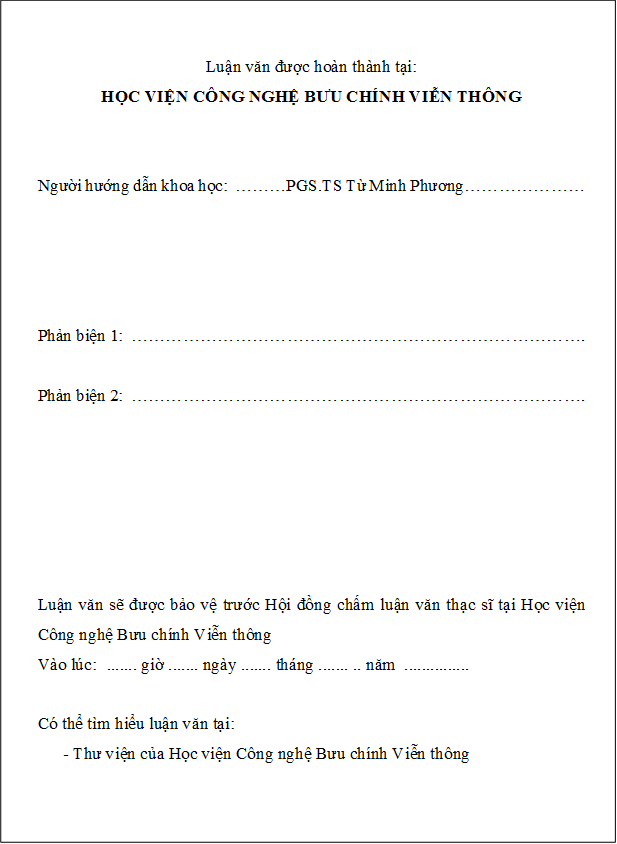
|  |  |
| --- | --- |
|  |  |
| **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  **---------------------------------------** | |
|  | |
|  | |
| **Hoàng Văn Thành** | |
|  | |
|  | |
| XÁC ĐỊNH ĐOẠN ĐIỀU HÒA GEN TRÊN TRÌNH TỰ ADN BẰNG PHƯƠNG PHÁP TÍNH TOÁN | |
|  | |
| **Chuyên ngành: Khoa học máy tính** | |
| **Mã số: 60.48.01**  **TÓM TẮT LUẬN VĂN THẠC SĨ**  HÀ NỘI - 2013 | |
|  | |



**MỤC LỤC**

[**MỤC LỤC** 1](#_Toc366071537)

[**MỞ ĐẦU** 3](#_Toc366071538)

[**CHƯƠNG 1: CÁC KHÁI NIỆM VÀ VẤN ĐỀ LIÊN QUAN ĐẾN BÀI TOÁN XÁC ĐỊNH VÙNG TĂNG CƯỜNG** 5](#_Toc366071539)

[**1.1.** **Các khái niệm** 5](#_Toc366071540)

[**1.1.1.** **Sự hoạt động của gen** 5](#_Toc366071541)

[**1.1.2.** **Điều hòa sự hoạt động của gen** 6](#_Toc366071542)

[**1.1.3.** **Nhân tố phiên mã (transcription factor)** 6](#_Toc366071543)

[**1.1.4.** **Vùng tăng cường (enhancer)** 6](#_Toc366071544)

[**1.2.** **Các nhóm phương pháp xác định vùng tăng cường.** 7](#_Toc366071545)

[**1.3.** **Kết luận chương** 7](#_Toc366071546)

[**CHƯƠNG 2: PHƯƠNG PHÁP XÁC ĐỊNH VÙNG TĂNG CƯỜNG DỰA TRÊN SVM** 8](#_Toc366071547)

[**2.1.** **Phương pháp xác định vùng tăng cường dựa trên SVM** 8](#_Toc366071548)

[**2.1.1.** **Giới thiệu về SVM** 8](#_Toc366071549)

[**2.1.2.** **Giới thiệu về hàm kernel trong SVM** 8](#_Toc366071550)

[**2.1.3.** **Phương pháp xác định vùng tăng cường dựa trên SVM** 8](#_Toc366071551)

[**2.1.4.** **Một số hàm kernel đã được sử dụng để giải quyết bài toán** 9](#_Toc366071552)

[**2.2.** **Các hàm kernel được đề xuất** 10](#_Toc366071553)

[**2.2.1.** **Hàm kernel dựa trên entropy** 11](#_Toc366071554)

[**2.2.2.** **Hàm kernel phân cấp (hierarchical kernel)** 12](#_Toc366071555)

[**CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ** 14](#_Toc366071556)

[**3.1.** **Cài đặt phương pháp** 14](#_Toc366071557)

[**3.1.1.** **Xây dựng bộ dữ liệu mẫu** 14](#_Toc366071558)

[**3.1.2.** **Cài đặt phương pháp** 15](#_Toc366071559)

[**3.2.** **Tiêu chí đánh giá** 16](#_Toc366071560)

[**3.3.** **Đánh giá kết quả** 17](#_Toc366071561)

[**3.4.** **So sánh kết quả với các phương pháp khác** 19](#_Toc366071562)

[**3.5.** **Nhận xét về kết quả và hướng phát triển.** 20](#_Toc366071563)

[**KẾT LUẬN** 21](#_Toc366071564)

[**TÀI LIỆU THAM KHẢO** 22](#_Toc366071565)

# **MỞ ĐẦU**

Trong khuôn khổ luận văn, em tập trung nghiên cứu phương pháp xác định vùng tăng cường (enhancer) từ trình tự ADN của động vật bậc cao, sử dụng phương pháp học máy vector tựa SVM (support vector machines). Em đã nghiên cứu một số phương pháp mới để xây dựng hàm kernel dùng để biểu diễn trình tự ADN khi sử dụng với SVM và thử nghiệm trên một số bộ dữ liệu thực. Phương pháp đề xuất có kết quả tốt, cho phép cải thiện độ chính xác so với phương pháp đã có. *Các kết quả được công bố trong bài báo “Enhancer prediction using distance aware kernels”. Bài báo đã được chấp nhận đăng tại kỷ yếu hội nghị RIVF 2013 sẽ diễn ra vào tháng 11/2013.*

Trải qua nhiều thập kỉ, khoa học máy tính đã có những bước phát triển vượt bậc. Ngày nay, khoa học máy tính đã khám phá ra những khả năng mới trong một lĩnh vực đầy hứa hẹn như tin sinh học. Sinh học và những nguyên lý liên quan của nó là một vùng đất màu mỡ để các nhà nghiên cứu tiếp tục đào sâu tìm hiểu. Tin sinh học hay sinh học tính toán liên quan đến việc sử dụng các kỹ thuật từ toán học ứng dụng, tin học, thống kê… để giải quyết các vấn đề sinh học. Một số mảng nghiên cứu chính đang được phát triển như là phân tích chuỗi, phân tích biểu diễn gen, dự đoán cấu trúc ADN và protein, các thuật toán dùng cho tin sinh học…

Công nghệ sinh học phân tử hiện đại cho phép thu thập rất nhiều loại dữ liệu, thông tin liên quan tới gen và protein. Có một vấn đề đặt ra là sau khi xác định được chức năng của gen, nếu phát hiện được gen đấy có nằm trong một chuỗi ADN, thì làm thế nào để ta có thể xác định được nó hoạt động hay không? Đây không phải là một vấn đề mới trong sinh học, nhưng có vai trò quan trọng trong việc tìm hiểu cơ chế hoạt động của các quá trình sinh học.

