TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

─────── \* ───────

ĐỒ ÁN

**TỐT NGHIỆP ĐẠI HỌC**

NGÀNH CÔNG NGHỆ THÔNG TIN

**PHÂN VÙNG ẢNH MÀU SỬ DỤNG THUẬT TOÁN K-MEANS**

Sinh viên thực hiện: **Trần Văn Thành**

Lớp CNTT2.01 – K58

Giáo viên hướng dẫn: **PGS. Phạm Văn Hải**

HÀ NỘI 1 - 2019

# Phiếu giao nhiệm vụ đồ án tốt nghiệp

*1. Thông tin về sinh viên*

Họ và tên sinh viên: Trần Văn Thành

Điện thoại liên lạc: 0978779701

Email: thanhtranvan.hust@gmail.com

Lớp: CNTT2.01 - K58 Hệ đào tạo: Đại học chính quy

Đồ án tốt nghiệp được thực hiện tại: Viện CNTT&TT - Đại học Bách Khoa Hà Nội

Từ ngày 31/ 08/ 2018 đến ngày 28/ 12/ 2018

*2. Mục đích nội dung của ĐATN*

Phần 1: Thuật toán K-means clustering và ứng dụng cho bài toán phân vùng ảnh.

Phần 2: Phát hiện đám cháy trong ảnh và video thời gian thực.

*3. Các nhiệm vụ cụ thể của ĐATN*

1. Tìm hiểu và cài đặt các ứng dụng của K-means cho các bài toán phân loại chữ số viết tay.
2. Cải tiến thuật toán K-means clusstering, áp dụngcho bài toán phân vùng ảnh màu, đánh giá và so sánh kết quả với các công trình nghiên cứu khác.
3. Xử lý ảnh phát hiện đám cháy trong video thời gian thực.

*4. Lời cam đoan của sinh viên*

Tôi – Trần Văn Thành - cam kết các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm 2019*  *Trần Văn Thành* |

*5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành và cho phép bảo vệ*

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm 2019*  *PSG. Phạm Văn Hải* |

# Lời cảm ơn

Lời đầu tiên, em xin chân thành cảm ơn tới thầy Phạm Văn Hải giảng viên bộ môn HTTT và thầy Đỗ Phan Thuận giảng viên bộ môn KHMT đã chấp nhận đề tài và tận tình chỉ bảo em hoàn thành đồ án này. Thầy đã cung cấp kiến thức, hướng dẫn tận tình, chỉ bảo cho em từ những bước đầu tiếp cận đề tài và trong suốt quá trình thực hiện đề tài này. Qua sự hướng dẫn của thầy em không chỉ học được nhiều kiến thức bổ ích mà còn học được những phương pháp nghiên cứu khoa học, thái độ và tinh thần nghiêm túc hết mình trong công việc.

Mặc dù em đã có những cố gắng hoàn thành đề tài này, xong do thời gian có hạn nên em cũng không thể tìm hiểu kỹ và đầy đủ hết các khía cạnh mà đề tài đề cập đến. Trong thời gian sắp tới em rất mong nhận được sự chỉ bảo tận tình của thầy hơn nữa để giúp em có thể hoàn thiện được đề tài hơn và áp dụng đề tài vào thực tế để đem lại lợi ích thiết thực theo đúng định hướng và mục tiêu mà đề tài đã đưa ra.

Em xin trân trọng cảm ơn thầy!

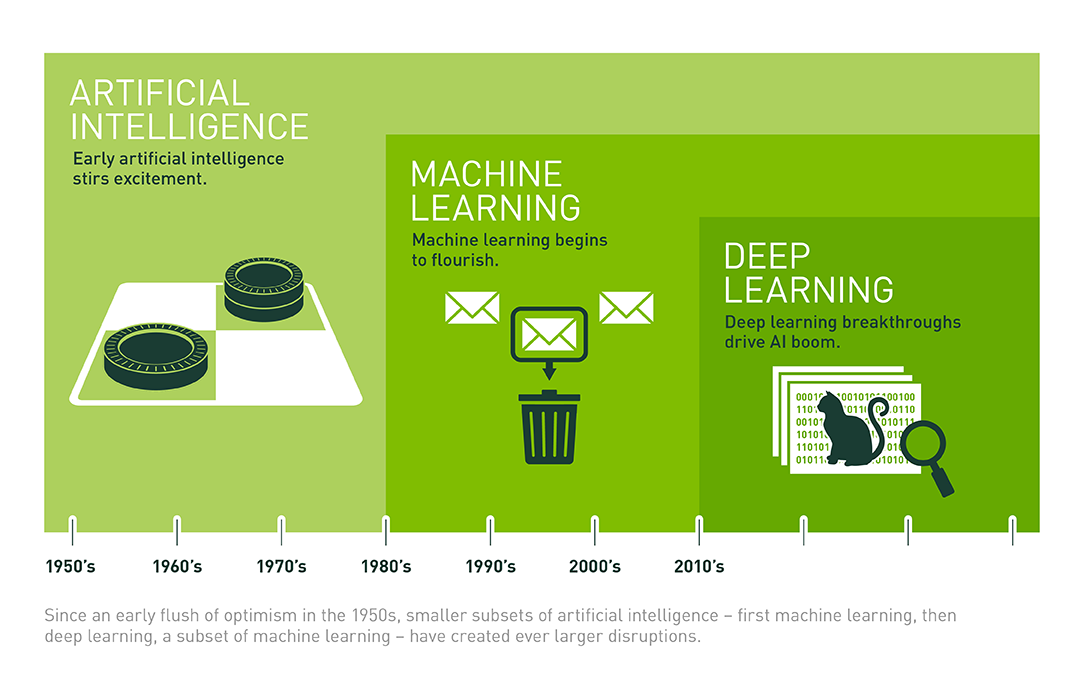
# CHƯƠNG 1: GIỚI THIỆU

## 1.1. Tổng quan

Những năm gần đây, AI - Artificial Intelligence (Trí Tuệ Nhân Tạo), và cụ thể hơn là Machine Learning (Học Máy) nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư 4.0 (1 - động cơ hơi nước, 2 - năng lượng điện và động cơ đốt trong, 3 – khoa học máy tính và internet). Trí Tuệ Nhân Tạo đang xuất hiện trong mọi lĩnh vực của đời sống mà có thể chúng ta không nhận ra. Xe tự lái của Google và Tesla, hệ thống tự gắng thẻ (tag) khuôn mặt người dùng của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, …, đó chỉ là một vài trong vô vàn những ứng dụng của AI/Machine Learning.

Học máy (machine learning - ML) là một tập con của AI. Nói đơn giản, Machine Learning là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể.

Sự phát triển của các hệ thống tính toán cùng với đó, lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn đã giúp Machine Learning tiến thêm một bước dài. Một lĩnh vực mới được ra đời gọi là học sâu (deep learning - DL). Deep Learning đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 10 năm trước: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn hay âm nhạc.



Hình: Mối quan hệ giữa AI, Machine Learning và Deep Learning.

Lĩnh vực machine learning và deep learning là cực kỳ rộng lớn và có nhiều nhánh nhỏ. Chúng ta luôn nhớ rằng điều đơn giản nhất là điều quan trọng nhất. Khi bắt tay vào giải quyết một bài toán machine learning hay bất cứ một bài toán nào, chúng ta nên bắt đầu từ những thuật toán đơn giản nhất. Ta không nên nghĩ rằng chỉ có những thuật toán phức tạp mới có thể giải quyết được vấn đề. Những thuật toán phức tạp thường yêu cầu độ tính toán cao và nhạy cảm với cách chọn tham số đầu vào. Thay vào đó, những thuật toán đơn giản giúp chúng ta sớm có một mô hình tổng quá cho bài toán. Các thuật toán đơn giản thường được gọi là baseline, sẽ giúp chúng ta có cái nhìn ban đầu về sự phức tạp của mỗi bài toán. Việc cải thiện kết quả sẽ được dần cải thiện ở các bước sau.

Từ những lý do trên, em quyết định sử dụng một trong những thuật toán cơ bản nhất trong Unsupervised learning - thuật toán K-means clustering (phân cụm K-means) để giải quyết bài toán.

**1.2. Mục đích đồ án**

Nghiên cứu về bài toán phân vùng ảnh (Image Segmentation) và ứng dụng của thuật toán phân cụm K-means để áp dụng nó vào giải quyết các bài toán xử lý ảnh trong thực tế. Đồng thời điều chỉnh các tham số thích hợp cho mô hình để đạt được kết quả có độ chính xác tốt nhất.

