Classification DNA sequences using Gap–Weighted Subsequences Kernel

Wilson Soto

Department of System Engineering and Industrial Engineering
Research Group on Algorithms and Combinatorics (ALGOS–UN)
National University of Colombia, Bogotá, Colombia
wesotof@unal.edu.co

Abstract

The aim of this paper is to show experimental results of classification DNA sequences using gap–weighted subsequences kernel including the assess the expected error rate of a classification algorithm. The process involve a type of kernel specific with a classification algorithm for learn to recognize sites that regulate transcription, sites that can be detected in the laboratory as DNaseI hypersensitive sites (HSs) on DNA sequences. The classification algorithm is support vector machine (SVM), which learns by example to discriminate between two given classes of data. The DNA sequences are converted using gap–weighted subsequences kernel in a matrix kernel, which is processed by the classification algorithm to produce a model with the which we can predict the classification of new examples. It is important to know that a high accuracy with computational methods for the identification of the DNaseI hypersensitive sites would to help to speed up the functional annotation of the human genome.

Keywords: Bioinformatics, Classification Algorithm, Gap–Weighted Subsequences Kernel, Pattern Algorithm, Transcription

Resumen

El fin de este artículo es mostrar los resultados experimentales de la clasificación de secuencias de ADN usando el kernel de subsecuencias con penalización incluyendo la evaluación de la tasa de error esperada de un algoritmo de clasificación. El proceso envuelve un tipo específico de kernel con un algoritmo de clasificación para aprender a reconocer sitios que regulan la transcripción, sitios que pueden ser detectados en el laboratorio como híper sensitivos de Deoxirribonucleasa I sobre secuencias de ADN. El algoritmo de clasificación es máquinas de vectores de soporte, el cual aprende por ejemplos a discriminar entre dos

clases de datos dadas. Las secuencias de ADN son convertidas usando el kernel de subsecuencias con penalización en una matriz de kernel, la cual es procesada por el algoritmo de clasificacin para producir un modelo con el cual se puede predecir la clasificación de nuevos ejemplos. Es importante saber que una alta exactitud con métodos computacionales para la identificación de sitios híper sensitivos de Deoxirribonucleasa I podría ayudar a acelerar la anotación funcional del genoma humano.

Palabras claves: Bioinformática, Algoritmo de Clasificación, Kernel de Subsecuencias con Penalización, Algoritmo de Patrones, Transcripción

1 INTRODUCTION

The human genome is composed of 23 pairs of chromosomes, each of which contain hundreds of genes. There are estimated $\approx 25,000 - 30,000$ human protein–coding genes. The regulatory sequences are crucial to controlling gene expression. These are typically short sequences that appear near or within genes.

A gene consists of a transcriptional region and a regulatory region. The regulatory region can be divided into cis–regulatory (or cis–acting) elements and trans–regulatory (or trans–acting) elements. The cis–regulatory elements are the binding sites of transcription factors which are the proteins that, upon binding with cis–regulatory elements, can affect (either enhance or repress) transcription. The identification of sequences that regulate transcription is one of the major goals of genome biology.

DNaseI hypersensitive sites (HSs) have since proven to be extremely reliable and generic markers of cis–regulatory sequences. Mapping DNaseI hypersensitive (HS) sites has traditionally represented the gold–standard experimental method for discovering functional non-coding elements involved in gene regulation, but the labor intensive nature of this technique has limited its application to only a small number of human genes [2].

Computational methods such as Support Vector Machines (SVMs) and related kernel methods are extremely good at solving such problems. SVMs are widely used in computational biology due to their high accuracy, their ability to deal with high–dimensional and large datasets, and their flexibility in modeling diverse sources of data. SVMs use two key concepts to solve this problem: large margin separation and kernel functions [1].

Viewing the input data as two sets of vectors in an n–dimensional space, an SVM will construct a separating hyperplane in that space, one which maximizes the "margin" between the two data sets. To calculate the margin, we construct two parallel hyperplanes, one on each side of the separating one, which are "pushed up against" the two data sets. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the neighboring data points of both classes. The hope is that, the larger the margin or distance between these parallel hyperplanes, the better the generalization error of the classifier will be.

Noble *et al.* [6] was proven that a classification algorithm (Support Vector Machine) can learn recognize DNaseI HSs with high accuracy using the spectrum

kernel for embedding the sequences into a vector space. The results was 85% in predicting HSs [4]. Them hypothesize that the difference between HS and non–HS sequences can be well characterized based on matching subsequences of characters without gaps.