Một trong những tác nhân quan trọng tác động lớn đến sự hoạt động của gen là vùng tăng cường. Vì vậy, bài toán về xác định sự hoạt động của gen có thể được đưa về bài toán nhỏ hơn là xác định trong chuỗi ADN chứa gen ấy có chứa các vùng tăng cường tương ứng của nó hay không. Nhưng làm thế nào để có thể tìm ra được các vùng tăng cường tương ứng với một gen?

Công việc này, trước đây, có thể được thực hiện thông qua quá trình thực nghiệm. Phương pháp này rất chính xác, tuy nhiên, nó rất chậm và tốn rất nhiều công. Đặc biệt là với tình hình hiện nay, khi mà số lượng gen tìm được càng ngày càng nhiều, thì yêu cầu được đặt ra là cần tìm ra được những phương pháp mới nhanh và rẻ hơn, và phương pháp tính toán (tin sinh học) là một trong những phương pháp có thể đáp ứng được những yêu cầu đó. Đã có nhiều phương pháp tính toán được đưa ra để giải quyết bài toán, mỗi phương pháp có những ưu điểm, nhược điểm riêng. Trong luận văn này, em sẽ tìm hiểu về nhóm phương pháp sử dụng máy vector hỗ trợ quyết định SVM để tìm ra các vùng tăng cường tương ứng với gen. Do đặc trưng của SVM, khi ta áp dụng các hàm kernel khác nhau thì kết quả được trả về cũng sẽ khác nhau. Trong luận văn này, em xin đề xuất một cách xây dựng hàm kernel mà khi áp dụng vào SVM để xác định một chuỗi ADN có phải là vùng tăng cường của một gen hay không sẽ đưa ra được kết quả với độ chính xác cao hơn khi so sánh với những phương pháp khác.

# **CHƯƠNG 1: CÁC KHÁI NIỆM VÀ VẤN ĐỀ LIÊN QUAN ĐẾN BÀI TOÁN XÁC ĐỊNH VÙNG TĂNG CƯỜNG**

Trong chương này, em sẽ giới thiệu về các khái niệm trong tin sinh học liên quan đến bài toán xác định vùng tăng cường như sự hoạt động của gen, điều hòa hoạt động của gen, nhân tố phiên mã (transcription factor), vùng tăng cường (enhancer) và trình bày về các nhóm phương pháp tìm kiếm vùng tăng cường đã có.

## **Các khái niệm**

## **Sự hoạt động của gen**

Các gen là đơn vị mang thông tin di truyền của sinh vật. Tùy vào loại tế bào và điều kiện môi trường cụ thế, một gen có thể hoạt động hay không hoạt động. Sự hoạt động của gen là quá trình tổng hợp thành các sản phẩm gen từ những thông tin được mã hóa trong gen. Các sản phẩm gen thường là các protein, nhưng cũng tồn tại các gen không mã hóa thông tin cho protein như rARN, tARN hay snARN, sản phẩm của những gen này là các ARN chức năng.

Quá trình hoạt động của gen bao gồm 2 giai đoạn chính là phiên mã và giải mã.

Phiên mã là quá trình sao chép thông tin di truyền được mã hoá dưới dạng trình tự các nucleotide trên gen thành dạng trình tự các ribonucleotide trên ARN thông tin (mARN) để mARN trực tiếp thực hiện chức năng truyền đạt thông tin di truyền đến cấu trúc phân tử protein trong quá trình giải mã nhờ đó mà tổng hợp những protein đặc thù cho gene.

Còn ở quá trình dịch mã, tế bào phải “phiên dịch” trình tự các bazơ của một phân tử mARN thành trình tự các axit amin của một chuỗi polypeptit. Vị trí diễn ra sự dịch mã là các ribosome; đó là phức hệ dạng hạt tạo điều kiện thuận lợi cho sự kết nối các axit amin theo một trật tự nhất định để hình thành nên các chuỗi polypeptit. Sau đó, từ các chuỗi polypeptit này, sẽ tạo thành protein đơn giản (chỉ bao gồm các chuỗi polypeptit) hoặc protein phức tạp (ngoài các chuỗi polypeptit ra còn có thêm một số thành phần phi protein như: axit nucleic, lipit, cacbonhidrat).

## **Điều hòa sự hoạt động của gen**

Sự hoạt động của gen là khác nhau giữa các tế bào. Đây là kết quả của một quá trình gọi là điều hòa sự hoạt động của gen. Quá trình này sẽ quyết định sự phát triển và phản ứng của gen đối với môi trường.

Ở quá trình phiên mã, sự hoạt động của gen bị điều khiển bởi các nhân tố phiên mã (transcription factor – TF). Các TF sẽ tác động đến gen dựa vào việc nhận diện các chuỗi nucleotide đặc trưng đối với gen đó, các chuỗi nucleotide này được gọi là các mô típ. Những mô típ này được gọi là với cái tên chung là các vùng liên kết nhân tố phiên mã (transcription factor binding site – TFBS). Các TFBS thường có vị trí ở gần nhau trên chuỗi ADN. Những TFBS ở gần gen mà nó tác động được gọi là các module điều hòa ở trên cùng một phía (cis-regulatory module – CRM), còn những TFBS ở xa gen thì được gọi là các vùng tăng cường (enhancer). Chính xác hơn, ta có thể xem các vùng tăng cường chính là các CRM có thể tăng cường khả năng hoạt động của gen từ xa. Việc xác định được các vùng tăng cường và giải mã được những quy luật liên quan đến chức năng của nó sẽ cung cấp cho ta những thông tin cần thiết để có thể lý giải được cách thức điều hòa sự hoạt động của gen diễn ra trong tế bào.

## **Nhân tố phiên mã (transcription factor)**

Nhân tố phiên mã (TF) là một thành phần quan trọng trong quá trình điều hòa sự hoạt động của gen ở giai đoạn phiên mã. Chúng thường là các protein, nhưng cũng có thể bao gồm một đoạn ARN ngắn, và không được mã hóa. Những TF có liên quan đến nhau thường nhóm lại và thực hiện tương tác lẫn nhau để điều khiển quá trình phiên mã theo các mức độ khác nhau. Các gen luôn có những vùng có những mô típ đặc trưng riêng biệt (được gọi chung là các CRM) và có thể được nhận diện bởi rất nhiều loại TF. Các TF này sẽ liên kết, tác động lên một loạt các TF khác và tạo thành một chuỗi tác động phức tạp. Cuối cùng sẽ tác động đến enzim ARN polymerase, từ đó bắt đầu (hoặc gây ức chế) quá trình phiên mã.

## **Vùng tăng cường (enhancer)**

Vùng tăng cường (enhancer) là một đoạn ADN ngắn có thể liên kết với các TF để tăng cường khả năng phiên mã của gen trong một nhóm gen (gene cluster – là một tập gồm hai hoặc nhiều gen được dùng để mã hóa những sản phẩm (sản phẩm của gen gồm protein và ARN) giống nhau hoặc tương tự nhau). Nhưng khác với các CRM khác, vùng tăng cường không cần phải ở vị trí gần với gen mà nó tác động đến, thậm chí trong một số trường hợp còn không nằm cùng một sợi nhiễm sắc thể.