**1.3. Phát biểu bài toán**

Phần 1: Xử lý ảnh vệ tinh để tính toán được sự thay đổi của đô thị hóa, hay sự sói mòn đất, nạn chặt phá rừng, mức độ thiệt hại của thiên tai. Theo phương pháp truyền thống từ xưa dùng các bản vẽ, dựa vào bản đồ và các chuyến đi thực nghiệm, nhưng không phải nơi đâu trên trái đất con người cũng có thể đặt chân đến để kiểm tra đo đạc. Vì vậy ngày nay với các công cụ khoa học hiện đại, từ vệ tinh con người có thể thu được các hình ảnh từ bất cứ nơi đâu trên trái đất. Qua các ảnh vệ tinh chụp được một khu rừng, sau khi xử lý phân vùng ảnh thành các vùng như cây, đất, nước… Ta tính được tỷ lệ phần trăm diện tích của các vùng trong tấm ảnh, qua đó tính được diện tích các vùng, và so sánh kết quả từ đó kết luận được độ che phủ rừng, hay sự đô thị hóa… thay đổi qua các mốc thời gian.

Bài toán đặt ra: xây dựng mô hình học máy cho bài toán phân vùng ảnh (Image Segmentation) sử dụng thuật toán K-means clustering và một số ứng dụng cho các bài toán thực tế như : phân nhóm chữ số viết tay.

Phần 2: Phát hiện đám cháy trong ảnh và video thời gian thực. Hiện nay, các hệ thống báo cháy tự động trong các tòa nhà cao tầng, văn phòng,… là các cảm biến khói – thiết bị dùng để phát hiện khói – một thành phần khởi đầu của sự cháy. Có 2 loại cảm biến khói là quang điện và ion. Các cảm biến này thưởng gây ra các hiện tượng báo cháy giả và có thể làm tốn kém tiền của, khiến người dùng thiếu đi sự tin tưởng, dẫn đến hậu quả nghiêm trọng.

Bài toán đặt ra: Tách vùng cháy trong ảnh, phát hiện đám cháy trong video thời gian thực lấy từ các camera báo cháy, kết hợp với các cảm biến khói để giảm thiểu báo cháy giả.

**1.4. Dữ liệu đầu vào**

Tập dữ liệu MNIST chứa 70 000 hình ảnh về chữ số viết tay:

<http://yann.lecun.com/exdb/mnist/>

Tập dữ liệu gồm 600 ảnh Natural Images và 1000 ảnh Ground Truth (ảnh phân đoạn bằng tay), tất cả các ảnh đều có kích thước 481x321 pixel:

<https://people.eecs.berkeley.edu/~yang/software/lossy_segmentation/>

Tập dữ liệu ảnh vệ tinh thu thập từ:

<https://earthengine.google.com/timelapse/>

Tập dữ liệu gồm 2400 ảnh có chứa đám cháy được gán nhãn 1, và 1589 ảnh không có đám cháy được được gán nhãn 0.

**1.5. Dữ liệu đầu ra**

Đối với bài toán phân vùng ảnh vệ tinh: ảnh đầu ra sau khi được phân vùng sẽ được gán nhãn và tỷ lệ % diện tích các vùng ảnh, đánh giá và so sánh kết quả mô hình.

Với bài toán phát hiện đám cháy trong ảnh và video realtime: đầu ra sẽ là 1 ảnh hoặc tập các ảnh là các frame được cắt từ video đầu vào sau đó được gán nhãn là cháy hoặc không cháy.

**1.6. Công cụ lựa chọn**

Thuật toán sử lý ảnh được sử dụng để giải quyết bài toán là thuật toán phân cụm K-means, được cài đặt và triển khai trên môi trường hệ điều hành Windows. Ngôn ngữ để triển khai bài toán là python v2.7 và matlab v2015b cùng thư viện sử dụng là OpenCV, và các thư viện toán học khác, các thư viện này sẽ giúp việc triển khai các hàm toán học và xử lý ảnh trong python một cách dễ dàng hơn.

**1.7. Tóm tắt nội dung đồ án**

Đồ án bao gồm phụ lục và 5 chương:

Chương 1: Giới thiệu: Chương này trình bày về nhiệm vụ cần thực hiện trong khuôn khổ đồ án tốt nghiệp, trình bày định hướng giải quyết các vấn đề đặt ra. Giới thiệu tóm tắt về cơ sở lý thuyết và công cụ được lựa chọn để giải quyết các vấn đề đặt ra.

Chương 2: Unsupervised Learning và K-means Clustering: Chương này trình bày những lý thuyết cơ bản về Unsupervised Learning , cung cấp một cách nhìn tổng quát, và những vấn đề về học không giám sát, thuật toán phân cụm K-means Clustering.

Chương 3: Image Segmentation: Chương này chủ yếu trình bày việc áp dụng thuật toán phân cụm K-means và cải tiến cho bài toán phân vùng ảnh.

Chương 4: Phát hiện đám cháy trong ảnh và video realtime: Chương này tình bày các quy tắc để phát hiện và phân loại ảnh có chứa đám cháy, áp dụng cho bài toán thực tế là phát hiện đám cháy trong video thời gian thực qua webcam trên laptop.

Chương 5: Kết luận và định hướng phát triển: Ở chương này, chúng ta đưa ra kết luận tổng kết các kết quả ban đầu, khó khăn và thuận lợi của đồ án. Các định hướng phát triển tiếp theo.

Phụ lục: Ở phần này sẽ nói về cách cài đặt, chạy chương trình giải quyết bài toán.

**Chương 2: Unsupervised Learning và K-means Clustering**

**2.1. Học không giám sát - Unsupervised Learning**

Supervised learning (học có giám sát)

Chúng ta cần nhắc lại về học có giám sát Supervised learning. Như chúng ta đã biết Supervised learning là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên các cặp (input, outcome) đã biết từ trước. Cặp dữ liệu này còn được gọi là (data, label), tức (dữ liệu, nhãn). Supervised learning là nhóm phổ biến nhất trong các thuật toán Machine Learning. Về mặt toán học, Supervised learning là khi chúng ra có một tập hợp biến đầu vào **X =**  và một tập hợp nhãn tương ứng **Y =** , trong đó **,**  là các vector. Các cặp dữ liệu biết trước được gọi là tập *training data* (dữ liệu huấn luyện). Từ tập traing data này, chúng ta cần tạo ra một hàm số ánh xạ mỗi phần tử từ tập **X** sang một phần tử (xấp xỉ) tương ứng của tập **Y**:

**f(), i 1, 2, …, N**

Mục đích là xấp xỉ hàm số **f** thật tốt để khi có một dữ liệu **x** mới, chúng ta có thể tính được nhãn tương ứng của nó **y = f(x)**.

Thuật toán supervised learning còn được tiếp tục chia nhỏ ra thành hai loại chính:

Classification (Phân loại)

Một bài toán được gọi là *classification* nếu các *label* của *input data* được chia thành một số hữu hạn nhóm. Với **y** chỉ nhận giá trị từ một tập rời rạc, chẳng hạn {cá, cây, quả, mèo}. Ví dụ: Gmail xác định xem một email có phải là *spam* hay *non\_spam*.

Regression (Hồi quy)

Nếu *label* không được chia thành các nhóm mà là một giá trị thực cụ thể. Ví dụ: một căn nhà rộng **x** , có **y** phòng ngủ và cách trung tâm thành phố **z** *km* sẽ có giá là bao nhiêu? Với **y** nhận giá trị là số thực.

Unsupervised Learning (Học không giám sát)

Trong thuật toán này, chúng ta không biết được *outcome* hay nhãn *label* mà chỉ có dữ liệu đầu vào. Thuật toán *unsupervised learning* sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (*clustering*) hoặc giảm số chiều của dữ liệu (*dimension reduction*) để thuận tiện trong việc lưu trữ và tính toán.

Về mặt toán học, *unsupervised learning* có nghĩa là khi chúng ta chỉ có dữ liệu vào **X** mà không biết nhãn **Y** tương ứng. Máy tính cần học một hàm **y = f(x)** từ tập học cho trước . **Y** có thể là các cụm dữ liệu hoặc các cấu trúc ẩn.

Những thuật toán loại này được gọi là *unsupervised learning* vì không giống như *supervised learning*, chúng ta không biết câu trả lời chính xác cho mỗi dữ liệu đầu vào.

Các bài toán Unsupervised learning được tiếp tục chia nhỏ thành ba loại:

Clustering (phân nhóm)

Một bài toán phân nhóm toàn bộ dữ liệu X thành các nhóm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi nhóm. Ví dụ: phân nhóm khách hàng dựa trên hành vi mua hàng. Điều này cũng giống như việc ta đưa cho một đứa trẻ rất nhiều mảnh ghép với các hình thù và màu sắc khác nhau, ví dụ tam giác, vuông, tròn với màu xanh và đỏ, sau đó yêu cầu trẻ phân chúng thành từng nhóm. Mặc dù không cho trẻ biết mảnh nào tương ứng với hình nào hoặc màu nào, nhiều khả năng chúng vẫn có thể phân loại các mảnh ghép theo màu hoặc hình dạng.

Community detection: phát hiện các cộng đồng trong mạng xã hội.

