**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**──────── \* ───────**

**PROJECT 2**

**Tìm hiểu một số thuật toán và mô hình học máy, học sâu**

**Tên: Lê Thành Chỉnh**

**Lớp: 727524**

**Giáo viên hướng dẫn: Trịnh Văn Loan**

**Mục lục**

[**Chương 1: Linear Regression 4**](#_Toc143000654)

[**1.Linear Regression là gì? 4**](#_Toc143000655)

[**2. Bài toán hồi quy: 5**](#_Toc143000656)

[**3. Triển khai thuật toán: 7**](#_Toc143000657)

[**3.1 Dữ liệu sử dụng: 7**](#_Toc143000658)

[**3.2 Cross-validation: 7**](#_Toc143000659)

[**3.3 Triển khai: 8**](#_Toc143000660)

[**3.4 Tiền sứ lý dữ liệu: 13**](#_Toc143000661)

[**Chương 2: Kmeans 21**](#_Toc143000662)

[**1. Kmeans là gì? 21**](#_Toc143000663)

[**2. Thuật toán: 22**](#_Toc143000664)

[**3. Triển khai: 23**](#_Toc143000665)

[**Chương 3: Multi-layer Perceptron 31**](#_Toc143000666)

[**1. Tensorflow 31**](#_Toc143000667)

[**2. Multi-layer Perceptron 34**](#_Toc143000668)

[**2.1 Multi-layer Perceptron là gì? 34**](#_Toc143000669)

[**2.2 Triển khai: 35**](#_Toc143000670)

[**Tài liệu tham khảo 44**](#_Toc143000671)

**Lời mở đầu**

**Trong thời đại của cuộc cách mạng công nghệ 4.0, lĩnh vực học máy và học sâu đã nhanh chóng trở thành những nguồn cảm hứng sáng tạo và phát triển không ngừng cho cộng đồng khoa học và công nghệ. Những khả năng vượt bậc của các thuật toán và mô hình trong lĩnh vực này đã góp phần tạo ra những ứng dụng đột phá, từ phân tích dữ liệu đến trí tuệ nhân tạo, từ y học đến ô tô tự hành.**

**Đồ án này được thực hiện nhằm mục đích tìm hiểu và nắm vững một số thuật toán và mô hình quan trọng trong học máy và học sâu. Việc nắm bắt sâu sắc về cách hoạt động, ưu điểm và hạn chế của các thuật toán và mô hình này sẽ là chìa khóa để áp dụng chúng một cách hiệu quả vào các bài toán thực tế.**

**Qua quá trình nghiên cứu và thực hành, chúng tôi đã đạt được cái nhìn sâu hơn về sự phức tạp và sự thú vị của học máy và học sâu. Sự kết hợp giữa kiến thức lý thuyết và kỹ năng thực tế trong việc xây dựng và đánh giá các mô hình đã mang lại cho chúng tôi nhiều bài học quý báu.**

**Chúng tôi xin chân thành cảm ơn sự hướng dẫn và hỗ trợ của giáo viên hướng dẫn cũng như sự động viên từ bạn bè và gia đình trong suốt thời gian thực hiện đồ án này. Hy vọng rằng bài viết sau đây sẽ truyền tải được phần nào kiến thức và cảm xúc mà chúng tôi đã trải qua trong hành trình tìm hiểu về học máy và học sâu.**

**Trân trọng**

# Chương 1: Linear Regression

## 1.Linear Regression là gì?

**Linear Regression là một phương pháp trong thống kê và máy học để mô hình hóa mối quan hệ tuyến tính giữa các biến. Đây là một phương pháp cơ bản và quan trọng trong việc dự đoán và phân tích dữ liệu. Dưới đây là một số điểm cơ bản về lý thuyết Linear Regression:**

**Khái niệm: Linear Regression (Hồi quy tuyến tính) là một phương pháp thống kê để tìm một mô hình tuyến tính giữa biến độc lập (X) và biến phụ thuộc (Y) sao cho mối quan hệ này có thể được sử dụng để dự đoán giá trị của Y dựa trên giá trị của X.**

**Công thức: Mô hình Linear Regression thường được biểu diễn bằng công thức: Y = β₀ + β₁X + ε, trong đó Y là biến phụ thuộc, X là biến độc lập, β₀ là hệ số chặn của đường thẳng hồi quy, β₁ là hệ số góc của đường thẳng hồi quy, và ε biểu thị sai số ngẫu nhiên.**

**Mục tiêu: Mục tiêu của Linear Regression là tìm các hệ số β₀ và β₁ sao cho tổng bình phương của sai số ε là nhỏ nhất (phương pháp bình phương tối tiểu).**

**Phân tích sai số: Có nhiều cách để đánh giá mô hình Linear Regression, bao gồm SSE (Sum of Squared Errors), R-squared (hệ số xác định), và nhiều chỉ số khác.**

**Tiến hóa: Linear Regression có thể mở rộng để xử lý các biến đa chiều (Multiple Linear Regression), và cũng có thể áp dụng các biến đổi để tạo ra các phiên bản phức tạp hơn như Polynomial Regression.**

**Giả định: Một số giả định của Linear Regression bao gồm: sự tương quan tuyến tính giữa các biến, độc lập tuyệt đối giữa các sai số (điều kiện không tương quan), phân phối chuẩn của sai số, và độc lập của sai số.**

**Áp dụng: Linear Regression được sử dụng rộng rãi trong nhiều lĩnh vực, bao gồm kinh tế học, thống kê, khoa học xã hội, khoa học dữ liệu và machine learning.**

**2. Bài toán hồi quy:**

**Cho tập dữ liệu D = {(x1, y1), (x2, y2), …, (xN, yN)}, trong đó mỗi**

**điểm dữ liệu (xi, yi) bao gồm 2 thành phần:**

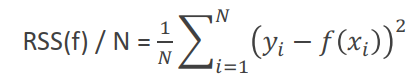
**xi = [xi1, xi2, …, xiK]T là một vector K chiều**

**yi ∈ ℝ là một số thực**

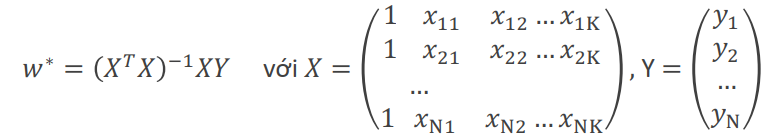
**Giả thiết rằng tồn tại hàm f tuyến tính sao cho yi ≅ f(xi):**

**f(xi) = w0 + w1xi1 + … + wK xiK = wxi**

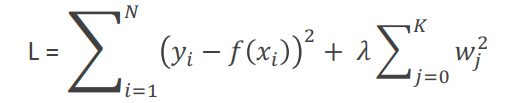
**Lỗi trên tập dữ liệu D:**

****

**Nghiệm w\* tối thiểu hóa L:**

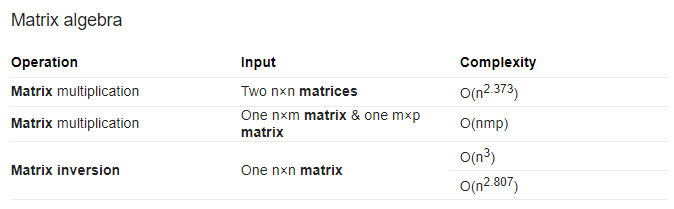
****

**Ridge Regression: thêm vào đại lượng phạt 𝜆 ||𝑤 ||22 vào RSS(f)**

****

**Minimize L ta có w\* lúc này là:**

****

****

**Khi kích thước dữ liệu quá lớn việc tính nghịch đảo ma trận quá tốn kém. Các giải quyết là tối ưu sử dụng stochastic gradient**

**Các lược đồ tối ưu dựa vào gradient:**

****

**∇w𝐿 = 𝐸q [𝐵( 𝑤)} , lấy mẫu ngẫu nhiên data từ phân phối q và gọi b(w) là**

**gradient trên tập mẫu:**

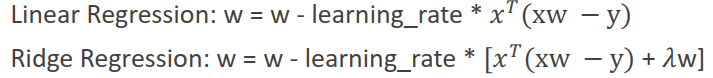
**w = w – learning\_rate\*𝑏(𝑤) . b là lấy mẫu độc lập từ B.**

**Việc tối ưu theo stochastic gradient đảm bảo tính hội tụ và về mặt thực nghiệm cho kết quả tốt hơn trên các hàm non-convex so với gradient thông thường.**

**Hiểu đơn giản là ta chia dữ liệu thành các minibatch rồi tối ưu parameter theo gradient của minibatch đó. Lặp lại trên dữ liệu nhiều epoch**

**Có thể sử dụng các phương pháp tối ưu dựa trên gradient để minimize hàm lỗi. Áp dụng khi kích thước dữ liệu quá lớn việc tính nghịch đảo ma trận quá tốn kém**

**Lược đồ tối ưu:**

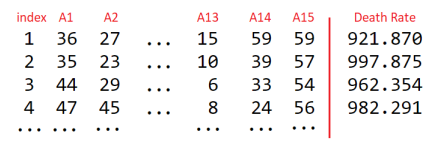
****

**3. Triển khai thuật toán:**

**3.1 Dữ liệu sử dụng:**

**Có tất cả 60 điểm dữ liệu, mỗi điểm dữ liệu có 15 thuộc tính và 1**

**giá trị death rate tương ứng**

****

**3.2 Cross-validation:**

**Một tập dữ liệu D thường có 2 phần: Dtrain và Dtest**

***Dtrain dùng để huấn luyện mô hình***

***Dtest để đánh giá hiệu quả của mô hình***

**Cross-validation (k-fold cross-validation) dùng để lựa chọn tham số cho mô hình (với ridge regression, đó là giá trị LAMBDA 𝜆).**