In this work, the difference between two sequences is assess the number of (possibly non–contiguous) matching subsequences them shared. Non–contiguous occurrences are penalized according to the number of gaps they contain. This is accomplished using the gap–weighted subsequences kernel presented in [3, 5]. In the experiments reported here, we using the factor decay for $\lambda = \{0.3, 0.7\}$ for penalise non–contiguous subsequences.

2 BACKGROUND

2.1 Strings

A string is finite sequence of symbols which belongs to an alphabet, is a function S of $\{1 \dots m\}$ over Σ, $S = a_1 a_2 \dots a_m$ where $a_i = S(i) \in \Sigma$. Here Σ denotes the alphabet containing σ symbols. The length of a string S is denoted by $\mid S \mid$ or when the input strings are known, by m for sequence x and n sequence for y.

A subsequence $s'\{1 \dots j\}$ of S is obtained by deleting $m - j$ symbols from S. A common subsequence (CS) of x and y is an ordered sequence of symbols (not necessarily contiguous), which occurs in both strings denoted by $\mathrm{CS}(x, y)$.

2.2 Kernels

A *kernel* is a function k that for all $\mathbf{p}$,$\mathbf{q} \in X$ satisfies

$$k(p, q) = \langle \phi(p), \phi(q) \rangle , \tag{1}$$

where ϕ is a mapping from X to an (inner product) feature space F

$$\phi : p \longmapsto \phi(p) \in F \tag{2}$$

2.3 Spectrum Kernels

Compare strings in many applications is to count how many (contiguous) substrings of length p they have in common. Then *spectrum of order p* (or *p–spectrum*) of a sequence x to be the histogram of frequencies of all its (contiguous) substrings of length p [3].

The feature space F associated with the p–spectrum kernel is indexed by $I = \Sigma^p$, with the embedding given by

$$\phi_u^p(x) = |\{(v_1, v_2) : x = v_1 u v_2\}| , u \in \Sigma^p. \tag{3}$$

The associated kernel is defined as

$$k_p(x, y) = \langle \phi^p(x), \phi^p(y) \rangle = \sum_{u \in \Sigma^p} \phi_u^p(x) \phi_u^p(y). \tag{4}$$

2.4 Gap–Weighted Subsequences Kernels

The gap–weighted subsequences kernel compare strings by means of the subsequences they contain, namely, between more subsequences in common, the more similar they are. Moreover, the kernel include a degree of contiguity of the subsequence (weight of the occurrences) that contribute to the comparison.

The feature space has the same coordinates as for the fixed subsequences kernel and hence the same dimension. In order to deal with non–contiguous substrings, it is neccesary to introduce a decay factor $\lambda \in (0, 1)$ that can be used to weight the presence of a certain feature in a string. Recall that for an index sequence $\mathbf{i}$ identifying the occurrence of a subsequence $u = x(\mathbf{i})$ in a string x, we use $l(\mathbf{i})$ to denote the length of the string in x. In the gap–weighted kernel, we weight the occurrence of u with the exponentially decaying weight $\lambda^{l(\mathbf{i})}$ [3].

The feature space associated with the gap–weighted subsequences kernel of length p is indexed by $I = \Sigma^p$, with the embedding given by

$$\phi_u^p(x) = \sum_{i:u=x(i)} \lambda^{l(i)}, u \in \Sigma^p. \tag{5}$$

The algorithm 1 show efficient implementation for computing the gap–weighted subsequences kernel for two strings (*e.g.* see Fig. 1 for the strings $x =$ `cata` and $y =$ `gatta` with the gap–weighted subsequences kernel of length $p = \{1, 2, 3\}$).

Algorithm 1 The algorithm for computing the gap–weighted subsequences kernel for two strings x and y

Require: (x, y, n, m, p, λ)
  DPS$(1 : n, 1 : m) = 0$;
  for $i = 1 : n$ **do**
    for $j = 1 : m$ **do**
      if $x_i = y_j$ **then**
        DPS$(i, j) = \lambda^2$;
  DP$(0, 0 : m) = 0$;
  DP$(1 : n, 0) = 0$;
  for $l = 2 : p$ **do**
    Kern$(l) = 0$;
    for $i = 1 : n - 1$ **do**
      for $j = 1 : m - 1$ **do**
        DP(i, j) = DPS(i, j) + λDP$(i - 1, j)$ + λDP$(i, j - 1)$ - λ^2DP$(i - 1, j - 1)$;
        if $x_i = y_j$ **then**
          DPS$(i, j) = \lambda^2$DP$(i - 1, j - 1)$;
          Kern(l) =Kern(l) + DPS(i, j);
  return Kern(p)

