1. SVM example: Computational Biology

Assume a fixed species $\mathcal{S}$ (e.g. baker's yeast, s. *cerevisae*) has genome $\mathcal{G}$ (collection of genes).

Typically a *transcription factor (TF)* t binds to the promoter (upstream) DNA near g and initiates transcription.

In this case we say g is a *target* of t.
Question: given a fixed TF t, for which genes $g \in \mathcal{G}$ are its targets?
Chemically hard to solve:

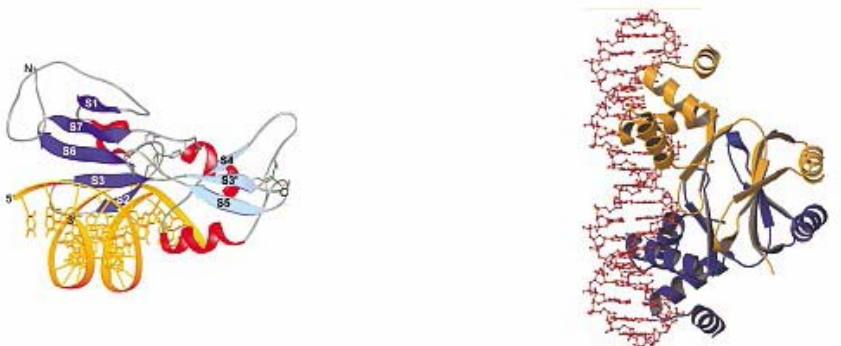


Fig. 1: Left: DNA binding of GCM; right: binding of Fur (C. Ehmke, E. Pohl, EMBL, 2005)

Try machine learning:

Consider a *training data set* (genes known as targets or non-targts of t)

$$D_0 = \{(g_i, y_i)\}_{i=1}^{n},$$

where g_i is a sample gene and

$$y_i = \begin{cases} 1 & \text{if } g_i \text{ is target} \\ -1 & \text{otherwise} \end{cases}.$$

For all genes g define function

$$f_0(g) = y = \begin{cases} 1 & \text{if } g_i \text{ is target} \\ -1 & \text{otherwise} \end{cases}$$

How to learn the general function $f_0 : \mathcal{G} \rightarrow \mathbb{B}$ from examples in D?

Start by representing g as a *feature vector*

$$\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$$

with 'useful' information about g and its promoter region.

What is useful here?

Consider

$ACGGTCTGGT...CGT =$ promoter DNA sequence of g

$=$ upstream region:

This is where TF typically binds; has ~1000 bases

2. Feature maps

One useful choice of feature vector:

Example: Consider an ordered list of possible strings of length 6:

Feature maps

string1	AAAAAA
string2	AAAAAC
string3	AAAAAG
string4	AAAAAT
string5	AAAACA
⋮	⋮

of all sequences of 6 base pairs.

Feature maps

Given gene g, choose feature vector $\mathbf{x} = [x_1, \ldots, x_d]$, where

$x_1 =$ # appearances of string 1 in promoter of g

$x_2 =$ # appearances of string 2 in promoter of g

etc.

Consider *feature map* Φ which takes g to $\mathbf{x}$:

$$\Phi(g) = \mathbf{x}.$$

Number of possible strings of length 6 is $4^6 = 4{,}096$.

Feature maps

Thus $d = 4{,}096$.

$$\Rightarrow \mathbf{x} = [x_1, \ldots, x_d]^T \in \mathbb{R}^{4{,}096} \equiv F$$

Thus let $F =$ all possible $\mathbf{x} =$ *feature space* (a vector space)

Henceforth replace g by its feature vector $\mathbf{x} = \mathbf{x}(g)$.

To classify g we classify $\mathbf{x}$.

Using $\mathbf{x} = \mathbf{x}(g)$ goal is to find

Feature maps

$$f(\mathbf{x}) = y = \begin{cases} 1 & \text{if } g \text{ is target} \\ -1 & \text{if } g \text{ is not target} \end{cases}$$

Thus: f maps a sequence **x** of string counts in F into a yes or no.

Replace g_i by $\mathbf{x}_i = \mathbf{x}(g_i)$

$$D_0 = \{(g_i, y_i)\}_i \to D = \{(\mathbf{x}_i, y_i)\},$$

Thus given examples $f(\mathbf{x}_i) = y_i$ for the *feature vectors* $\mathbf{x}_i$ *in our sample*, want to generalize and find $f(\mathbf{x})$ for *all* **x**.

