# NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM

**(Của người hướng dẫn)**

**Điểm:** (**bằng chữ:** )

**Đồng ý/Không đồng ý** cho sinh viên bảo vệ trước hội đồng chấm đồ án tốt nghiệm?

Hà Nội, ngày … tháng 12 năm 2019

CÁN BỘ - GIẢNG VIÊN HƯỚNG DẪN

*(Ký, ghi rõ họ tên)*

# LỜI CẢM ƠN

Lời đầu tiên, ein xin gửi lời cảm ơn chân thành và sâu sắc tới tất cả các thầy cô trong khoa Công nghệ thông tin 1 nói chung và các thầy trong Bộ môn An toàn thông tin nói riêng tại Học viện Công nghệ Bưu chính Viễn Thông. Với sự dạy dỗ, quan tâm, chỉ bảo chu đáo, truyền đạt đầy đủ các kiến thức và kinh nghiệm quý báu của các thầy cô đã giúp em hoàn thành đồ án này và có đầy đủ hành trang vững bước trong tương lai.

Đặc biệt, em xin gửi lời cảm ơn chân thành nhất tới giảng viên, PGS.TS Phạm Văn Cường, người đã quan tâm, tận tình hướng dẫn, chỉ bảo em trong suốt quá trình nghiên cứu và thực hiện đồ án. Những lời khuyên và chỉ dẫn của thầy đã giúp em có thêm nhiều kiến thức và kinh nghiệm không chỉ trong việc thực hiện đồ án mà còn úng dụng rất nhiều vào lĩnh việc nghiên cứu khoa học.

Em xin gửi lời cảm ơn đến anh Nguyễn Thái Sơn, Học viện Công nghệ Bưu chính Viễn thông đã giúp đỡ, chia sẻ những kinh nghiệm quý báu để giúp em hoàn thành đồ án này.

Mình xin gửi lời cảm ơn đến các bạn trong team Pacific và Savage\_AF đã cùng mình học tập và tham gia các cuộc thi về Tin học, một trong những trải nghiệm đáng nhớ của thời sinh viên.

Cuối cùng, em xin cảm ơn tới gia đình, ban bè, anh em trong Câu lạc bộ ITPTIT, đã luôn ở bên quan tâm, chia sẻ và là chỗ dựa tinh thần đã giúp em rất nhiều trong quá trình học tập và thực hiện đồ án.

Hà Nội, ngày 12 tháng 12 năm 2019

Tác giả

*Phạm Trung Thành*

# MỤC LỤC

[NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM i](#_Toc26740566)

[LỜI CẢM ƠN ii](#_Toc26740567)

[MỤC LỤC iii](#_Toc26740568)

[DANH SÁCH BẢNG v](#_Toc26740569)

[DANH SÁCH HÌNH VẼ vi](#_Toc26740570)

[DANH MỤC TỪ VIẾT TẮT vii](#_Toc26740571)

[MỞ ĐẦU 1](#_Toc26740572)

[Chương 1: KHÁI QUÁT VỀ BÀI TOÁN PHÁT HIỆN TIẾNG NGÁY 2](#_Toc26740573)

[1.1 Khát quát về âm thanh 3](#_Toc26740574)

[1.2 Bài toán về phát hiện tiếng ngáy 3](#_Toc26740575)

[1.3 Tổng kết chương 1 4](#_Toc26740576)

[Chương 2: CÁC ĐẶC TRƯNG CỦA ÂM THANH 5](#_Toc26740577)

[2.1 Phổ âm thanh 6](#_Toc26740578)

[2.2 Mel Frequency Cestral Coefficient 8](#_Toc26740579)

[**a)** **Chi tiết các bước trích chọn MFCC:** 9](#_Toc26740580)

[**b)** **Tính toán bộ lọc Mel** 10](#_Toc26740581)

[2.3 Gammatone Frequency Cepstral Coefficient 13](#_Toc26740582)

[2.4 Linear Predictive Code 15](#_Toc26740583)

[2.5 Một số đặc trưng khác 17](#_Toc26740584)

[**a)** **Perceptual Linear Prediction** 17](#_Toc26740585)

[**b)** **Linear Prediction Cepstral Coefficient** 18](#_Toc26740586)

[2.6 Tổng kết chương 2 19](#_Toc26740587)

[Chương 3: MÔ HÌNH HỌC MÁY SỬ DỤNG 20](#_Toc26740588)

[3.1 Giới thiệu về học máy 21](#_Toc26740589)

[3.2 Decision Tree 24](#_Toc26740590)

[3.3 Random Forest 34](#_Toc26740591)

[3.4 AdaBoost 35](#_Toc26740592)

[3.5 XGBoost 37](#_Toc26740593)

[3.6 Tổng kết chương 3 38](#_Toc26740594)

[Chương 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ 39](#_Toc26740595)

[4.1 Thu thập và gán nhãn dữ liệu 40](#_Toc26740596)

[**a)** **Thu thập dữ liệu** 40](#_Toc26740597)

[**b)** **Chuẩn lưu trữ âm thanh** 40](#_Toc26740598)

[**c)** **Gán nhãn dữ liệu** 41](#_Toc26740599)

[4.2 Xử lý dữ liệu 42](#_Toc26740600)

[4.3 Phương pháp đánh giá và so sánh 42](#_Toc26740601)

[4.4 Kết quả thực nghiệm 43](#_Toc26740602)

[4.5 Tổng kết chương 4 43](#_Toc26740603)

[TỔNG KẾT 44](#_Toc26740604)

[TÀI LIỆU THAM KHẢO 45](#_Toc26740605)

# DANH SÁCH BẢNG

[Bảng 3.1: Bảng dữ liệu về kết quả của việc chơi hay không dựa trên thời tiết 29](#_Toc26740103)

[Bảng 3.2: Bảng dữ liệu của việc chơi hay không khi outlook là sunny 30](#_Toc26740104)

[Bảng 3.3: Bảng dữ liệu của việc chơi hay không khi outlook là overcast 30](#_Toc26740105)

[Bảng 3.4: Bảng dữ liệu của việc chơi hay không khi outlook là rainy 30](#_Toc26740106)

[Bảng 3.5: Bảng dữ liệu với temperature là hot 31](#_Toc26740107)

[Bảng 3.6: Bảng dữ liệu với temperature là mild 31](#_Toc26740108)

[Bảng 3.7: Bảng dữ liệu với temperature là cool 31](#_Toc26740109)

[Bảng 4.1: Bảng thông kê số lượng nhãn dữ liệu 43](#_Toc26740110)

[Bảng 4.2: Bảng so sánh kết quả giữa các đặc trưng và mô hình học máy 44](#_Toc26740111)

# DANH SÁCH HÌNH VẼ

[Hình 1.1: Phân loại các loại âm tần 4](file:////Users/sonnt/Desktop/thesis_phamtrungthanh_v2.docx#_Toc26740129)

[Hình 2.1: Mô tả quá trình biến đổi 1 đoạn âm thanh sang phổ âm thanh 7](#_Toc26740130)

[Hình 2.2: Hình ảnh quá trình biến đổi đầy đủ phổ âm thanh 7](file:////Users/sonnt/Desktop/thesis_phamtrungthanh_v2.docx#_Toc26740131)

[Hình 2.3: Hình ảnh phổ trên thực tế sau khi biến đổi 8](#_Toc26740132)

[Hình 2.4: Hình ảnh về sự chuyển dịch âm thanh trong phổ âm thanh 8](#_Toc26740133)

[Hình 2.5: Sơ đồ các bước thực hiện biến đổi MFCC 9](#_Toc26740134)

[Hình 2.6: Hình ảnh mô tả phổ âm thanh sau khi đưa qua bộ lọc Mel 11](#_Toc26740135)

[Hình 2.7: Hình ảnh của 8 bộ lọc Mel được xây dựng 12](#_Toc26740136)

[Hình 2.8: Bộ lọc Gammatone được xác định bởi công thức 14](#_Toc26740137)

[Hình 2.9: Hình ảnh so sánh bộ lọc Mel và bộ lọc Bark 19](#_Toc26740138)

[Hình 3.1: Quá trình phát triển của trí tuệ nhân tạo 22](#_Toc26740139)

[Hình 3.2: Quá trình phát triển của thuật toán Boosting 24](#_Toc26740140)

[Hình 3.3: Ví dụ về cây quyết định cho việc học hoặc chơi 25](file:////Users/sonnt/Desktop/thesis_phamtrungthanh_v2.docx#_Toc26740141)

[Hình 3.4: Hình ảnh về phân loại 2 lớp xanh và đỏ 26](#_Toc26740142)

[Hình 3.5: Hình ảnh cây quyết định cho phân loại 2 lớp xanh và đỏ 26](#_Toc26740143)

[Hình 3.6: Cây quyết định được xây dựng dựa trên ID3 32](#_Toc26740144)

[Hình 3.7: Sơ đồ các bước thực hiện của Random Forest 35](#_Toc26740145)

[Hình 3.8: Sơ đồ các bước thực hiện của AdaBoost 36](#_Toc26740146)

[Hình 3.9: Sơ đồ các bước thực hiện của XGBoost 39](#_Toc26740147)

[Hình 4.1: Hình ảnh minh họa tín hiệu âm thanh 41](#_Toc26740148)

[Hình 4.2: Hình ảnh tín hiệu âm thanh hiển thị trong Audacity 42](#_Toc26740149)

[Hình 4.3: Số liệu gán nhãn của từng file âm thanh 43](#_Toc26740150)

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Từ viết tắt** | **Tiếng Anh** | **Tiếng Việt/Giải thích** |
| 1 | AI | Artificial Intelligence | Trí tuệ nhân tạo |
| 2 | MFCC | Mel Frequency Cepstral Coefficient | Hệ số đặc trưng Mel |
| 3 | LPC | Linear Predictive Coding | Đặc trưng mã hóa tuyến tính |
| 4 | GFCC | Gammatone Frequency Cepstral Coefficient | Hệ số đặc trưng Gammatone |
| 5 | PLP | Perceptual Linear Prediction | Đặc trưng mã hóa tuyến tính dữa trên giác quan |
| 6 | LPCC | Linear Prediction Cepstral Coefficient | Hệ số đặc trưng của mã hóa tuyến tính |
| 7 | AdaBoost | Adaptive Boosting | Mô hình học máy Adaptive Boosting |
| 8 | XGBoost | Extreme Grandient Boosting | Mô hình học máy Extreme Gradient Boosting |
| 9 | FFT | Fast Fourier transform | Biến đổi Fourier nhanh |
| 10 | DCT | Discrete cosine transform | Biến đổi cosin rời rạc |

# MỞ ĐẦU

Cuộc sống của con người càng ngày càng phát triển, kèm theo đó chất lượng cuộc sống càng ngày càng được nâng cao. Ngáy là môt hiện tượng khá phổ biến xuất hiện khi con người đi vào giấc ngủ. Ngáy không những tiềm tàng nhiều nguy cơ về bệnh lý mà còn khiến cho cuộc sống của bản thân và người xunh quanh bị ảnh hưởng. Vì vậy việc phát hiện tiếng ngáy được phát triển và càng ngày càng được nghiên cứu sâu hơn để phục vụ vào lĩnh vực y tế.

Hiện nay các phương pháp phát hiện tiếng ngáy có thể sử dụng đều sử dụng các mô hình học máy và học sâu để thực hiện nghiên cứu và phát triển. Trong đồ án này em sẽ tập trung vào nghiên cứu các đặc trưng âm thanh được sử dụng trong việc nhận diện âm thanh và các mô hình học máy cơ bản để hỗ trợ quá trình huấn luyện dữ liệu.

Vì vậy trong đồ án em sẽ tập trung trình bày các nội dung sau:

* **Chương 1: Khát quát về bài toán phát hiện tiếng ngáy**

Nội dung chương 1 sẽ tập trung vào việc nếu ra khát niệm về âm thanh và bước đầu xác định bài toán phát hiện tiếng ngáy

* **Chương 2: Các đặc trưng âm thanh**

Nội dung chương 2 sẽ tập trung vào việc đưa ra các đặc trưng âm thanh và các bước tiến hành trích chọn đặc trưng âm thanh từ dữ liệu âm thanh ban đầu.