**Áp dụng 5-fold cross-validation vào việc lựa chọn LAMBDA**

**5-fold cross-validation được tiến hành như sau:**

* **Chia Dtrain thành 5 phần (xấp xỉ) bằng nhau: D1, D2, D3, D4, D5**
* **Với mỗi Di (i = 1,2,3,4,5), ta thực hiện:**
  + **Huấn luyện mô hình trên Dtrain \ Di**
  + **Tính lỗi trên Di**
* **Tính lỗi trung bình qua 5 lần**
* **Lựa chọn LAMBDA đem lại lỗi trung bình nhỏ nhất.**
* **Huấn luyện mô hình trên toàn bộ Dtrain với LAMBDA tìm được và đánh giá hiệu quả mô hình trên Dtest**

**3.3 Triển khai:**

**Triển khai thuật toán Ridge Regression (trường hợp tổng quát của Linear Regression)**

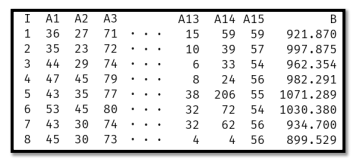
**> Đọc dữ liệu**

**> Chuẩn hóa dữ liệu**

**> Xây dựng mô hình**

**Lựa chọn LAMBDA theo phương pháp cross-validation**

**Đọc dữ liệu:**

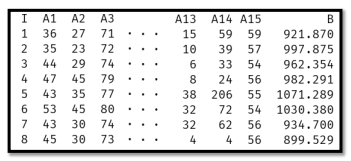
**> Đọc file**

**> Chia nội dung thành từng dòng**

**> Chia mỗi dòng thành các features**

**> X: features từ A1 –> A15**

**> Y: feature cuối cùng, B**

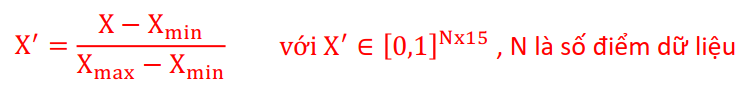
**Chuẩn hóa dữ liệu:**

**> Các features có miền giá trị lệch nhau**

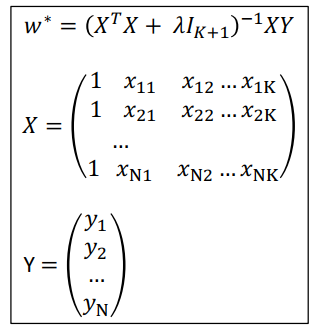
**> Chuẩn hóa để đưa về 1 miền chung**

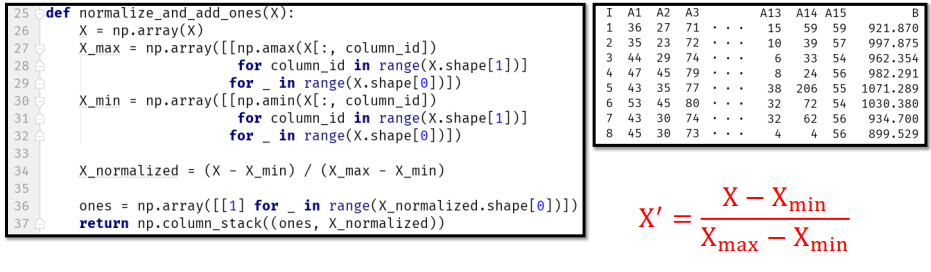
**> Có nhiều phương pháp, ta chọn**

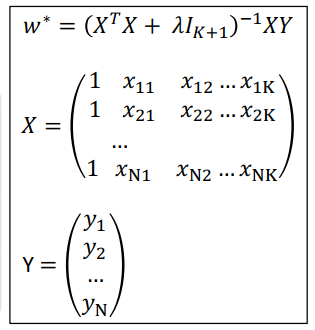
**“Feature Scaling”:**

****

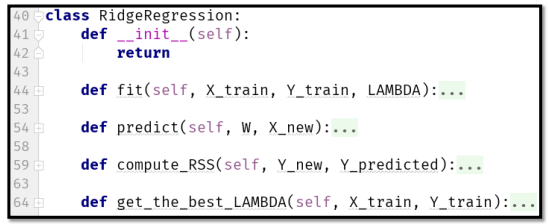
**=> Ta cần thêm feature xi0 = 1 vào mỗi điểm dữ liệu**

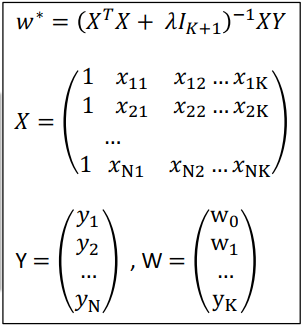
****

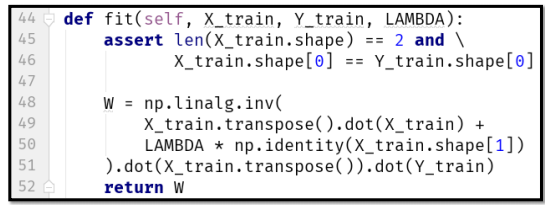
****

**Triển khai mô hình:**

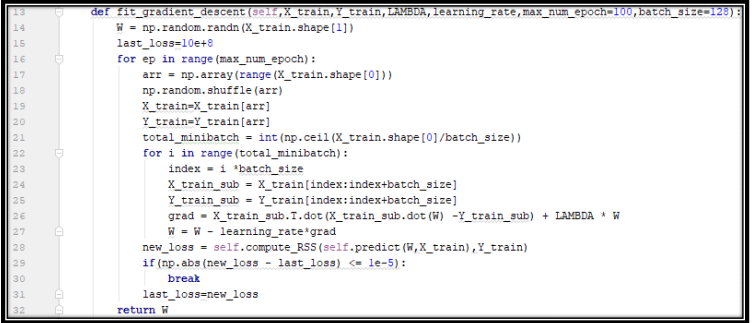
**> Xây dựng lớp RidgeRegression**

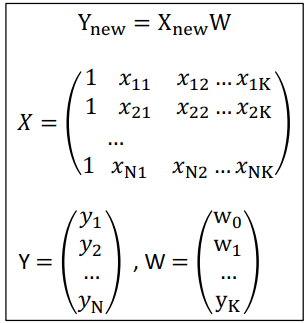
****

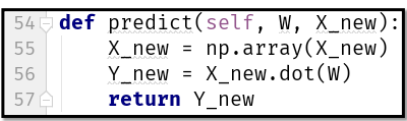
**> Hàm fit**

****

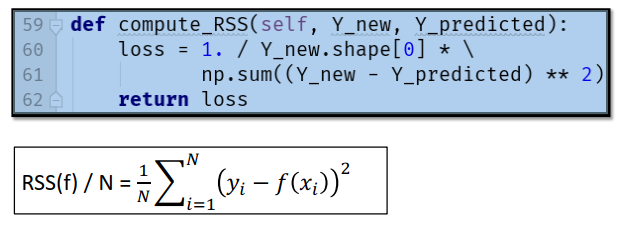
**> Hàm fit\_gradient**

****

**> Hàm predict**

****

**> Hàm compute\_RSS**

****

**> Xác định giá trị LAMBDA tốt nhất:**

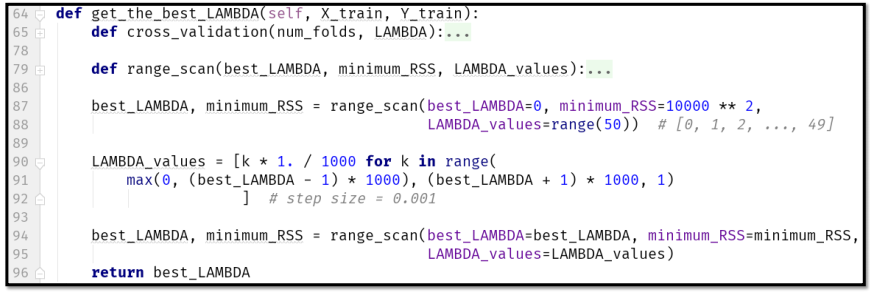
**\* B1: Xác định miền giá trị tìm kiếm**

**\* B2: Thực hiện cross-validation với từng giá trị LAMBDA có thể**

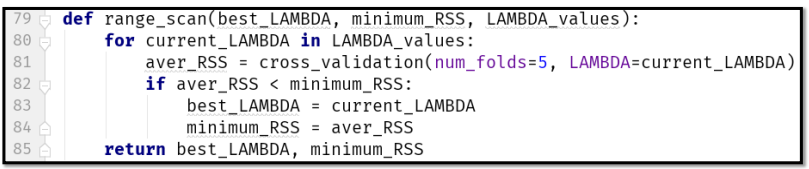
**\* B3: Xác định giá trị LAMBDA tốt nhất trong miền**