Feature maps

With data set D, can we find the right function $f : F \to \pm 1$ which generalizes the above examples, so that $f(\mathbf{x}) = y$ for all feature vectors?

Easier: find a $f : F \to$ real numbers, where

$$f(\mathbf{x}) > 0 \text{ if } y = 1; \;\; f(\mathbf{x}) < 0 \;\; \text{if} \;\; y = -1.$$

Resulting predictive accuracy depends on the number of features used, i.e., what the components x_i of **x** mean.

For example, x_{249} can be how many times ACGGAT appears.

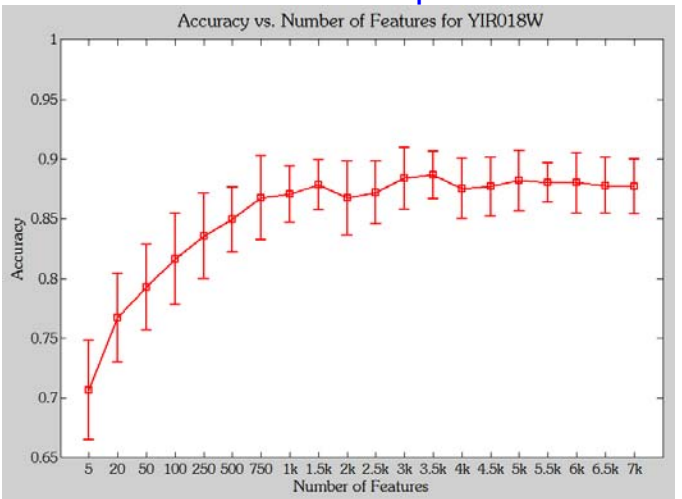
Feature maps

$x_{4,598}$ can be how many times ACG_ _ _GAT appears (i.e., any 3 letters allowed to go in middle)

Generally choose the most significant features - the most helpful ones in discrimination.

For TF YIR018W, accuracy in prediction of targets vs. number of features:

Feature maps



Nonlinear kernels

SVM: Nonlinear Feature Maps and Kernels

http://www.youtube.com/watch?v=3liCbRZPrZA

1. General SVM: when $K(\mathbf{x}, \mathbf{y})$ is not an ordinary dot product

Recall: In yeast, $\mathcal{G} =$ genome (all genes).

Given fixed transcription factor t, want to determine which genes $g \in \mathcal{G}$ bind to t.

Nonlinear kernels

Have a feature map $\mathbf{x} : \mathcal{G} \to F = \mathbb{R}^d$ with F the feature space, with

$$\mathbf{x}(g) = \mathbf{x} = \text{feature vector}$$

for gene g.

For each g define

$$y(g) = \begin{cases} 1 & \text{if } g \text{ binds} \\ -1 & \text{otherwise} \end{cases}.$$

We want a map $f(\mathbf{x})$ which classifies genes. That is, for $g \in \mathcal{G}$, with feature vector $\mathbf{x} = \Phi(g)$ we want

Nonlinear kernels

$$f(\mathbf{x}) \begin{cases} > 0 & \text{if } y(g) = 1 \\ \leq 0 & \text{if } y(g) = -1 \end{cases}.$$

Have examples $\{g_i\}_{i=1}^n \subset \mathcal{G}$ of genes with known binding, together with $y(g_i)$. Define

$$\mathbf{x}_i = \mathbf{x}(g_i)$$

to be feature vectors of the examples.

The SVM provides f of the form

Nonlinear kernels

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b;$$

$f(\mathbf{x}) > 0$ yields conclusion $y = 1$ (binding gene) and otherwise $y = -1$.

Thus have linear separation of points in F.

What about nonlinear separations?

1. Replace base space $\mathcal{G}$ by F (i.e., replace gene by its feature vector)

Nonlinear kernels

Thus have collection of examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ for of feature vectors $\mathbf{x}_i$ for which binding y_i is known.

Desire a new (possibly nonlinear) function $f(\mathbf{x})$ which is positive when **x** is feature vector of binding gene and negative otherwise.

2. With F now as base space, define *new* feature map $\Phi : F \to F_1$ (now may be nonlinear but continuous).

Map the collection of examples into F_1. Thus new set of examples is

Nonlinear kernels

$$\{\mathbf{z}_i \equiv (\Phi(\mathbf{x}_i), y_i)\}_{i=1}^{n}.$$

Induce a *linear* SVM in F_1 (SVM algorithm above).