DP: K_1^0	g	a	t	t	a
c	0	0	0	0	0
a	0	λ^2	0	0	λ^2
t	0	0	λ^2	λ^2	0
a	0	λ^2	0	0	λ^2

DP_2	g	a	t	t
c	0	0	0	0
a	0	λ^2	λ^3	λ^4
t	0	λ^3	$\lambda^4+\lambda^2$	$\lambda^7+\lambda^5+\lambda^4$

DP: K_2^0	g	a	t	t	a
c	0	0	0	0	0
a	0	0	0	0	0
t	0	0	λ^4	λ^5	0
a	0	0	0	0	$\lambda^7+\lambda^5+\lambda^4$

DP_3	g	a	t	t
c	0	0	0	0
a	0	0	0	0
t	0	0	λ^4	$2\lambda^5$

DP: K_3^0	g	a	t	t	a
c	0	0	0	0	0
a	0	0	0	0	0
t	0	0	0	0	0
a	0	0	0	0	$2\lambda^7$

Fig. 1. Computations for the strings $x =$ `cata` and $y =$ `gatta` with the gap–weighted subsequences kernel of length $p = \{1, 2, 3\}$ using dynamic programming (DP) tables

3 MAPPING DNASE HYPERSENSITIVE

One major goal in genomics is to identify the location regulatory elements and to understand how genes are regulated in different tissues, diseases and species. DNA sequences that regulate transcription and other chromosomal processes are associated with focal alteration (local disruptions, or "openings") in chromatin structure *in vivo*, detectable through hypersensitivity to DNaseI and other nucleases.

Mapping DNase hypersensitive sites within nuclear chromatin is a powerful and well–established method of identifying many different types of transcriptional regulatory elements including enhancers, promoters, insulators, and locus control regions. Mapping DNase hypersensitive sites has been used to identify the precise location of regulatory elements [2]. The regulatory elements can be detected experimentally as DNaseI hypersensitive sites (HSs) *in vivo*, though the process is extremely laborious and costly. The ability to discriminate DNaseI HSs computationally would have a major impact on the annotation and utilization of the human genome [6].

4 EXPERIMENTAL FRAMEWORK

The experiments were run on a Dell of 2.0 GHz Clock, with 3GB RAM which is a 32 bit machine. During all experiments, this machine was not performing other heavy tasks (or process).

4.1 Methodology

First implement the gap–weighted subsequences kernel using Java IDE Eclipse 3.2. Next, use the library *Libsvm* for the support vector machine (SVM). The *Libsvm* is available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`. The *Libsvm* tools may precomputed kernel values and input them as training and testing files.

Assume there are L training instances $x_1, \ldots x_L$ and $K(x, y)$ be the kernel value of two instances x and y. The input formats are:

New training instance for x_i:

$$< label > \mathbf{0} : i \ \mathbf{1} : k(x_i, x_1) \ldots \mathbf{L} : k(x_i, x_L)$$

New testing instance for any x:

$$< label > \mathbf{0} : ? \ \mathbf{1} : k(x, x_1) \ldots \mathbf{L} : k(x, x_L)$$

That is, in the training file the first column must be the "ID" of x_i. In testing, ? can be any value.

Above mentioned the gap–weighted subsequences kernel have two parameters λ and p. In this experiments, the decay factors are $\lambda = 0.3$ and $\lambda = 0.7$ with p in range 1 to 3.

Using Cross–validation find optimal C for the SVM. SVM models have a cost parameter, C, that allow some flexibility in separating the categories and controls the trade off between allowing training errors and forcing rigid margins. It creates a soft margin that permits some misclassifications. Increasing the value of C increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well.

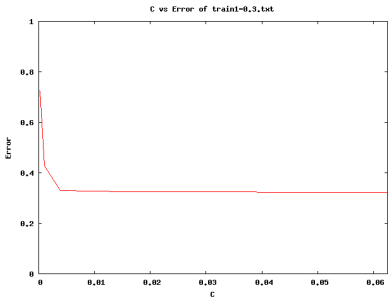
4.2 Types of Data

Experiments were carried out over sequences of DNA alphabet in format FASTA. The DNA alphabet is $\Sigma = \{\texttt{A,C,G,T}\}$, each character represents a nucleotide (standing for Adenine, Cytosine, Guanine and Thymine, respectively) used to encode DNA.