**\* B4: Quay trở lại bước 1**

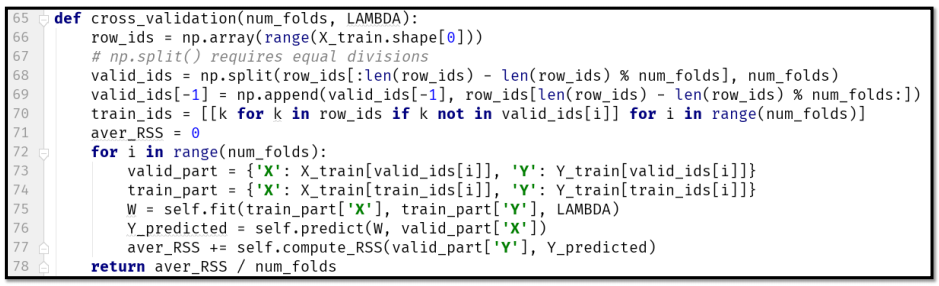
**> Hàm get\_the\_best\_LAMBDA**

****

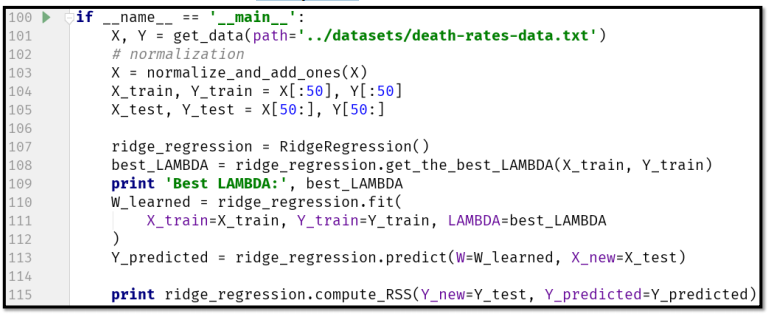
**>Hàm range\_scan**

****

**>Hàm cross\_validation**

****

**Chạy thử**

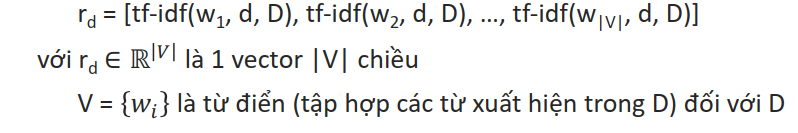
****

### 3.4 Tiền sứ lý dữ liệu:

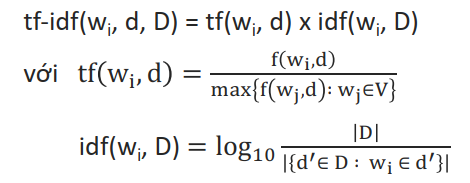
**3.4.1 Biểu diễn Bag of words và TF-IDF cho doc:**

**TF-IDF = term frequency–inverse document frequency**

**Được sử dụng cho dữ liệu dạng văn bản (text)**

**Biểu diễn TF-IDF đối với 1 văn bản d trong một tập văn bản (corpus) D:**

**Trong đó, mỗi giá trị tf-idf(wi, d, D) được tính như sau:**

****

**Trong đó, f(wi, d) là số lần xuất hiện của từ wi trong văn bản d.**

**Biểu diễn TF-IDF**

**Xác định từ điển V:**

**Với mỗi văn bản d trong D:**

**\* B1: Tách d thành các từ theo punctuations ta thu được Wd:**

**‘Data-Science Lab;2018’ -> [‘Data’, ‘Science’, ‘Lab’, ‘2018’]**

**\* B2: Loại bỏ từ dừng (stop words) khỏi Wd :**

**Wd = Wd\ {stop\_words}**

**a, an, the, have, for, ….**

**\* B3: Đưa các từ về dạng gốc (stemming) :**

**Wd = {stem(w) : w ∈ Wd }**

**trong đó stem(w) là dạng gốc của w**

**Một cách xác định từ điển V:**

**Với mỗi văn bản d trong D: thu được Wd**

**Cuối cùng, ta có:**

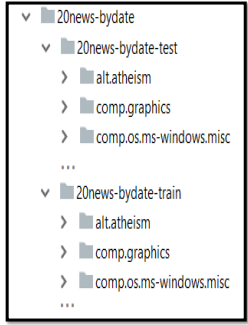
**V = ⋃d∈D Wd**

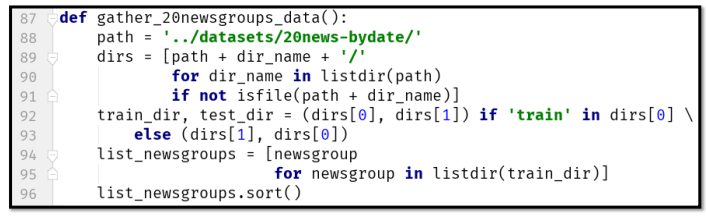
**Tập dữ liệu sử dụng: 20newsgroups**

**Bao gồm xấp xỉ 20,000 bài báo, thuộc 20 nhóm tin tức khác nhau.**

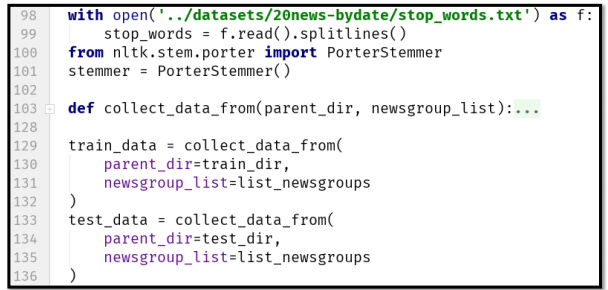
**cấu trúc cây thư mục:**

****

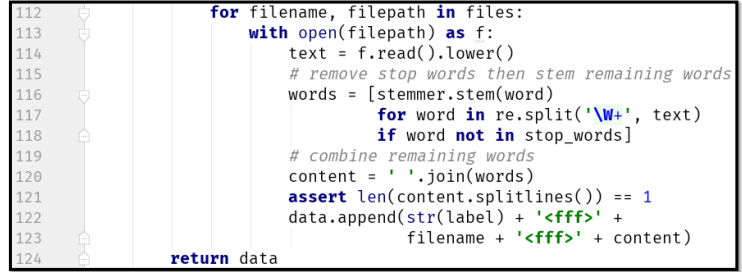
**Đọc dữ liệu và tập hợp dữ liệu:**

**> Lấy danh sách các thư mục và newsgroups**

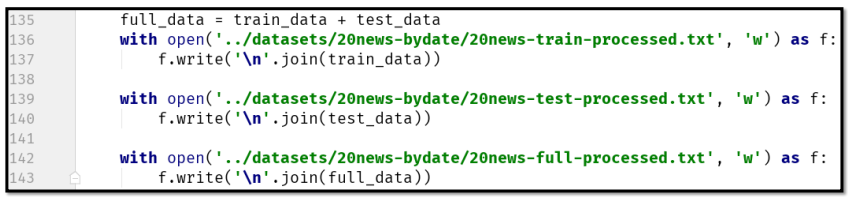
**> Thu thập dữ liệu**

****

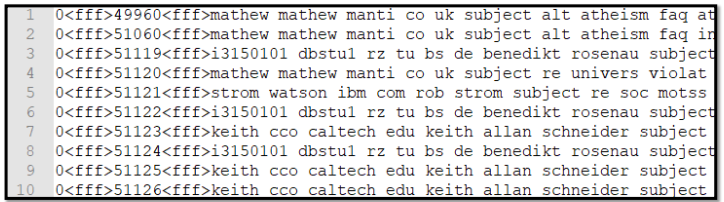
**> Thu thập dữ liệu: hàm collect\_data\_from**

****

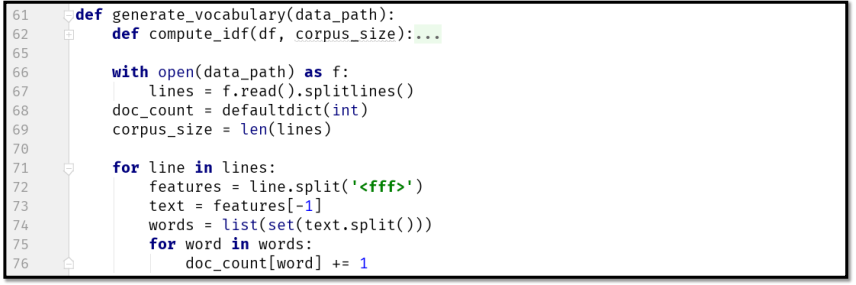
**> Ghi ra file:**

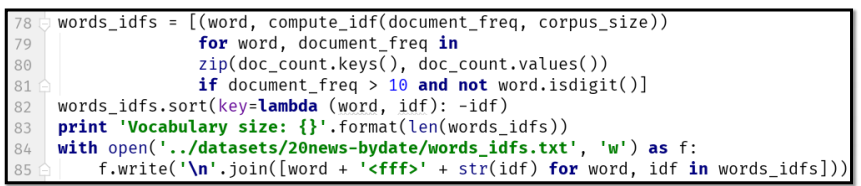
****

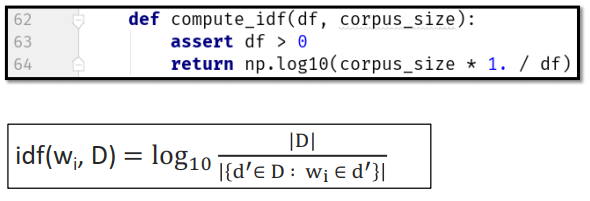
**> Output: 20news-train-processed.txt**