Nonlinear kernels

3. New decision rule:

$$f_1(\Phi(\mathbf{x})) \equiv \mathbf{w}_1 \cdot \Phi(\mathbf{x}) + b.$$

If $f_1(\Phi(\mathbf{x})) > 0$ we conclude $y = 1$ (gene binds) and otherwise $y = -1$.

Equivalent rule on original F:

$$f(\mathbf{x}) = f_1(\Phi(\mathbf{x})).$$

Allows arbitrary nonlinear separating surfaces on F.

2. The kernel trick

Equivalently to above: assume $\mathbf{w}_1 = \Phi(\mathbf{w})$ for some $\mathbf{w}$. pNew decision rule:

$$f(\mathbf{x}) = \mathbf{w}_1 \cdot \Phi(\mathbf{x}) + b = \Phi(\mathbf{w}) \cdot \Phi(\mathbf{x}) + b.$$

Define standard linear kernel function on F:

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$$

(as before ordinary dot product).

Now back in F, can show $K(\mathbf{x}, \mathbf{y})$ is a Mercer kernel:

Kernel trick

(a) $K(\mathbf{x}, \mathbf{y}) = K(\mathbf{y}, \mathbf{x})$

(b) $K(\mathbf{x}, \mathbf{y})$ is positive definite.

Indeed, given any set $\{\mathbf{x}_i\}_{i=1}^n$,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = \mathbf{u}_i \cdot \mathbf{u}_j$$

with $\mathbf{u}_i = \Phi(\mathbf{x}_i)$. We already know ordinary dot product makes pos. def. kernel.

(c) K is continuous because Φ is cont.

Kernel trick

Like any kernel function $K(\mathbf{x}, \mathbf{y})$ satisfies certain properties of inner product, and so can be thought of as a *new* dot product on F.

Thus

$$f(\mathbf{x}) = K(\mathbf{w}, \mathbf{x}) + b.$$

With the redefined dot product $\mathbf{w} \cdot \mathbf{x} \equiv K(\mathbf{w}, \mathbf{x})$, training SVM is identical to before - we have already developed the algorithm here - just replace old dot product by the new one.

Conclusion: The introduction of nonlinear separators for SVM via replacement of $\mathbf{x} \in F$ by a nonlinear function $\Phi(\mathbf{x})$

Kernel trick

is exactly equivalent to replacement of the dot product $\mathbf{w} \cdot \mathbf{x}$ by $K(\mathbf{w}, \mathbf{x})$ with K a Mercer kernel!

This is equivalent to replacing the standard linear kernel

$$K(\mathbf{w}, \mathbf{x}) = \mathbf{w} \cdot \mathbf{x} \text{ (linear)}$$

by a general nonlinear kernel, e.g., the Gaussian kernel:

$$K(\mathbf{w}, \mathbf{x}) = e^{-|\mathbf{w}-\mathbf{x}|^2}$$

Recall advantages: the calculation of $\mathbf{w}$ and b involve linear algebra involving the matrix $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

3. Examples

Ex 1: Gaussian kernel

$$K_\sigma(\mathbf{x}, \mathbf{y}) = e^{-\frac{|\mathbf{x}-\mathbf{y}|^2}{2\sigma^2}}$$

[can show pos. def. Mercer kernel]

Gaussian kernel

SVM: from (4) above have

$$f(\mathbf{x}) = \sum_j a_j K(\mathbf{x}, \mathbf{x}_j) + b = \sum_j a_j e^{-\frac{|\mathbf{x}-\mathbf{x}_j|^2}{2\sigma^2}} + b,$$

where examples $\mathbf{x}_j$ in F have known classifications y_j, and a_j, b are obtained by quadratic programming.

What kind of classifier is this? It depends on σ (see Vert movie).

Note Movie1 varies σ in the Gaussian ($\sigma = \infty$ corresponds to a linear SVM); then movie2 varies the margin $\frac{1}{|\mathbf{w}|}$ (in

Gaussian kernel

Gaussian feature space F_2) as determined by changing λ or equivalently $C = \frac{1}{2\lambda n}$.

