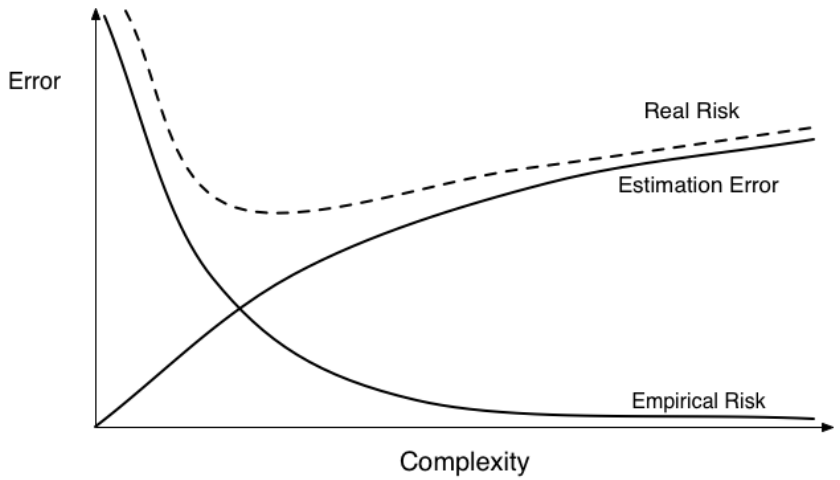
Simple vs Complex Functions



For a given set of training data, there are many possible functions which can explain it. However, some functions are “simple” and others are “complex”. We want to estimate a functional dependence from a set of examples.

Which function is preferable?

Structural Risk Minimization



The complexity or capacity is a property of the function class, and not any individual function f.

VC Dimension

A model class shatters a set of data points if it can correctly classify any possible labelling.



Lines shatter any 3 points in $\mathbb{R}^2$, but not 4 points.

VC dimension [Vapnik, 1995]

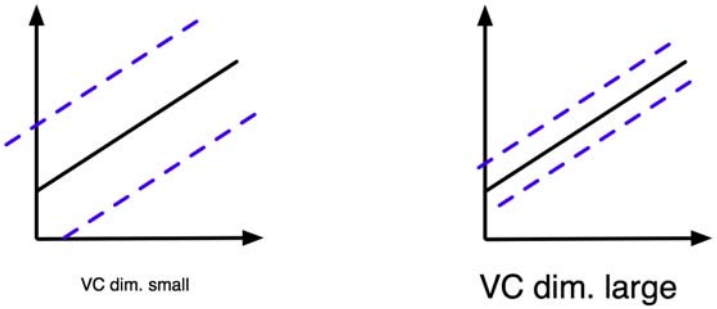
The VC dimension of a model class is the maximum h such that some data point set of size h can be shattered by the model. (e.g. VC dimension of $\mathbb{R}^2$ is 3.)

Complex model classes have large VC dimension.

Larger Margin $\Rightarrow$ Less Complex

Large Margin $\Rightarrow$ Small VC dimension

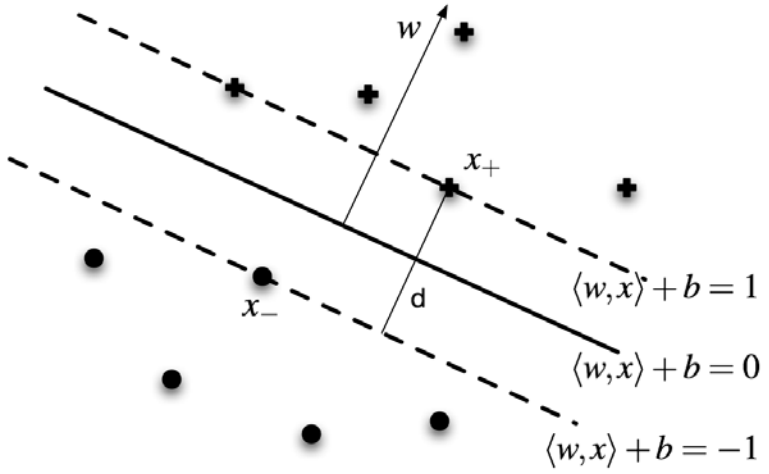
Hyperplane classifiers with large margin have small VC dimension [Vapnik, 1995].