## **Các nhóm phương pháp xác định vùng tăng cường.**

Để có thể xác định được một vùng tăng cường, ta có thể dựa vào sự ràng buộc về mặt không gian (nghĩa là một vùng tăng cường phải ở gần những vùng ADN khác liên kết với nó) và sự ràng buộc về mặt hệ thống loài (nghĩa là một vùng tăng cường có thể tồn tại ở nhiều loài có mối liên quan chặt chẽ đến nhau). Hiện nay có rất nhiều phương pháp được đưa ra để tìm kiếm vùng tăng cường. Những phương pháp này có thể chia một cách tương đối thành 4 nhóm sau: phân nhóm cửa sổ (windows clustering), mô hình xác suất (probabilistic modeling), dựa vào mô hình hệ thống sinh vật (phylogenetic footprinting) và mô hình phân biệt (discriminative modeling).

## **Kết luận chương**

Ở trong chương 1, em đã giới thiệu về các khái niệm trong tin sinh học liên quan đến bài toán xác định vùng tăng cường như sự hoạt động của gen, nhân tố phiên mã (transcription factor), vùng tăng cường (enhancer) và đã trình bày về các nhóm phương pháp tìm kiếm vùng tăng cường đã có. Trong các nhóm phương pháp đã trình bày thì nhóm phương pháp mô hình phân biệt có những điểm nổi trội hơn so với những nhóm phương pháp khác. Chính vì thế, phương pháp em đề xuất ở trong báo cáo này cũng thuộc về nhóm này. Nó tương tự với các phương pháp được trình bày trong [4], [5] nhưng có sử dụng thêm cả thông tin về vị trí của các vùng tăng cường khi xây dựng hàm kernel, từ đó giúp cho kết quả thu được có độ chính xác cao hơn. Ở phần kế tiếp, em sẽ trình bày về nhóm phương pháp xác định vùng tăng cường dựa trên SVM [1], [14] , cũng như cách xây dựng hàm kernel của mình để đạt được kết quả tối ưu.

# **CHƯƠNG 2: PHƯƠNG PHÁP XÁC ĐỊNH VÙNG TĂNG CƯỜNG DỰA TRÊN SVM**

Trong chương này, em sẽ trình bày về nhóm phương pháp xác định vùng tăng cường dựa trên máy vector hỗ trợ quyết định SVM [1], [14], giới thiệu về các hàm kernel đã được sử dụng để giải quyết bài toán và đề xuất cách xây dựng hàm kernel để đạt được kết quả tối ưu.

* 1. **Phương pháp xác định vùng tăng cường dựa trên SVM**
     1. **Giới thiệu về SVM**
     2. **Giới thiệu về hàm kernel trong SVM**

Để mở rộng khả năng phân lớp của thuật toán SVM, thay vì sử dụng hàm tích nội (inner product), người ta đưa ra khái niệm kernel để đo độ tương đồng giữa 2 mẫu trong không gian dữ liệu huấn luyện đầu vào.

Đầu tiên, dữ liệu huấn luyện sẽ được chuyển sang không gian *H* (chính là không gian đặc trưng (feature space)) bằng hàm ánh xạ như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (2.10) |

Để tính độ tương đồng giữa các mẫu học trong *H*, ta có thể sử dụng hàm tích nội tương ứng trong không gian *H*, ký hiệu . Để tiện lợi, ta định nghĩa hàm tương ứng như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (2.11) |

mà thỏa điều kiện:

|  |  |  |
| --- | --- | --- |
|  |  | (2.12) |

Hàm như ở công thức (2.12) chính là hàm kernel. Với cách thức thay hàm tích nội bởi hàm kernel như trên, chúng ta có thể sửa dụng SVM trong không gian đặc trưng *H* mà hơn nữa, nếu chúng ta có thể tính được trực tiếp giá trị của mà không cần phải tính chính xác các vector đặc trưng (feature vector), từ đó có thể giúp cho việc tính toán sẽ trở nên nhẹ nhàng hơn rất nhiều đối với không gian đặc trưng có nhiều chiều.

* + 1. **Phương pháp xác định vùng tăng cường dựa trên SVM**

SVM là một phương pháp học máy được đánh giá khá cao và được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau. Điểm mạnh của phương pháp này đó là tính chính xác và sự linh hoạt, nó có thể làm việc với những nguồn và loại dữ liệu khác nhau, thậm chí là với những dữ liệu không được biểu diễn dưới dạng các vector đặc trưng. Điều này thực sự hữu ích vì những các chuỗi gen của sinh vật thường là những dữ liệu không được biểu diễn dưới dạng vector.

Đối với phương pháp xác định vùng tăng cường dựa trên SVM, bài toán được đưa về dạng bài toán phân loại nhị phân, nghĩa là xác định xem một chuỗi ADN là vùng tăng cường hay không dựa vào một bộ phân loại được huấn luyện trước. Lúc này, bộ phân loại sẽ kiểm tra các chuỗi gen đầu vào xác định vùng nào là vùng tăng cường ta cần tìm kiếm.

Như đã giới thiệu ở trên, quá trình tìm ra một bộ phân loại dựa vào SVM, cũng chính là quá trình ta định nghĩa ra một hàm kernel có thể tính ra được độ tương tự giữa hai đối tượng x và y. Hàm kernel được định nghĩa ra nên phản ánh được miền tri thức mà ta đang sử dụng và điều này là cực kỳ quan trọng, nó sẽ ảnh hưởng đến mức độ hiệu quả của phương pháp. Dưới đây, em sẽ giới thiệu về một số hàm kernel đã được giới thiệu và sử dụng trong các phương pháp xác định vùng tăng cường dựa vào SVM đã có trước đó.

* + 1. **Một số hàm kernel đã được sử dụng để giải quyết bài toán**

Hàm kernel đầu tiên em muốn nói đến là hàm spectrum kernel. Đây là hàm kernel được giới thiệu bởi Leslie [7] và được xem là một trong những hàm kernel đơn giản và dễ dàng sử dụng nhất khi tính toán. Hơn nữa, nó có thể phù hợp với rất nhiều mô hình dữ liệu khác nhau, và khi áp dụng hàm kernel vào SVM thì thời gian để phân loại dữ liệu đầu vào đều ở dạng tuyến tính. Đối với dữ liệu dạng chuỗi, hàm kernel này lại càng được sử dụng rộng rãi do những ưu điểm của nó.

Leslie đã đưa ra định nghĩa *k-mer* là tất cả những đoạn có độ dài *k* có thể tạo được từ tất cả các ký tự thuộc tập *Ʃ (Ʃ = {A, C, T, G}* đối với các chuỗi ADN). Khi đó, vector đặc trưng cho một chuỗi *x* có độ dài *l* (trong chuỗi *x* có chứa các ký tự thuộc tập *Ʃ*) có thể được biểu diễn như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (2.13) |

Trong đó, là số lần *k-mer* xuất hiện ở trong chuỗi *x*. Đối với trường hợp của ADN, vector đặc trưng sẽ có trường, tương ứng với *k-mer*.