4. Software available

Software which implements the quadratic programming algorithm above includes:

- SVMLight: http://svmlight.joachims.org
- SVMTorch: http://www.idiap.ch/learning/SVMTorch.html
- LIBSVM: http://wws.csie.ntu.edu.tw/~cjlin/libsvm

A Matlab package which implements most of these is Spider:

http://www.kyb.mpg.de/bs/people/spider/whatisit.html

Computational Biology Applications

References:

T. Golub et al Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression. Science 1999.

S. Ramaswamy et al Multiclass Cancer Diagnosis Using Tumor Gene Expression Signatures. PNAS 2001.

B. Schölkopf, T. Tsuda, and J.P. Vert, Kernel Methods in Computational Biology. MIT Press, 2004.

J.P. Vert, http://cg.ensmp.fr/~vert/talks/060921icgi/icgi.pdf
http://cg.ensmp.fr/%7Evert/svn/bibli/html/biosvm.html

Transcription factor binding

1. Matching genes and TF's:

For yeast gene $g \in \mathcal{G}$, feature vector is $\mathbf{x} = \Phi(g)$.

Examples: data set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ of known examples of feature vectors $\mathbf{x}_i$ (together with binding $y_i = \pm 1$) used to form kernel matrix

$$K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$$

$K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$ is *linear kernel*

Transcription factor binding

$K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2}$ is *gaussian kernel*

$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^k$ is *polynomial kernel*

Can show all of these are Mercer Kernels.

Form discriminant function

$$f(\mathbf{x}) = K(\mathbf{w}, \mathbf{x}) + b,$$

with $\mathbf{w}, b$ determined by quadratic programming algorithm using matrix K_{ij} formed from data set D.

Transcription factor binding

$f(\mathbf{x}) > 0$ means corresponding gene g with feature vector $\Phi(g) = \mathbf{x}$ binds to transcription factor.

What features are we interested in? Characterize g by its *upstream region*

of about 1000 bases (reading from the $5'$ end of DNA).

Transcription factor binding

Interesting feature maps:

$$\Phi_1(g) = \mathbf{x} = \text{vector of } 6\text{-string counts}$$

Specifically: take all possible 6-strings (strings of 6 consecutive bases, e.g., $ATGAAC$), index them with $i = 1$ to $4^6 = 4096$, and form vector **x** with

$$x_i = \#\text{ appearances of } i^{th}\ 6\text{-mer in upstream region of corresponding gene } g$$

(large space!).

Or:

Transcription factor binding

$\Phi_2(g) = \mathbf{x} =$ vector of microarray experiment results

Specifically

$x_i = \begin{cases} 1 & \text{if gene } g \text{ is expressed in the } i^{th} \text{ microarray experimen} \\ 0 & \text{otherwise} \end{cases}$

Or:

Transcription factor binding

$\Phi_3(g) =$ vector of gene ontology appearances of g

i.e., $x_i = 1$ if i^{th} ontology term applies to gene g.

$\Phi_4(g) =$ vector of melting temperatures of DNA along consecutive positions in upstream region

Each feature map Φ_k yields a different kernel $K_k(\mathbf{x}, \mathbf{y})$ and kernel matrix $K_{ij}^{(k)}$.

Transcription factor binding

Combination of features: integrate all information into a large vector (i.e., concatenate the vector strings $\Phi_k(\mathbf{x})$ into one:

$$\Phi_{\text{comb}}(\mathbf{x}) = (\Phi_1(\mathbf{x}), \ldots, \Phi_l(\mathbf{x})).$$

This is equivalent to taking a direct sum F of the corresponding feature spaces $F_1, \ldots, F_k$.

How to define inner product in the large feature space F? In the obvious way for a concatenation of vectors:

Transcription factor binding

$$\Phi_{\text{comb}}(\mathbf{x}) \cdot \Phi_{\text{comb}}(\mathbf{y}) = \sum_k \Phi_k(\mathbf{x}) \cdot \Phi_k(\mathbf{y}).$$

Thus the kernel corresponding to feature map Φ_{comb} is given by

$$K_{\text{comb}}(\mathbf{x}, \mathbf{y}) = \Phi_{\text{comb}}(\mathbf{x}) \cdot \Phi_{\text{comb}}(\mathbf{y})$$

$$= \sum_k \Phi_k(\mathbf{x}) \cdot \Phi_k(\mathbf{y}) = \sum_k K_k(\mathbf{x}, \mathbf{y}).$$

Thus SVM kernel K_{comb} which combines *all* feature information is the sum of individual kernels K_k!