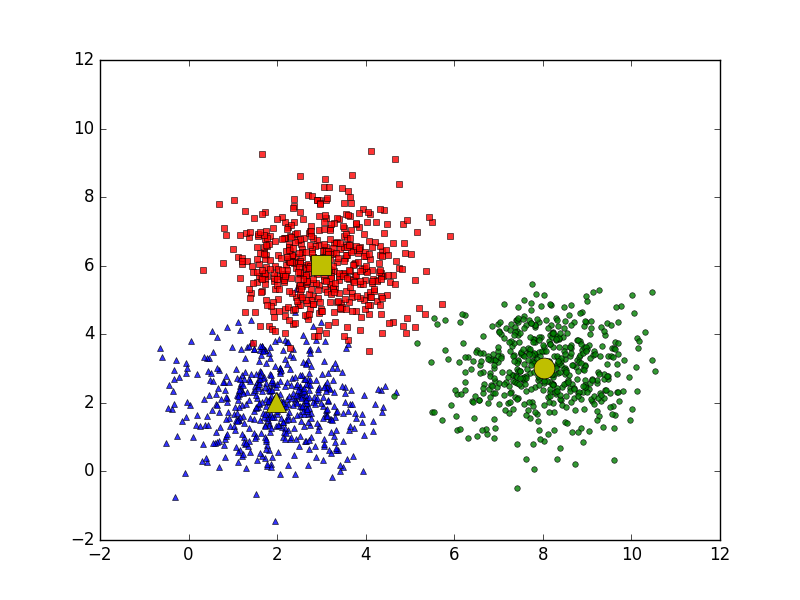
Trends detection: phát hiện xu hướng thị yếu.

**2.2. K-means Clustering**

**2.2.1. Giới thiệu**

Trong thuật toán K-means clustering, chúng ta không biết nhãn *label* của từng điểm dữ liệu. Mục đích là làm thế nào để phân dữ liệu thành các cụm *cluster* khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau.

Ý tưởng đơn giản nhất về *cluster* là tập hợp các điểm ở gần nhau trong một không gian nào đó (không gian này có thể có rất nhiều chiều trong trường hợp thông tin về một điểm dữ liệu là rất lớn). Hình bên dưới là một ví dụ về 3 cụm dữ liệu trong không gian 2 chiều.



Hình: Bài toán phân cụm với 3cluster.

**2.2.2. Phân tích toán học**

Mục đích cuối cùng của thuật toán phân nhóm này là: từ dữ liệu đầu vào và số lượng nhóm chúng ta muốn tìm, hãy chỉ ra *center* của mỗi nhóm và phân các điểm dữ liệu vào các nhóm tương ứng. Giả sử rằng mỗi điểm dữ liệu chỉ thuộc vào đúng một nhóm.

Giả sử có **N** điểm dữ liệu là **X =**  và **K N** là số cluster chúng ta muốn phân chia. Chúng ta cần tìm các center **, , …,** và label của mỗi điểm dữ liệu.

Với mỗi điểm dữ liệu đặt **=** là label vector của nó, trong đó nếu được phân vào *cluster* **k** thì **= 1** và **= 0**, . Điều này có nghĩa là có đúng một phần tử của vector là bằng 1 (tương ứng với cluster của ), các phần tử còn lại bằng 0. Ví dụ: nếu một điểm dữ liệu có *label* vector là thì nó thuộc vào *cluster* 1, nếu là thì nó thuộc vào *cluster* 2, .. Ràng buộc của có thể viết dưới dạng toán học như sau:

Hàm mất mát và bài toán tối ưu

Nếu ta coi center là center (hoặc representative) của mỗi *cluster* và ước lượng tất cả các điểm được phân vào *cluster* này bởi , thì một điểm dữ liệu được phân vào cluster ***k*** sẽ bị sai số là . Ta mong muốn sai số này có trị tuyệt đối nhỏ nhất nên sẽ tìm cách để đại lượng sau đây đạt giá trị nhỏ nhất:

Hơn nữa, vì được phân vào *cluster* ***k*** nên **= 1, = 0,** . Khi đó, biểu thức bên trên sẽ được viết lại là:

Sai số cho toàn bộ dữ liệu sẽ là:

Trong đó **Y =** , **M =** lần lượt là các ma trận được tạo bởi *label* vector của mỗi điểm dữ liệu và *center* của mỗi *cluster*. Hàm số mất mát trong bài toán K-means clustering là hàm với ràng buộc như được nêu trong phương trình .

Ta cần tối ưu bài toán sau:

Ta cần tối ưu 2 biến số Y và M để hàm mất mát *đạt giá trị nhỏ nhất*.

Thuật toán tối ưu hàm mất mát

Bài toán là một bài toán khó tìm điểm tối ưu vì nó có thêm các điều kiện ràng buộc. Bài toán này thuộc loại *mix-integer programming* (điều kiện biến là số nguyên) - là loại rất khó tìm nghiệm tối ưu toàn cục (*global optimal point*, tức nghiệm làm cho hàm mất mát *đạt giá trị nhỏ nhất* có thể). Tuy nhiên, trong một số trường hợp ta vẫn có thể tìm được phương pháp để tìm được nghiệm gần đúng hoặc điểm cực tiểu. Điểm cực tiểu ở đây chưa chắc đã phải là điểm làm cho hàm số đạt giá trị nhỏ nhất (cực tiểu toàn cục), mà nó có thể là điểm cực tiểu địa phương (điểm cực tiểu cục bộ).

Một cách đơn giản để giải bài toán là xen kẽ giải Y và M khi biến còn lại được cố định làm tham số.

Cố định M và tìm Y

Giả sử đã tìm được các centers, hãy tìm các label vector để hàm mất mát đạt giá trị nhỏ nhất. Điều này tương đương với việc tìm cluster cho mỗi điểm dữ liệu.

Khi các centers là cố định, bài toán tìm label vector cho toàn bộ dữ liệu có thể được chia nhỏ thành bài toán tìm label vector cho từng điểm dữ liệu như sau:

Vì chỉ có một phần tử của *label* *vector* bằng 1 nên bài toán có thể tiếp tục được viết dưới dạng đơn giản hơn:

Vì chính là bình phương khoảng cách tính từ điểm đến *center* , ta có thể kết luận rằng mỗi điểm thuộc vào *cluster* có *center* gần nó nhất! Từ đó ta có thể dễ dàng suy ra *label vector* của từng điểm dữ liệu.

Cố định Y và tìm M

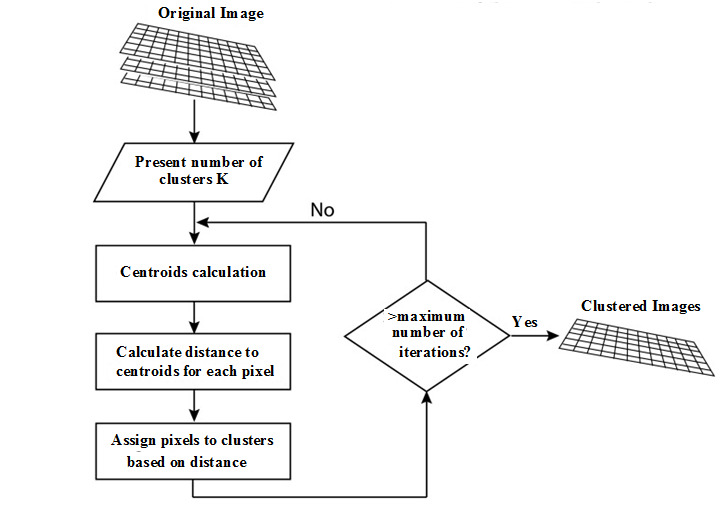
Giả sử đã tìm được cluster cho từng điểm, hãy tìm center mới cho mỗi cluster để hàm mất mát đạt giá trị nhỏ nhất. Một khi chúng ta đã xác định được label vector cho từng điểm dữ liệu, bài toán tìm center cho mỗi cluster được rút gọn thành:

Lúc này, ta có thể tìm nghiệm bằng phương pháp giải phương trình đạo hàm bằng 0, vì hàm cần tối ưu là một hàm liên tục và có đạo hàm xác định tại mọi điểm. Và quan trọng hơn, hàm này là hàm lồi (convex) theo nên chúng ta sẽ tìm được giá trị nhỏ nhất và điểm tối ưu tương ứng.

Giải phương trình đạo hàm bằng 0 ta có:

Ở đây mẫu số chính là phép đếm số lượng các điểm dữ liệu trong cluster ***j***. Còn tử số chính là tổng các điểm dữ liệu trong cluster ***j***. Nghĩa là chính là trung bình cộng của các điểm trong cluster ***j***.

**2.2.3. Thuật toán K-means**



Hình: Sơ đồ khối thuật toán K-means cho bài toán phân vùng ảnh.

Đầu vào: Dữ liệu X và số cụm cluster cần tìm K.

Đầu ra: Các center M và label vector cho từng điểm dữ liệu Y.

1. Chọn K điểm bất kỳ làm các center ban đầu.
2. Phân mỗi điểm dữ liệu vào cluster có center gần nó nhất.
3. Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước nó thì ta dừng thuật toán.
4. Cập nhật center cho từng cluster bằng cách lấy trung bình cộng của tất các các điểm dữ liệu đã được gán vào cluster đó sau bước 2.
5. Quay lại bước 2.