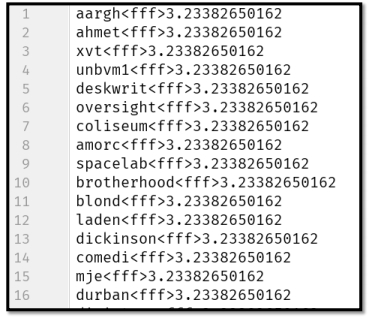
****

**>tạo từ điển và tính trước giá trị idf**

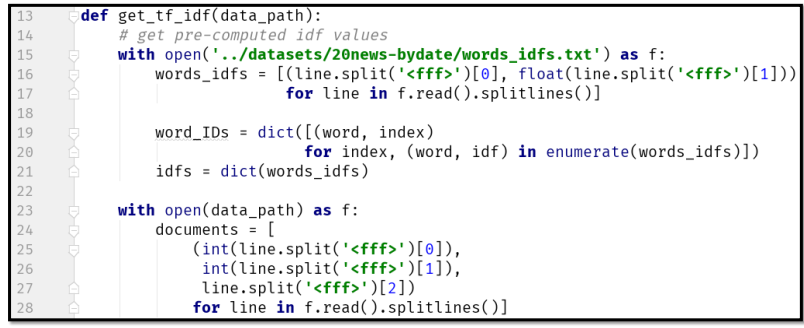
****

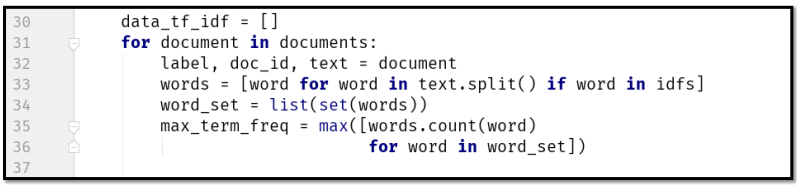
****

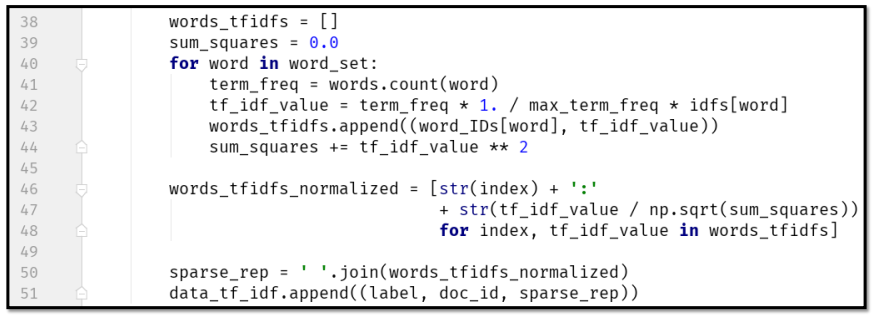
****

****

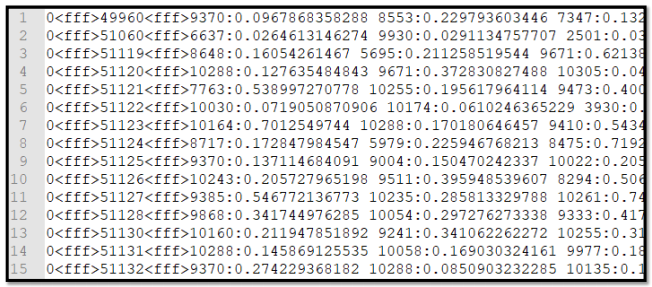
**>tính tf-idf**

****

****

****

**>Ghi data\_tf\_idf ra file**

****

**3.4.2 Word2vec: Biểu diễn vector cho từ:**

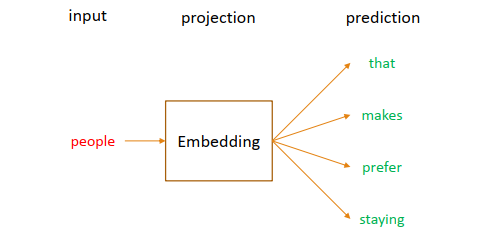
**Để thu được biểu diễn word2vec của từ, có 2 mô hình:**

**1. Skip-Gram**

**2. CBOW (Continuous Bag-of-Word Model)**

**3.4.2.1. Skip-Gram:**

**Sử dụng center word làm input và context words làm target**

****

**3.4.2.2 CBOW**

**Sử dụng context words làm input và center word làm target**

****

**Sau khi huấn luyện, thu được ma trận Word Embedding, mỗi hàng là một vector biểu diễn cho một từ.**

**Lưu ý: ma trận Word Embedding cũng thay đổi khi training**

**Word Embedding thường là tầng đầu tiên trong rất nhiều mô hình Deeplearning hiện nay**

# Chương 2: Kmeans

1. **Kmeans là gì?**

**K-means là một thuật toán trong lĩnh vực máy học và thống kê được sử dụng để phân cụm dữ liệu. Thuật toán này giúp tách một tập dữ liệu thành các nhóm (clusters) dựa trên các đặc điểm tương tự của các điểm dữ liệu. Mục tiêu của K-means là tìm ra các điểm trung tâm của các nhóm sao cho tổng bình phương khoảng cách từ các điểm dữ liệu tới điểm trung tâm gần nhất là nhỏ nhất.**

**Cách hoạt động của K-means như sau:**

**Khởi tạo các điểm trung tâm ban đầu của các nhóm (thường là ngẫu nhiên hoặc dựa trên kiến thức tiền định).**

**Gán từng điểm dữ liệu vào nhóm có điểm trung tâm gần nhất.**

**Cập nhật lại điểm trung tâm của mỗi nhóm dựa trên các điểm dữ liệu đã được gán vào nhóm đó.**

**Lặp lại bước 2 và 3 cho đến khi không có sự thay đổi đáng kể trong việc gán điểm dữ liệu và cập nhật điểm trung tâm.**

**Kết quả cuối cùng của thuật toán là các nhóm dữ liệu có đặc điểm tương tự được phân chia dựa trên sự tối ưu hóa khoảng cách giữa điểm dữ liệu và điểm trung tâm của nhóm. K-means thường được sử dụng trong nhiều ứng dụng khác nhau như phân cụm dữ liệu, nén ảnh, phân loại và khám phá dữ liệu.**