Transcription factor binding

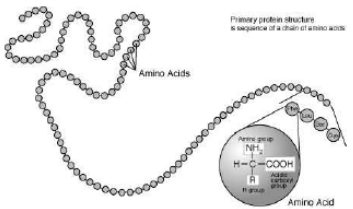
So addition of individual kernel matrices automatically combines their feature information.

Positive predictive values (probability of correct positive prediction) for combined kernel K reaches approximately 90%.

Reference: Machine learning for regulatory analysis and transcription factor target prediction in yeast (with D. Holloway, C. DeLisi), Systems and Synthetic Biology, 2007.

Protein characterization

2. Application: protein sequences:



A : Alanine	V : Valine	L : Leucine
F : Phenylalanine	P : Proline	M : Méthionine
E : Acide glutamique	K : Lysine	R : Arginine
T : Threonine	C : Cysteine	N : Asparagine
H : Histidine	V : Thyrosine	W : Tryptophane
I : Isoleucine	S : Sérine	Q : Glutamine
D : Acide aspartique	G : Glycine	

JP Vert

Protein characterization

Jakkola, et al. (1998) developed feature space kernel methods for anlyzing and classifying protein sequences.

Applications: classification of proteins into functional vs. structural classes, cellular localization of proteins, and knowledge of protein interactions.

Derive kernels (equivalently, appropriate feature maps!) by starting with choices of feature spaces F.

Protein characterization

Choices of F: we map protein p into $\Phi(p) \in F$, where $\Phi(p)$ has information on:

- physical chemistry properties of protein p
- strings of amino acids in p (see DNA examples earlier)
- motifs (what standard functional portions appear in p?)
- similarity measures, local alignment measures with other standard proteins

Protein characterization

Additional relevant protein features $\Phi(p)$ would be

- sequence length
- time series of chemical properties of amino acids in sequence, e.g.,
  hydrophilic properties, polarity
- do transforms on these series, e.g.
  Autocorrelation functions $\sum_t a_t a_{t+k}$, with a_t running parameter
  Fourier transforms

String map: a useful feature map

Protein characterization

Consider a fixed string of length 6: e.g. $S_k = PKTHDR$

Define k^{th} component x_k of feature map $\Phi(p) = \mathbf{x}(p)$ by

$x_k =$ # occurrences of S_k in protein p

(spectrum kernel, Leslie, et al. 2002)

or

$x_k =$ # occurrences of S_k in p with up to M mismatches (mismatch kernel, Leslie, et al. 2004)

Protein characterization

or gapped string kernels (have gaps with weights which decay with number of gaps; substring kernel, Lohdi, et al., 2002)

For example, given string

$$LIMASGCVGFSC$$

we have spectrum of 3-mers:

$\{LIM, IMA,$
$MAS, ASG, SGC, GCV, CVG, VGF, GFS, FSC\}$

spectrum (string) kernel:

Protein characterization

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) = \sum_k \Phi_k(\mathbf{x})\Phi_k(\mathbf{y})$$

where **x, y** are feature vectors

General Observations about kernel methods:

(1) Above examples illustrate advantage of feature space methods: we are able to summarize amorphous-seeming information in an object g in a feature vector $\Phi(g)$.

Protein characterization

(2) After that a very important advantage is the *kernel trick:* we summarize *all* information about our sample data $\{\mathbf{x}_j\}$ in a kernel matrix $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

This allows representation of *very* high dimesional data $\Phi(\mathbf{x})$ in matrix **K** with size equal to number of examples (sometimes *much* smaller than dimension)
Matrix **K** is all we need to find the a_i in the discriminator function

$$f(\mathbf{x}) = \sum_i a_i K(\mathbf{x}, \mathbf{x}_i) + b$$

Protein characterization

Another approach to forming kernels: similarity kernels

Start with a known collection (dictionary) of sequences

$$D = (\mathbf{x}_1, \ldots, \mathbf{x}_n).$$

Define a similarity measure

$$s(\mathbf{x}, \mathbf{y}).$$

Define a feature vector by similarities to objects in D:

$$\Phi(\mathbf{x}) = (s(\mathbf{x}, \mathbf{x}_1), \ldots, s(\mathbf{x}, \mathbf{x}_n)).$$

Protein characterization

- Known as pairwise kernels (Liao, Noble, 2003):
  $s =$ standard distance between strings
- Motif kernels (Logan, et al., 2001):
  $s(\mathbf{x}, \mathbf{x}_j) =$ distance measure between string and motif (standard signature sequence) $\mathbf{x}_j$

3. Jakkola's Fisher score map

Jakkola, et al. (1998) studied HMM models for protein sequences and combined with kernel methods.