* **Chương 3: Các mô hình học máy được sử dụng**

Nội dung chương 3 sẽ trình bày khat quát về học máy, sự phát triển của học máy, đưa ra cái nhìn tổng quan về 4 mô hình học máy trong đó có Cây quyết định được xây dựng như là một thành phần con trong 3 mô hình còn lại là Random Forest, AdaBoost, XGBoost.

* **Chương 4: Thực nghiệm và đánh giá**

Nội dung chương 4 đưa ra cách thức thu thập dữ liệu và kết quả thực nghiệm bộ dữ liệu dựa trên các đặc trưng âm thanh và mô hình học máy đã đề suất

* **Chương 5: Tổng kết**

Nội dung chương 5 tổng kết lại kết quả đã đạt được và chưa đạt được. Từ đó sẽ đưa ra hướng phát triển, nghiên cứu tiếp theo cho đồ án.

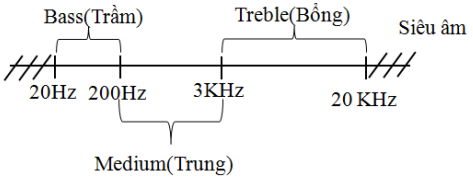
# KHÁI QUÁT VỀ BÀI TOÁN PHÁT HIỆN TIẾNG NGÁY

Trong chương 1, đồ án sẽ trình bày về khát quát về âm thanh và bài toán phát hiện tiếng ngáy, nói rõ về tầm ảnh hưởng của tiếng ngáy trong sức khỏe và đời sống hoạt động bình thường của con người, bao gồm các phần sau:

* + **Khát quát về âm thanh**
  + **Bài toán về phát hiện tiếng ngáy**
  1. **Khát quát về âm thanh**

**Âm thanh** [1] là các dao động cơ học (biến đổi vị trí qua lại) của các phân tử, nguyên tử hay các hạt làm nên vật chất và lan truyền trong vật chất như các sóng. Âm thanh, giống như nhiều sóng, được đặc trưng bởi **tần số**, **bước sóng**, **chu kỳ**, **biên độ** và **vận tốc lan truyền** (tốc độ âm thanh).

**Âm tần** [2] được định nghĩa là khoảng tần số âm thanh mà tai người cảm nhận được 20 Hz 20 KHz. Phổ âm tần được chia làm 3 vùng Bass, Medium và Trebel:



Hình 1.1: Phân loại các loại âm tần

* 1. **Bài toán về phát hiện tiếng ngáy**

Ngáy [3] là những âm thanh phát ra khi các mô đường hô hấp trên rung lên khi ngủ. Ngáy hay xuất hiện trong thì hít vào nhưng cũng có thể xảy ra trong thì thở ra. Ngáy là hành vi phổ biến, xuất hiện ở 44% nam giới và 28% nữ giới trong độ tuổi 30 – 60. Khi hít thở trong lúc ngủ, do lượng khí vào đi qua vùng họng hẹp phía sau, khiến các niêm mạc mô xung quanh rung lên, tạo ra tiếng ngáy.

Trong hầu hết các trường hợp gây cản trở, khiến cho không khí giữa thanh quản và mũi lưu thông không được bình thường đều gây ra hiện tượng ngủ ngáy. Ngủ ngáy nguyên nhân còn là do mắc các bệnh như dị ứng, nghẹt mũi, amidan quá lớn , ..  Nguyên nhân ngủ ngáy cũng có thể là do các dị tật bẩm sinh gây ra như hẹp cổ họng, cuống lưỡi lớn, cuống họng quá dài, chân lưỡi dày, ...

Ngoài việc gây khó chịu đối với người ngủ cùng thì bản thân người mắc bệnh ngủ ngáy cũng có thể cảnh báo một số nguy cơ sức khỏe.

Trẻ em bị ngủ ngáy thường khó đi vào giấc ngủ, do đó làm giảm chất lượng giấc ngủ, như ngủ không say, không sâu, ảnh hưởng đến sự phát triển cả về thể chất lẫn trí tuệ, do não bị thiếu oxy khi ngủ. **Ngủ ngáy ở trẻ em** còn có thể gây ra tình trạng ngưng thở lúc ngủ. Trẻ ngủ ngáy cũng sẽ ảnh hưởng đến hình dáng khuôn mặt do phải miệng thường há ra để hít thở khi ngủ.

Đối với người lớn, khi mắc bệnh ngủ ngáy, ngưng thở trong lúc ngủ, do các phần mềm, cũng như niêm mạc cuống họng làm khí quản bị nghẹt, sẽ dẫn đến phổi và não bị thiếu dưỡng khí. Để quá trình hô hấp diễn ra lại như bình thường, não sẽ phát tín hiệu để làm giãn nở cuống họng, khí quản. Một người mắc bệnh ngủ ngáy tức là gặp những rối loạn như này, sẽ có nguy cơ ngưng thở khi ngủ cao hơn, ảnh hưởng đến chất lượng giấc ngủ, khiến người bệnh mệt mỏi do não bộ không được nghỉ ngơi hoàn toàn. Ở mức độ nghiêm trọng, ngủ ngáy còn có thể làm xáo trộn sóng điện não, làm giảm trí nhớ, năng suất làm việc, khả năng tập trung, tinh thần mệt mỏi ...

Bên cạnh đó, người mắc bệnh ngủ ngáy cũng có nguy cơ bị các bệnh khác như cao huyết áp, nhồi máu cơ tim, loạn nhịp tim, đột quỵ trong lúc ngủ.

Chính vì thế phát hiện tiếng ngáy khá quan trọng trong việc cải thiện chất lượng cuộc sống của con người. Mục đích của nghiên cứu này là đưa ra được một phương pháp tốt để phát hiện tiếng ngáy và có thể phát triển để tích hợp vào các thiết bị di động.

* 1. **Tổng kết chương 1**

Trong chương 1, đồ án đã trình bày được về âm thanh là gì, khát quát về âm thanh,đã đưa ra được các vấn đề về tầm quan trọng trong việc phát hiện tiếng ngáy.

Trong chương tiếp theo, đồ án sẽ đề cập tới các đặc trưng và các bước trích chọn các dặc trưng của âm thanh.

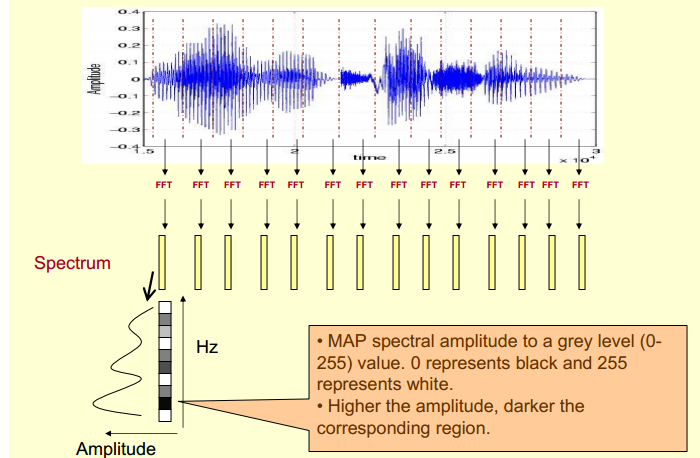
# CÁC ĐẶC TRƯNG CỦA ÂM THANH

Trong chương 2, đồ án sẽ trình bày chi tiết về các bước trích chọn đặc trưng trong âm thanh, bào gồm 3 đặc trưng chính là MFCC, LPC, GFCC và 2 đặc trưng phụ là PLP, LPCC thông qua các phần sau:

* **Phổ âm thanh**
* **Mel Frequency Cestral Coefficient**
* **Linear Predictive Code**
* **Gammatone Frequency Cepstral Coefficient**
* **Một số đặc trung khác**
  1. **Phổ âm thanh**

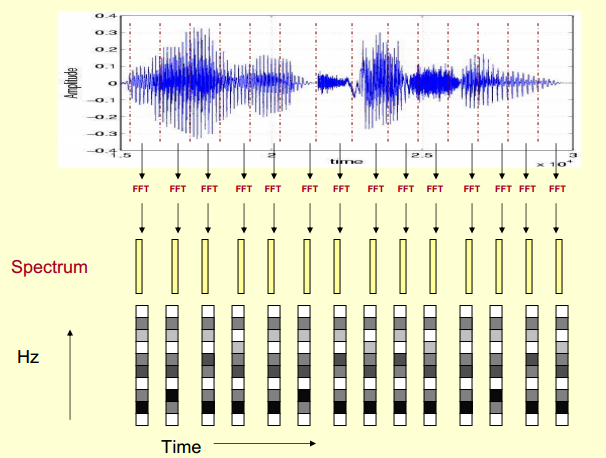
Phổ [4] là biểu diễn âm thanh theo tần số và thời gian.

Cách biến đổi tín hiệu âm thanh từ **biên độ - thời gian** thành dạng **tần số - biên độ**



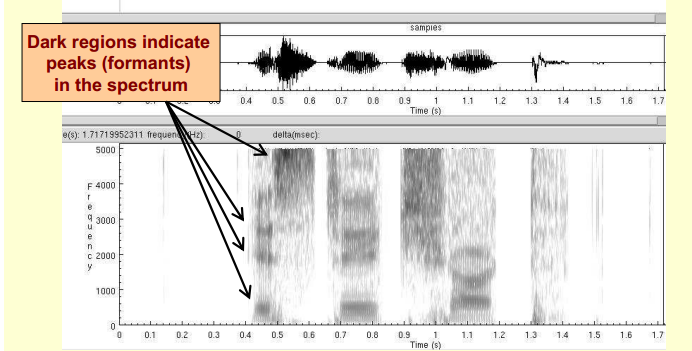
Hình 2.1: Mô tả quá trình biến đổi 1 đoạn âm thanh sang phổ âm thanh

Từ hình 2.1, có thể thấy từ dữ liệu âm thanh ban đầ ta thực hiện biến đổi FFT [5] trên từng windows để thu được phổ (biên độ- tần số), cường độ phổ được tính trong khoảng từ 0-255, cường độ càng cao thì thang màu càng đậm. Phép biến đổi FFT sẽ được đề cập chi tiết trong chương 2 của đồ án.

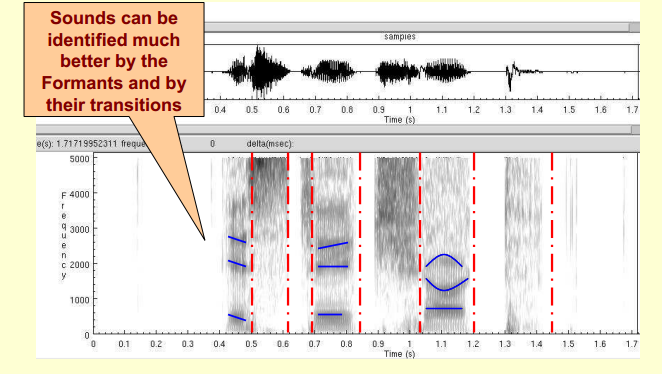


Hình 2.2: Hình ảnh quá trình biến đổi đầy đủ phổ âm thanh

Thực hiện các phép biến đổi FFT đến hết đoạn tín hiệu âm thanh ta thu được hình ảnh của phổ như sau:



Hình 2.3: Hình ảnh phổ trên thực tế sau khi biến đổi



Hình 2.4: Hình ảnh về sự chuyển dịch âm thanh trong phổ âm thanh

Âm thanh có thể được định danh bằng hình dạng và sự chuyển dịch của nó, nhìn vào hình 2.4 có thể dễ dàng nhìn thấy được sự khác nhau của các đoạn phổ âm thanh dựa vào cường độ của phổ.