Thuật toán sẽ dừng lại sau một số hữu hạn vòng lặp (luôn hội tụ sau một số hữu hạn lần lặp). Vì hàm mất mát là một số dương và sau mỗi bước 2 hoặc 3, giá trị của hàm mất mát bị giảm đi. Số lượng cách phân nhóm cho toàn bộ dữ liệu là hữu hạn nên đến một lúc nào đó, hàm mất mát sẽ không thể thay đổi, và khi đó thuật toán sẽ dừng.

**Chương 3: Image Segmentation**

Trước khi đi vào bài toán chính, tôi sẽ đi vào một bài toán cũng rất thực tế và cơ bản sau, để có thể thấy rõ hơn cách biểu diễn dữ liệu và xây dựng mô hình cho bào troán phân vùng ảnh với thuật toán phân cụm K-means clusstering.

**3.1. Phân nhóm chữ số viết tay**

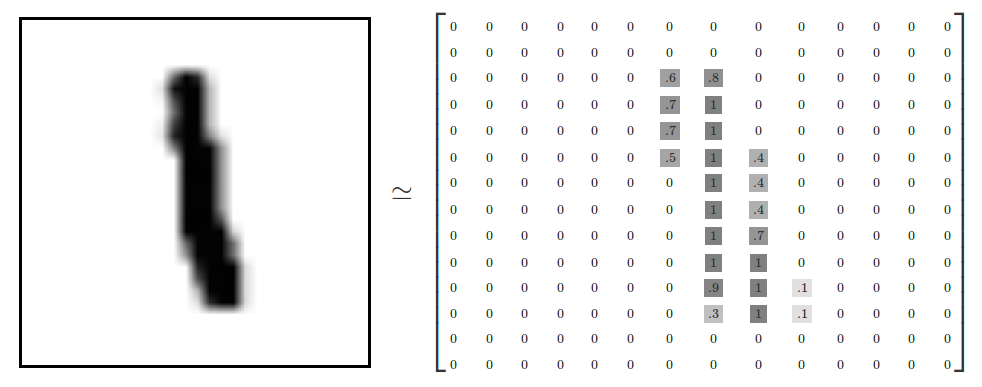
**3.1.1. Biểu diễn dữ liệu**

Trước hết ta đi vào bài toán Image Classification cho bộ cơ sở dữ liệu chữ số viết tay MNIST gồm hai tập con: tập dữ liệu huấn luyện (*training\_set*) có tổng cộng 60000, và tập dữ liệu thử nghiệm (*test\_set*) có tổng cộng 10000 ví dụ khác nhau. Tất cả đều đã được gán nhãn.



Hình: MNIST bộ cơ sở dữ liệu chữ số viết tay.

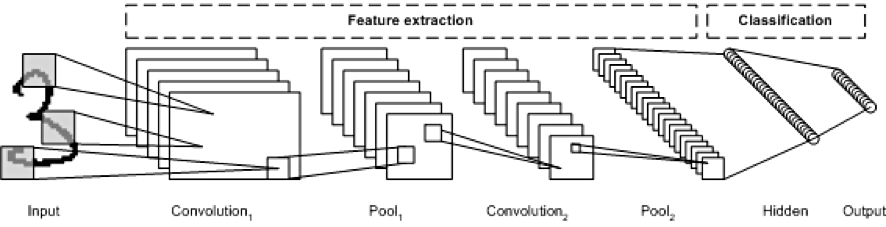
Mỗi bức ảnh là một ảnh đen trắng (có 1 channel), có kích thước 28x28 pixel. Khi đó mỗi bức ảnh sẽ là 1 vector 1x784, mà mỗi phần tử của vector là một pixel mang một giá trị là một số tự nhiên từ 0 đến 255. Các pixel màu đen có giá trị bằng 0, các pixel càng trắng thì có giá trị càng cao. Tất cả các chữ số được đặt trên nền đen, và chữ số màu sáng. Dưới đây là một ví dụ về chữ số 1 và giá trị các pixel của nó (ảnh nền đã được thay đổi thành nền trắng).



Hình: Biểu diễn ảnh đen trắng dưới dạng ma trận pixel.

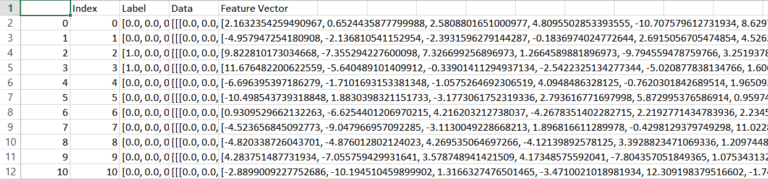
Ta coi mỗi bức ảnh là một điểm dữ liệu. Và mỗi điểm dữ liệu là 1 vector có kích thước 1x784 biểu diễn 1 bức ảnh chữ số.

Ở đây chúng ta sẽ sử dụng các vector đặc trưng được trích xuất bởi các mô hình CNN để phân cụm dữ liệu MNIST:



Hình: Sơ đồ kiến trúc mạng LeNet.

Trích xuất đặc trưng từ ảnh thu được các vector đặc trưng và lưu vào file .csv có dạng như sau:



Hình: Mô tả vector đặc trưng trích xuất từ ảnh lưu thành file \*.csv.

**3.1.2. K-Means clustering of MNIST data**

Tối ưu số cụm K

Trước khi áp dụng thuật toán K-means clustring cho bộ dữ liệu ta cần dự đoán số cụm K thích hợp để có được kết quả tốt nhất.

Phương pháp khuỷu tay (Elbow): ý tưởng là đối với một phạm vi các giá trị thích hợp của K, ta tính sai số của tổng lỗi trong cụm (wcss - within-cluster sum of errors) để tìm ra giá trị K tối ưu nhất. Ta có thể tìm thấy K ở vị trí khuỷu tay, tức là trước khuỷu tay lỗi giảm nhanh dần và sau khi qua khuỷu tay lỗi giảm không đáng kể.

Phương pháp Average Silhouette: phương pháp này thể hiện tính nhất quán trong các *cluster*, giá trị *silhouette* là thước đo mức độ tương tự của một đối tượng với cụm (sự gắn kết) của chính nó so với các cụm khác (sự tách biệt) hay mức độ phân tách giữa các cụm. Silhouette nằm trong phạm vi −1 đến +1, trong đó giá trị cao biểu thị rằng đối tượng được kết hợp tốt với cụm của chính nó và tách rời với các cụm lân cận. Nếu hầu hết các đối tượng có giá trị cao, thì cấu hình phân cụm là phù hợp. Nếu nhiều điểm có giá trị thấp hoặc âm, thì cấu hình phân cụm có thể có quá nhiều hoặc quá ít cụm. Silhouette được tính như sau:

Tính khoảng cách trung bình từ tất cả các điểm dữ liệu trong cùng một cụm ().

Tính khoảng cách trung bình từ tất cả các điểm dữ liệu trong cụm gần nhất ().

Tính hệ số:

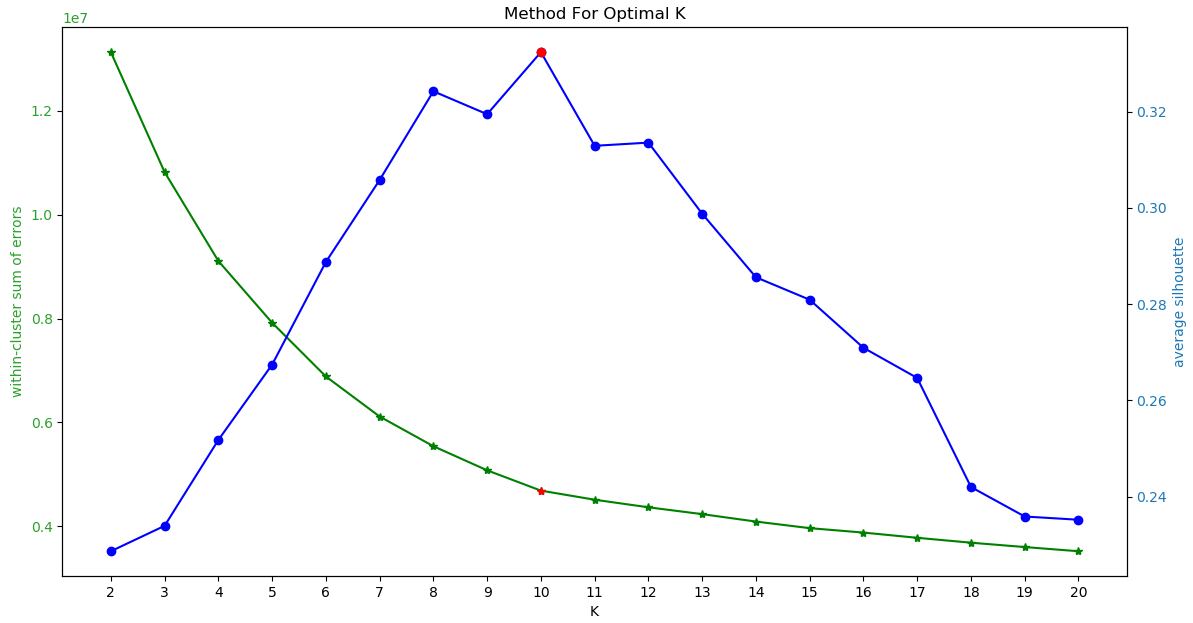
Hoặc là

Nếu mẫu i rất gần với các cụm lân cận.