Lúc này, hàm spectrum kernel sẽ trở thành như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (2.14) |

Mặc dù số trường trong vector đặc trưng có thể rất lớn do tăng theo *k* với cấp số mũ, nhưng số giá trị khác 0 ở trong vector đặc trưng tối đa chỉ là . Hơn nữa, bản chất của một hàm kernel là bắt cặp tự do (alignment-free) bởi vì nó không cần phải tiến thành bắt cặp hai chuỗi khi tính độ tương đồng giữa chúng, điều này giúp cho việc tính toán hàm kernel này trở nên dễ dàng và thuận tiện hơn rất nhiều.

Từ hàm spectrum kernel, Leslie đã mở rộng nó để tạo thành hàm kernel ghép đôi bất đối xứng (mismatch kernel) [8], được ký hiệu là , trong đó, *k* chính là dộ dài của *k-mer*, còn *m* quy định số lượng ký tự tối đa có thể khác khi đếm số lần xuất hiện của một *k-mer* trên một chuỗi. Nghĩa là, với một *k-mer* , khi đếm số lần xuất hiện của trên một chuỗi, ta sẽ tính thêm cả số lần xuất hiện của các chuỗi khác sao cho chỉ khác tối đa *m* ký tự. Điều này cho phép quá trình so sánh giữa các chuỗi có thể ghi nhận được cả các mô típ có thể bị biến đổi trong quá trình phát triển và tiến hóa.

Cũng được mở rộng từ hàm spectrum kernel, có một hàm kernel khác được đánh giá cao là hàm kernel của Lee et al., được giới thiệu ở [6]. Như ta đã biết, các chuỗi ADN được tạo thành từ các bazơ A, T, G, C, các bazơ này bắt cặp theo quy tắc: A bắt cặp với T và ngược lại, G bắt cặp với C và ngược lại. Do đó, ứng với mỗi *k-mer*, ta sẽ có một *k-mer* khác ở dạng đảo ngược với nó. Ví dụ như ta có chuỗi TTGCGAT, thì dạng đảo ngược của nó sẽ là ATCGCAA. Khi xây dựng vector đặc trưng cho hàm kernel mới, Lee et al. [6] đã xem một *k-mer* và *k-mer* khác ở dạng đảo ngược của nó là như nhau và sẽ chỉ sử dụng những *k-mer* khác biệt hoàn toàn để làm giảm độ lớn của không gian đặc trưng, từ đó giúp cho quá trình tính toán càng trở nên dễ dàng hơn và đồng thời cũng thu được kết quả tốt hơn.

* 1. **Các hàm kernel được đề xuất**

### **Hàm kernel dựa trên entropy**

Hàm spectrum kernel và những hàm mở rộng của nó như hàm kernel ghép đôi bất đối xứng (mismatch kernel) dựa trên sự xuất hiện cùng nhau của các *k-mer* trong các chuỗi khác nhau mà không quan tâm đến vị trí của các *k-mer* này. Vì thế, nếu như nhiều *k-mer* thường xuất hiện cùng với nhau hoặc ở vị trí gần giống nhau ở các chuỗi đầu vào thì các hàm kernel này cũng không thể ghi nhận được các đặc điểm này. Trong thực tế, để quá trình liên kết có thể diễn ra, trong một vài trường hợp, các TFBS của cùng một TF cần phải nằm gần nhau (khoảng cách giữa chúng thường trong khoảng 10 bp).

Trong phần này, em sẽ giới thiệu một hàm kernel mới có kết hợp với thông tin về khoảng cách giữa các *k-mer* giống nhau. Khi xem xét đến số lần mà mỗi *k-mer* xuất hiện ở trên chuỗi đầu vào, em sử dụng một đơn vị đo lường để mô tả mức độ tập trung của nó, nghĩa là xem xét nó xuất hiện gần nhau hay là trải đều trên toàn bộ chuỗi. Đơn vị đo lường được chọn là entropy.

Cách thức tính entropy như sau: cho một chuỗi có độ dài *l*, đầu tiên em chia chuỗi đó thành *n* phần có độ dài bằng nhau (các phần này có thể chồng lên nhau để đảm bảo các *k-mer* quan trọng không bị bỏ qua khi chúng nằm ở vùng ranh giới giữa các phần). Tiếp đó, lần lượt với mỗi *k-mer*, em đếm số lần mà nó xuất hiện trên *n* phần này, sau đó em chuẩn hóa các số đã thu được để tổng của chúng = 1, giả sử khi đó, các giá trị thu được là . Nếu các vị trí xuất hiện của một *k-mer* ở trên chuỗi nằm gần nhau, thì chỉ có một hoặc một vài là có giá trị khác 0. Ngược lại, giải giá trị của sẽ rất gần với giải giá trị chuẩn. Để có thể lượng hóa đặc trưng của sự phân bố này, em tính entropy như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (2.15) |

Giá trị của entropy này sẽ trải đều từ (khi *k-mer* chỉ xuất hiện ở trong cùng một phần), đến , khi *k-mer* này trải đều trên toàn bộ *n* phần. Từ giá trị entropy này, em đưa ra giá trị đặc trưng cho một *k-mer* theo công thức sau:

|  |  |  |
| --- | --- | --- |
|  |  | (2.16) |

Từ công thức trên, ta có thể thấy rằng, giá trị của sẽ nằm trong đoạn . Sau đó, giá trị đặc trưng sẽ được kết hợp với đặc trưng của hàm spectrum. Có khá nhiều cách để kết hợp hai đặc trưng với nhau, như ta có thể tính từng hàm kernel cho từng đặc trưng riêng lẻ, rồi sau đó cộng các hàm kernel đó lại với nhau. Ở đây, em chỉ đơn giản là nối vector của đặc trưng dựa trên entropy với đặc trưng của spectrum để tạo thành vector đặc trưng. Tiếp đó, hàm kernel được tính như là hàm tích nội của vector đặc trưng được kết hợp này.

### **Hàm kernel phân cấp (hierarchical kernel)**

Đặc trưng dựa trên entropy được giới thiệu ở phần trước chỉ có thể ghi nhận được những thông tin về vị trí tương đối của các vị trí xuất hiện của cùng 1 *k-mer*. Tuy nhiên, trên thực tế, có nhiều vùng tăng cường chứa nhiều nhóm của các TFBS khác nhau và các TFBS này thường vị trí gần nhau. Đặc trưng dựa trên entropy chưa thể hiện được các ràng buộc về khoảng cách giữa TFBS khác nhau này. Trong phần này, em sẽ giới thiệu về một loại đặc trưng và hàm kernel khác có thể ghi nhận được thông tin này.

Ý tưởng chính của phương pháp này là: 1) chia chuỗi đầu vào thành các phần có độ dài tương tự nhau; 2) đếm số lần xuất hiện của các *k-mer* trong mỗi phần; và 3) chia các phần này thành các phần con và lặp lại quá trình này cho đến khi độ dài của các phần này là nhỏ nhất để có thể chứa các *k-mer*. Khi quá trình này kết thúc, ta sẽ thu được các con số và chúng sẽ được dùng như là các đặc trưng để tạo thành hàm kernel. Kết quả của quá trình này được thể hiện theo sự phân cấp của các đặc trưng mà trong đó, các đặc trưng ở mức thấp hơn sẽ ghi nhận thông tin về sự xuất hiện cùng nhau của các *k-mer* thường có vị trí gần nhau, còn các đặc trưng ở mức cao hơn sẽ cho thấy sự xuất hiện của các k-mer trong chuỗi đầu vào với ràng buộc về vị trí sẽ yếu hơn.