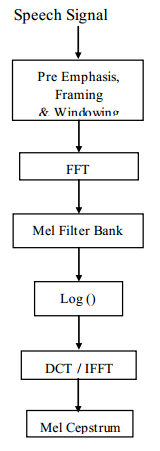
* 1. **Mel Frequency Cestral Coefficient**

Phương pháp phổ biến và phổ biến nhất được sử dụng để trích xuất các tính năng phổ là tính toán các hệ số cepstral Mel-Tần số (MFCC). MFCC là một trong những kỹ thuật trích xuất tính năng phổ biến nhất được sử dụng trong nhận dạng giọng nói dựa trên miền tần số sử dụng thang đo Mel dựa trên thang đo tai người.

MFCC được coi là tính năng miền tần số chính xác hơn nhiều so với tính năng miền thời gian.

Các hệ số cepstral tần số Mel (MFCC) [6] là một đại diện của cepstral thực của tín hiệu thời gian ngắn có cửa sổ xuất phát từ Biến đổi Fourier nhanh (FFT) của tín hiệu đó. Sự khác biệt so với thực tế

cepstral là thang đo tần số phi tuyến được sử dụng, gần đúng với hành vi của hệ thống thính giác. MFCC là một kỹ thuật trích xuất tính năng âm thanh, trích xuất các tham số từ lời nói tương tự như thông tin được con người sử dụng để nghe lời nói, đồng thời, làm mất đi tất cả các thông tin khác.



Hình 2.5: Sơ đồ các bước thực hiện biến đổi MFCC

Mel scale là miền tần số của âm thanh, đặc trưng cho sự cảm nhận âm thanh, âm lượng của 1 giai điệu. Tai người vô cùng nhạy cảm với những sự thay đổi nhỏ về tần số ở miền tần số thấp và độ nhạy cảm sẽ giảm dần khi miền tần số tăng cao. Đây cũng chính là ý tưởng chính để chúng ta rút trích đặc trưng MFCC, chỉ giữ lại những âm thanh nằm trong vùng tần số mà tai người nhạy cảm, để giảm độ lớn dữ liệu, qua đó giảm độ phức tạp tính toán.

Công thức chuyển đổi từ tần số sang Mel scale:

Và công thức biến đổi từ Mel scale về tần số:

1. **Chi tiết các bước trích chọn MFCC:**

Giả sử ta đang có 1 đoạn âm thanh và tốc độ mẫu của nó là 16kHz.

1. Chia tín hiệu đầu vào thành các khung có độ dài khoảng 20-40ms, tiêu chuẩn là 25ms. Giả sử tốc độ lấy mẫu là 16kHz thì 1 khung sẽ chứa 0.025 \* 16000 = 400 mẫu. Khi chia tách tín hiệu thì các khung sẽ đè lên nhau 10ms (160 mẫu), tức là 400 mẫu đầu tiên sẽ bắt đầu từ vị trí 0, 400 mẫu tiếp theo sẽ bắt đầu từ vị trí 160, cứ như thế cho tới khung cuối cùng.

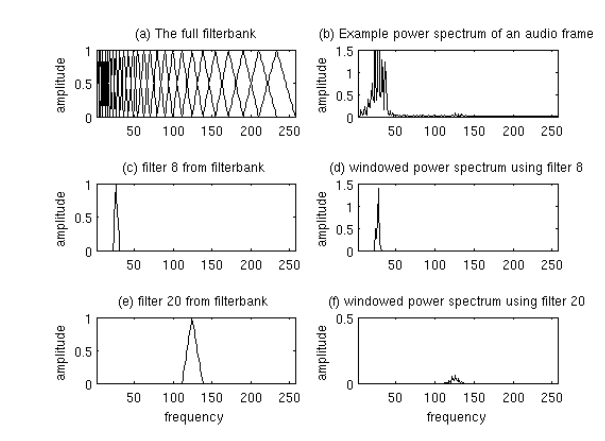
Tiếp theo với từng khung ta sẽ trính xuất ra 13 hệ số MFCC, ta gọi tín hiệu theo miền thời gian là **s(n)**, thì với mỗi khung ta sẽ có **si(n)** với n trong đoạn [1,400] (giả sử mỗi khung chúng ta có 400 mẫu) và i là số thứ tự của khung. Tiếp đến ta thực hiện biến đổi biến đổi DFT ta có **Si(k)** và **Pi(k)** là năng lượng phổ tại i.

1. Công thức biến đổi DFT như sau:

Trong đó **h(n)** là giá trị của cửa sổ hamming tại n và **K** là độ dài của DFT. Sau khi thu được **Si(k)** thì ta tính năng lượng phổ theo công thức sau:

Thông thường ta sẽ tạo ra 512 điểm FFT nhưng sẽ chỉ giữ lại 257 hệ số đầu tiên.

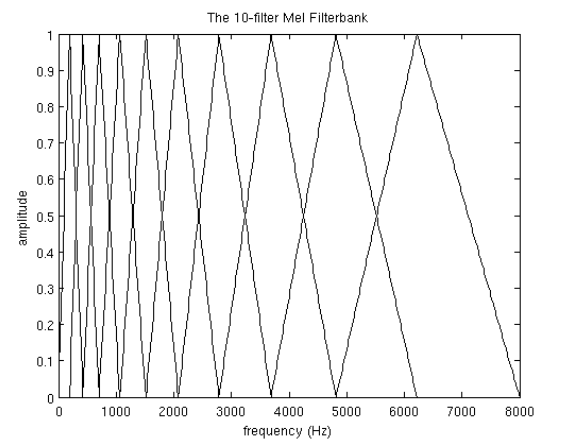
1. Thực hiện tính toán bộ lọc Mel. Đó là 1 tập hợp các bộ lọc hình tam giác, thường là 20-40 bộ lọc ( tiêu chuẩn là 26). Ta sẽ thực hiện lọc các giá trị năng lượng phổ thu được ở bước thứ 2, mỗi bộ lọc là 1 vector dài 257. Khi thực hiện lọc thì sẽ chỉ giữ lại các giá trị nằm trong bộ lọc. Hình ảnh mình họa cho việc sử dụng bộ lọc lên năng lượng phổ:



Hình 2.6: Hình ảnh mô tả phổ âm thanh sau khi đưa qua bộ lọc Mel

1. Lấy log của 26 vector thu được sau khi sử dụng 26 bộ lọc lên năng lượng phổ
2. Ở bước này ta sẽ thực hiện biến đổi DCT [7] để thu được 26 hệ số cestral và ta chỉ giữ lại 12-13 hệ số nhỏ nhất là đại diện.
3. **Tính toán bộ lọc Mel**

Trong phần này ta sẽ sử dụng 10 bộ lọc để làm ví dụ cụ thể, còn trong thực tế ta sẽ sử dụng 26-40 bộ lọc.



Hình 2.7: Hình ảnh của 8 bộ lọc Mel được xây dựng

Để tạo được bộ lọc như hình trên ta cần phải chọn giới hạn cho tần số. Như trong ví dụ này ta sẽ chọn tần số nằm trong đoạn [300, 8000]. Nếu tốc độ lấy mẫu là 8000 Hz thì khoảng giới hạn của bộ lọc cao nhất sẽ là 4000 Hz.

1. Sử dụng công thức biến đổi từ tần số sang hệ số Mel ta có 300Hz và 8000Hz lần lượt tưởng ứng với 401.25 Mels và 2834.99 Mels.
2. Ta cần 12 điểm để tạo ra 10 bộ lọc nên ta sẽ chọn ra 12 điểm Mel trong khoảng 401.25 đến 2834.99
3. Sau đó ta dung công thức biến đổi từ hệ số Mel về tần số ta có:
4. Ta sử dụng công thức sau để biến đổi tần số về gần với hệ số FFT:

Ghi chú:

* *Trong trường hợp này nfft =512 và samplerate = 16000*

Khi đó ta thu được dãy như sau:

1. Bây giờ ta bắt đầu xây dựng bộ lọc, mỗi bộ lọc sẽ bắt đầu từ điểm thứ nhất, đỉnh ở điểm thứ 2 vào kết thúc ở điểm thứ 3. Từ đó ta xây dựng được công thức sau cho bộ lọc:

Với m tương ứng với số bộ lọc k nằm trong [0,256]

Với từng bộ lọc ta sẽ thực hiện lọc bằng cách nhân vô hướng vector thu được từ mỗi bộ lọc với vector năng lượng phổ.

Để tăng hiệu quả của MFCC ta sử dụng đạo hàm bậc 1 và đạo hàm bậc 2 để tăng số lượng hệ số MFCC thư được của mỗi khung từ 13 lên 39.

Công thức tính:

Trong đó:

* t là số thú tự khung, thông thường N sẽ dc lấy bằng 2.
  1. **Gammatone Frequency Cepstral Coefficient**

Bộ lọc gammatone [8] được xác định hoàn toàn trong miền thời gian. Cụ thể, các bộ lọc được áp dụng trực tiếp vào chuỗi thời gian của tín hiệu giọng nói bằng các thao tác đơn giản như độ trễ, tổng và nhân. Điều này khá khác biệt so với thiết kế miền tần số được áp dụng rộng rãi, trong đó các tín hiệu được chuyển thành phổ tần số trước tiên và các bộ ghi gamma sau đó được áp dụng cho chúng. Việc thực hiện miền thời gian tránh sự gần đúng không cần thiết được đưa ra bằng phân tích quang phổ thời gian ngắn và tiết kiệm một tỷ lệ đáng kể tính toán liên quan đến FFT.

A screenshot of a cell phone

Description automatically generated

Hình 2.8: Bộ lọc Gammatone được xác định bởi công thức

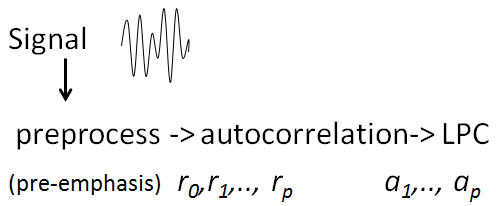
Trong đó *varphi* là pha (thường là *varphi* = 0), *n* là thứ tự của bộ lọc thường được đặt thành giá trị nhỏ hơn 4, *alpha* là hằng số (thường là *α* = 1) kiểm soát mức tăng và *β* = 1*.*019*ERB*( *fc*) là hệ số suy giảm của bộ lọc, có liên quan đến dải của bộ lọc và là yếu tố quyết định của tốc độ phân rã đáp ứng xung. ERB là một biện pháp âm thanh đo, cung cấp một xấp xỉ với băng thông của các bộ lọc trong thính giác của con người, được mô tả qua công thức:

Do chi tiết các bước thực hiện của GFCC khá giống các bước biến đổi MFCC nên ta sẽ không đi sâu vào GFCC nữa.

* 1. **Linear Predictive Code**

LPC [9] nén tín hiệu để truyền và lưu trữ hiệu quả. Tín hiệu số được nén trước khi truyền để sử dụng hiệu quả các kênh trên phương tiện không dây. Đối với bộ mã hóa tốc độ bit trung bình hoặc thấp, LPC được sử dụng rộng rãi nhất. LPC tính toán phổ công suất của tín hiệu. Nó được sử dụng để phân tích định dạng. LPC là một trong những kỹ thuật phân tích giọng nói mạnh mẽ nhất và nó đã trở nên phổ biến như một kỹ thuật ước lượng định dạng.

Quy trình tách LPC:



Bước đầu tiên ta lọc nhiễu (pre-emphasis) bằng công thức sau:

Trong đó:

* S là tín hiệu đầu vào.
* Thường hệ số a thường được chọn là 0.95

Ý tưởng của LPC là với mỗi khung ta thực hiện tính toán để thu được 8 hệ số LPC a1, …, a8 thỏa mãn các điểm kiên sau:

và

Từ đó ta có:

Với:

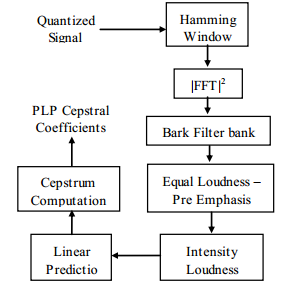
Ta hoàn toàn có thể tính đượ các giá trị r0, …, rp để tính ra a1, …, ap.