1. Form a parametric family of probabilistic models P_Θ, e.g., HMM models of a family of protein sequences, with $\Theta \in \Omega \subset \mathbb{R}^m$ a family of parameters.

2. Find an estimate Θ_0 (e.g., Baum-Welsh, maximum likelihood from some training set)

3. Form the feature map (Fisher score vector) on sequence

Jakkola's feature map

vector **x**

$$\Phi_0(\mathbf{x}) = \nabla_\Theta \ln P_\Theta(\mathbf{x})\Big|_{\Theta=\Theta_0}.$$

4. In feature space define the inner product (kernel K) by

$$K_0(\mathbf{x},\mathbf{y}) = \Phi_0(\mathbf{x}) \cdot \left(I_0^{-1}\Phi_0(\mathbf{y})\right),$$

where

$$I_0 = E_{\Theta_0}\left[\Phi_0(\mathbf{x})\Phi_0(\mathbf{x})^T\right]$$

Jakkola's feature map

(expectation over **x** under Θ_0) is the Fisher information matrix (expectation assuming parameter Θ_0).

Advantages of Fisher kernel:

- Fisher score shows how strongly the probability $P_\Theta(\mathbf{x})$ depends on each parameter θ_i in Θ
- Fisher score $\Phi_\Theta(\mathbf{x})$ can be computed explicitly, e.g., for HMM
- Different models Θ_i can be trained and their kernels K_i combined

Jakkola's feature map

Results for correct classification of proteins in the G-protein family as subset of SCOP (nucleotide triphosphate hydrolases) superfamily:

Jakkola's feature map

Sequence	BLAST	B-Hom	S-T98	SVM-F
5p21	0.043	0.010	0.001	0.000
1guaA	0.179	0.031	0.000	0.000
1etu	0.307	0.404	0.428	0.038
1hurA	0.378	0.007	0.007	0.000
1eft(3)	0.431	0.568	0.041	0.051
1dar(2)	0.565	0.391	0.289	0.019
1tadA(2)	0.797	0.330	0.004	0.000
1gia(2)	0.867	0.421	0.017	0.000

Table 1: Rate of false positives for *G proteins* family. BLAST = BLAST:SCOP-only, B-Hom = BLAST:SCOP+SAMT-98-homologs, S-T98 = SAMT-98, and SVM-F = SVM-Fisher method.

4. Finding kernels - how do we decide

We can find kernels by:

- Finding a feature map Φ into F_1 which separates positive and negative examples well.

  Then

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}).$$

Finding kernels

- Defining kernel as a "similarity measure" $K(\mathbf{x}, \mathbf{y})$ which is large when **x** and **y** are "similar", given by a positive definite function $K(\mathbf{x}, \mathbf{y})$ with $K(\mathbf{x}, \mathbf{x}) = 1$ for all **x**.

Rationale: Note here we require for all **x** in feature space F_1 :

$$|\Phi(\mathbf{x})| = \sqrt{K(\mathbf{x}, \mathbf{x})} = 1$$

Finding kernels

So

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}) = |\Phi(\mathbf{x})||\Phi(\mathbf{y})|\cos\,\theta = \cos\,\theta. \quad (2)$$

where

$$\theta \equiv \text{angle between } \Phi(\mathbf{x}) \text{ and } \Phi(\mathbf{y}).$$

Finding kernels

So if

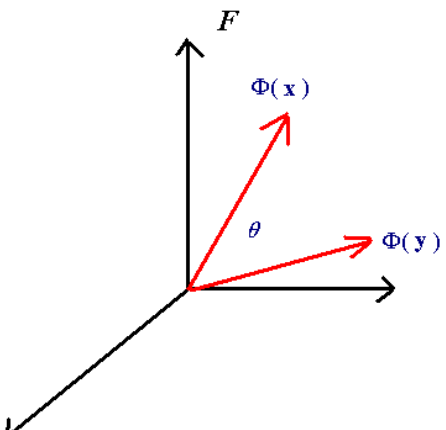
x and **y** are similar

(by desired criterion) then $K(\mathbf{x}, \mathbf{y})$ large and by (2) θ small, i.e.