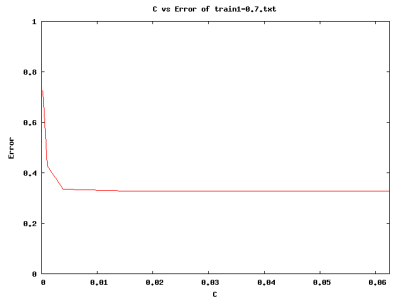
The original dataset (available at `http://noble.gs.washington.edu/proj/hs/`) contain 280 HS sequences and 737 non–HS sequences from erythroid cells. For the experiments the dataset are: the training set (916 sequences) of 252 HS sequences and 664 non–HS sequences and testing set (101 sequences) of 28 HS sequences and 73 non–HS sequences.

5 Analysis Results

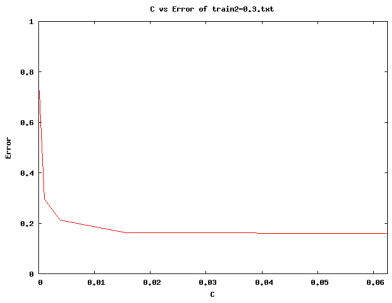
The graphics (see Figure 2) shown the relation between the parameter C and validation error, for the kernel with parameters $p = \{1, 2, 3\}$ and $\lambda = \{0.3, 0.7\}$. Where the error of validation is minimum, the parameter C is selected. The receiver operating characteristic (ROC) curve (see Figure 3) present the score (area under this curve AUC) of classification or performance measure for each kernel.



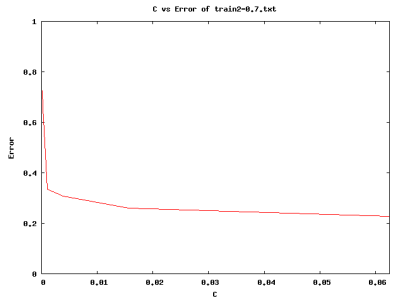
(a) $p = 1; \lambda = 0.3$



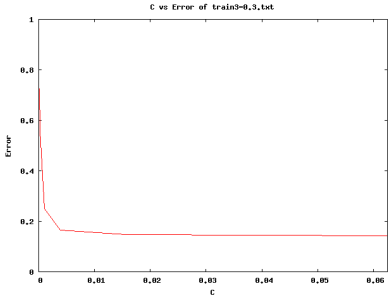
(b) $p = 1; \lambda = 0.7$



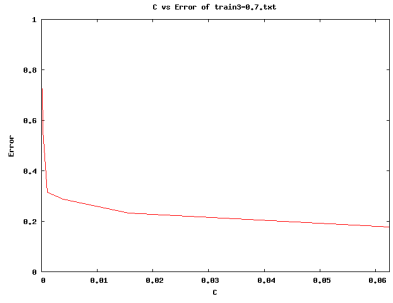
(c) $p = 2; \lambda = 0.3$



(d) $p = 2; \lambda = 0.7$

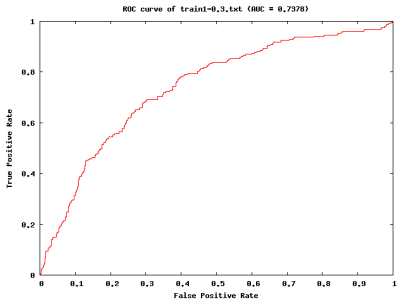


(e) $p = 3; \lambda = 0.3$

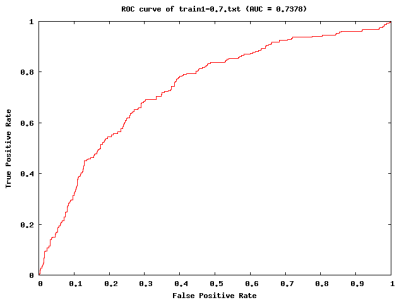


(f) $p = 3; \lambda = 0.7$

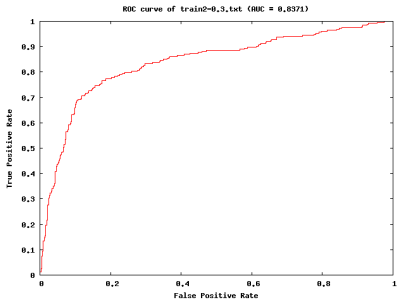
Fig. 2. Parameter C *vs.* Validation Error for $p = \{1, 2, 3\}$ and $\lambda = \{0.3, 0.7\}$.