Trong đồ án ta sẽ thực hiện trích xuất LPC với p = 11 để thu được 11 hệ số LPC tương ứng với mỗi khung.

* 1. **Một số đặc trưng khác**

1. **Perceptual Linear Prediction**

Quy trình các bước của PLP

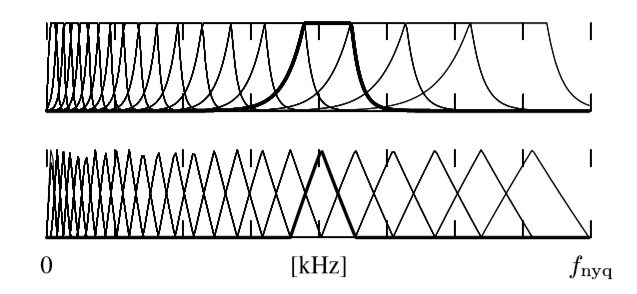


Có thể thấy rằng PLP [9] là 1 đặc trưng kết hợp từ MFCC và LPCC. Điều cần chú ý ở đây là PLP sử dụng Bark Filter Bank để lọc năng lượng phổ.

Ta có công thức biến đổi từ tần số sang Bark Scale:

Và công thức biến đổi từ Bark scale về tần số là:

Hình ảnh mình họa về sự khác nhau giữa Bark Filter và Mel Filter:

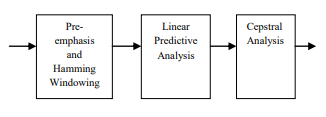


Hình 2.9: Hình ảnh so sánh bộ lọc Mel và bộ lọc Bark

Trong thực tế thì hiệu quả của 2 bộ lọc đều không khác nhau đáng kể, giữa chúng chỉ khác nhau ở hình dạng và độ rộng của Bark filter lớn hơn Mel filter.

1. **Linear Prediction Cepstral Coefficient**

Quy trình các bước của LPCC:



LPCC [10] là 1 kĩ thuật sử dụng LP (linear prediction) và được phát triển từ LPC, nhìn vào sơ đồ trên có thể thấy rằng LPCC kế thừa gần như toàn bộ các phương pháp biến đổi của LPC.

Sau khi thu được các tham số LPC là ai thì ta thực hiện biến đổi để thu được các tham số của LPCC là ci được xác định bới công thức truy hồi sau:

* 1. **Tổng kết chương 2**

Trong chương 2, đồ án đã trình bày chi tiết về phổ âm thanh và các đặc trưng âm thanh MFCC, LPC và GFCC, đồng thời giới thiệu khát quát về 2 đặc trưng PLP và LPCC.

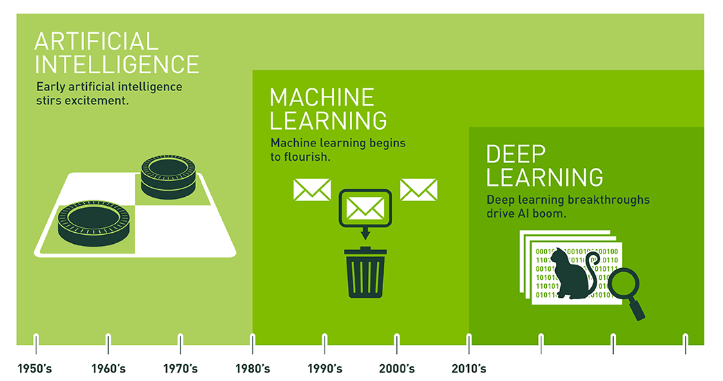
Trong chương tiếp theo, đồ án sẽ đề cập tới các mô hình học máy sử dụng đầu vào là các đặc trưng âm thanh để thực hiện huấn luyện.

# MÔ HÌNH HỌC MÁY SỬ DỤNG

Trong chương 3, đồ án sẽ đề cập khát quát về học máy và đề cập đến 3 mô hình học máy chính là Random Forest, AdaBoost, XGBoost thông qua các phần sau:

* **Giới thiệu về học máy**
* **Cây quyết định**
* **Random Forest**
* **AdaBoost**
* **XGBoost**
  1. **Giới thiệu về học máy**

**AI - Trí tuệ nhân tạo** (Artificial Intelligence) [11] là các kỹ thuật giúp cho máy tính thực hiện được những công việc của con người chúng ta. Ví dụ như một chương trình cờ vua tự động có thể coi là một chương trình có sử dụng AI hay viết tắt là một chương trình AI.



Hình 3.1: Quá trình phát triển của trí tuệ nhân tạo

Trong lĩnh vực AI có một nhánh nghiên cứu về khả năng tự học của máy tính được gọi là **học máy** (machine learning). Hiện nay không có 1 định nghĩa chính thức nào về học máy cả nhưng có thể hiểu rằng nó là các kỹ thuật giúp cho máy tính có thể tự học mà không cần phải cài đặt các luật quyết định. Thường một chương trình máy tính cần các quy tắc, luật lệ để có thể thực thi được một tác vụ nào đó như dán nhãn cho các email là thư rác nếu nội dung email có chứ từ khoá “quảng cáo”. Nhưng với học máy, các máy tính có thể tự động phân lại các thư rác thành mà không cần chỉ trước bất kỳ quy tắc nào cả. Nói một cách dễ hiểu, đơn giản là nó giúp cho máy tính có được cảm quan và suy nghĩ được như con người. Còn nếu nói theo cách kỹ thuật một chút thì học máy là phương pháp vẽ các đường thể hiện mối quan hệ của tập dữ liệu. Ví dụ như đường ngăn cách 2 loại dữ liệu cho nhãn khác nhau, đường thể hiện xu hướng của giá nhà phụ thuộc vào diện tích và trí hay các đường phân cụm dữ liệu.

Một nhánh nhỏ trong học máy gần đây rất được ưu chuộng là **học sâu** (deep learning). Học sâu là kỹ thuật sử dụng các mạng nơ-ron tương tự như các nơ-ron của não người để xây dựng hệ thống học máy. Đây là một sự kết hợp tuyệt vời giữa toán học và khoa học thần kinh. Kết quả của nó mang lại cực kỳ to lớn, có thể coi là khởi nguyên của ngành công nghiệp mới. Tại thời điểm này, hầu hết các anh lớn cả trong ngành công nghệ lẫn các ngành khác như ô tô, điện tử đều đang tập trung phát triển và ứng dụng kỹ thuật học sâu cho bài toán của mình. Ví dụ như AlphaGo của Google đã chiến thắng nhà vô địch cờ vây Lee Sedol vào tháng 3 năm 2016. Tính năng nhận diện khuôn mặt khá chính xác của Facebook được triển khai vào năm 2016. Trợ lý ảo Siri của Apple được giới thiệu từ năm 2006. Xe tự lái của Google được thử nghiệm chính thức trên đường phố vào năm 2015…

Để phân loại học máy, ta phải xem cách mà máy "học" như thế nào. Có 4 cách chính: học có giám sát, học không giám sát, học bán giám sát và học củng cố.

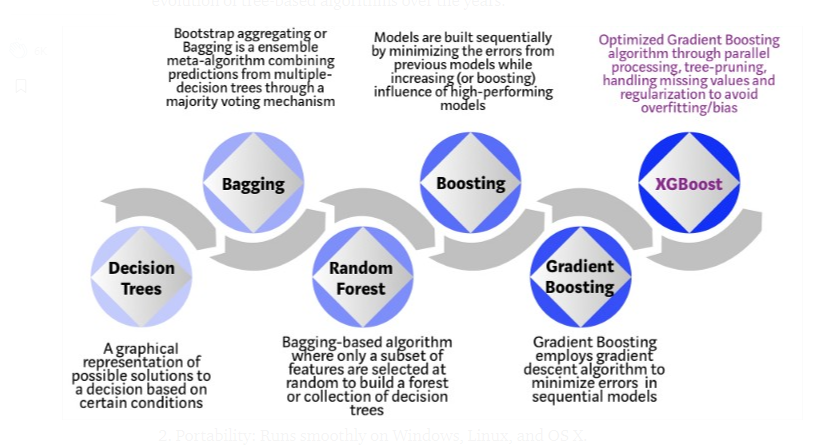
* **Học có giám sát** (supervised learning) cần có các dữ liệu dùng để "học", và phải đưa ra được một kết quả nhất định (như thuật toán lọc spam ở trên).
* **Học không giám sát** (unsupervised learning) cũng cần có các dữ liệu dùng để "học", nhưng không phải đưa ra các kết quả cụ thể nào.
* **Học bán giám sát** (semi-supervised learning) cũng cần có các dữ liệu dùng để "học", và có thể đưa ra được một vài kết quả khác nhau.
* **Học củng cố** (reinforcement learning) là một cách "học" đang trở nên vô cùng phổ biến hiện nay. Trong học củng cố, AI sẽ được thưởng dựa trên những gì nó làm được. Học củng cố có thể giúp máy móc đạt được những khả năng như hiểu được làm sao để chơi một game máy tính thông qua một quá trình thử và lỗi (trial-and-error), dựa trên việc nghiên cứu làm cách nào để tăng điểm số của mình.

**Học đồng bộ** [12] là một mô hình đưa ra dự đoán dựa trên một số mô hình khác nhau, bằng cách kết hợp các mô hình riêng lẻ, mô hình tập hợp có xu hướng linh hoạt hơn (ít sai lệch) và ít nhạy cảm dữ liệu (ít phương sai hơn).

Hai phương pháp đồng bộ phổ biến là **Bagging** và **Boosting**.

* **Bagging**: Các mô hình được huấn luyện một cách song song, mỗi mô hình huấn luyện 1 con ngẫu nhiên dữ liệu từ tập dữ liệu ban đầu.
* **Boosting**: Các mô hình được huấn luyện một cách tuần tự, mỗi mô hình các nhân học từ những sai lầm của mô hình trước đó

Sơ đồ quá trình phát triển của học đồng bộ:



Hình 3.2: Quá trình phát triển của thuật toán Boosting

Hiện nay để phân loại các âm thanh, đặc biệt là tiếng ngáy thì ta có thể sử dụng rất nhiều các mô hình học máy như Support Vector Machine, K-Means, K-nearest neighbor,… Nhưng để tiếp cận một cách đơn giản với bài toán phát hiện tiếng ngáy cũng như bài toán về binary classification thì đồ án sẽ sử dụng đến các mô hình học đồng bộ và trong đó sử dụng cây quyết định làm mô hình chính.

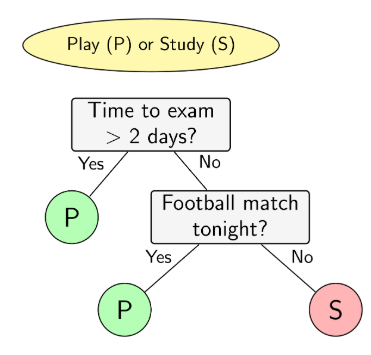
Trong các phần tiếp theo của chương sẽ đi sâu vào tìm hiểu cây quyết định, phân biệt và tìm hiểu về sự phát triển của 3 mô hình học đồng bộ là Random Forest, Adaptive Boosting, XGBoosting.

* 1. **Decision Tree**

**Decision Tree** hay **Cây quyết định** [13] là một công cụ hỗ trợ quyết định sử dụng đồ thị hoặc mô hình quyết định giống như cây.

Các thuật toán học tập dựa trên cây được coi là một trong những phương pháp học có giám sát tốt nhất và được sử dụng nhiều nhất. Các phương pháp dựa trên cây trao quyền cho các mô hình dự đoán với độ chính xác cao, ổn định và dễ giải thích. Không giống như các mô hình tuyến tính, chúng ánh xạ các mối quan hệ phi tuyến tính khá tốt. Họ có thể thích nghi trong việc giải quyết bất kỳ loại vấn đề nào (phân loại hoặc hồi quy). Các thuật toán Cây quyết định được gọi là CART (Classification and Regression Trees).