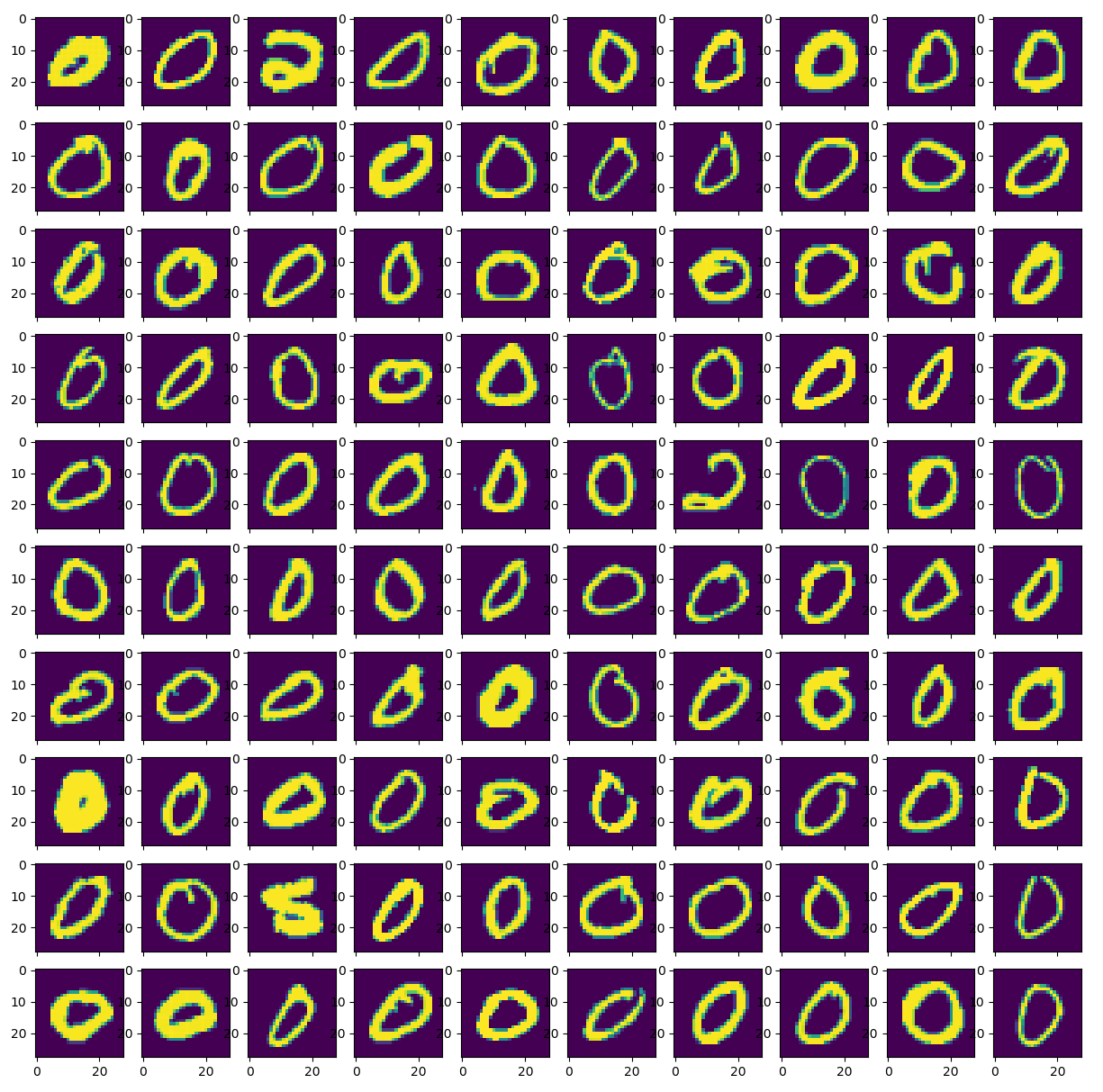
Nếu mẫu nằm cách xa các cụm lân cận.

Nếu mẫu được gán cho các cụm sai.

Dưới đây là kết quả khi thực hiện tối ưu số cụm K sử dụng 2 phương pháp Average Silhouette và Elbow, kết quả khá rõ cho thấy khi chọn số cụm K = 10 là tốt nhất.



Triển khai thuật toán K-means clustering cho bộ dữ liệu features matrix ta được kết quả:



Đánh giá độ chính xác

Trong đó: là số chữ số giống nhau nhiều nhất trong cụm *j*

K là số cụm, N tổng số bức ảnh, là số bức ảnh thuộc cụm *j*.

Tôi thực hiện chia số lượng những bức ảnh chữ số được phân đúng (số lượng nhiều nhất trong 1 cụm) chia cho tổng các chữ số trong cụm đó. Trong hình trên, hiển thị 1 cụm có kích thước 100 chữ số, gồm có 97 chữ số 0 và 3 chữ số bị phân nhóm sai là 2 chữ số 2 và 1 chữ số 5. Như vậy độ chính xác của cụm này là 97%. Dưới đây là kết quả phân loại trên toàn bộ tập dữ liệu:

Bảng: Kết quả độ chính xác khi phân loại chữ số viết tay MNIST sử dụng K-means

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cluster | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Accuracy (%) | 94.68 | 93.54 | 80.69 | 91.56 | 96.71 | 92.66 | 90.10 | 86.82 | 94.84 | 85.47 |
| Average (%) | 90.71 | | | | | | | | | |
| Time  (s) | 4.640 | | | | | | | | | |

Nhận thấy độ chính xác trung bình là 90.71%, kết quả này ở mức khá tốt khi thực hiện phân loại chữ số viết tay bằng phương pháp K-meas clustering. So với các kết quả từ các bài báo khác sử dụng mô hình CNN sẽ có độ chính xác cao từ 95-98% với cùng 1 tập dữ liệu MNIST, thì đó là kết quả rất tốt đối với 1 thuật toán học không giám sát.

**3.2. Image Segmentation và cải tiến K-means**

**3.2.1. Bài toán phân vùng ảnh màu (color image segmentation)**

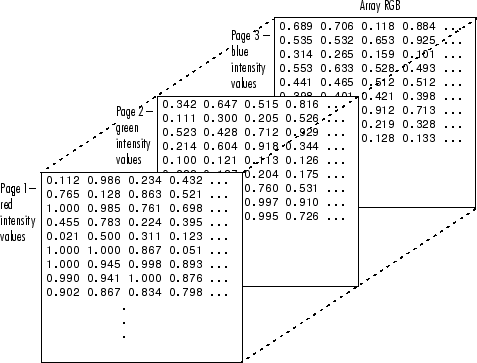
Thuật toán K-means đã được sử dụng rất nhiều trong nghiên cứu và được cài đặt trong các phần mềm xử lý ảnh viễn thám. Tuy nhiên, khi phân vùng ảnh viễn thám kích thước lớn, tốc độ hội tụ của thuật toán vẫn rất chậm. Trong đồ án này, tôi đề xuất một cải tiến thuật toán phân cụm K-Means, cải tiến việc dự đoán tối ưu số cụm K và khởi tạo tâm cụm ban đầu hiệu quả, kết hợp với bước tiền xử lý ảnh đầu vào nhằm tăng tốc độ phân vùng ảnh viễn thám có kích thước lớn, và cho kết quả đầu ra tốt đối với ảnh bị nhiễu.

**3.2.2. Biểu diễn dữ liệu**

Ý tưởng cơ bản để phân đoạn hình ảnh bằng thuật toán phân cụm K-means là tạo các cụm dựa trên giá trị màu của mỗi pixel, vì thế độ chói (hay độ sáng) của mỗi pixel sẽ bị bỏ qua, ta chỉ quan tâm đến các thành phần về mầu của pixel. Chính vì thế nên trong phần này, ta áp dụng phân cụm K-means cho ảnh trên không gian mầu L\*a\*b\*. Trong không gian màu L\*a\*b\* ta loại bỏ thành phần về độ sáng L, và chỉ giữ lại 2 thành phần về màu là kênh màu a và b. Lúc này kích thước dữ liệu đầu vào giảm, giúp cho thuật toán chạy nhanh hơn và phân vùng ảnh chính xác hơn. Sau đó ta cẩn *reshape* lại kích thước của ma trận ảnh từ ma trận 3 chiều sang ma trận 2 chiều. Ảnh số thực chất là một ma trận các điểm ảnh. Đối với ảnh màu, mỗi điểm ảnh sẽ có 3 thuộc tính tương ứng với 3 chiều dữ liệu. Hai chiều về không gian chính là tọa độ hay vị trí của pixel đó trong ma trận điểm ảnh, chiều thứ ba là giá trị về màu sắc (màu ở đây có thể là không gian màu RGB) của pixel.