Maximum Margin $\Rightarrow$ Minimum Complexity

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

Margin Maximization



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

SVM: Geometric View

$$\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \qquad (1)$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geqslant 1 - \xi_i \text{ and } \xi_i \geqslant 0 \qquad (2)$$

for all $i = 1, \ldots, N$.

Objective function (1) Maximize the margin.

Constraints (2) Correctly classify the training data.

The slack variables ξ allow points to be in the margin, but penalize them in the objective.

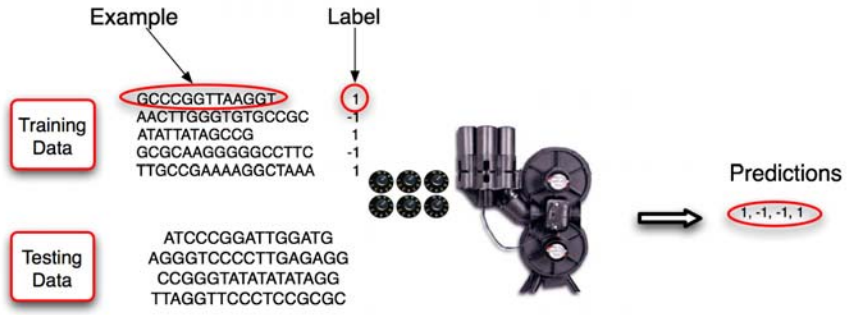
Why maximize the margin?

- 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.
- There is one global maximum, i.e. the problem is convex.

Tutorial Outline

- Machine learning & support vector machines
- Kernels
  - Substring kernels (Spectrum, WD, . . .)
  - Other kernels (Fisher Kernel, . . .)
- Some theoretical aspects
  - Margins & Complexity Control
  - Model Selection
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

Review: Generalization Error



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

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

Empirical Risk $\mathrm{R}_{emp(f)}$. The average loss on training data.

Measuring performance



What to do in practice

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

A. Cross validation

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

B. Random splits

Randomly split the data set into two parts, for example 80% of the data for training and 20% for validation. This is usually repeated many times.

See e.g. 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

- SVM parameter C.
- Kernel parameters: e.g. subsequence length, degree of kernel, amount of shift.

Tutorial Outline

- Machine learning & support vector machines
- Kernels
  - Substring kernels (Spectrum, WD, . . .)
  - Other kernels (Fisher Kernel, . . .)
- Some theoretical aspects
  - Margins & Complexity Control
  - Model Selection
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

Estimators

Basic Notion

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

Formally

We want to choose an estimator

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

Intuition

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

Question

How do we measure how well we are doing?

Loss Function



Basic Notion

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

Formally

We define a loss function

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

Intuition

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

Example: Error rate

For binary classification,

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

Soft Margin SVM



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

Objective function

By minimizing the squared norm of the weight vector, we maximize the margin.

Constraints

We can express the constraints in terms of a loss function.

SVM: Loss View



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

where

$$\begin{aligned} \ell(f_{\mathbf{w},b}(\mathbf{x}), y_i) &:= C\max\{0, 1 - y_i(f_{\mathbf{w},b}(\mathbf{x}))\} \\ f_{\mathbf{w},b}(\mathbf{x}) &:= \mathbf{w}^\top\mathbf{x}_i + b. \end{aligned}$$

The above loss function is known as the hinge loss.

Regularizer = $\frac{1}{2}\|\mathbf{w}\|^2$.

Empirical Risk = $\sum_{i=1}^{N}\ell(\mathbf{w}^\top\mathbf{x}_i + b, y_i)$.

How much does a mistake cost us?