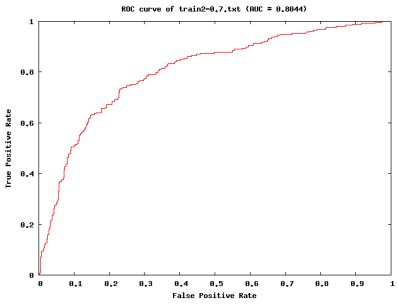
(a) $p = 1; \lambda = 0.3$



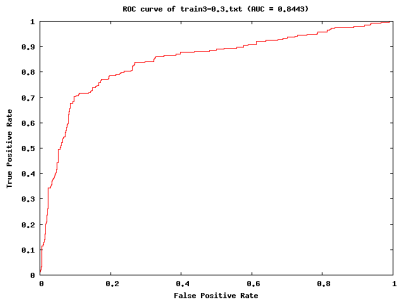
(b) $p = 1; \lambda = 0.7$



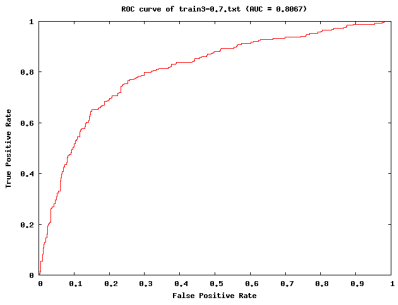
(c) $p = 2; \lambda = 0.3$



(d) $p = 2; \lambda = 0.7$



(e) $p = 3; \lambda = 0.3$



(f) $p = 3; \lambda = 0.7$

Fig. 3. Curve ROC and AUC for $p = \{1, 2, 3\}$ and $\lambda = \{0.3, 0.7\}$.

6 Conclusions

For the data set $p = \{2, 3\}$ and $\lambda = \{0.3, 0.7\}$ the mean AUC of the ROC curve was upper the 0.8, indicative of excellent performance, knowing that in the data set selected existing a number greater of examples non–HS sequences that DNaseI HS sequences. For the data set $p = \{1\}$ and $\lambda = \{0.3, 0.7\}$ the AUC of the ROC curve is 0.73, this value is lower due to that the parameter $p = 1$ is similarity a to count the matches of the characters between the DNA sequences.

The computational tools provide a good alternative for discovery and analysis of functional non–coding elements involved in gene regulation, moreover, the approach described here should be applicable to other type of problems relationated with the annotation of genomes.

The SVM leads to good generalization, however, from the practical the most serious problem with SVM is the high computational complexity of the required quadratic programming in large–scale tasks.

References

[1] Ben–Hur, Asa; Ong, Cheng–Soon; Sonnenburg, Sören; Schölkopf, Bernhard and Rätsch, Gunnar: *Support Vector Machines and Kernels for Computational Biology.* PLoS Computational Biology **4**(10) : 2008. 1 – 10.

[2] Crawford, Gregory E.; Holt, Ingeborg E.; Mullikin, James C.; Tai, Denise and *et al.*: *Identifying gene regulatory elements by genome–wide recovery of DNase hypersensitive sites.* PNAS **101**(4) : 2004. 992 – 997.

[3] Cristianini, Nello and Shawe–Taylor, John: *Kernel Methods for Pattern Analysis*, (2004), Cambridge University Press.New York, NY, USA.

[4] Leslie, Cristina; Eskin, Eleazar and Noble, William Stafford: *The Spectrum Kernel: A String Kernel for SVM protein classification.* Pacific Symposium on Biocomputing **7** : 2002. 566 – 575.

[5] Lodhi, Huma; Saunders, Craig; Shawe–Taylor, John; Cristianini, Nello and Watkins, Chris: *Text Classification using String Kernels.* Journal of Machine Learning Research **2** : 2002. 419 – 444.

[6] Noble, William Stafford; Kuehn, Scott; Thurman, Robert; Yu, Man and Stamatoyannopoulos, John: *Predicting the in vivo signature of human gene regulatory sequences.* Bioinformatics **21**(1) : 2005. 338 – 343.