Mô hình cây quyết định được thể hiện thông qua ví dụ sau:



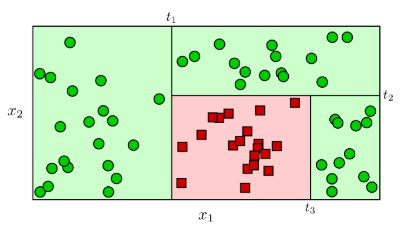
Hình 3.3: Ví dụ về cây quyết định cho việc học hoặc chơi

Giả sử có một cây quyết định xem ta sẽ chơi (P) hay học (S).

Đầu tiên xét xem nếu thời gian đến ngày thi lớn hơn 2 ngày hay không, nếu có thì sẽ chơi, ngược lại thì sẽ xét tiếp nếu tối nay có bóng đá thì sẽ chơi còn nếu không thì sẽ học để ôn thi

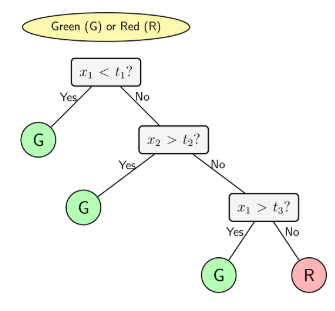
Có thể thấy rằng quyết định của con người cũng dựa trên việc trả lời các câu hỏi nên trong học máy việc đưa ra 1 kết quả - quyết định cũng dựa trên các câu hỏi.

Tiếp đến ta xét đến 1 ví dụ về việc phân loại 2 class màu đỏ và màu xanh



Hình 3.4: Hình ảnh về phân loại 2 lớp xanh và đỏ

Quan sát vào hình trên ta thấy ranh giới để phân loại của 2 class đỏ và xanh khá đơn giản, từ đó ta có thể xây dựng được 1 cây quyết định như sau:



Hình 3.5: Hình ảnh cây quyết định cho phân loại 2 lớp xanh và đỏ

Trong decision tree, các ô màu xám, lục, đỏ trên Hình 2 được gọi là các node. Các node thể hiện đầu ra (màu lục và đỏ) được gọi là node lá (leaf node hoặc terminal node). Các node thể hiện câu hỏi là các non-leaf node. Non-leaf node trên cùng (câu hỏi đầu tiên) được gọi là node gốc (root node). Các non-leaf node thường có hai hoặc nhiều node con (child node). Các child node này có thể là một leaf node hoặc một non-leaf node khác. Các child node có cùng bố mẹ được gọi là sibling node. Nếu tất cả các non-leaf node chỉ có hai child node, ta nói rằng đó là một binary decision tree (cây quyết định nhị phân). Các câu hỏi trong binary decision tree đều có thể đưa được về dạng câu hỏi đúng hay sai. Các decision tree mà một leaf node có nhiều child node cũng có thể được đưa về dạng một binary decision tree. Điều này có thể đạt được vì hầu hết các câu hỏi đều có thể được đưa về dạng câu hỏi đúng sai.

Vậy các *câu hỏi* nên được xây dựng như thế nào, và thứ tự của chúng ra sao. Các câu hỏi này thường được áp dụng lên từng thuộc tính, hoặc một tổ hợp tuyến tính của các thuộc tính. Cách thứ nhất, áp dụng lên từng thuộc tính, được sử dụng nhiều hơn vì tính đơn giản của nó. Với các thuộc tính dạng categorical, câu hỏi sẽ là *Nó rơi vào category nào?* hoặc *Nó có rơi vào category nào đó không?* với trường hợp nhị phân. Với các thuộc tính dạng liên tục, câu hỏi có thể là *Nó nằm vào khoảng giá trị nào?* hoặc *Nó có lớn hơn một ngưỡng nào đó không*?

ID3 là một thuật toán được áp dụng cho các bài toán phân loại mà tất cả các thuộc tính đều ở dạng categorical.

**Thuật toán ID3**

Trong ID3, *tổng có trọng số của entropy tại các leaf-node* sau khi xây dựng decision tree được coi là hàm mất mát của decision tree đó. Các trọng số ở đây tỉ lệ với số điểm dữ liệu được phân vào mỗi node. Công việc của ID3 là tìm các cách phân chia hợp lý (thứ tự chọn thuộc tính hợp lý) sao cho hàm mất mát cuối cùng đạt giá trị càng nhỏ càng tốt. Như đã đề cập, việc này đạt được bằng cách chọn ra thuộc tính sao cho nếu dùng thuộc tính đó để phân chia, entropy tại mỗi bước giảm đi một lượng lớn nhất. Bài toán xây dựng một decision tree bằng ID3 có thể chia thành các bài toán nhỏ, trong mỗi bài toán, ta chỉ cần chọn ra thuộc tính giúp cho việc phân chia đạt kết quả tốt nhất. Mỗi bài toán nhỏ này tương ứng với việc phân chia dữ liệu trong một *non-leaf node*. Chúng ta sẽ xây dựng phương pháp tính toán dựa trên mỗi node này.

Xét một bài toán với C class khác nhau. Giả sử ta đang làm việc với một non-leaf node với các điểm dữ liệu tạo thành một tập S với số phần tử của |S| = N. Giả sử thêm rằng trong số N diểm dữ liệu này, Nc với c=1, 2, …, C điểm thuộc vào class C. Xác suất để mỗi điểm dữ liệu rơi vào một class C được xấp xỉ bằng . Như vậy, entropy tại node này được tính bởi:

Tiếp theo, giả sử thuộc tính được chọn là x. Dựa trên x các điểm dữ liệu trong S được phân ra thành K child node S1, S2, …, Sk với số điểm trong mỗi child node lần lượt là m1, m2, …, mK. Ta định nghĩa:

Là tổng trong số entropy của mỗi childnode. Việc lấy trọng số này là quan trọng vì node thường có số lượng điểm khác nhau.

Tiếp theo ta định nghĩa information gian dựa trên thuộc tính x:

Trong ID3, tại mỗi node thuộc tính được chọn được xác định dữa trên:

Để hiểu rõ hơn về thuật toán ID3 ta đi vào ví dụ sau:

Giả sử ta có bảng dữ liệu về mối quan hệ thời tiết và việc có quyết định đi chơi hay không như sau:

| **id** | **outlook** | **temperature** | **humidity** | **wind** | **play** |
| --- | --- | --- | --- | --- | --- |
| 1 | sunny | hot | high | weak | no |
| 2 | sunny | hot | high | strong | no |
| 3 | overcast | hot | high | weak | yes |
| 4 | rainy | mild | high | weak | yes |
| 5 | rainy | cool | normal | weak | yes |
| 6 | rainy | cool | normal | strong | no |
| 7 | overcast | cool | normal | strong | yes |
| 8 | sunny | mild | high | weak | no |
| 9 | sunny | cool | normal | weak | yes |
| 10 | rainy | mild | normal | weak | yes |
| 11 | sunny | mild | normal | strong | yes |
| 12 | overcast | mild | high | strong | yes |
| 13 | overcast | hot | normal | weak | yes |
| 14 | rainy | mild | high | strong | no |

Bảng 3.1: Bảng dữ liệu về kết quả của việc chơi hay không dựa trên thời tiết

Có bốn thuộc tính thời tiết:

1. Outlook nhận một trong ba giá trị: sunny, overcast, rainy.
2. Temperature nhận một trong ba giá trị: hot, cool, mild.
3. Humidity nhận một trong hai giá trị: high, normal.
4. Wind nhận một trong hai giá trị: weak, strong.

(Tổng cộng có 3×3×2×2=36 loại thời tiết khác nhau, trong đó 14 loại được thể hiện trong bảng.)

Đây có thể được coi là một bài toán dự đoán liệu đội bóng có chơi bóng không dựa trên các quan sát thời tiết. Ở đây, các quan sát đều ở dạng categorical. Cách dự đoán dưới đây tương đối đơn giản và khá chính xác, có thể không phải là cách ra quyết định tốt nhất:

* Nếu outlook = sunny và humidity = high thì play = no.
* Nếu outlook = rainy và windy = true thì play = no.
* Nếu outlook = overcast thì play = yes.
* Ngoài ra, nếu humidity = normal thì play = yes.
* Ngoài ra, play = yes.

Chúng ta sẽ cùng tìm thứ tự các thuộc tính bằng thuật toán ID3.

Trong 14 giá trị đầu ra ở Bảng trên, có năm giá trị bằng no và chín giá trị bằng yes. Entroy tại root node của bài toán là:

Tiếp theo, chúng ta tính tổng có trọng số entropy của các *child node* nếu chọn một trong các thuộc tính *outlook, temperature, humidity, wind, play* để phân chia dữ liệu.

Xét thuộc tính *outlook*. Thuộc tính này có thể nhận một trong ba giá trị *sunny, overcast, rainy*. Mỗi một giá trị sẽ tương ứng với một *child node*. Gọi tập hợp các điểm trong mỗi child node này lần lượt là Ss,So,Sr với tương ứng ms,mo,mr phần tử. Sắp xếp lại Bảng ban đầu theo thuộc tính outlook ta đạt được ba Bảng nhỏ sau đây.

| **id** | **outlook** | **temperature** | **humidity** | **wind** | **play** |
| --- | --- | --- | --- | --- | --- |
| 1 | sunny | hot | high | weak | no |
| 2 | sunny | hot | high | strong | no |
| 8 | sunny | mild | high | weak | no |
| 9 | sunny | cool | normal | weak | yes |
| 11 | sunny | mild | normal | strong | yes |

Bảng 3.2: Bảng dữ liệu của việc chơi hay không khi outlook là sunny

| **id** | **outlook** | **temperature** | **humidity** | **wind** | **play** |
| --- | --- | --- | --- | --- | --- |
| 3 | overcast | hot | high | weak | yes |
| 7 | overcast | cool | normal | strong | yes |
| 12 | overcast | mild | high | strong | yes |
| 13 | overcast | hot | normal | weak | yes |

Bảng 3.3: Bảng dữ liệu của việc chơi hay không khi outlook là overcast

| **id** | **outlook** | **temperature** | **humidity** | **wind** | **play** |
| --- | --- | --- | --- | --- | --- |
| 4 | rainy | mild | high | weak | yes |
| 5 | rainy | cool | normal | weak | yes |
| 6 | rainy | cool | normal | strong | no |
| 10 | rainy | mild | normal | weak | yes |
| 14 | rainy | mild | high | strong | no |

Bảng 3.4: Bảng dữ liệu của việc chơi hay không khi outlook là rainy

Quan sát nhanh ta thấy rằng child node ứng với outlook = overcast sẽ có entropy bằng 0 vì tất cả mo=4 output đều là yes. Hai child node còn lại với ms=mr=5 có entropy khá cao vì tần suất output bằng yes hoặc no là xấp xỉ nhau. Tuy nhiên, hai child node này có thể được phân chia tiếp dựa trên hai thuộc tính humidity và wind.