Với trường hợp phân cấp ở mức 2 lớp, là trường hợp mà em sẽ sử dụng để thực nghiệm, các đặc trưng và hàm kernel sẽ được tính như sau. Đầu tiên, em chia chuỗi đầu vào thành *n* phần có độ dài bằng nhau giống như với trường hợp đặc trưng dựa trên entropy. Tiếp đó, em đếm số lần xuất hiện của mỗi *k-mer* trên toàn bộ chuỗi và trên mỗi phần (Hình 2.1b). Như vậy, ta sẽ có *n + 1* vector chứa số lần xuất hiện của các *k-mer* trên chuỗi đầu vào và các phần của nó. Để làm giảm sự ảnh hưởng của việc các chuỗi đầu vào có thể có độ dài khác nhau, em chuẩn hóa các vector về dạng đơn vị, tức là sao cho . Khi đó, giá trị kernel giữa hai chuỗi *x* và *x’* sẽ được tính như là tổng của *n + 1* kernel, với mỗi kernel trong đó tương ứng với vector đếm số lần xuất hiện của các *k-mer* trên 2 chuỗi x và x’, như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (2.17) |

Trong đó, là hàm kernel giữa các vector đếm từ hai chuỗi *x* và *x’*. Nếu là hàm kernel tuyến tính, nghĩa là , thì hàm kernel ở trên có thể được tính bằng cách nối tất cả *n + 1* vector đếm vào một vector dài, và sẽ tính hàm tích nội giữa các vector dài của 2 chuỗi này. Em gọi chúng là các đặc trưng và hàm kernel phân cấp, bởi vì chúng được xây dựng dựa trên quy tắc phân cấp khi chia chuỗi đầu vào thành các phần bé hơn và tiếp tục chia nhỏ trên các phần này.

Điểm chú ý cuối cùng là khi xác định các *k-mer*, tương tự như phương pháp của Lee được đưa ra ở [5], trong phương pháp này, một *k-mer* và *k-mer* ở dạng đảo ngược của nó sẽ được xem là như nhau và sẽ chỉ sử dụng những *k-mer* khác biệt. Ví dụ, AACTTG và CAAGTT sẽ được xem là các *6-mer* giống nhau. Vì thế với phương pháp được cài đặt với , sẽ chỉ có 2080 *6-mer* khác biệt được sử dụng, thay vì là *6-mer* như ở hàm spectrum kernel.

# **CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ**

Trong chương này, em sẽ trình bày về cách thức xây dựng bộ dữ liệu mẫu, quá trình cài đặt phương pháp xác định vùng tăng cường đã được giới thiệu ở chương 2, tiêu chí để đánh giá tính hiệu quả của phương pháp, đánh giá kết quả, so sánh kết quả với các phương pháp khác, đưa ra nhận xét về phương pháp cũng như hướng phát triển trong tương lai.

1. **Cài đặt phương pháp**

### **Xây dựng bộ dữ liệu mẫu**

Để đánh giá phương pháp, em sử dụng bộ dữ liệu về vùng tăng cường của nhiều loại TF khác nhau của người và loài giun tròn. Cụ thể, em sử dụng các dữ liệu có sẵn từ các phương pháp ChIP-chip và ChIP-seq để tạo ra một tập các enhancer cho các TF sau: TAL1 (được lấy từ [10]), HNF4A, GATA6, CDX2, H3K4me2 (được lấy từ [17]) và PHA-4 (được lấy từ [21]). Trừ dữ liệu của TF PHA-4 là dữ liệu của loài giun tròn, các bộ dữ liệu khác đều là của người.

Để có được các tập dữ liệu dương (tức là những tập dữ liệu chứa các vùng tăng cường), em thực hiện các bước tương tự như cách thức đã được sử dụng bởi Yanez-Cuna [19]. Dữ liệu thô ChIP-seq được lấy từ [17], [10] và [21] sẽ được ánh xạ lên trình tự hệ gen của người và giun tròn được lấy từ UCSD Genome Browser (<http://genome.ucsd.edu>). Công đoạn ánh xạ được thực hiện bằng chương trình Bowtie được giới thiệu ở [5]. Bằng cách sử dụng MACS [20], em có thể lấy được danh sách các peak của ChIP (peak là những vùng mang tín hiệu để định danh các vùng chức năng trên ADN trong các quá trình liên kết protein). Sau đó, đối với các TF của người, em sẽ lấy 1000 peak có giá trị p-value cao nhất, còn với loài giun tròn thì sẽ là 500 peak. Với những peak bị trùng lên nhau, là những peak mà khoảng cách giữa chúng chỉ cách nhau tầm khoảng 300bp, em sẽ chỉ lấy 1 peak, còn những vùng khác trong nhóm đó sẽ bị loại. Sau khi đã xác định được một peak, em sẽ mở rộng nó về 2 phía ở trên gen với độ dài khoảng từ 300 đến 500 bp để có được 1 chuỗi vùng tăng cường có độ dài xấp xỉ 1000 bp. Độ dài của phần được mở rộng này được chọn theo cách này để đảm bảo rằng vị trí của peak sẽ không chỉ ở vị trí trung tâm của chuỗi vùng tăng cường, điều đó sẽ làm cho việc dự đoán trở nên khó khăn hơn, giúp cho quá trình đánh giá phương pháp được chính xác hơn.

Để tạo ra các tập dữ liệu âm (nghĩa là các tập dữ liệu gồm các chuỗi đã được xác định không phải là vùng tăng cường), em sử dụng thuật toán tương tự như phương pháp của Lee [5]. Ứng với mỗi tập dữ liệu dương đã có ở trước, em sử dụng thuật toán lấy mẫu – loại bỏ dưới đây để chọn ra ngẫu nhiên các chuỗi gen của loài sinh vật tương ứng, từ đó tạo thành tập dữ liệu âm tương ứng với tập dữ liệu dương. Các tập dữ liệu này được tạo ra sao cho phù hợp với sự phân bố về độ dài chuỗi và trị số lặp lại (repeat fraction) các thành phần của tập dữ liệu dương tương ứng. Để có được trị số lặp lại, ta dựa vào dữ liệu về các chuỗi lặp lại (repeat sequence) được lấy từ UCSD Genome Browser. Dưới đây là thuật toán để có được tập dữ liệu âm, gồm các bước như sau:

1. Lấy ngẫu nhiên một chuỗi dương (là chuỗi vùng tăng cường) có độ dài *l*.
2. Lấy ngẫu nhiên một chuỗi có độ dài *l* từ bộ gen tương ứng với chuỗi dương đã có ở trước và tính ra trị số lặp của nó.
3. Xác suất để chuỗi có ở bước 2 được chọn vào tập dữ liệu là , trong đó, và là xác suất mà có khả năng xuất hiện ở trên chuỗi enhancer và chuỗi gen tương ứng, còn là số tiêu chuẩn.
4. Lặp đi lặp lại các bước từ 1-3 cho đến khi số lượng các chuỗi âm gấp đôi số lượng các chuỗi dương.