Loss Functions



0-1 loss

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

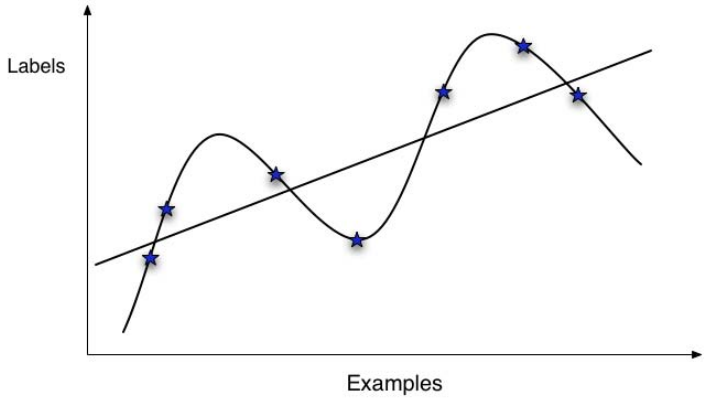
hinge loss

$$\ell(f(\mathbf{x}_i), y_i) := \max\{0, 1 - y_i f(\mathbf{x}_i)\}$$

logistic loss

$$\ell(f(\mathbf{x}_i), y_i) := \log(1 + \exp(-y_i f(\mathbf{x}_i)))$$

Regression



examples $x \in \mathcal{X}$

labels $y \in \mathbb{R}$

Regression

ε-insensitive loss function

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

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

Squared loss

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

Huber's loss

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

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

Multiclass

Real problems often have more than 2 classes

Generalize the SVM to multiclass, for $c > 2$.

Three approaches [Schölkopf and Smola, 2002]

one-vs-rest

For each class, label all other classes as "negative" (c binary problems).

one-vs-one

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

multiclass loss

Define a new empirical risk term.

Multiclass Loss for SVM

Two class SVM

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

Multiclass SVM

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

Convex Optimization



- SVMs are a special case of Quadratic Programs (QPs)
- QPs can be efficiently solved via constrained optimization.
  For $f_i : \mathbb{R}^N \to \mathbb{R}$ and $g_j : \mathbb{R}^N \to \mathbb{R}$:

$$\begin{array}{ll} \min_{x \in \mathbb{R}^N} & f_0(x) \\ \text{subject to} & f_i(x) \leqslant 0 \text{ for } i = 1, \ldots, m \,, \\ & g_j(x) = 0 \text{ for } j = 1, \ldots, p \end{array}$$

- There exists many open source and commercial packages for solving convex optimization problems.

QPs for SVMs



- General Purpose QP solver (e.g. CPLEX [CPL, 1994])
  - Does not exploit problem structure.
- Chunking Methods [Osuna et al., 1997]
  - Select subsets, solve QPs, join the sets, ...
- SVM-Light [Joachims, 1999]
  - Select n variables, solve QP, ...
- SMO Algorithm [Platt, 1999]
  - Select two variables, solve QP analytically, ...
- ...

Shogun toolbox Sonnenburg et al. [2006a]

- SVM-Light type QP optimization
- Many string kernels implementations

Tutorial Outline



- Machine learning & support vector machines
- Kernels
  - Substring kernels (Spectrum, WD, ...)
  - Other kernels (Fisher Kernel, ...)
- Some theoretical aspects
  - Margins & Complexity Control
  - Model Selection
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

Risk and Regularization



Basic Notion

In general, we can think of an SVM as optimizing a particular cost function,

$$\Omega(\mathbf{w}) + \mathbf{R}_{emp}(\mathbf{w}),$$

where $\mathbf{R}_{emp}(\mathbf{w})$ is the empirical risk measured on the training data, and $\Omega(\mathbf{w})$ is the regularizer.

Regularization

The regularizer is a function which measures the complexity of the function.

General principle

There is a trade-off between fitting the training set well (low empirical risk) and having a "simple" function (small regularization term).

Soft Margin SVM



General principle

There is a trade-off between fitting the training set well (low empirical risk) and having a "simple" function (small regularization term).

General equation

$$\Omega(\mathbf{w}) + \mathbf{R}_{emp}(\mathbf{w}),$$

Soft Margin SVM

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

Representer Theorem



Let $\Omega : [0, \infty) \to \mathbb{R}$ be a strictly monotonic increasing function and $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ a loss function. Then each minimizer $(\mathbf{w}, b)$ of the regularized risk

$$\sum_{i=1}^{N} \ell\left(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b, y_i\right) + \Omega\left(\|\mathbf{w}\|\right) \tag{3}$$

admits a representation of the form

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i) \quad \Rightarrow \quad f_{\mathbf{w},b}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b. \tag{4}$$

where k is the reproducing kernel of $\mathcal{H}$, and $\alpha_i \in \mathbb{R}$ for all $i = 1, \ldots, m$.

$\|w\|^2$ term in SVM allows us to use kernels.