Ta có thể kiểm tra được rằng:

Xét thuộc tính temperature, ta có phân chia như các Bảng dưới đây.

| **id** | **outlook** | **temperature** | **humidity** | **wind** | **play** |
| --- | --- | --- | --- | --- | --- |
| 1 | sunny | hot | high | weak | no |
| 2 | sunny | hot | high | strong | no |
| 3 | overcast | hot | high | weak | yes |
| 13 | overcast | hot | normal | weak | yes |

Bảng 3.5: Bảng dữ liệu với temperature là hot

| **id** | **outlook** | **temperature** | **humidity** | **wind** | **play** |
| --- | --- | --- | --- | --- | --- |
| 4 | rainy | mild | high | weak | yes |
| 8 | sunny | mild | high | weak | no |
| 10 | rainy | mild | normal | weak | yes |
| 11 | sunny | mild | normal | strong | yes |
| 12 | overcast | mild | high | strong | yes |
| 14 | rainy | mild | high | strong | no |

Bảng 3.6: Bảng dữ liệu với temperature là mild

| **id** | **outlook** | **temperature** | **Humidity** | **wind** | **play** |
| --- | --- | --- | --- | --- | --- |
| 5 | rainy | cool | Normal | weak | yes |
| 6 | rainy | cool | Normal | strong | no |
| 7 | overcast | cool | Normal | strong | yes |
| 9 | sunny | cool | Normal | weak | yes |

Bảng 3.7: Bảng dữ liệu với temperature là cool

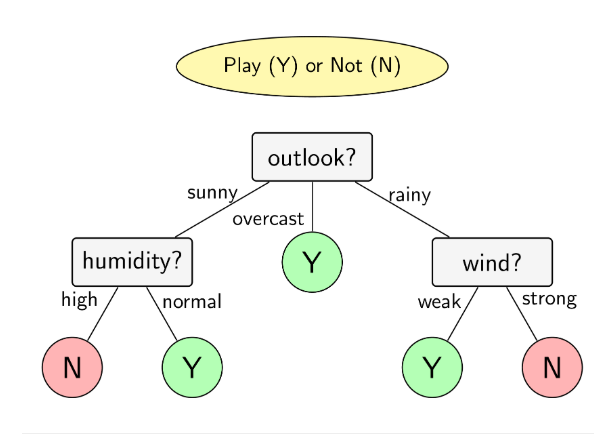
Gọi Sh, Sm, Sc là ba tập con tương ứng với temperature bằng hot, mild, cool. Ta có thể tính được:

Nếu các kết quả là giống nhau, chúng sẽ bằng:

Như vậy, thuộc tính cần chọn ở các bước đầu tiên là outlook vì H(outlook,S) đạt giá trị nhỏ nhất (information gain lớn nhất);

Sau bước phân chia đầu tiên này, ta nhận được ba child node với các phần tử như trong ba Bảng phân chia theo *outlook*. Child node thứ hai không cần phân chia tiếp vì nó đã *tinh khiết*. Với child node thứ nhất, ứng với *outlook = sunny*, kết quả tính được bằng ID3 sẽ cho chúng ta thuộc tính *humidity* vì tổng trọng số của entropy sau bước này sẽ bằng 0 với output bằng *yes* khi và chỉ khi *humidity = normal*. Tương tự, child node ứng với *outlook = wind* sẽ được tiếp tục phân chia bởi thuộc tính *wind* với output bằng *yes* khi và chỉ khi *wind = weak*.

Như vậy, cây quyết định cho bài toán này dựa trên ID3 sẽ có dạng như sau:



Hình 3.6: Cây quyết định được xây dựng dựa trên ID3

Trong các thuật toán decision tree nói chung và ID3 nói riêng, nếu ta tiếp tục phân chia các node chưa tinh khiết, ta sẽ thu được một tree mà mọi điểm trong tập huấn luyện đều được dự đoán đúng (giả sử rằng không có hai input giống nhau nào cho output khác nhau). Khi đó, tree có thể sẽ rất phức tạp (nhiều node) với nhiều leaf node chỉ có một vài điểm dữ liệu. Như vậy, nhiều khả năng overfitting sẽ xảy ra.

Để tránh overfitting, một trong số các phương pháp sau có thể được sử dụng. Tại một node, nếu một trong số các điều kiện sau đây xảy ra, ta không tiếp tục phân chia node đó và coi nó là một leaf node:

* nếu node đó có entropy bằng 0, tức mọi điểm trong node đều thuộc một class.
* nếu node đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này, ta chấp nhận có một số điểm bị phân lớp sai để tránh overfitting. Class cho leaf node này có thể được xác định dựa trên class chiếm đa số trong node.
* nếu khoảng cách từ node đó đến root node đạt tới một giá trị nào đó. Việc hạn chế chiều sâu của tree này làm giảm độ phức tạp của tree và phần nào giúp tránh overfitting.
* nếu tổng số leaf node vượt quá một ngưỡng nào đó.
* nếu việc phân chia node đó không làm giảm entropy quá nhiều (information gain nhỏ hơn một ngưỡng nào đó).

Ngoài các phương pháp trên, một phương pháp phổ biến khác được sử dụng để tránh overfitting là pruning, tạm dịch là cắt tỉa.

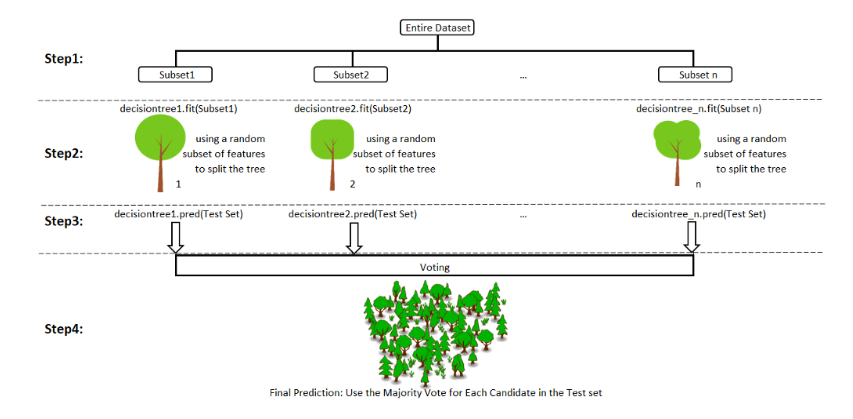
Pruning [13] là một kỹ thuật regularization để tránh overfitting cho decision tree nói chung. Trong pruning, một decision tree sẽ được xây dựng tới khi mọi điểm trong training set đều được phân lớp đúng. Sau đó, các leaf node có chung một non-leaf node sẽ được cắt tỉa và non-leaf node đó trở thành một leaf-node, với class tương ứng với class chiếm đa số trong số mọi điểm được phân vào node đó. Việc cắt tỉa cây quyết định này có thể được xác định dựa vào các cách sau.

1. Dựa vào một validation set. Trước tiên, training set được tách ra thành một training set nhỏ hơn và một validation set. Decision tree được xây dựng trên training set cho tới khi mọi điểm trong training set được phân lớp đúng. Sau đó, đi ngược từ các leaf node, cắt tỉa các sibling node của nó và giữ lại node bố mẹ nếu độ chính xác trên validation set được cải thiện. Khi nào độ chính xác trên validation set không được cải thiện nữa, quá trình pruning dừng lại. Phương pháp này còn được gọi là reduced error pruning.
2. Dựa vào toàn bộ data set. Trong phương pháp này, ta không tách tập training ban đầu ra mà sử dụng toàn bộ dữ liệu trong tập này cho việc xây dựng decision tree. Một ví dụ cho việc này là cộng thêm một đại lượng regularization vào hàm mất mát. Đại lượng regularization sẽ lớn nếu số leaf node là lớn. Cụ thể, giả sử decision tree cuối cùng có K leaf node, tập hợp các điểm huấn luyện rơi vào mỗi leaf node lần lượt là S1, …, SK. Khi đó, regularized loss của ID3 có thể được tính tương tự như :

với |Sk| ký hiệu số phần tử của tập hợp Sk và H(Sk) chính là entropy của leaf node tương ứng với Sk, và λ là một số thực dương không quá lớn. Giá trị của hàm số này nhỏ nếu cả data loss–số hạng thứ nhất–nhỏ (entropy tại mỗi node là thấp) và regularization–số hạng thứ hai–cũng nhỏ (số leaf node là ít). Việc tối ưu có thể được thực hiện thông qua pruning như sau. Trước hết, xây dựng một decision tree mà mọi điểm trong tập huấn luyện đều được phân loại đúng (toàn bộ các entopy của các node bằng 0). Lúc này data loss bằng 0 nhưng regularization có thể lớn, khiến cho L lớn. Sau đó, ta có thể tỉa dần các leaf node sao cho L giảm. Việc cắt tỉa được lặp lại đến khi L không thể giảm được nữa.

* 1. **Random Forest**

Random Forest [12] hay rừng ngẫu nhiên là mô hình sử dụng bagging hay mô hình tập hợp sử dụng cây quyết định làm các mô hình con.



Hình 3.7: Sơ đồ các bước thực hiện của Random Forest

Các bước thực hiện của random forest:

Bước 1: Chọn N tập con ngẫu nhiên từ tập huấn luyện

Bước 2: Huấn luyện từng tập con với từng cây quyết định

- Một tập hợp con ngẫu nhiên được sử dụng để huấn luyện một cây quyết định

- Các phân chia tối ưu cho mỗi cây quyết định dựa trên một tập hợp con các tính năng ngẫu nhiên (ví dụ: tổng cộng 10 tính năng, chọn ngẫu nhiên 5 trên 10 tính năng để phân tách)

Bước 3: Mỗi cây riêng lẻ sẽ dự đoán kết quả trong bộ dữ liệu test

Bước 4: Đưa ra quyết định cuối cùng, ở bước này thì rừng ngẫu nhiên sẽ đưa ra quyết định dựa vào số dự đoán có số phiếu nhiều nhất.

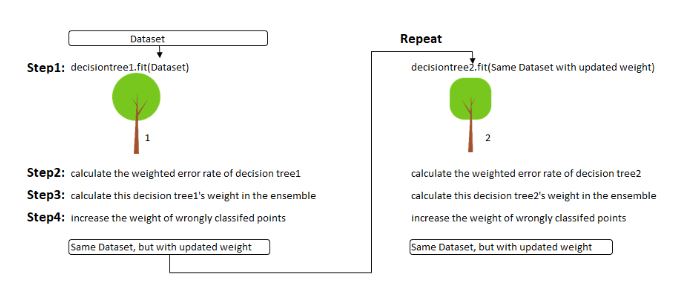
Có thể thấy rằng Random Forest là một thuật toán đơn thuần sử dụng nhiều cây quyết định để tạo ra nhiều kết quả hơn và tăng độ chính xác của một cây quyết định.

* 1. **AdaBoost**

AdaBoost [12], viết tắt của Adaptive Boosting, là một thuật toán học máy được xây dựng bởi Yoav Freund và Robert Schapire, người đã giành giải thưởng Gôdel 2003 cho công việc của họ. Nó có thể được sử dụng kết hợp với nhiều loại thuật toán học tập khác để cải thiện hiệu suất. Đầu ra của các thuật toán học tập khác (weak leaner) được kết hợp thành một tổng có trọng số đại diện cho đầu ra cuối cùng của trình phân loại được tăng cường. AdaBoost thích nghi theo nghĩa là những mô hình yếu được điều chỉnh theo hướng có lợi cho những trường hợp bị phân loại sai bởi các phân loại trước đó. AdaBoost nhạy cảm với dữ liệu ồn ào.

Mọi thuật toán học tập có xu hướng phù hợp với một số loại vấn đề tốt hơn các loại vấn đề khác và thường có nhiều tham số và cấu hình khác nhau để điều chỉnh trước khi đạt được hiệu suất tối ưu trên tập dữ liệu, AdaBoost (với cây quyết định là weak leaner) là một trong những mô hình phân loại khá tốt. Khi được sử dụng với cây quyết định, thông tin được thu thập ở mỗi giai đoạn của thuật toán AdaBoost về 'độ cứng' tương đối của từng mẫu đào tạo được đưa vào các cây quyết định tiếp theo sao cho các cây sau này có xu hướng tập trung vào các ví dụ khó phân loại hơn.

Các bước tiến hành của AdaBoost:



Hình 3.8: Sơ đồ các bước thực hiện của AdaBoost

Bước 0: Khởi tạo trọng số của các điểm dữ liệu. nếu tập huấn luyện có 100 điểm dữ liệu, thì mỗi điểm trọng lượng ban đầu phải là 1/100 = 0,01.