Dữ liệu các chuỗi thuộc tập dữ liệu âm và dương sẽ được lưu dưới định dạng file fasta để làm đầu vào cho chương trình được cài đặt nhằm kiểm tra tính hiệu quả của phương pháp.

### **Cài đặt phương pháp**

Phương pháp đề xuất được cài đặt bằng ngôn ngữ Python và có sử dụng bộ công cụ hỗ trợ học máy Shogun [12]. Em sử dụng hàm nhân CommWordStringKernel có sẵn trong bộ công cụ Shogun để cài đặt hàm spectrum kernel. Hàm nhân mới đề xuất được em cài đặt hoàn toàn mới. Em sử dụng thư viện SVM của Shogun với giá trị . Dựa trên kết quả của phương pháp của Lee [5], khi ta có thể thu được kết quả tốt nhất với , tức là sử dụng *6-mer*, em cũng sẽ chỉ sử dụng *6-mer* mà không tiến hành thực nhiệm với những độ dài khác.

1. **Tiêu chí đánh giá**

Lâu nay, đường cong ROC (Receiver Operating Characteristics) vẫn được sử dụng để đánh giá các thuật toán học máy. Ở phương pháp đánh giá này, ta cần sắp xếp các giá trị sau khi phân loại thành thành một dãy tăng dần, sau đó biểu diễn chúng trên đồ thị hai chiều. Khi đó, các điểm này sẽ tạo thành một đường cong, được gọi là đường cong ROC. Tuy nhiên, thông thường, khi so sánh hiệu năng của 2 thuật toán, sự phân biệt giữa hai đường cong ROC tương ứng với chúng ở trên đồ thị là không rõ ràng. Chính vì thế, để so sánh chính xác hơn, người ta đã đưa ra một phép đo khác dựa trên đường cong ROC, đó là AUC (area under the ROC curve). Giá trị AUC chính là diện tích của vùng nằm dưới đường cong ROC (area under the ROC curve), và cũng là một con số thể hiện hiệu năng của thuật toán [9]. Giá trị của AUC nằm trong khoảng (0, 1). Nếu giá trị AUC = 1 thì khi đó ta có thể khẳng định rằng bộ phân loại được đưa ra hoạt động hoàn hảo, nghĩa là tất cả những chuỗi vùng tăng cường sẽ được tách biệt với những chuỗi không phải là vùng tăng cường, khi giá trị AUC = 0.5 thì sẽ tương ứng với quá trình lựa chọn ngẫu nhiên, còn nếu AUC bé hơn 0.5 thì bộ phân loại bị đánh giá là không có hiệu quả.

Thông thường, các SVM sử dụng mức ngưỡng = 0 để quyết định nhãn mà dữ liệu đầu vào có thể được nhận. Tuy nhiên, trong một vài trường hợp, chúng ta có thể phải thay đổi giá trị ngưỡng này để có thể cân bằng giữa độ nhạy và độ đặc trưng. Trong những trường hợp đó, AUC là một tiêu chí phù hợp bởi vì nó có thể tổng hợp cả hiệu năng của bộ phân loại (mà ở đây là SVM) trên toàn bộ dải giá trị của mức ngưỡng và ta không cần phải chọn ra ngưỡng giới hạn cụ thể. Hơn nữa, khi ta sử dụng bộ phân bộ phân loại để gán nhãn cho một tập dữ liệu đầu vào, độ đo AUC sẽ tương đương với xác suất mà bộ phân loại sẽ xếp các dữ liệu dương ở vị trí cao hơn các dữ liệu âm khác [3]. Và như trong [9] đã chứng minh, nếu chúng ta xây dựng một bộ phân loại được tối ưu theo AUC, thì không chỉ giá trị AUC, mà độ chính xác của nó cũng sẽ cao hơn so với những bộ phân loại khác được tối ưu theo độ chính xác.

1. **Đánh giá kết quả**

Để đánh giá hiệu quả của phương pháp, em sử dụng phương thức năm lần xác nhận chéo (five-fold cross-validation). Nghĩa là mỗi tập dữ liệu sẽ được chia ngẫu nhiên thành 5 tập con có số lượng chuỗi dương và âm gần như bằng nhau. Một tập con sẽ được giữ lại để làm tập dữ liệu kiểm tra, còn 4 tập còn lại sẽ được sử dụng để huấn luyện SVM, sau đó SVM này sẽ được dùng để dự đoán trên tập dữ liệu kiểm tra. Quá trình này sẽ được lặp đi lặp lại 5 lần sao cho tất cả các tập con đều sẽ được chọn làm tập dữ liệu kiểm tra. Độ đo AUC sẽ được tính trung bình từ các giá trị có được từ 5 lần lặp đó.

Đầu tiên, em so sánh hai hàm kernel đã đề xuất là hàm kernel phân cấp và hàm kernel dựa trên entropy. Để so sánh hai hàm kernel này, em thay đổi số lượng phần con *n* từ 3 đến 11 phần và các phần con không chồng lên nhau. Giá trị AUC ứng với hàm kernel phân cấp và hàm kernel dựa trên entropy được tổng kết lần lượt ở Bảng 3.1 và Bảng 3.2.

**Bảng 3.1. Giá trị AUC () đối với hàm kernel phân cấp ứng với số lượng các phần khác nhau và giữa các phần không chồng lên nhau. Giá trị AUC cao nhất ứng với mỗi TF được bôi đậm**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TF** | **n=3** | **n=4** | **n=5** | **n=6** | **n=7** | **n=8** | **n=9** | **n=10** | **n=11** |
| TAL1\_erythroid | 89.01 | 86.57 | 92.48 | 89.35 | 93.50 | 90.83 | **93.67** | 91.28 | 93.17 |
| HNF4A\_prolCell | 82.16 | 79.83 | 84.50 | 82.45 | 85.31 | 83.13 | **85.60** | 83.81 | 84.79 |
| GATA6\_prolCell | 97.99 | 96.48 | **98.53** | 97.09 | 98.46 | 97.23 | 98.37 | 97.76 | 97.97 |
| CDX2\_prolCell | 88.00 | 86.96 | **90.55** | 89.16 | 90.11 | 90.10 | 90.05 | 90.15 | 89.65 |
| PHA4\_embryo | 89.68 | 87.36 | 93.15 | 90.40 | **93.46** | 92.22 | 93.45 | 92.97 | 92.96 |
| H3K4me2\_prolCell | 78.90 | 78.67 | 81.77 | 80.62 | 82.74 | 82.10 | 82.40 | **83.16** | 82.13 |
| Trung bình | 87.62 | 85.97 | 90.16 | 88.18 | **90.60** | 89.27 | 90.59 | 89.86 | 90.11 |