See e.g. Kimeldorf and Wahba [1971], Vapnik [1995], Schölkopf and Smola [2002].

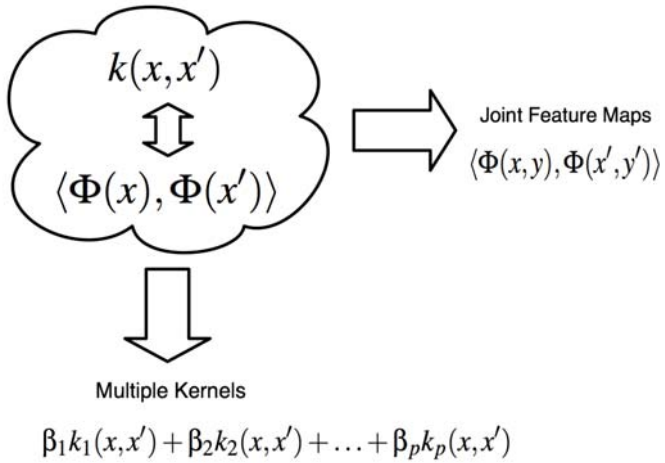
Tutorial Outline



- Machine learning & support vector machines
- Kernels
  - Substring kernels (Spectrum, WD, ...)
  - Other kernels (Fisher Kernel, ...)
- Some theoretical aspects
  - Margins & Complexity Control
  - Model Selection
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications

Generalizing kernels





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

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.

Interdependent Outputs

For example a hierarchy of classes like the EC classes or part of speech tagging.

Label Sequence Learning

An example of a very large set of $\mathcal{Y}$ is all possible labellings of the secondary structure elements of the amino acid sequence.

- Protein secondary structure prediction (α/β/coils)
- Gene structure prediction (intergenic/exon/intron)

Joint Feature Map



Recall the kernel trick

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

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

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

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

For multiclass

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

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

Learning with two kernels



Kernel Methods

For a particular kernel $k(\mathbf{x}, \mathbf{x}')$, we can find the optimal separating hyperplane using a SVM.

What if we have two kernels?

For example, we may have a kernel measuring the amino acid sequence similarity and another kernel measuring the secondary structure similarity.

Possible solution

We can add the two kernels, that is

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

Multiple Kernel Learning (MKL)



Better solution

We can mix the two kernels,

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

where $t$should be estimated from the training data.

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

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

Applications

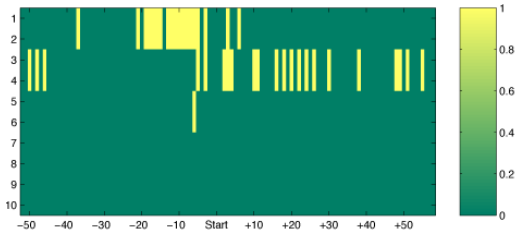
- Heterogeneous data
- Improving interpretability

Method for Interpreting SVMs

- Weighted Degree kernel: linear comb. of LD kernels

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

- Example: Classifying splice sites



See Rätsch et al. [2006] for more details.

Summary of Kernel Methods

- The capacity or complexity of a function class.
- Principle of structural risk minimization
- Two views of SVM:
  - Maximum margin algorithm.
  - Minimization of a loss function.
- Estimating expected risk from empirical risk (validation).
- Convex optimization
- Further generalizations for bioinformatics.

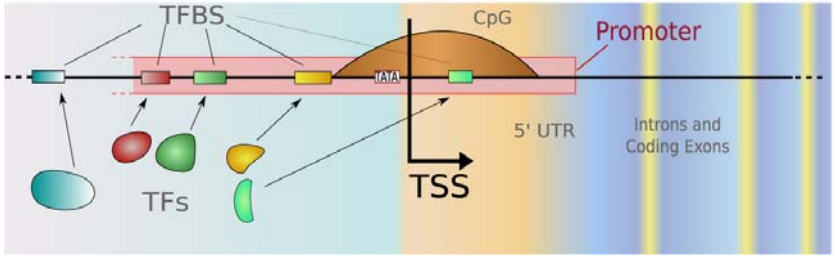
Tutorial Outline

- Machine learning & support vector machines
- Kernels
  - Substring kernels (Spectrum, WD, . . .)
  - Other kernels (Fisher Kernel, . . .)
- Some theoretical aspects
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications
  - Transcription start site prediction
  - Prediction of alternative splicing

Applications

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

Transcription Start Sites - Properties



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

SVMs with 5 sub-kernels

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

Training – Data Generation

True TSS:

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

Decoy TSS:

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

Processing:

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

Training & Model Selection

16 kernel parameters + SVM regularization to be tuned!