Bước 1: Bắt đầu training với cây quyết định

Bước 2: Tính tỷ lệ lỗi trọng số (e) của cây quyết định. Tỷ lệ lỗi có trọng số (e) chỉ là có bao nhiêu dự đoán sai trong tổng số và bạn xử lý các dự đoán sai khác nhau dựa trên trọng số điểm dữ liệu của nó. Trọng số càng cao, sai số tương ứng sẽ càng được tính trọng số trong quá trình tính toán (e)

Bước 3: Tính trọng số cây quyết định này trong tập hợp

trọng số của cây này = tốc độ học tập \* log ((1 - e) / e)

* Tỷ lệ lỗi có trọng số cao hơn của cây, cây càng ít quyền quyết định trong lần bỏ phiếu sau
* Tỷ lệ lỗi có trọng số thấp hơn của cây, quyền quyết định cao hơn cây sẽ được đưa ra trong lần bỏ phiếu sau

Bước 4: Cập nhật trọng số của các điểm được phân loại sai

trọng số của từng điểm dữ liệu =

* Nếu mô hình có điểm dữ liệu này chính xác, trọng lượng vẫn giữ nguyên
* Nếu mô hình có điểm dữ liệu này sai, trọng số mới của điểm này = trọng lượng cũ \* np.exp (trọng lượng của cây này)

Lưu ý: Trọng số của cây càng cao (cây này thực hiện càng chính xác), càng tăng (mức độ quan trọng) thì điểm dữ liệu được phân loại sai bởi cây này sẽ nhận được. Trọng số của các điểm dữ liệu được chuẩn hóa sau khi tất cả các điểm phân loại sai được cập nhật.

Bước 5: Lặp lại bước 1 cho đến khi đủ số lượng cây cần

Bước 6: Đưa ra dự đoán cuối cùng

AdaBoost đưa ra một dự đoán mới bằng cách cộng trọng số (của mỗi cây) nhân với dự đoán (của mỗi cây). Rõ ràng, cây có trọng số cao hơn sẽ có nhiều sức mạnh ảnh hưởng đến quyết định cuối cùng.

* 1. **XGBoost**

XGBoost [14] là một thuật toán Machine Learning dựa trên cây quyết định sử dụng khung tăng cường độ dốc. Trong các vấn đề dự đoán liên quan đến dữ liệu phi cấu trúc (hình ảnh, văn bản, v.v.), mạng nơ ron nhân tạo có xu hướng vượt trội hơn tất cả các thuật toán hoặc khung khác. Tuy nhiên, khi nói đến dữ liệu dạng bảng / cấu trúc từ nhỏ đến trung bình, các thuật toán dựa trên cây quyết định được coi là tốt nhất trong lớp ngay bây giờ.

Thuật toán XGBoost được phát triển như một dự án nghiên cứu tại Đại học Washington. Tianqi Chen và Carlos Guestrin đã trình bày bài báo của họ tại Hội nghị SIGKDD năm 2016 và bắt lửa thế giới Machine Learning. Kể từ khi được giới thiệu, thuật toán này không chỉ được ghi nhận là đã chiến thắng nhiều cuộc thi Kaggle mà còn là động lực thúc đẩy cho một số ứng dụng công nghiệp tiên tiến. Kết quả là, có một cộng đồng mạnh mẽ gồm các nhà khoa học dữ liệu đóng góp cho các dự án nguồn mở XGBoost với ~ 350 người đóng góp và ~ 3.600 commit trên GitHub. Thuật toán phân biệt chính nó theo các cách sau:

* Một loạt các ứng dụng: Có thể được sử dụng để giải quyết hồi quy, phân loại, xếp hạng và các vấn đề dự đoán do người dùng xác định.
* Tính di động: Chạy trơn tru trên Windows, Linux và OS X.
* Ngôn ngữ: Hỗ trợ tất cả các ngôn ngữ lập trình chính bao gồm C ++, Python, R, Java, Scala và Julia.
* Tích hợp đám mây: Hỗ trợ các cụm AWS, Azure và Sợi và hoạt động tốt với Flink, Spark và các hệ sinh thái khác.

XGBoost là viết tắt của Extreme Gradient Boosting; đó là một triển khai cụ thể của phương pháp Gradient Boosting sử dụng các xấp xỉ chính xác hơn để tìm mô hình cây tốt nhất. Nó sử dụng một số thủ thuật tiện lợi làm cho nó thành công đặc biệt, đặc biệt là với dữ liệu có cấu trúc. Quan trọng nhất là

* Tính toán độ dốc bậc hai, tức là các đạo hàm riêng thứ hai của hàm mất (tương tự như phương pháp Newton,), cung cấp thêm thông tin về hướng của độ dốc và cách đạt đến mức tối thiểu của hàm mất mát của chúng ta. Mặc dù tăng cường độ dốc thông thường sử dụng chức năng mất của mô hình cơ sở (ví dụ: cây quyết định) để giảm thiểu lỗi của mô hình tổng thể, XGBoost sử dụng đạo hàm bậc 2 làm xấp xỉ.
* Và chính quy hóa nâng cao (L1 & L2), giúp cải thiện khái quát hóa mô hình.

A screenshot of a cell phone

Description automatically generated

Hình 3.9: Sơ đồ các bước thực hiện của XGBoost

Các bước thực hiện của Gradient Boosting:

* Bước 1: Bắt đầu huấn luyện với cây quyết định
* Bước 2: Áp dụng cây quyết định vừa được huấn luyện để dự đoán
* Bước 3: Tính toán phần dư của cây quyết định này, Lưu các lỗi còn lại là y mới
* Bước 4: Lặp lại bước 1 cho tới khi đủ số lượng cây đề ra
* Bước 5: Đưa ra quyết định cuối cùng

Gradient Boosting đưa ra một dự đoán mới bằng cách thêm các dự đoán (của tất cả các cây).

* 1. **Tổng kết chương 3**

Trong chương 3, đồ án đã trình bày chi tiết về cây quyết định, một mô hình học con được sử dụng trong 3 mô hình Random Forest, AdaBoost, XGBoost. Đồng thời cũng đưa ra được các bước hoạt động cơ bản của 3 mô hình chính.

Trong chương tiếp theo, đồ án sẽ đề cập đến quá trình thu thập và gán nhãn cho dữ liệu để phục vụ cho quá trình thực nghiệm, tiến hành các thực nghiệm với bộ dữ liệu đã thu thập được với các mô hình học máy đã đề ra

# THỰC NGHIỆM VÀ ĐÁNH GIÁ

Trong chương 4, đồ án sẽ đề cặp tới quá trình thu thập dữ liệu và đưa ra kết quả thực nghiệm dữ trên bộ dữ liệu thu thập được thông qua các phần sau:

* **Thu thập và gán nhãn dữ liệu**
* **Xử lý dữ liệu**
* **Phương pháp đánh giá độ chính xác**
* **Kết quả thực nghiệm**
  1. **Thu thập và gán nhãn dữ liệu**

Đồ án đề xuất xây dựng mô hình phát hiện và phân loại tiếng ngáy giữa các âm thanh khác. Tuy nhiên tiếng ngáy chỉ có thể thu được khi con người chìm vào giấc ngủ, nên không gian rất yên tĩnh, không chứa nhiều tạp âm hay nhiễu. Vì vậy trong phạm vi đồ án này sẽ chỉ tập trung vào việc thu thập tiếng ngáy và thực nghiệm trên các mô hình học máy đã đề ra.

1. **Thu thập dữ liệu**

Đồ án thu thập tiếng ngáy của 3 người khác nhau. Mỗi người tham gia thu dữ liệu sẽ sử dụng điện thoại đặt cách vị trí ngủ khoảng 20-30 cm, các file dữ liệu sẽ chỉ giữ lại các đoạn âm thanh có chứa tiếng ngáy. Mỗi người sẽ chia ra làm 3 lần thu dữ liệu mỗi lần trung bình khoảng 35 phút.

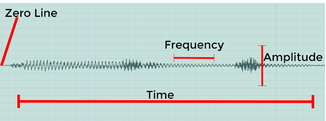
File dữ liệu sẽ được lưu dưới dạng ngay\_{stt}.wav, trong đó stt là số thứ tự của file.

1. **Chuẩn lưu trữ âm thanh**

Khi lưu trữ âm thanh ta tuân theo các chuẩn sau:

* Định dạng: **WAV File**
* Tốc độ lấy mẫu (**Sample Rate**): **48kHz**
* Số bit để biểu diễn 1 mẫu (**Bit depth**): **16**

**Ghi âm** là quá trình chuyển đổi âm thanh (chuyển động áp suất không khí hoặc năng lượng âm thanh) thành âm thanh / thông tin kỹ thuật số. Sau khi thu được âm thanh kỹ thuật số được biểu diễn dưới dạng sóng.



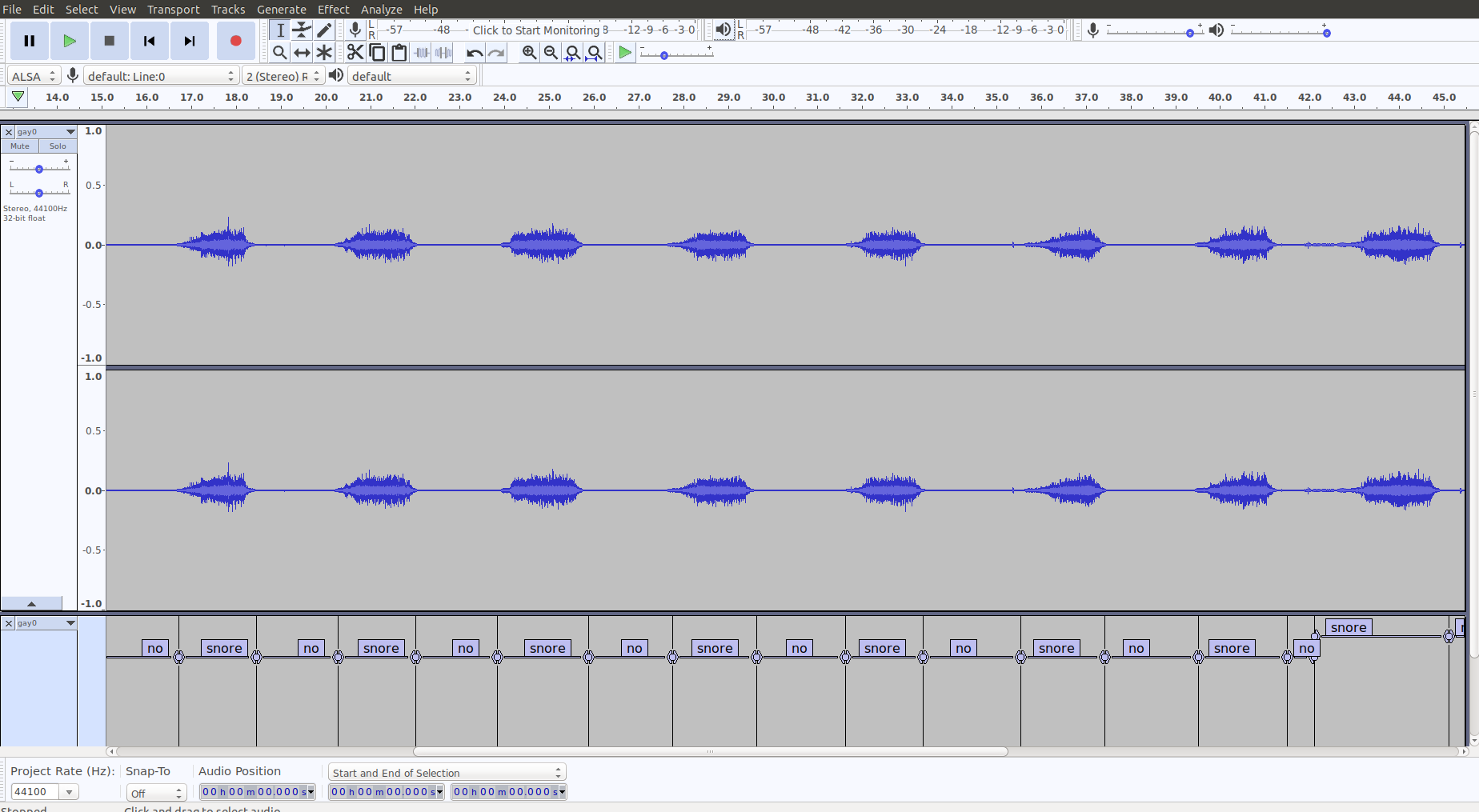
Hình 4.1: Hình ảnh minh họa tín hiệu âm thanh

* **Zero Line**: Đây là nơi không có âm thanh hoặc chuyển động trong áp suất không khí.
* **Thời gian**: Thời gian theo nghĩa đen là thời gian ghi hoặc trục thời gian (ngang).
* **Biên độ**: Dạng sóng càng lớn theo chiều dọc thì âm thanh càng lớn, được đo theo chiều dọc trên trục biên độ.
* **Tần số**: Đây là đại diện cho âm trầm và âm bổng và xảy ra theo chiều ngang dọc theo dạng sóng. Nếu thấy sóng quá chặt hoặc gần, đó là âm cao (treble), ngược lại sóng phát ra là âm thấp (bass).