**Bảng 3.2. Bảng giá trị AUC () đối với hàm kernel dựa trên entropy ứng với số lượng các phần khác nhau và giữa các phần không chồng lên nhau.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TF** | **n=3** | **n=4** | **n=5** | **n=6** | **n=7** | **n=8** | **n=9** | **n=10** | **n=11** |
| TAL1\_erythroid | 86.22 | 86.14 | 86.05 | 86.00 | 85.87 | 85.80 | 85.62 | 85.57 | 85.65 |
| HNF4A\_prolCell | 78.27 | 78.21 | 78.20 | 78.08 | 78.13 | 78.12 | 78.08 | 77.94 | 77.85 |
| GATA6\_prolCell | 94.85 | 94.89 | 94.71 | 94.86 | 94.62 | 94.85 | 94.54 | 94.82 | 94.29 |
| CDX2\_prolCell | 85.15 | 85.17 | 85.08 | 85.10 | 84.93 | 85.01 | 84.86 | 84.71 | 84.54 |
| PHA4\_embryo | 86.34 | 86.34 | 86.38 | 86.19 | 86.27 | 86.32 | 86.00 | 86.10 | 86.09 |
| H3K4me2\_prolCell | 79.89 | 79.85 | 79.84 | 79.83 | 79.98 | 79.73 | 79.58 | 79.58 | 79.60 |
| Trung bình | 85.12 | 85.1 | 85.04 | 85.01 | 84.96 | 84.97 | 84.78 | 84.78 | 84.67 |

Từ kết quả có được, ta có thể thấy rằng hai phương pháp được đề xuất có điểm số AUC trung bình khá cao với 85% cho hàm kernel dựa trên entropy và 90% cho hàm kernel phân cấp. Tuy nhiên, khoảng cách điểm số giữa hai phương pháp là khá đáng kể khi điểm của hàm kernel dựa trên entropy thấp hơn khoảng 5% so với hàm kernel phân cấp. Nguyên nhân có thể là do những tập dữ liệu được sử dụng đều chứa các vùng tăng cường khác nhau của một TF, trong khi hàm kernel dựa trên entropy được thiết kế cho bài toán xác định một loại vùng tăng cường của một TF. Điểm số AUC trung bình của hàm kernel phân cấp vào khoảng 90.6% và điểm cao nhất là 98.53%, đạt được khi kiểm tra với tập dữ liệu vùng tăng cường được liên kết bởi TF GATA6 và các nhân tố phối hợp (co-factor). Ta cũng có thể nhận thấy rằng, các điểm số AUC tốt nhất mà phương pháp có thể đạt được đều là khi số phần con là từ 5 đến 9 phân, tương đương với độ dài của từng phần con vào khoảng từ 100 bp đến 200 bp.

**Bảng 3.3. Bảng giá trị AUC () đối với hàm kernel phân cấp ứng với số lượng phần con là 7 và các độ dài vùng chồng lên nhau khác nhau.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TF** | **L = 0** | **L = 10** | **L = 20** | **L = 30** | **L = 40** |
| TAL1\_erythroid | 93.50 | 93.11 | 92.39 | 91.70 | 91.09 |
| HNF4A\_prolCell | 85.31 | 85.08 | 84.66 | 84.17 | 83.52 |
| GATA6\_prolCell | 98.46 | 98.51 | 98.52 | 98.35 | 98.17 |
| CDX2\_prolCell | 90.11 | 90.46 | 90.36 | 90.33 | 89.95 |
| PHA4\_embryo | 93.46 | 93.48 | 93.06 | 92.33 | 91.66 |
| H3K4me2\_prolCell | 82.74 | 82.37 | 82.15 | 81.47 | 80.87 |
| Trung bình | 90.59 | 90.50 | 90.19 | 89.72 | 89.21 |

Tiếp theo, em đánh giá hiệu quả của phương pháp hàm kernel phân cấp với các cách chia có độ dài vùng chồng lên nhau khác nhau. Với bước đánh giá này, như kết quả đã có ở trên thì hàm kernel phân cấp có điểm số AUC khá cao khi n có giá trị từ 5 đến 9 nên trong phép so sánh này, em lấy số phần con *n* = 7, còn độ dài *L* của phần chồng lên nhau sẽ được thay đổi từ 0 (tức là các phần con không chồng lên nhau) đến 40 bp. Những điểm số AUC thu được ứng với các trường hợp có độ dài *L* thay đổi được tổng kết ở Bảng 3.3. Như ta có thể thấy, trong hầu hết các trường hợp, việc chia thành các phần con có độ dài phần chồng lên nhau lớn hơn 10 bp sẽ làm giảm hiệu năng của phương pháp. Vì thế, khi sử dụng hàm kernel phân cấp, ta chỉ nên chia phần con với vùng chồng lên nhau có độ dài bằng 0 hoặc hoặc bé hơn 10 bp.

1. **So sánh kết quả với các phương pháp khác**

Để chứng minh tính hiệu quả của phương pháp đề xuất so với những phương pháp hiện có, em so sánh kết quả với phương pháp được đưa ra bởi Lee et al. ở [6], đây là một trong những phương pháp được đánh giá có hiệu quả tốt nhất đến hiện tại khi áp dụng cho các bài toán tương tự như bài toán được đặt ra của luận văn.

**Bảng 3.4. Bảng giá trị AUC () đối với hàm kernel phân cấp ứng với số lượng phần là 7 và phương pháp đưa ra bởi Lee et al.** **[6].**

|  |  |  |
| --- | --- | --- |
| **TF** | **Lee et al. [6]** | **n = 7** |
| TAL1\_erythroid | 86.26 | 93.50 |
| HNF4A\_prolCell | 78.29 | 85.31 |
| GATA6\_prolCell | 94.88 | 98.46 |
| CDX2\_prolCell | 85.18 | 90.11 |
| PHA4\_embryo | 86.33 | 93.46 |
| H3K4me2\_prolCell | 79.88 | 82.74 |
| Trung bình | 85.14 | 90.60 |

Đối với phương pháp đề xuất, em chọn đại diện tiêu biểu nhất là hàm kernel phân cấp với số phần con *n* = 7 và giữa các phần này không có vùng chồng lên nhau. Với phương pháp của Lee et al. [6], do chương trình cài đặt được công khai nên em sẽ chạy chương trình đó với các cài đặt khác đều được để mặc định. Kết quả của quá trình so sánh được tổng kết ở Bảng 3.4. Từ bảng kết quả, ta có thể thấy rằng, kết quả của phương pháp đề xuất có điểm số AUC cao hơn phương pháp của Lee et al. [6] ở tất cả các trường hợp, với mức chênh lệch khoảng 5%. Đây thực sự là một bước cải tiến tuyệt vời khi phương pháp của Lee et al. [6] là một phương pháp được đánh giá khá cao.

1. **Nhận xét về kết quả và hướng phát triển.**

Từ những kết quả thu được từ thực nghiêm, ta có thể thấy rằng, đối với 2 hàm kernel được đề xuất để giải quyết bài toán thì hàm kernel phân cấp hiệu quả hơn so với hàm kernel dựa trên entropy. Đối với cách chia phần con các chuỗi đầu vào, thì ta chỉ nên chia các thành các phần với vùng chồng lên nhau bé hơn 10bp, không nên lớn hơn vì nếu như thế, hiệu năng của phương pháp sẽ bị giảm xuống. Và khi so sánh với những phương pháp khác, phương pháp đề xuất đã chứng tỏ được hiệu năng cao hơn so với phương pháp của Lee et al. [6], một trong những phương pháp được đánh giá có hiệu quả tốt nhất đến hiện tại, mức chênh lệch khoảng 5% với những dữ liệu của người và loài giun tròn. Các hàm kernel được đề xuất đều dựa trên những đặc điểm về chuỗi gen cơ bản và rất đơn giản để cài đặt, tính toán, nó không cần thêm những thông tin khác, cũng như quá trình chuẩn bị dữ liệu đầu vào đơn giản, không phức tạp. Điều này giúp cho phương pháp đề xuất có thể áp dụng vào nhiều bài toán khác nữa.