1. **Thuật toán:**

**Input:**

**> Tập dữ liệu R = {rd : d ∈ D} với rd ∈ ℝ|V| là biểu diễn tf-idf của d**

**> Số cụm K**

**Output: A = {ad : d ∈ D} với ad ∈ {1, 2, …, K} cho biết d được phân vào**

**cụm nào**

**Procedure:**

**> B1: Khởi tạo tâm cho K cụm:**

**E = { ek } với ek là tâm của cụm k ,**

**k ∈ {1, 2, …, K} và |E| = K ,**

**E là 1 tập con gồm K phần tử được lấy từ R = {rd : d ∈ D}**

**> B2: Lặp cho tới khi hội tụ:**

**\* Với mỗi d ∈ D :**

**+ Tính similarity(rd , ek)**

**+ Gán d vào cụm k\* với k\* = argmaxk(similarity(rd , ek))**

**\* Cập nhật lại E**

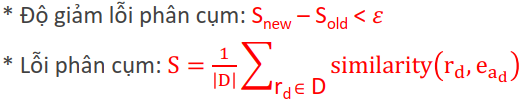
**Lựa chọn điều kiện dừng:**

**> Số bước lặp vượt quá 1 ngưỡng đặt trước: iteration > max\_iters**

**> E = { ek } thay đổi không đáng kể:**

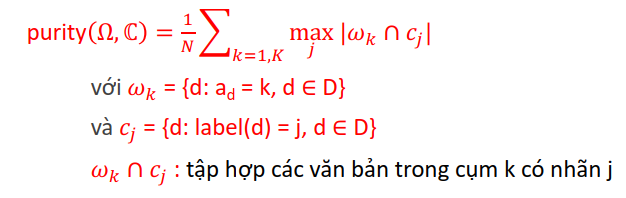
**|Enew \ Eold| < n0 với n0 << K**

**> Độ tương đồng trung bình không tăng hoặc tăng không đáng kể**

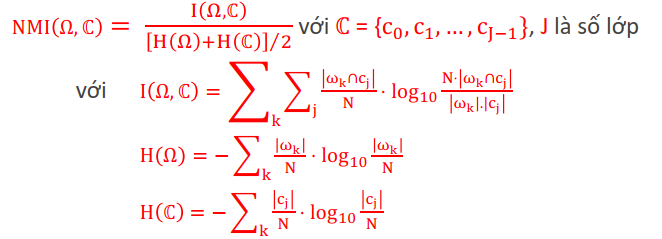
****

**Đánh giá chất lượng phân cụm:**

**> Purity:**

****

**> NMI (normalized mutual information ):**

****

1. **Triển khai:**

**20newsgroups dataset đã tiền xử lý:**

**Mỗi cụm ta lưu trữ các thông tin sau:**

**> centroid: tâm cụm**

**> members: danh sách các điểm dữ liệu trong cụm**

**Mỗi điểm dữ liệu d ta sẽ lưu trữ thông tin sau:**

**> r\_d: biểu diễn tf-idf rd của văn bản d**

**> label: newsgroup của văn bản d**

**> doc\_id: tên file chứa văn bản d**

**Ta sẽ xây dựng 3 lớp:**

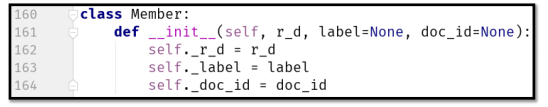
**> 2 lớp cho lưu trữ thông tin:**

**\* class Cluster**

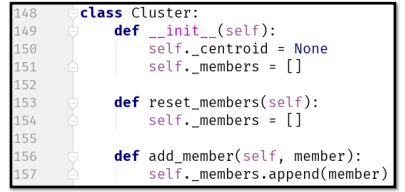
**\* class Member**

**> 1 lớp Kmeans cho triển khai thuật toán**

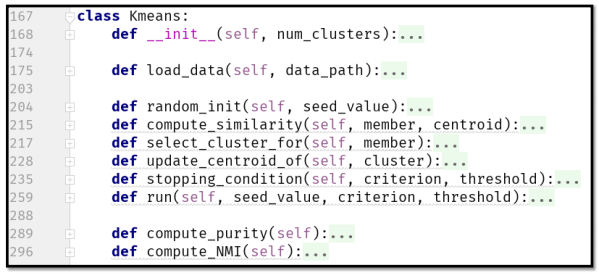
**\* class Kmeans**

**class Member: **

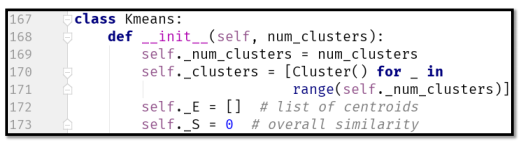
**class Cluster:**

****

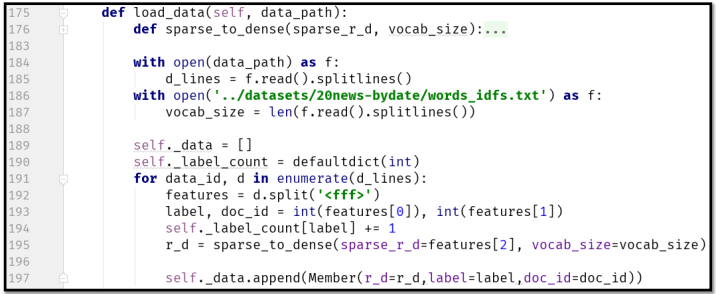
**class Kmeans:**

****

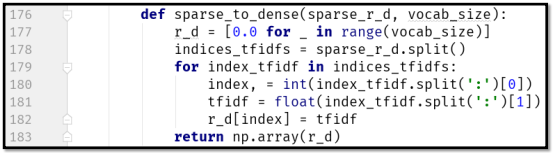
**Hàm khởi tạo:**

****

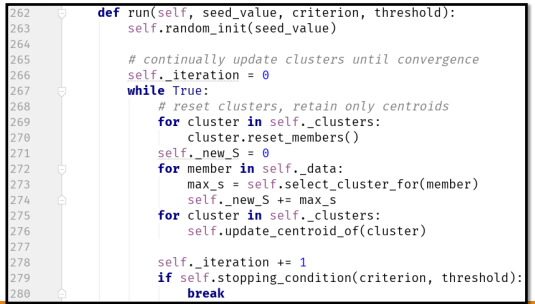
**Đọc dữ liệu:**

****

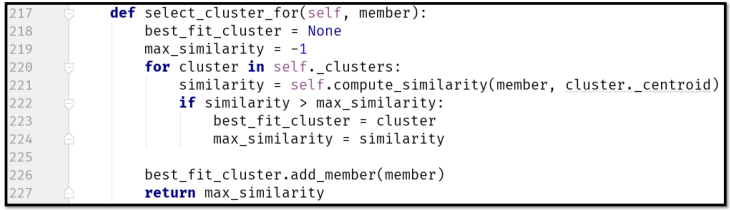
**Đọc dữ liệu: Hàm sparse\_to\_dense**

****

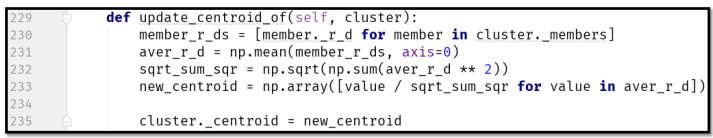
**Chạy thuật toán: Hàm run**

****

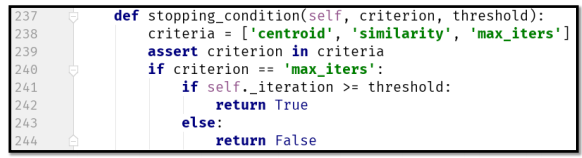
**Chạy thuật toán: xác định cụm cho từng điểm dữ liệu**

****

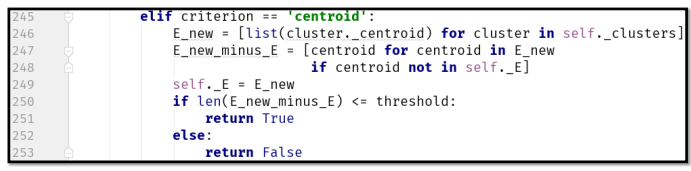
**Chạy thuật toán: cập nhật lại tâm cụm**

****

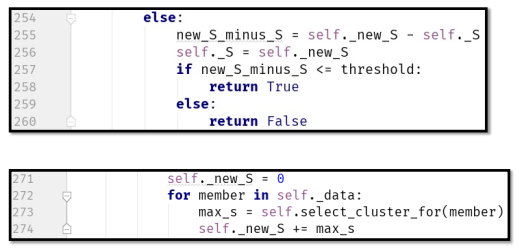
**Chạy thuật toán: Kiểm tra điều kiện dừng – max\_iters**

****

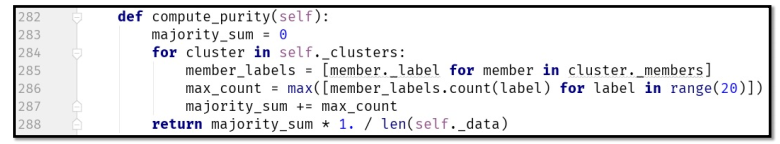
**Chạy thuật toán: Kiểm tra điều kiện dừng – centroid**

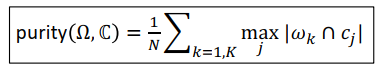
****

**Chạy thuật toán: Kiểm tra điều kiện dừng – similarity**

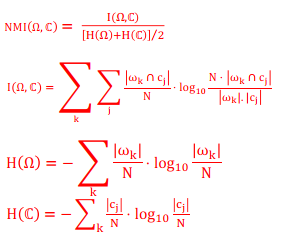
****

**Đánh giá chất lượng phân cụm: Tính purity**

****

****

**Đánh giá chất lượng phân cụm: Tính NMI**

****

****

**Vấn đề khởi tạo tâm cụm:**

**> Kết quả của Kmeans phụ thuộc vào việc khởi tạo tâm cụm**

**=> Làm vài lần và chọn lấy lần tốt nhất**

**hoặc Khởi tạo theo chiến lược:**

**\* Dùng Kmeans++**

**\* Cluster center initialization algorithm for Kmeans clustering**

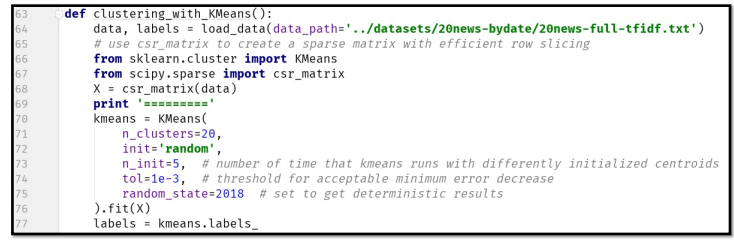
**Sử dụng Scikit-learn**

**>Kmeans**

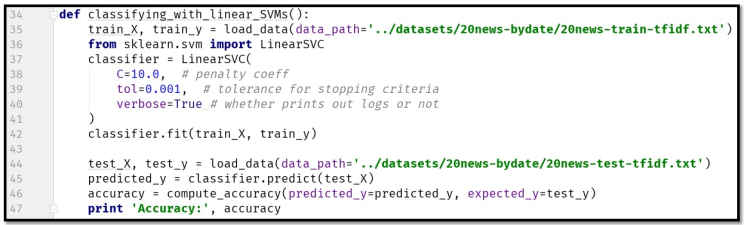
**>SVMs: > Linear SVMs**

**> kernel SVMs**

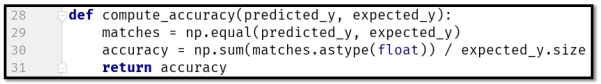
**Kmeans:**

****

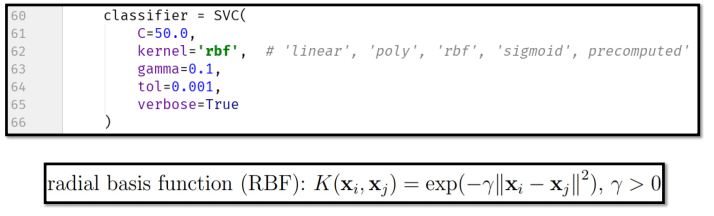
**SVMs: Linear SVMs:**

****

**Hàm compute\_accuracy**

****

**Kernel SVMs:**

****

# Chương 3: Multi-layer Perceptron

1. **Tensorflow**

**Tensorflow là gì – Với sự bùng nổ của lĩnh vực Trí Tuệ Nhân Tạo – A.I. trong thập kỷ vừa qua, machine learning và deep learning rõ ràng cũng phát triển theo cùng. Và ở thời điểm hiện tại, TensorFlow chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.**