Trong đó: *h, w* là kích thước và *c* là số kênh màu của ảnh *I*.

Mỗi điểm ảnh lúc này sẽ là 1 hàng của ma trận A. ma trận A sẽ có hàng (ứng với tổng số pixel trong ảnh), và có *c* cột (c = 3 nếu là không gian màu RGB).



Hình: Biểu diễn ảnh màu RGB bằng ma trận 2 chiều.

Công thức chuyển đổi từ hệ màu RGB sang hệ màu L\*a\*b\*:

Trong đó:

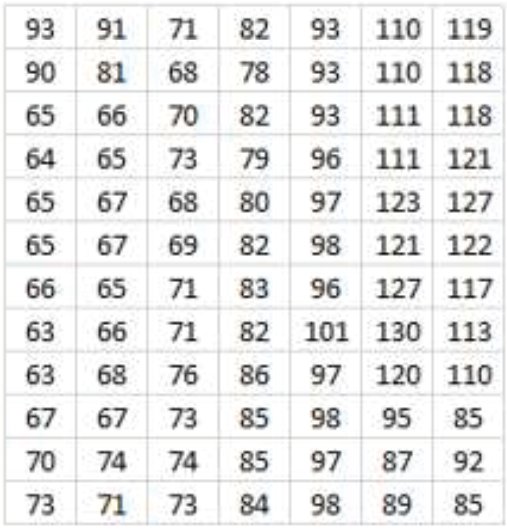
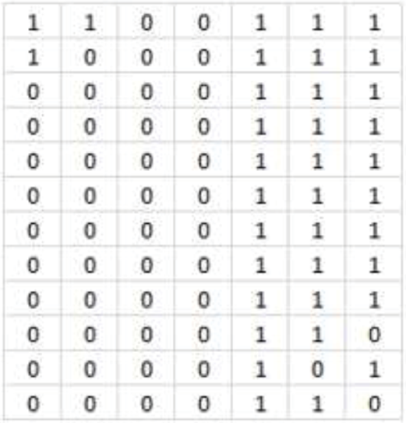
và

**3.2.3. Tiền xử lý ảnh**

Là quá trình sử dụng các kỹ thuật xử lý ảnh để nâng cao chất lượng ảnh. Phương pháp sử dụng SMQT (Successive Mean Quantization Transform). Đơn vị cơ bản của SMQT là MQU (Mean Quantization Unit) trong đó bao gồm việc tính giá trị trung bình của tất cả các pixel trong ảnh, sau đó giá trị trung bình được sử dụng để định lượng giá trị của dữ liệu thành 0 hoặc 1, tùy thuộc vào giá trị của pixel thấp hơn hoặc cao hơn giá trị trung bình.

MQU:

Pixel (V) Mean

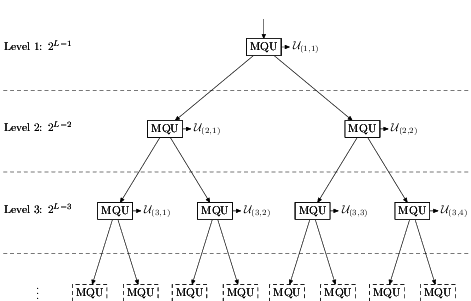
Hình: Đơn vị lượng tử trung bình MQU (Mean Quantization Unit).

Mean = 87.5595

SMQT có thể được coi là một cây của các phép tính MQU, trong đó trọng số

được đưa ra tùy thuộc vào mức độ hiện tại của cây.

Trong đó L là mức tối đa của cây, i là mức hiện tại.



Hình: Cấu trúc cây trong SMQT.

Trong ảnh RGB, SMQT có thể được áp dụng theo hai cách khác nhau. Đầu tiên, áp dụng SMQT cho ba kênh RGB. Cho là tất cả các giá trị dữ liệu bất kể kênh nào, thì:

Điều này sẽ dẫn đến sự tăng cường độ tương phản phi tuyến bảo tồn thứ tự của các giá trị RGB cho mỗi pixel, nhưng với khoảng cách thay đổi giữa các giá trị R, G và B trong mỗi pixel. Cách thứ hai là áp dụng nó một cách riêng biệt trong mỗi ba kênh. Đặt , và là tất cả các giá trị dữ liệu của kênh màu R, G và B tương ứng, thì:

Cuối cùng, các pixel được tăng cường thu được bằng cách ghép các kênh màu , và . Dưới đây là hình ảnh sau quá trình nâng cao chất lượng ảnh bằng phương pháp SMQT:

Bảng: Kết quả ảnh sau tăng cường

|  |  |  |  |
| --- | --- | --- | --- |
| Ảnh gốc |  |  |  |
| Ảnh tăng cường | D:\workspace\matlab\ImageSegmentation_FireDetection\output_images\pre_2.jpg | D:\workspace\matlab\ImageSegmentation_FireDetection\output_images\pre_copy.jpg |  |

**3.2.4. Xây dựng mô hình**

Hiệu năng của thuật toán K-means bị ảnh hưởng rất nhiều ở bước khởi tạo trung tậm K cụm đầu tiên. Tùy vào các center ban đầu mà thuật toán có thể có tốc độ hội tụ rất chậm, hoặc thậm chí cho chúng ta nghiệm không chính xác (chỉ là local minimum - điểm cực tiểu địa phương - mà không phải giá trị nhỏ nhất). Đó là một nhược điểm lớn của K-means cho bài toán phân đoạn ảnh nói riêng và phân cụm nói chung. Vì thế ta sẽ đi khắc phục nhược điểm này để có được khởi tạo trung tâm K ban đầu tốt nhất.

Một nhược điểm nữa ta sẽ cần khắc phục đó là K-means cho kết quả không tốt đối với các hình ảnh bị nhiễu, hay có chất lượng thấp. Vì thế trước khi thực hiện phân cụm ảnh với K-means, ta sẽ nâng cao chất lượng ảnh đầu vào, tăng cường đặc trưng cho các điểm ảnh.

Thuật toán K-means được chứng minh là hội tụ và có độ phức tạp tính toán là O(tkn)  
với **t** là số lần lặp, **k** là số cụm, **n** là số đối tượng của tập dữ liệu vào. Thông thường  
**k<<n** và **t<<n** thường kết thúc tại một điểm tối ưu cục bộ.

Công việc cải tiến phương pháp cho bài toán phân vùng ảnh màu sử dụng K-means clusstering:

1. Đọc ảnh màu RGB.
2. Nâng cao chất lượng ảnh dùng phương pháp SMQT.
3. Chuyển đổi hình ảnh từ định dạng RGB sang Lab. (Trong không gian L\*a\*b loại bỏ kênh độ sáng L, chỉ lấy 2 kênh màu a\* và b\*)
4. Biểu diễn dữ liệu: reshape ảnh thành ma trận hai chiều. Vì ảnh ban đầu ở không gian 3D, ta chuyển về không gian 2D. (sử dụng công thức 12)
5. Áp dụng phương pháp Elbow và Silhouette để tối ưu số cụm K từ ma trận dữ liệu.
6. Sắp xếp ma trận dữ liệu ma trận hình ảnh theo thứ tự không giảm, chia toàn bộ dữ liệu hình ảnh được sắp xếp thành K phần (K đã được dự đoán ở bước 4).
7. Khởi tạo K trung tậm cụm đầu tiên: Tìm giá trị trung bình của từng phần và giá trị trung bình đó trở thành trọng tâm cho cụm tương ứng và lưu lại trung tậm cụm ban đầu.
8. Áp dụng thuật toán phân cụm K-means cho ma trận dữ liệu hình ảnh với trung tâm cụm ban đầu và số cụm K đã tối ưu ở trên.

**3.2.5. Phương pháp đánh giá**

* Sai số bình phương trung bình: Mean Squared Error (MSE)

Trong đó: – input image, I – segmented image,

M, N là kích thước của ảnh I.

Giá trị MSE càng nhỏ thì ảnh đầu ra sẽ có chất lượng càng tốt.

* Tỷ lệ nhiễu tín hiệu cực đại: Peak Signal-to-Noise Ratio (PSNR)

Trong đó: là giá trị pixel lớn nhất của ảnh I.

Giá trị PSNR cao hơn tương ứng với một hình ảnh chất lượng tốt hơn.

* Chỉ số tương đồng về cấu trúc: Structural Similarity Index Metric (SSIM)

Trong đó:

N là tổng số pixel, *x* – segmented image, *y* – input image.

**3.2.6. Kết quả thử nghiệm**

Các thuật toán phân cụm có thể khó đánh giá mà không có trước thông tin hoặc giả định về dữ liệu. Vì vậy trước khi thực hiện thuật toán phân cụm, tôi đã thực hiện phân tích và tối ưu số cụm K từ dữ liệu ảnh đầu vào.