# **KẾT LUẬN**

Trong luận văn, em đã trình bày những đặc điểm cơ bản, cũng như các khái niệm cơ bản liên quan đến bài toán tìm kiếm vùng tăng cường của một chuỗi gen, giới thiệu về những nhóm phương pháp tìm kiếm vùng tăng cường đã có. Từ đó, luận văn giới thiệu nhóm phương pháp tìm kiếm vùng tăng cường dựa trên SVM. Phương pháp này đã được nhiều nhà nghiên cứu áp dụng bằng cách sử dụng nhiều hàm kernel khác nhau và đã thu được những kết quả tốt. Em cũng đã đề xuất hàm kernel mới chỉ sử dụng tập dữ liệu chuỗi gen và có thể ghi nhận được những thông tin về vị trí của các TFBS là hàm kernel dựa trên entropy và hàm kernel phân cấp.

Khi tiến hành thực nghiệm, do những tập dữ liệu được sử dụng đều chứa các vùng tăng cường khác nhau của một TF, trong khi hàm kernel dựa trên entropy được thiết kế cho bài toán xác định một vùng tăng cường của một TF nên kết quả của hàm kernel phân cấp cao hơn so với của hàm kernel dựa trên entropy. Khi áp dụng các hàm này vào thực tế, tùy vào loại dữ liệu đầu vào, ta sẽ chọn hàm kernel phù hợp. Kết quả thực nghiệm cũng đã chứng tỏ rằng phương pháp đề xuất có hiệu năng cao hơn so với những phương pháp khác đối với những dữ liệu của người và loài giun tròn. Hơn nữa, phương pháp đề xuất chỉ sử dụng dữ liệu về chuỗi gen mà không cần những dữ liệu khác nên rất đơn giản và dễ dàng khi tính toán, cài đặt, cũng như không phức tạp ở khâu chuẩn bị dữ liệu. Điều này giúp cho phương pháp này có thể áp dụng vào nhiều bài toán thực tế khác trong tương lai.

Kết quả của được luận văn được tóm tắt trong bài báo “Enhancer prediction using distance aware kernels”. Bài báo đã được chấp nhận đăng tại kỷ yếu hội nghị RIVF 2013 sẽ diễn ra vào tháng 11/2013.

# **TÀI LIỆU THAM KHẢO**

1. BE. Boser, IM. Guyon, VN. Vapnik, (1992), “*A training algorithm for optimal margin classifiers*”*,* trong *“Proceedings of the Fifth Annual Workshop on Computational Learning Theory”.* Association for Computing Machinery (ACM), New York.
2. D. Carter, L. Chakalova, CS. Osborne, Y. Dai, P. Fraser, (2002), *“Long-range chromatin regulatory interactions in vivo”*. Nat Genet 32: 623–626.
3. T. Fawcett, (2005). *“An introduction to ROC analysis”,* Pattern Recognition Letters, Vol. 27, pp: 861-874.
4. J. Göke, MH. Schulz, J. Lasserre, M. Vingron, (2012), “*Estimation of Pairwise Sequence Similarity of Mammalian Enhancers with Word Neighbourhood Counts. Bioinformatics*”, Vol. 28(5), pp: 656-663.
5. B. Langmead, C. Trapnell, M. Pop, SL. Salzberg, (2009), *“Ultrafast and memory-efficient alignment of short DNA sequences to the human genome”*, Genome Biol 10: R25.
6. D. Lee, R. Karchin and MA. Beer, (2011), “*Discriminative prediction of mammalian enhancers from DNA sequence*”, Genome research, Vol 21(12), pp: 2167-2180.
7. C. Leslie, E. Eskin, WS. Noble, (2002), *“The spectrum kernel: A string kernel for SVM protein classification”*. Pac Symp Biocomput 7, pp: 564–575.
8. C. Leslie, E. Eskin, A. Cohen, J. Weston, WS. Noble, (2003), *“Mismatch string kernels for discriminative protein classification”*, Bioinformatics 20, pp: 467–476.
9. CX. Ling, H. Jin, Z. Harry, (2003), *“AUC: a Statistically Consistent and more Discriminating Measure than Accuracy”*, IJCAI'03 Proceedings, pp: 519-524.
10. CG. Palii, C. Perez-Iratxeta, Z. Yao, Y. Cao, F. Dai, J. Davison, H. Atkins, D. Allan, FJ. Dilworth, R. Gentleman, (2011). *“Differential genomic targeting of the transcription factor TAL1 in alternate haematopoietic lineages”,* EMBO J.Vol 30, pp: 494–509.
11. SJ. Schultheiss, W. Busch, JU. Lohmann, O. Kohlbacher, G. Rätsch, (2009), “*KIRMES: Kernel-based identification of regulatory modules in euchromatic sequences. Bioinformatics*”, Vol 25(16), pp: 2126-2133.
12. S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, V. Franc. (2010), *“The SHOGUN Machine Learning Toolbox. J. Mach. Learn”,* Res. Vol. 11, pp: 1799-1802.
13. E. Ukkonen. (1995), *“On-line construction of suﬃx trees”*, Algorithmica, 14, pp: 249–260.
14. VN. Vapnik (1995), “*The nature of statistical learning theory”*, Springer, New York.
15. VN. Vapnik, A. Lerner. (1963), *“Pattern recognition using generalized portrait method”*, Automation and Remote Control, 24, pp: 774-780.
16. VN. Vapnik, A. Chervonenkis, (1964), *“A note on one class of perceptrons”*, Automation and Remote Control, 25.
17. MP. Verzi, H. Shin H, HH. He, R. Sulahian, CA. Meyer, RK. Montgomery, JC. Fleet, Brown M, Liu XS, and Shivdasani RA, (2010), *“Differentiation-Specific Histone Modifications Reveal Dynamic Chromatin Interactions and Partners for the Intestinal Transcription Factor CDX2”*, Developmental Cell, Vol 19, pp: 713–726.
18. ST. Whiteside, S. Goodbourn, (1993), *“Signal transduction and nuclear targeting: regulation of transcription factor activity by subcellular localization”*, Journal of Cell Science 104 (4), pp: 949–55.
19. JO. Yanez-Cuna, HQ. Dinh, EZ. Kvon, (2012), *“Uncovering cis-regulatory sequence requirements for context specific transcription factor binding”*, Genome research, Vol 22, pp: 2018-2030.
20. Y. Zhang, T. Liu, CA. Meyer, J. Eeckhoute, DS. Johnson, BE. Bernstein, C. Nussbaum, RM. Myers, M. Brown, W. Li, (2008), *“Model-based analysis of ChIP-Seq (MACS)”*. Genome Biol 9, pp: R137.
21. M. Zhong, W. Niu, ZJ. Lu, M. Sarov, JI. Murray, J. Janette, D. Raha, KL. Sheaffer, HYK. Lam, E. Preston, (2010), *“Genome-wide identification of binding sites defines distinct functions for Caenorhabditis elegans PHA-4/FOXA in development and environmental response”*. PLoS Genet 6: e1000848.