**Các hàm được dựng sẵn trong thư viện cho từng bài toán cho phép TensorFlow xây dựng được nhiều neural network. Nó còn cho phép bạn tính toán song song trên nhiều máy tính khác nhau, thậm chí trên nhiều CPU, GPU trong cùng 1 máy hay tạo ra các dataflow graph – đồ thị luồng dữ liệu để dựng nên các model. Nếu bạn muốn chọn con đường sự nghiệp trong lĩnh vực A.I. này, nắm rõ những điều cơ bản của TensorFlow thực sự rất quan trọng.**

**Được viết bằng C++ và thao tác interface bằng Python nên phần performance của TensorFlow cực kỳ tốt. Đối tượng sử dụng nó cũng đa dạng không kém: từ các nhà nghiên cứu, nhà khoa học dữ liệu và dĩ nhiên không thể thiếu các lập trình viên.**

**Kiến trúc của TensorFlow:**

**Kiến trúc TensorFlow hoạt động được chia thành 3 phần:**

**– Tiền xử lý dữ liệu**

**– Dựng model**

**– Train và ước tính model**

**Cách TensorFlow hoạt động**

**TensorFlow cho phép các lập trình viên tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 sê-ri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học, và mỗi kết nối hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là ‘tensor’.**

**TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python. Vì Python khá dễ học và làm việc, ngoài ra còn cung cấp nhiều cách tiện lợi để ta hiểu được làm thế nào các high-level abstractions có thể kết hợp cùng nhau. Node và tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow bản thân chúng cũng là các ứng dụng Python.**

**Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau.**

**TensorFlow 2.0, được ra mắt vào tháng 10 năm 2019, cải tiến framework theo nhiều cách dựa trên phản hồi của người dùng, để dễ dàng và hiệu quả hơn khi làm việc cùng nó (ví dụ: bằng cách sử dụng các Keras API liên quan đơn giản cho việc train model). Train phân tán dễ chạy hơn nhờ vào API mới và sự hỗ trợ cho TensorFlow Lite cho phép triển khai các mô hình trên khá nhiều nền tảng khác nhau. Tuy nhiên, nếu đã viết code trên các phiên bản trước đó của TensorFlow thì bạn phải viết lại, đôi lúc 1 ít, đôi lúc cũng khá đáng kể, để tận dụng tối đa các tính năng mới của TensorFlow 2.0.**

**Lợi ích từ TensorFlow**

**Lợi ích dễ thấy nhưng quan trọng nhất mà TensorFlow cung cấp cho việc lập trình machine learning chính là abstraction. Thay vì phải đối phó với những tình huống rườm rà từ việc thực hiện triển khai các thuật toán, hay tìm ra cách hợp lý để chuyển output của 1 chức năng sang input của 1 chức năng khác, giờ đây bạn có thể tập trung vào phần logic tổng thể của 1 ứng dụng hơn. TensorFlow sẽ chăm sóc phần còn lại thay cho bạn.**

**Ngoài ra TensorFlow còn ung cấp các tiện ích bổ sung cho các lập trình viên cần debug cũng như giúp bạn tự suy xét các ứng dụng TensorFlow. Chế độ eager execution cho phép bạn đánh giá và sửa đổi từng operation của biểu đồ 1 cách riêng biệt và minh bạch, thay vì phải dựng toàn bộ biểu đồ dưới dạng 1 đối tượng độc lập vốn khá mơ hồ hay phải đánh giá chung tổng thể. Cuối cùng, 1 tính năng khá độc đáo của TensorFlow là TensorBoard. TensorBoard cho phép bạn quan sát 1 cách trực quan những gì TensorFlow đang làm.**

**TensorFlow còn có nhiều cải tiến từ sự hậu thuẫn từ các ekíp thương mại hạng A tại Google. Google không những tiếp lửa cho tiến độ nhanh chóng cho sự phát triển đằng sau dự án, mà còn tạo ra nhiều phục vụ độc đáo xung quanh TensorFlow để nó dễ dàng deploy và sử dụng: như silicon TPU mình đã nói ở trên để tăng tốc hiệu suất đám mây Google, 1 online hub cho việc chia sẻ các model được tạo với framework, sự hiện diện của in-browser và gần gũi với mobile của framework, và nhiều hơn thế nữa…**

**Lưu ý: Trong 1 số công việc training, vài chi tiết về việc triển khai của TensorFlow làm cho nó khó có thể quyết định được hoàn toàn kết quả training model . Đôi khi 1 model được train trên 1 hệ thống này sẽ có thay đổi 1 chút so với 1 model được train trên hệ thống khác, ngay cả khi chúng được cung cấp dữ liệu như nhau. Các nguyên nhân cho điều này cũng xê xích hay 1 số hành vi khi không được xác định khi sử dụng GPU. Điều này nói rằng, các vấn đề đó có thể giải quyết được, và đôi ngũ của TensorFlow cũng đang xem xét việc kiểm soát nhiều hơn để ảnh hưởng đến tính quyết định trong quy trình làm việc.**

**Các Component của TensorFlow**

**Tensor**

**Tên của TensorFlow được đưa ra trực tiếp là nhờ vào framework cốt lõi của nó: Tensor. Trong TensorFlow, tất cả các tính toán đều liên quan tới các tensor. 1 tensor là 1 vector hay ma trận của n-chiều không gian đại diện cho tất cả loại dữ liệu. Tất cả giá trị trong 1 tensor chứa đựng loại dữ liệu giống hệt nhau với 1 shape đã biết (hoặc đã biết 1 phần). Shape của dữ liệu chính là chiều của ma trận hay mảng.**

**1 tensor có thể được bắt nguồn từ dữ liệu input hay kết quả của 1 tính toán. Trong TensorFlow, tất cả các hoạt động được tiến hành bên trong 1 graph – biểu đồ. Biểu đồ là 1 tập hợp tính toán được diễn ra liên tiếp. Mỗi operation được gọi là 1 op node (operation node) và được kết nối với nhau.**

**Biểu đồ phát thảo các op và kết nối giữa các node. Tuy nhiên, nó không hiển thị các giá trị. Phần edge của các node chính là tensor, 1 cách để nhập operation với dữ liệu.**

**Graph**

**TensorFlow sử dụng framework dạng biểu đồ. Biểu đồ tập hợp và mô tả tất cả các chuỗi tính toán được thực hiện trong quá trình training. Biểu đồ cũng mang rất nhiều lợi thế:**

**– Nó được làm ra để chạy trên nhiều CPU hay GPU, ngay cả các hệ điều hành trên thiết bị điện thoại.**

**– Tính di động của biểu đồ cho phép bảo toàn các tính toán để bạn sử dụng ngay hay sau đó. Biểu đồ có thể được lưu lại để thực thi trong tương lai.**

**– Tất cả tính toán trong biểu đồ được thực hiện bằng cách kết nối các tensor lại với nhau. 1 tensor có 1 node và 1 edge. Node mang operation toán học và sản xuất các output ở đầu cuối. Các edge giải thích mối quan hệ input/output giữa các node.**

1. **Multi-layer Perceptron**
   1. **Multi-layer Perceptron là gì?**

**Một Multi-Layer Perceptron (MLP) là một loại mạng neural feedforward, tức là thông tin di chuyển qua từ lớp đầu tiên đến lớp cuối cùng mà không có các vòng lặp hay kết nối ngược. MLP là một thành phần cơ bản của deep learning và được sử dụng rộng rãi trong nhiều ứng dụng khác nhau như xử lý ngôn ngữ tự nhiên, nhận dạng hình ảnh, và dự đoán dữ liệu.**

**MLP gồm có ít nhất ba lớp:**

**- Lớp đầu vào (Input Layer): Lớp này nhận các đặc trưng hoặc dữ liệu đầu vào của mô hình. Mỗi nút trong lớp này thể hiện một đặc trưng riêng biệt của dữ liệu.**

**- Các lớp ẩn (Hidden Layers): Đây là các lớp nằm giữa lớp đầu vào và lớp đầu ra. Mỗi lớp ẩn có một số nút (neurons) và mỗi nút kết nối với tất cả các nút trong lớp trước đó và sau đó. Các lớp ẩn này giúp mô hình học được các biểu diễn phức tạp từ dữ liệu.**