Bảng: Đánh giá kết quả thử ngiệm trên bộ dữ liệu ảnh *truth*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ảnh | Cluster K | Time(s) | MSE | MAE | SNR | PSNR | SC | SSIM |
| 1 | 2 | 8.76 | 227.94 | 11.27 | 66.94 | 24.55 | 1.02 | 0.93 |
| 2 | 2 | 3.79 | 158.62 | 9.86 | 93.23 | 26.13 | 1.01 | 0.97 |
| 3 | 4 | 15.44 | 828.63 | 23.47 | 22.92 | 18.95 | 1.05 | 0.59 |
| 4 | 4 | 25.40 | 455.28 | 17.10 | 43.39 | 21.55 | 1.02 | 0.71 |
| 5 | 4 | 19.62 | 897.62 | 22.70 | 17.48 | 18.60 | 1.03 | 0.49 |
| 6 | 3 | 21.68 | 670.10 | 18.68 | 29.37 | 19.87 | 1.03 | 0.61 |
| 7 | 4 | 13.10 | 425.14 | 15.77 | 42.56 | 21.85 | 1.02 | 0.63 |

Bảng: So sánh kết quả phân vùng ảnh bằng KM\* và ảnh Truth Image

|  |  |  |
| --- | --- | --- |
| Ảnh gốc | Truth Image | KM\* |
| D:\download\CTM-mean-0.2\MDL-adaptive-0.2\3096.jpg |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Bảng: So sánh kết quả phương pháp đề xuất KM\* và kết quả trên bài báo khoa học

|  |  |  |
| --- | --- | --- |
| Origin Image |  |  |
| Truth Image |  |  |
| K-means |  |  |
| Paper |  |  |
| KM\* |  |  |

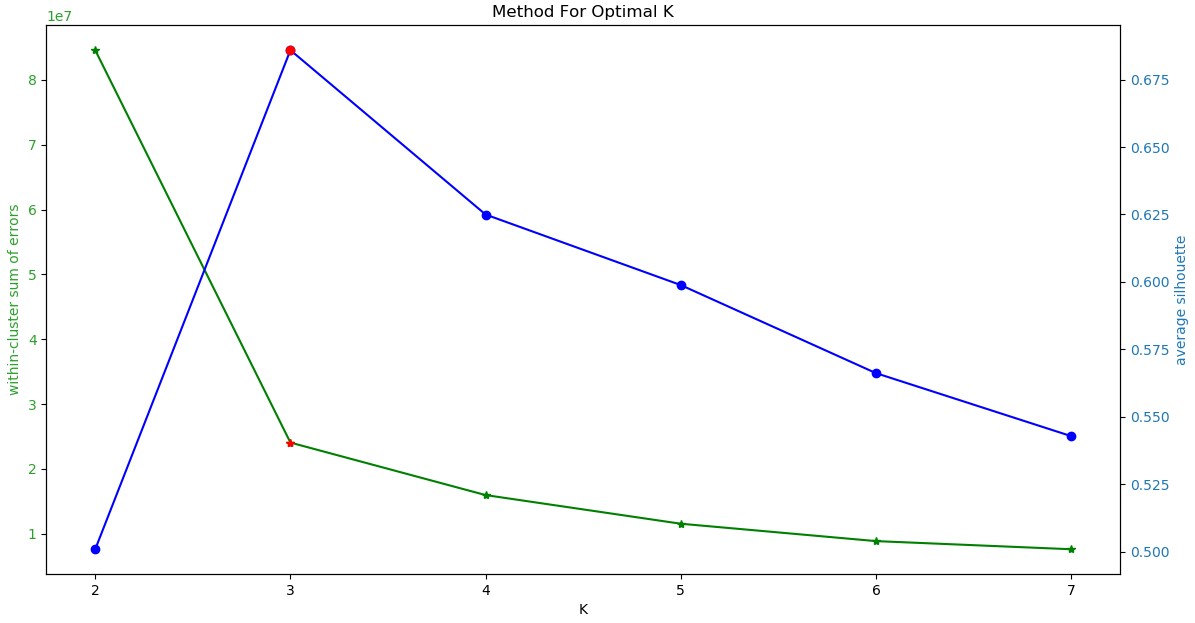
Bảng: So sánh kết quả của phương pháp đề xuất KM\* và thuật toán K-means.

|  |  |  |
| --- | --- | --- |
| Origin Image | K-means | KM\* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Tôi sẽ sử dụng hình ảnh dưới đây để thấy rõ nhất sự khác biệt thay đổi của phương pháp đề xuất KM\* so với phương pháp K-means cơ bản. Thực hiện áp dụng phương pháp đề xuất và thuật toán K-means để tự động nhận ra vùng khuôn mặt, vùng khăn, vùng hậu cảnh và tách chúng ra. Bằng trực quan chúng ta cũng có thể dự đoán được bức ảnh có 3 màu chủ đạo: hồng ở khăn và môi; đen ở mắt, tóc, và hậu cảnh; màu da ở vùng còn lại của khuôn mặt, điều đó tương ứng với 3 cluster.



Để chắc chắn, tôi thực hiện tối ưu số cụm K cho hình ảnh trên sử dụng 2 phương pháp Elbow và Average Silhouette, kết quả được biểu diễn bằng đồ thị dưới đây.

****

Hình: Phương pháp tìm số cụm tối ưu bằng đồ thị.

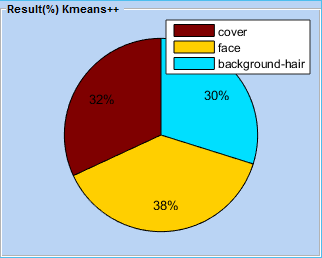
Khi số cụm K nằm trong khoảng [2,7], tại vị trí khuỷu tay (K=3) giá trị tổng lỗi trong cụm sẽ giảm chậm dần theo phương pháp Elbow, và cũng tại vị trí đó giá trị Average Silhouette là lớn nhất. Khi đó K=3 sẽ là số cụm tối ưu nhất cần tìm.

Bảng: Tìm số cụm tối ưu

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| K | 2 | 3 | 4 | 5 | 6 | 7 |
| wcss | 0.5009 | 0.6858 | 0.6249 | 0.5988 | 0.5662 | 0.5428 |
| silhouette | 8.4587 | 2.4055 | 1.5958 | 1.1538 | 0.8857 | 0.7603 |

Bảng: So sánh kết quả chi tiết phương pháp đề xuất KM\* và thuật toán K-means

|  |  |  |
| --- | --- | --- |
| Ảnh gốc | K-means | KM\* |
|  |  |  |
| Cover (%) | 35 | 32 |
| Face (%) | 33 | 38 |
| Background-hair (%) | 32 | 30 |
| Time (s) | 7.152 | 4.922 |
| Iterations | 14 | 9 |

Hình: Biểu đồ so sánh kết quả đề xuất KM\* và thuật toán K-means.

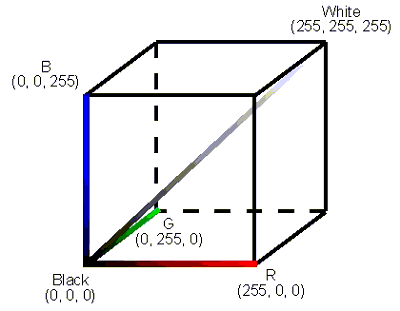
Từ các số liệu đánh giá và kết quả trên, nhận thấy phương pháp được đề xuất cho kết quả khá tốt, và tốt hơn so với phương pháp K-means truyền thống. Các hình ảnh được phần vùng bằng phương pháp đè xuất có độ chi tiết và chính xác cao hơn phương pháp K-means.

**Chương 4: Phát hiện đám cháy trong Ảnh và Video**

**4.1. Biểu diễn dữ liệu**

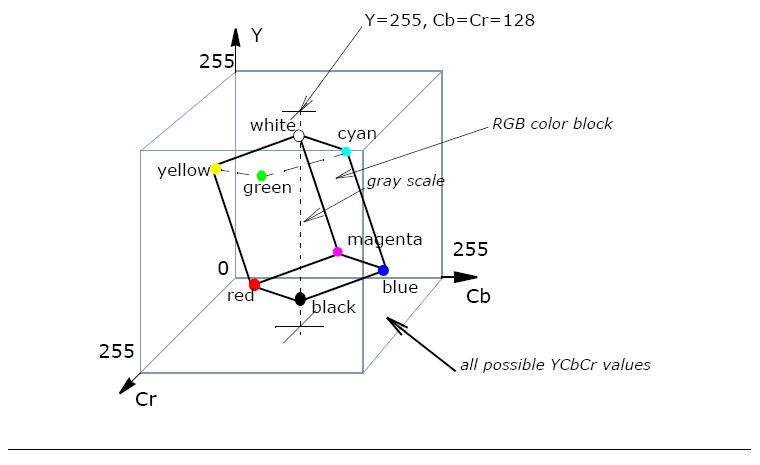
Sử dụng không gian màu RGB và YCbCr, ưu điểm khi sử dụng không gian màu YCbCr là nó có thể tách độ sáng hiệu quả hơn không gian màu RGB.

