

The “Kernel Trick”

A Computational Biology Example
using Support Vector Machines

Suzi Fei
January 13, 2009

Take-home message

Everything you need to remember in one slide

- Sometimes it improves your classifier if you map (i.e. transform) your data into a different feature space.
 - Example 1: you have nonlinear data but want to use a linear classifier
 - Example 2: your original data aren't numbers—for example they could be sequences
- Kernel methods map your original data into a different space so that you can use linear classifiers.
- This mapping can often substantially increase the number of features to consider.
 - This can be problematic as your number of dimensions grows.
- The “Kernel Trick” addresses this by putting a cap on the feature explosion so that the complexity of your classifier increases only linearly with the size of your original data.

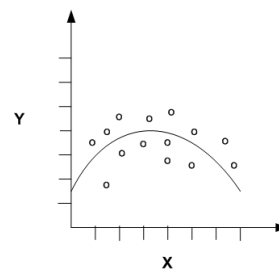
Introduction

- Kernel functions are *efficient* methods for computing the similarity between two objects using non-linear decision boundaries.
- Kernel methods are used in a number of statistical learning algorithms.
 - Support Vector Machines use them to determine the relative position or similarity of the points to each other.

3

Mapping from nonlinear to linear

- Nonlinear classifiers often provide better accuracy.
- However, it's handy to coerce the data into linear so that linear classifiers can be used because they have simple training algorithms and scale well with the size of the data.



$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \quad \rightarrow \quad f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b$$

Nonlinear Linear

4

Such a mapping often does not scale well with the number of dimensions.

- Examples:
 1. All the product pairs of features (monomials of degree 2).
 - Complexity grows quadratically with the number of dimensions in the input space.
 2. All the monomials of degree d .
 - Complexity grows exponentially.
- Kernel methods avoid this complexity by avoiding mapping data to high-dimensional feature space.

5

Using matrix voo doo to do the kernel trick

- The weight vector can be expressed as a linear combination of the training points.

$$f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b \quad \rightarrow \quad f(\mathbf{x}) = \sum_{i=1}^n y_i \alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b$$

- A kernel function can be constructed that utilizes this fact: $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$
 - Can be computed efficiently
 - Avoids mapping into very high dimensional space

6

One common kernel function: Polynomial

- Considers all monomials with degree up to d
(the one mentioned earlier where the complexity increased exponentially with the size of your data)
- In kernel form, the complexity increases only linearly
- Two parameters: d and κ
 - Simple linear discriminant function is $d=1$ and $\kappa=0$

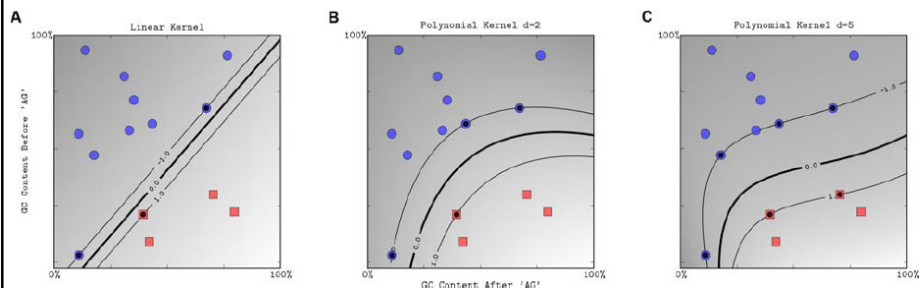


Figure 6. The effect of the degree of a polynomial kernel. The polynomial kernel of degree 1 leads to a linear separation (A). Higher-degree polynomial kernels allow a more flexible decision boundary (B,C). The style follows that of Figure 3.

Another common kernel function: Gaussian

- One parameter, σ , specifies the width of the Gaussian
 - A large σ encompasses many neighbors and leads to a smooth boundary.
 - As σ decreases, the boundary becomes more curved.
 - A tiny σ leads to overfitting.

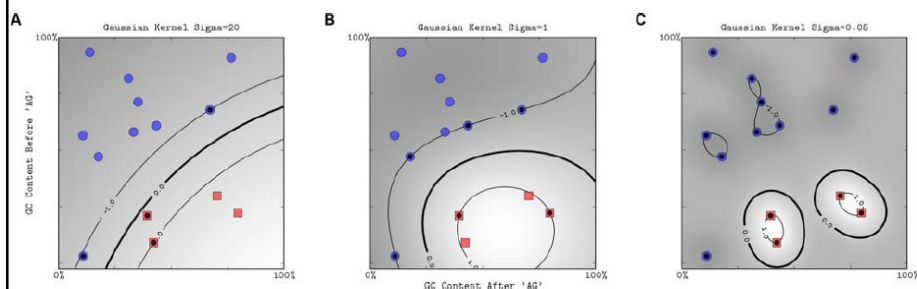


Figure 7. The effect of the width parameter of the Gaussian kernel (σ) for a fixed value of the soft-margin constant. For large values of σ (A), the decision boundary is nearly linear. As σ decreases, the flexibility of the decision boundary increases (B). Small values of σ lead to overfitting

Which kernel should you use?