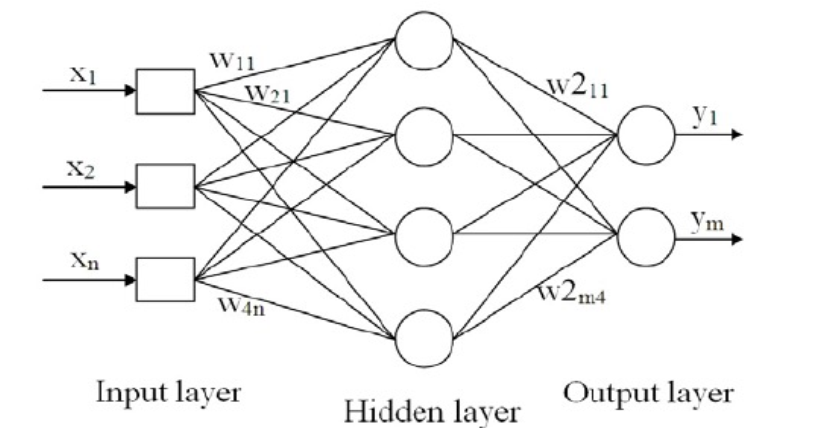
**- Lớp đầu ra (Output Layer): Lớp này cho ra kết quả của mô hình, ví dụ như dự đoán một lớp hoặc giá trị. Số nút trong lớp đầu ra thường phụ thuộc vào mục tiêu của bài toán (ví dụ, phân loại đa lớp sẽ có nhiều nút tương ứng với số lớp).**

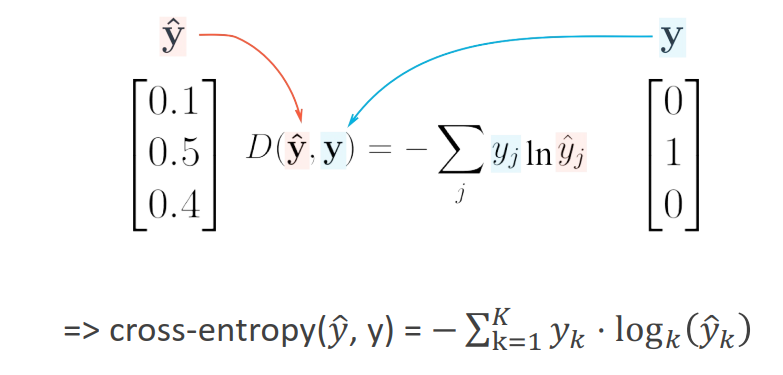
**Các kết nối giữa các nút trong các lớp là trọng số (weights) và mô hình học chính là quá trình tối ưu hóa các trọng số này để dự đoán đầu ra chính xác nhất có thể.**

**Để huấn luyện một MLP, bạn cần sử dụng các thuật toán tối ưu hóa như gradient descent để điều chỉnh trọng số dựa trên sai số giữa dự đoán và kết quả thực tế. Quá trình này được lặp lại qua nhiều lần cho đến khi mô hình hội tụ và có khả năng dự đoán tốt trên dữ liệu mới.**

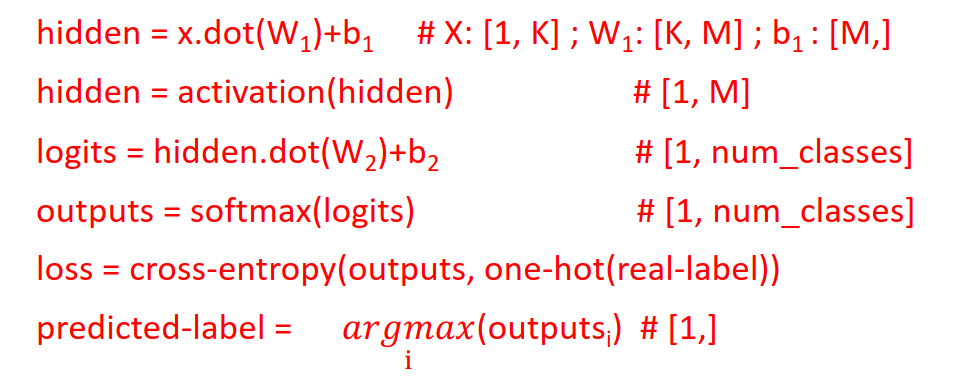
**MLP có thể mở rộng để chứa nhiều lớp ẩn và nhiều nút, tạo thành các kiến trúc mạng neural sâu (deep neural networks), làm cho chúng có khả năng học biểu diễn phức tạp từ dữ liệu lớn hơn và giải quyết các bài toán khó khăn hơn.**

* 1. **Triển khai:**

****

**Hàm lỗi phân lớp:**

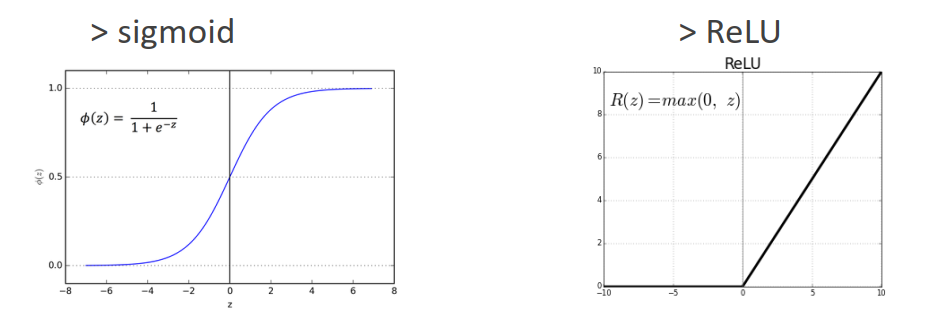
**Mô hình hóa:**

****

**Hàm softmax:**

****

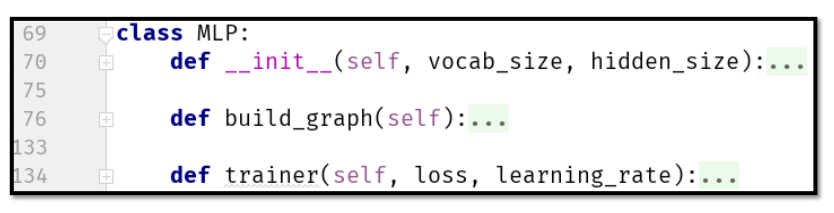
**Hàm activation:**

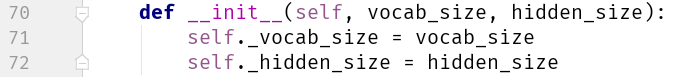
****

**Các bước để xây dựng mô hình trên tensorlfow:**

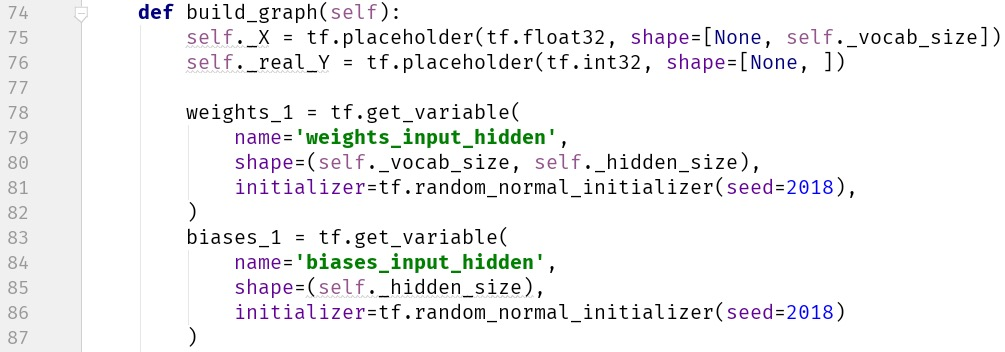
**> B1: Xây dựng computation graph**

**> B2: Mở một phiên làm việc (session), truyền (feed) dữ liệu vào graph và chạy**

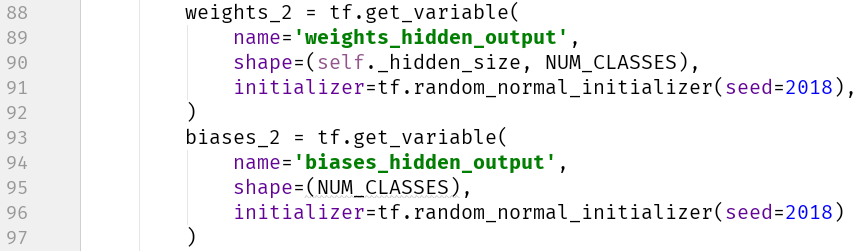
**Xây dựng class MLP **

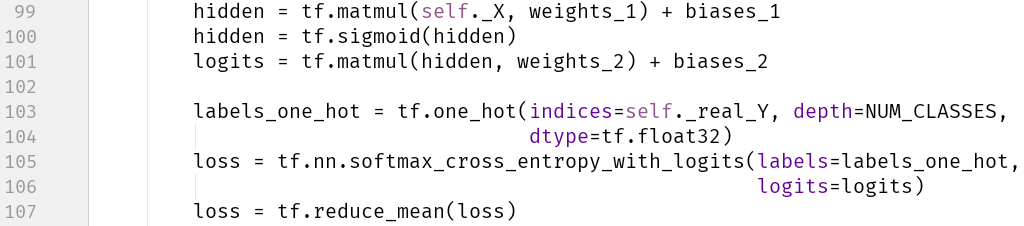
**Xây dựng class MLP: hàm init **

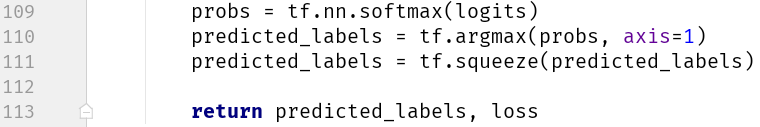
**Xây dựng class MLP: hàm build\_graph**