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

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

Speedup trick:

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

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

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

Experimental Comparison

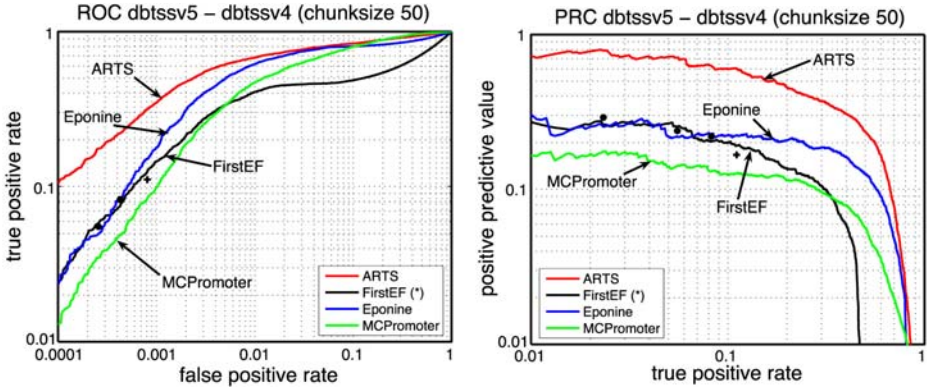
Current state-of-the-art methods:

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

⇒ **Do a genome wide evaluation!**
⇒ **How to do a fair comparison?**

Results

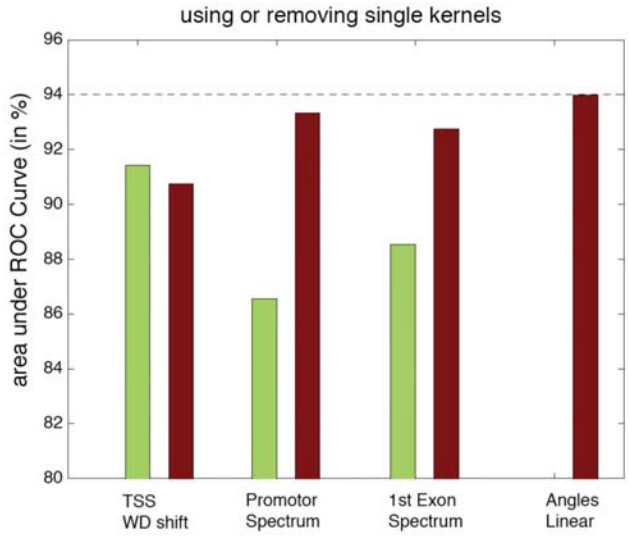
Receiver Operator Characteristic Curve *and* Precision Recall Curve



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

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

Which kernel is most important?

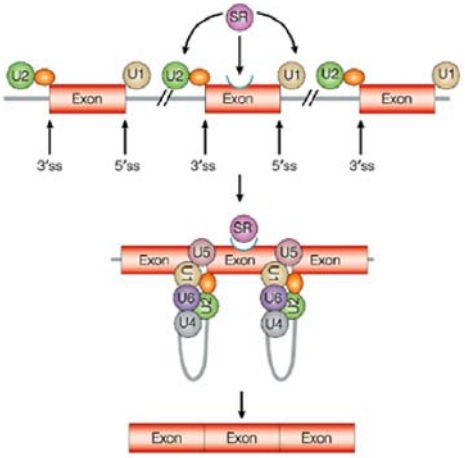


⇒ **Weighted Degree Shift kernel modeling TSS signal**

Tutorial Outline

- Machine learning & support vector machines
- Kernels
  - Substring kernels (Spectrum, WD, . . .)
  - Other kernels (Fisher Kernel, . . .)
- Some theoretical aspects
- Loss functions & Regularization
  - Regression & Multi-Class problems
  - Representer Theorem
- Extensions
- Applications
  - Transcription start site prediction
  - Prediction of alternative splicing