1. **Gán nhãn dữ liệu**

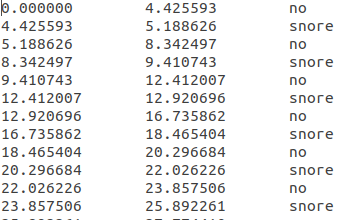
Công cụ gán nhãn được sử dụng là Audacity. Đây là một phần mềm rất trực quan, dễ sử dụng và tương thích với hầu hết các hệ điều hành phổ biến hiện nay như Windows, Mac OS và Linux. Nó hoàn toàn miễn phí và hỗ trợ cho nhiều định dạng file khác nhau. Chúng ta có thể gán nhãn (label) vào label track dễ dàng cho các đoạn âm thanh nhờ Audacity.

Tất cả các âm thanh được gán nhãn với một trong 3 nhãn sau: normal, deep, strong. Việc làm này được thực hiện tỉ mỉ và được thực hiện 2 lần. Lần đầu là một người thực hiện việc gán nhãn, lần sau là người khác sẽ kiếm tra lại việc gán nhãn này để đảm bảo không có dữ liệu nào bị gán nhãn sai. Việc thực hiện gán nhãn bằng công cụ Audacity.



Hình 4.2: Hình ảnh tín hiệu âm thanh hiển thị trong Audacity

Nhãn của dữ liệu sẽ được lưu dưới dạng file (.txt) và cấu trúc là “starttime endtime label”, hình dưới mình họa cấu trúc của nhãn



Hình 4.3: Số liệu gán nhãn của từng file âm thanh

* 1. **Xử lý dữ liệu**

Dữ liệu sẽ là các đoạn âm thanh rất dài nên ta cần tiến hành chia nhỏ âm thanh ra làm các đoạn 1 giây và đè lên nhau 0.5 giây. Trong 1 giây này sẽ chứa 1 phần hoặc cả 1 nhịp ngáy, với những đoạn chứa 1 phần tiếng ngáy nếu thời lượng chứa tiếng ngáy chiếm hơn 70% thì ta sẽ coi đó là tiếng ngáy – “snore” và ngược lại sẽ là “no”.

Sau khi tách đoạn âm thanh thì ta thu được bảng thống kê như sau:

|  |  |
| --- | --- |
| **Nhãn dữ liệu** | **Số lượng** |
| Snore | 17402 |
| No | 17854 |

Bảng 4.1: Bảng thông kê số lượng nhãn dữ liệu

Sau đó ta tiến hành tách đặc trưng của các đoạn âm thành và tiến hành quá trình huấn luyện trên Random Forest, AdaBoost, XGBoost để so sánh.

* 1. **Phương pháp đánh giá và so sánh**

Để đánh giá độ tốt của việc phát hiện cũng như so sánh giữa các mô hình ta sử dụng tham số **Accuracy**.

Công thức tính Accuracy được định nghĩa như sau:

* 1. **Kết quả thực nghiệm**

Trong quá trình thực nghiệm để so sánh được chính xác hiệu quả giữa các mô hình thì đồ án đều sử dụng tham số đầu vào cho cả 3 mô hình có cùng learning rate (tốc độ học) là 0.1 và số cây quyết định sử dụng trong quá trình huấn luyện là 100.

Sau khi huấn luyện qua 3 mô hình Random Forest, AdaBoost, XGBoost thì ta thu được bảng so sánh như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| Đặc trưng | Mô hình | | |
| Random Forest | AdaBoost | XGBoost |
| MFCC | 0.65 | 0.71 | 0.75 |
| GFCC | 0.61 | 0.62 | 0.66 |
| LPC | 0.55 | 0.59 | 0.60 |
| MFCC + GFCC+ LPC | 0.9 | 0.92 | 0.95 |

Bảng 4.2: Bảng so sánh kết quả giữa các đặc trưng và mô hình học máy

Từ kết quả thực nghiệm có thể thấy rằng MFCC là đặc trưng tốt nhất trong 3 loại (75%) và khi kết hợp 3 loại đặc trung lại với nhau thì ta thu được kết quả tốt nhất (95%).

Qua việc kết quả này có thể thầy rằng XGBoost là mô hình tốt nhất, luôn đưa ra độ chính xác cao nhất đối với từng loại đặc trưng lần lượt là 75%, 66%, 70%, 95%.

* 1. **Tổng kết chương 4**

Trong chường 4, đồ án đã trình bày được quá trình thu thập dữ liệu và kết quả thực nghiệm trên bộ dữ liệu thu được từ đó đưa ra kết luận về các đặc trưng âm thanh và mô hình học máy.

# TỔNG KẾT

Trên cơ sở tìm hiểu và phân tích những phương pháp để xác định tiếng ngáy. Đồ án đã đạt được những kết quả như sau:

* Giới thiệu khát quát được khái niệm về âm thanh, đặc biệt là về tiếng ngáy, nêu ra được những lợi ích của việc phát hiện tiếng ngáy.
* Trình bày và phân tích các đặc trưng âm thanh phổ biến sử dụng trong phát hiện tiếng ngáy như MFCC, GFCC, LPC, LPCC, PLP.
* Trình bày các khái niệm về trí tuệ nhân tạo và học máy, đi sâu vào các mô hình học máy như cây quyết định, Random Forest, AdaBoost, XGBoost.
* Chỉ ra rõ các bước tiền xử lý âm thanh từ dữ liệu thô đến trích xuất các đặc trung đưa vào các mô hình học máy để thực hiện huấn luyện xác định tiếng ngáy.
* Trình bày phương pháp đánh giá độ chính xác của kết quả thu được sau khi huyến luyện dữ liệu.
* Thực hiện thu thập tiếng ngáy để đưa vào quá trình thực nghiệm đánh giá hiệu quả của các đặc trưng và mô hình đã đề ra.

**Hướng phát triển trong tương lai**:

* Sử dụng các mô hình học sâu như **Convolutional Neuron Network**, **Long Short Term Memory** để cái độ hiệu quả của việc xác định tiếng ngáy
* Nghiên cứu thêm các đặc trưng âm thanh tốt phù hợp hơn với tiếng ngáy.
* Phát triển được ứng dụng tự động phát hiện tiếng ngáy trên thiết bị di động.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Wikipedia, "Âm thanh," [Online]. Available: https://vi.wikipedia.org/wiki/Am\_thanh. |
| [2] | A. Store, "Khái niệm cơ bản về âm trầm, âm cao, âm trung của âm thanh," [Online]. Available: http://audiostore.vn/t4838-khai-niem-co-ban-ve-am-tram-am-cao-am-trung-cua-am-thanh. |
| [3] | Vinmec, "Ngủ ngáy có thể cảnh báo một số nguy cơ sức khỏe sau," [Online]. Available: https://www.vinmec.com/vi/tin-tuc/thong-tin-suc-khoe/ngu-ngay-co-canh-bao-mot-so-nguy-co-suc-khoe-sau/. |
| [4] | K. Prahallad, "Spectrogram, Cepstrum and Mel-Frequency Analysis," Carnegie Mellon University. |
| [5] | Wikipedia, "Fast Fourier transform," [Online]. Available: https://en.wikipedia.org/wiki/Fast\_Fourier\_transform. |
| [6] | P. Cryptography, "Mel Frequency Cepstral Coefficient (MFCC) tutorial," [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. |
| [7] | Wikipedia, "Discrete cosine transform," [Online]. Available: https://en.wikipedia.org/wiki/Discrete\_cosine\_transform. |
| [8] | N. Dave, "Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition," *INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY,* vol. 1, no. VI, 2013. |
| [9] | S. S. Admuthe and P. H. Patil, "Feature extraction method - MFCC and GFCC used for Speaker Identification," *International Journal for Scientific Research & Development,* vol. 3, no. 4, pp. 2321-0613, 2015. |
| [10] | H. Gupta and D. Gupta, "LPC and LPCC method of feature extraction in Speech Recognition System," in *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, 2016. |
| [11] | T. D. Science, "Basic Ensemble Learning (Random Forest, AdaBoost, Gradient Boosting)- Step by Step Explained," [Online]. Available: https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725. |
| [12] | T. Vu, "Decision Trees," [Online]. Available: https://machinelearningcoban.com/2018/01/14/id3/. |
| [13] | T. Vu, "Giới thiệu về Machine Learning," [Online]. Available: https://machinelearningcoban.com/2016/12/26/introduce/. |
| [14] | XgBoost, "XgBoost," [Online]. Available: https://github.com/dmlc/xgboost. |
| [15] | L. Breiman, "Random Forests," in *Machine Learning*, Kluwer Academic Publishers, 2001, p. 5–32. |
| [16] | Kaggle, "Kaggle: Your Home for Data Science.," [Online]. Available: https://www.kaggle.com/. |
| [17] | Scikit-learn, "Scikit-learn: machine learning in Python," [Online]. Available: scikitlearn.org. |
| [18] | T. Chen and C. Guestrin, " XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, California, USA, 2016. |
| [19] | J. Li and Y. Hong, "Wheeze Detection Algorithm Based on Spectrogram Analysis," in *8th International Symposium on Computational Intelligence and Design (ISCID)*, 2015. |
| [20] | F. Jin, S. Krishnan and F. Sattar, "Adventitious Sounds Identification and Extraction Using Temporal–Spectral Dominance-Based Features," *IEEE Transactions on Biomedical Engineering,* vol. 58, no. 11, pp. 3078 - 3087, 2011. |
| [21] | C. O. Dumitru and I. Gavat, "A Comparative Study of Feature Extraction Methods Applied to Continuous Speech Recognition in Romanian Language," in *ELMAR 2006*, Zadar, Croatia, 2006. |
| [22] | U. o. T. I. F. Lahouti Dept. of Electr. & Comput. Eng., A. Fazel, A. Safavi-Naeini and A. Khandani, "Single and double frame coding of speech LPC parameters using a lattice-based quantization scheme," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 14, no. 6, pp. 1624 - 1632, 2006. |
| [23] | H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *The Journal of the Acoustical Society of America,* vol. 87, no. 4, p. 1738, 1990. |
| [24] | J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE,* vol. 63, no. 4, pp. 561 - 580, 1975. |
| [25] | A. Yadollahi and Z. Moussavi, "Acoustic obstructive sleep apnea detection," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Minneapolis, MN, USA, 2009. |
| [26] | M. CAVUSOGLU, M. KAMASAK, O. EROGUL, T. CILOGLU, T. AKCAM and Y. SERINAGAOGLU, "SASA: A SOFTWARE SYSTEM FOR SLEEP AND SNORING ANALYSIS," in *International Symposium on Health Informatics and Bioinformatics (HIBIT) 2007*, Belek, Antalya, Turkey, 2007. |
| [27] | N. A. K., K. T. S., B. E. and P. K., "Speech-like Analysis of Snore Signals for the Detection of Obstructive Sleep Apnea," in *International Conference on Biomedical and Pharmaceutical Engineering*, 2006. |