Một ảnh số trong không gian màu RGB có 3 mặt phẳng R, G, B. Các sự kết hợp giữa các mặt phẳng màu RGB cho phép các thiết bị hiển thị biểu diễn màu trong môi trường kỹ thuật số. Mỗi mặt phẳng màu được lượng hóa thành các mức rời rạc. Nói chung có 256 (8 bit cho mỗi mặt phẳng màu) lượng tử hóa được sử dụng cho mỗi mặt phẳng, ví dụ màu trắng được biểu thị bằng (R, G, B) = (255,255,255) và màu đen được biểu thị bằng (R, G, B) = (0, 0, 0). Một ảnh màu bao gồm nhiều pixel, trong đó mỗi pixel được biểu thị bằng vị trí không gian trong lưới hình chữ nhật (x, y) và vector màu (R (x, y), G (x, y), B (x, y)) tương ứng với vị trí không gian (x, y) là vị trí tọa độ của pixel.



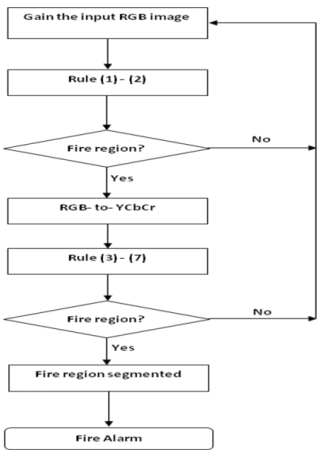
Hình: Không gian màu RGB.

Công thức chuyển đổi từ hệ màu RGB sang hệ màu YCbCr:



Hình: Không gian màu YCbCr.

**4.2. Xây dựng mô hình**



Hình: Mô hình phát hiện đám cháy trong ảnh.

*Một điểm ảnh là điểm ảnh cháy chỉ khi nó thỏa mãn đồng thời 7 quy tắc sau đây:*

**Quy tắc 1**

Có thể nhận thấy các vùng cháy trong ảnh, kênh R có giá trị cường độ cao hơn kênh G và kênh G có giá trị cường độ cao hơn giá trị cường độ kênh B.

Hình ảnh được phân vùng các điểm chớp cháy của nó như hình trên với kênh màu G. Sau đó chúng ta tính giá trị trung bình của các kênh màu R, G và B tại vị trí các vùng lửa được phân nhận dạng từ ảnh gốc. Vì vậy, đối với pixel tại vị trí tọa độ (x, y) nếu là pixel lửa thì quy tắc bên sau đây phải được thỏa mãn với pixel này.

**Quy tắc 2**

Xác định một số giá trị ngưỡng cho điểm ảnh lửa:

Khi hình ảnh được chuyển đổi từ không gian màu RGB sang YCbCr, cường độ và không gian màu sẽ dễ ràng phân biệt. Điều này giúp việc nhận dạng vùng lửa dễ dàng trong không gian màu YCbCr.

Trong đó: Y là độ sáng, Cb và Cr là các thành phần màu Blue và màu Red .

Giá trị trung bình của ba thành phần Y, Cb và Cr trên toàn bộ bức ảnh được biểu thị bằng Ymean, Cbmean và Crmean tương ứng được tính như sau:

Trong đó: (x, y) là vị trí tọa độ của pixel trong ảnh, M × N

là tổng số pixel trong hình ảnh đã cho.

**Quy tắc 3 và 4**

**Quy tắc 5**

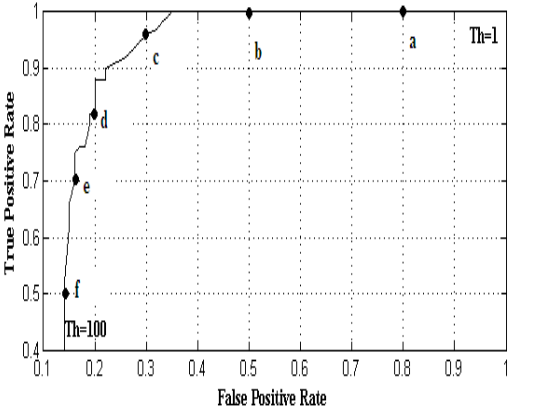
Vùng ngọn lửa nói chung là vùng sáng nhất trong bối cảnh nhìn thấy của bức ảnh hay camera. Giá trị trung bình trên toàn bộ tấm ảnh của ba kênh Ymean, Cbmean và Crmean chứa thông tin có giá trị. Đối với vùng ngọn lửa, giá trị của thành phần Y là lớn hơn thành phần Y trung bình trên toàn bộ bức ảnh. Trong khi giá trị của thành phần Cb tại vùng lửa nhỏ hơn giá trị Cb trung bình của trên toàn bộ bức ảnh. Hơn nữa, thành phần Cr của vùng lửa lớn hơn Cr trung bình trên toàn bộ bức ảnh:

Trong đó: chỉ ra rằng bất kỳ điểm ảnh nào thỏa mãn điều kiện được đưa ra trong phương trình trên được coi là điểm ảnh cháy.

**Quy tắc 6**

Có sự khác biệt đáng kể giữa Cb và các thành phần Cr của các điểm ảnh cháy. Đối với pixel lửa, thành phần Cb chủ yếu là "đen" (cường độ thấp hơn) trong khi thành phần Cr chủ yếu là "trắng" (cường độ cao hơn). Thực tế này có thể được dịch sang một quy tắc khác như sau:

Giá trị của Th được chọn sao cho tỷ lệ phát hiện vượt quá 95% và tỷ lệ báo động giả nhỏ hơn 30% (điểm c) tương ứng với Th = 70.



Hình: Giá trị ngưỡng cho tỷ lệ phát hiện cháy > 95% và tỷ lệ báo cháy giả < 30%.

**Quy tắc 7**

Chúng ta đã xác định một số giá trị ngưỡng cho điểm ảnh lửa. Ta chỉ xét giá trị ngưỡng cho 2 kênh màu Cb và Cr, không xét kênh độ sáng Y vì nó là thành phần độ sáng và nó phụ thuộc vào điều kiện chiếu sáng.

Một điểm ảnh là pixel vùng lửa nếu thỏa mãn đồng thời 7 quy tắc trên.

Bảng: Quy tắc phát hiện điểm ảnh cháy

|  |  |
| --- | --- |
| Color space | Rules |
| RGB | 1. RGB |
| 1. R Rmean G Gmean B Bmean |
| YCbCr | 1. Y(x,y) Cb(x,y) |
| 1. Cr(x,y) Cb(x,y) |
| 1. Y(x,y) Ymean Cb(x,y) Cbmean Cr(x,y) Crmean |
| 1. , *(Th = 70)* |
| 1. (Cb(x,y) 120) (Cr(x,y) 150) |

**4.3. Kết quả thử nghiệm và đánh giá**

Tập dữ liệu kiểm thử gồm 2 tập con, tập *test\_fire* chứa 2400 ảnh có đám cháy được gán nhãn 1 và tập *not\_fire* chứa 1589 ảnh không có đám cháy được gán nhãn 1.

|  |  |
| --- | --- |
| Origin Image | Fire Detection |
|  |  |
|  |  |
|  |  |

**Chương 5: Kết luận và định hướng phát triển**

5.1. Kết luận

5.2. Định hướng phát triển

**Phát hiện vùng khói**

Tương tự như phát hiện vùng lửa, ta có thể mô hình hóa các điểm ảnh khói. Nhưng điểm ảnh khói không hiển thị các đặc điểm màu sắc như pixel lửa. Lúc đầu, khi nhiệt độ khói thấp, dự kiến khói sẽ hiển thị từ màu trắng-xanh đến trắng. Về sau nhiệt độ của khói tăng lên và màu chuyển từ màu xám đen đến màu đen. Vì vậy, tôi có thể xây dựng các điểm ảnh khói như sau:

*Th*

*Th*

*Th*

Trong đó Th là ngưỡng toàn cầu dao động từ 15 đến 25, các điểm ảnh khói nên có tương tự cường độ trong các kênh màu RGB của chúng. Vì thông tin khói sẽ được sử dụng cho hệ thống phát hiện cháy sớm nhất, nên phát hiện các mẫu khói khi khói có nhiệt độ thấp. Đây là trường hợp, mà các mẫu khói có màu từ trắng-xanh đến trắng, có nghĩa là độ bão hòa của màu sắc nên càng thấp càng tốt. Sử dụng ý tưởng này, quy tắc sau được sử dụng khi không gian màu HSV được sử dụng:

S(x, y) 0.1

Để phân biệt được khói và các hình ảnh tương tự như sương mù hay mây trời, ta sẽ phát hiện sự chuyển động của khói và lửa trong video đám cháy.