Splicing



Splice sites are

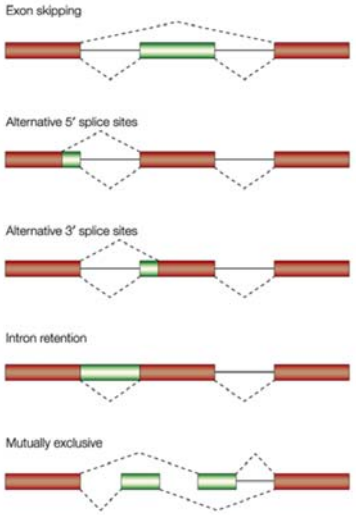
- exon/intron boundaries
- recognized by five snRNAs assembled in snRNPs
- flanked by regulatory elements

Spliceosomal Proteins

- interact with snRNPs/mRNA
- regulate recognition of splice sites
- can lead to alternative transcripts

Alternative splicing

One gene may correspond to several transcripts/proteins

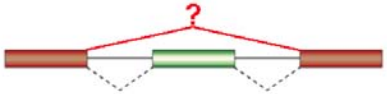


Use Machine Learning to

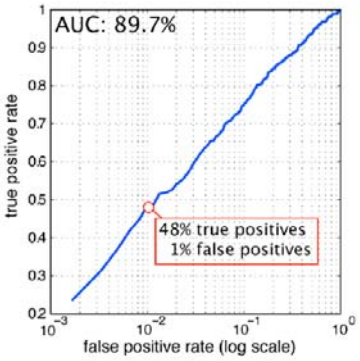
- analyze sequences near splice sites
- understand differences between alternative and constitutive splicing
- exploit and identify regulative splicing elements
- predict yet unknown alternative splicing events

Exon Skipping: Two tasks

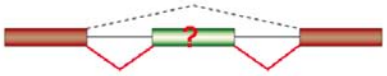
Exon is known,
can it be skipped?



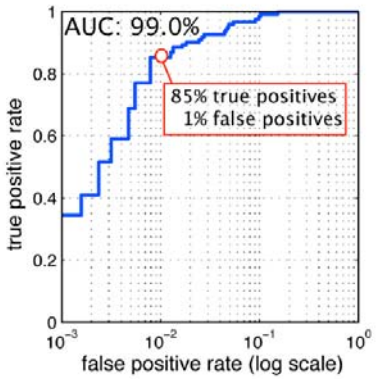
Task 1 (exon known)



Intron is known,
does it contain an exon?



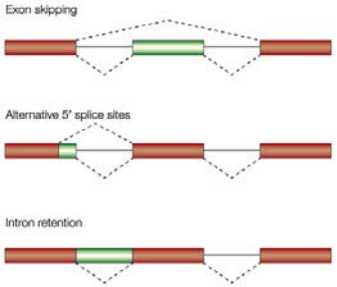
Task 2 (intron known)



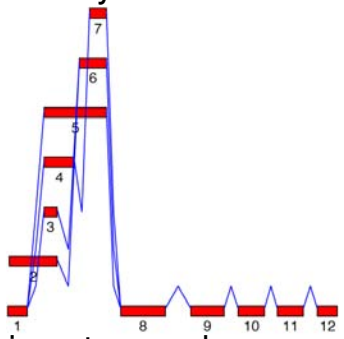
[Rätsch et al., 2005]

Empirical Inference Challenges

Simple classes:



Reality:



- Predicting the simple cases is not enough

$\Rightarrow$ need to predict the gene structure

- Difficult learning setting:
  - Input: DNA sequence
  - Output: Splicegraph (vertices & edges unknown)

References

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

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

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

Using the CPLEX Callable Library. CPLEX Optimization Incorporated, Incline Village, Nevada, 1994.

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

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

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

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

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

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

T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.

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

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

G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

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

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

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, pages 276–285, New York, 1997. IEEE.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

G. Rätsch and S. Sonnenburg. Accurate splice site detection for *Caenorhabditis elegans*. In K. Tsuda B. Schoelkopf and J.-P. Vert, editors, *Kernel Methods in Computational Biology*. MIT Press, 2004.

G. Rätsch, S. Sonnenburg, and B. Schölkopf. RASE: recognition of alternatively spliced exons in *C. elegans*. *Bioinformatics*, 21(Suppl. 1):i369–i377, June 2005.