$\Phi(\mathbf{x})$ close to $\Phi(\mathbf{y})$.

Thus *similar* **x** and **y** are close in the new feature space F_1, and *different* **x** *and* **y** are far, i.e., can be separated.

Finding kernels



5. Finding translation initiation sites (Zien, Ratsch, Mika, Schölkopf, et al., 2000)

A *translation initiation site* (*TIS*) is a DNA location where coding for a protein starts (i.e., beginning of gene).

Usually determined by codon ATG.

Question: How to determine whether particular ATG is start codon for a TIS?

Finding translation initiation sites

Strategy: Given a potential start codon ATG at location i in the genome :

1. Start by looking at 200 nucleotides (nt) around current ATG:

$$ACTGATGTG\ldots AC\underbrace{ATG}_{\text{center}}TAG\ldots ATGCACC \qquad (1)$$

Finding translation initiation sites

2. Use the *unary bit encoding* of nucleotides:

$= 10000;\ \ C = 01000;\ \ G = 00100;\quad T = 00010;\quad$ Unknown $= 0$

3. Concatenate the unary encodings: replace each nt in (1) by unary code:

$$\Phi(i) = \underbrace{10000\ 01000\ 00010\ 00100}_{A\qquad C\qquad T\qquad G}\ldots \in F$$

This becomes feature vector.

What kernel to use in this feature space?

Try polynomial kernel:

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^m = \left(\sum_i x_i y_i\right)^m$$

$$= \sum_{i_1} x_{i_1} y_{i_1} \cdot \sum_{i_2} x_{i_2} y_{i_2} \cdot \ldots \cdot \sum_{i_m} x_{i_m} y_{i_m}$$

$$= \sum_{i_1,\ldots,i_m} x_{i_1} y_{i_1} x_{i_2} y_{i_2} \ldots x_{i_m} y_{i_m}$$

with m fixed and $\mathbf{x}, \mathbf{y} \in F$.

Note:

Finding translation initiation sites

- If $m = 1$ then

$K(\mathbf{x}, \mathbf{y}) = \sum_i x_i y_i =$ number of common NT (nucleotides) in strings **x** and **y**

- If $m = 2$ then

$K(\mathbf{x}, \mathbf{y}) = \sum_{i,j} x_i y_i x_j y_j =$ total number of ***pairs*** of common NT

Finding translation initiation sites

in **x** and **y** (times 2)

- generally $K(\mathbf{x}, \mathbf{y}) = \#$ ***m*-tuples** of common NT in **x** and **y** (times $m!$) for fixed m

Note: This is a good *similarity measure* (see previous section).

6. Better Kernel: Local matches

Now define

Finding translation initiation sites

$$M_k(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } x_k = y_k \\ 0 & \text{otherwise} \end{cases}$$

Define *window kernel* W for fixed k :

$$W_r(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=-k}^{k} a_i M_{r+i}(\mathbf{x}, \mathbf{y}) \right)^{m_1}$$

$=$ [weighted # matches in window $(r-k, r+k)$]^{m_1}

Usefulness: measures correlations in a window of length $2k+1$ centered at r; less noise.

Finding translation initiation sites

Now add up the weighted matches over center positions r:

$$K_{\text{new}}(\mathbf{x}, \mathbf{y}) = \sum_{r=1}^{N} W_r(\mathbf{x}, \mathbf{y}).$$

Recognition error rates for TSS recognition (Zien, Ratsch, et al., 2000)

Neural Network	15.4%
Linear Kernel K with $m = 1$	13.2%
K_{new}	11.9%

7. General kernel construction

Many string algorithms in comp. bio. can lead to kernels, as long as they give similarlity scores $S(\mathbf{x}, \mathbf{y})$ for sequences **x** and **y** which translate to pos. def. kernel $K(\mathbf{x}, \mathbf{y})$.

1. Smith-Waterman score (pairwise alignment measure, kernelized in Gorodkin, 2001) gives similarity kernel $K(\mathbf{x}, \mathbf{y})$ for multiple alignments (separation of strings with hyperplane in string space.)

2. *Kernelization* of other algorithms can be done similarly:

- Kernel ICA (independent component analysis)

algorithms

- Kernel PCA (principal component analysis) algorithms
- Kernel logistic regression methods

For other examples of string kernel methods in computational biology see talks of J.P. Vert:

http://cg.ensmp.fr/~vert/talks/060921icgi/icgi.pdf