- Depends on your data—try several.
 - Kernels have even been developed for nonnumeric data like sequences, structures, and trees/graphs.
- Plain old linear usually works great if you have a lot of dimensions and not a lot of examples (common in bioinformatics).
 - Nonlinear can lead to overfitting when N is small
- May help to use a combination of several kernels.
- Don't touch your evaluation data while you're trying out different kernels and parameters.
 - Use cross-validation for this if you're short on data

9

Take-home message

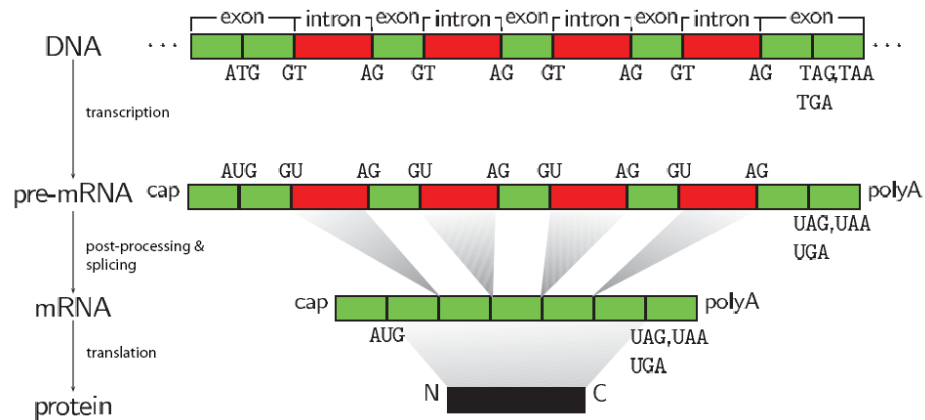
Everything you need to remember in one slide

- Sometimes it improves your classifier if you map (i.e. transform) your data into a different feature space.
 - Example 1: you have nonlinear data but want to use a linear classifier
 - Example 2: your original data aren't numbers—for example they could be sequences
- Kernel methods map your original data into a different space so that you can use linear classifiers.
- This mapping can often substantially increase the number of features to consider.
 - This can be problematic as your number of dimensions grows.
- The “Kernel Trick” addresses this by putting a cap on the feature explosion so that the complexity of your classifier increases only linearly with the size of your original data.

10

Computational Biology Example

- Problem: Find intron/exon boundaries in DNA sequence



11

The data

- Gene sequences
- Which AG dimers truly signal the beginning of an exon and which ones are just internal in an intron or exon?
 - There are MANY more internal ones than boundary ones
 - This becomes a classification problem

12

Useful features

- “Numeric” features
 - GC content of the preceding sequence (intron)
 - GC content of the following sequence (exon)
 - (exons typically have more GCs than introns)
- “Sequence” features
 - Count of A’s, C’s, T’s, & G’s in intron vs. exon
 - Count of dimers in intron vs. exon
 - Count of ℓ -mers in intron vs. exon
 - Location of ℓ -mers relative to potential splice site
 - Sequence similarity with other sequences in a database
 - Log-likelihood of the sequence on the parameters of a probabilistic model

13

An example kernel describing ℓ -mer content: the *spectrum* kernel

- Utilizes a count of all monomers, dimers, trimers, and so on in the exon vs. the intron.
 - This is useful because exons are made up of three-letter “codons” that are meaningful whereas introns are mostly random.
 - Looking at dimers captures GC information, too.
- The kernel parameter is ℓ : maximum length of the subsequence
 - Not feasible to calculate explicitly for large ℓ -- instead just use the ℓ -mers that are actually present in the sequences

14

Performance of Various Classifiers

Using just GC content

Table 1. SVM accuracy on the task of acceptor site recognition using polynomial and Gaussian kernels with different degrees d and widths σ .

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

Accuracy is measured using the area under the ROC curve (auROC) and is computed using 5-fold cross-validation (cf. the section Running Example: Splice Site Recognition for details).
doi:10.1371/journal.pcbi.1000173.t001

Using other sequence features

Table 2. The area under the ROC curve (auROC) of SVMs with the spectrum, mixed spectrum, and weighted degree kernels on the acceptor splice site recognition task for different substring lengths ℓ .

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

doi:10.1371/journal.pcbi.1000173.t002

Spectrum:
Counts of l-mers

Mixed Spectrum:
Counts of l-mers
weighted for length

WD:
location of l-mers