****

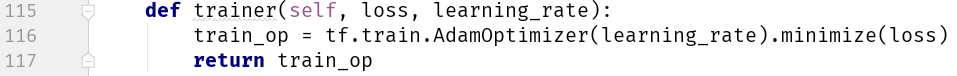
**Xây dựng class MLP: hàm build\_graph**

****

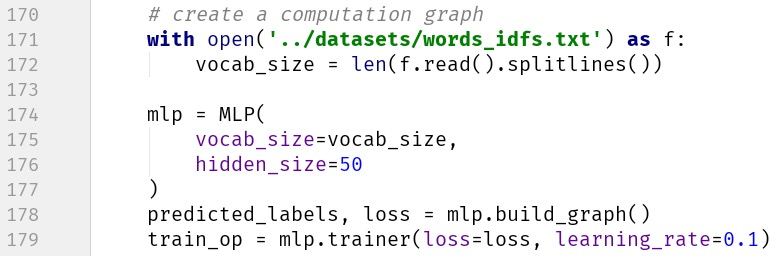
****

****

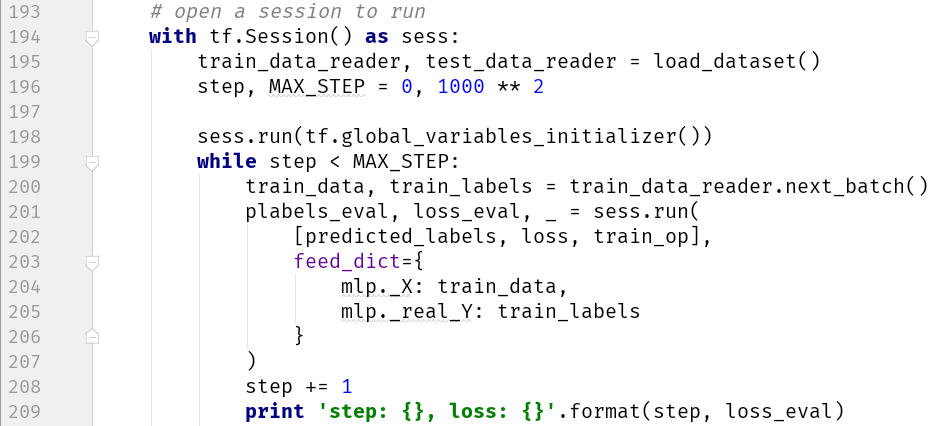
**Xây dựng class MLP: hàm trainer: chọn thuật toán để tối ưu hàm Loss**

****

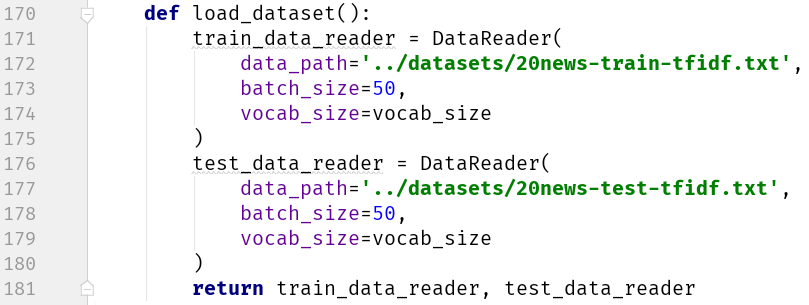
**Xây dựng computation graph**

****

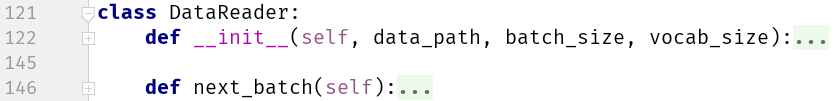
**Mở một phiên làm việc, truyền dữ liệu và chạy**

****

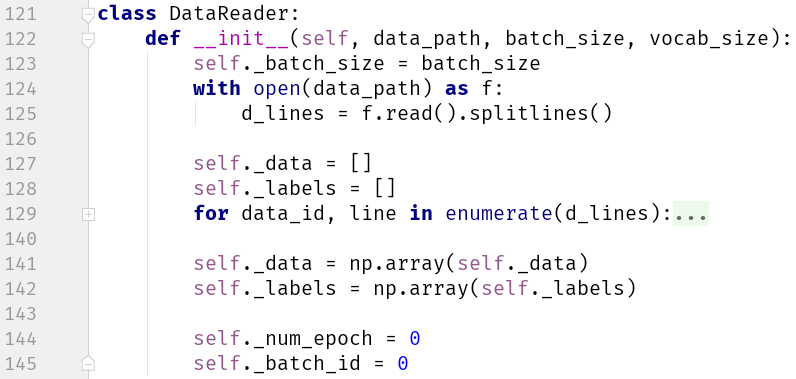
**> hàm load\_dataset**

****

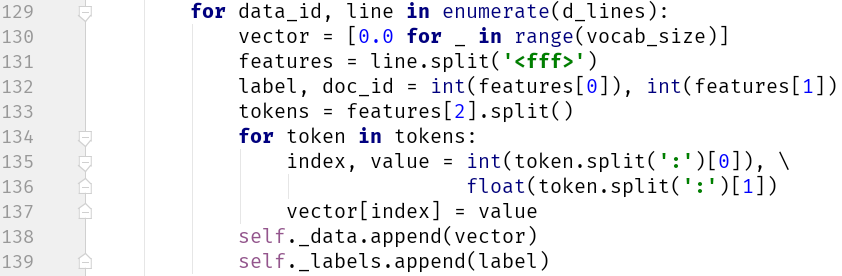
**> class DataReader**

****

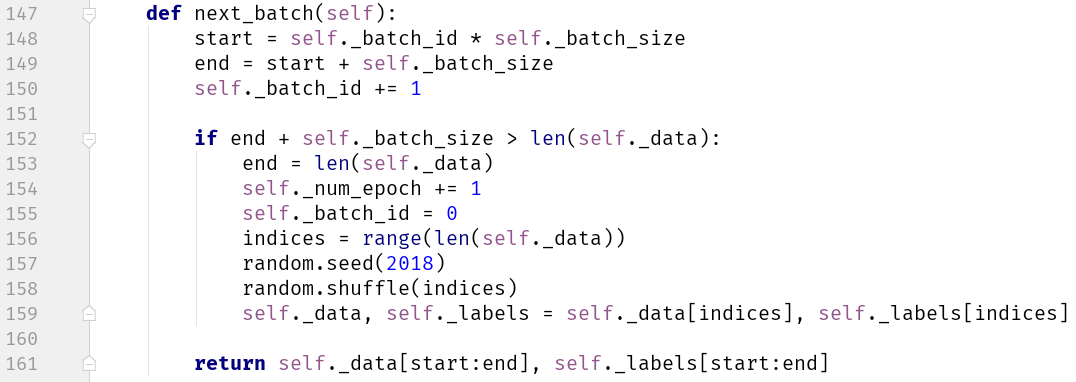
**class DataReader: hàm init**

****

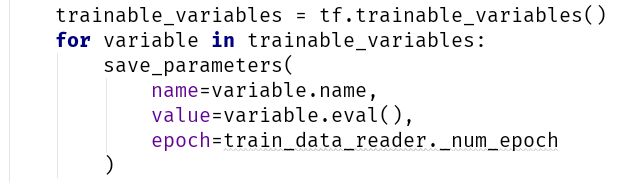
**class DataReader: hàm init:**

****

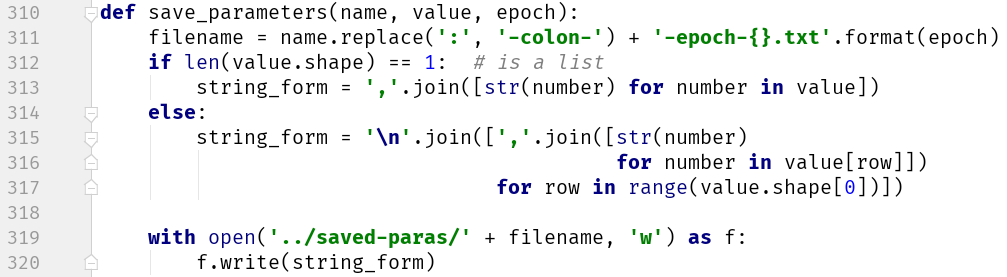
**class DataReader: hàm next\_batch**

****

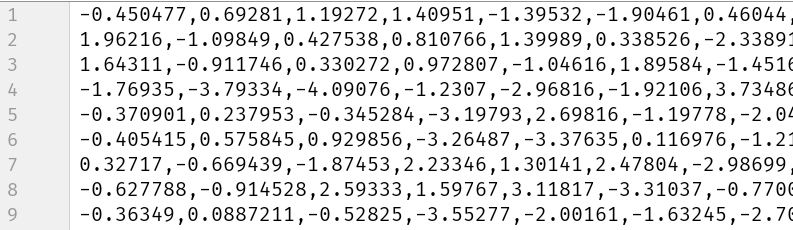
**Lưu các tham số mô hình: có thể lưu tại bất cứ bước lặp nào của quá trình training**

****

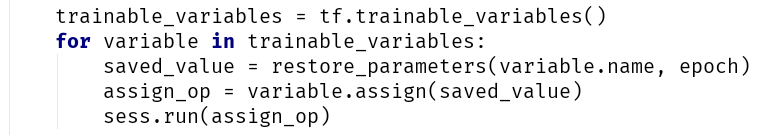
**Lưu các tham số mô hình: hàm save\_parameters**

****

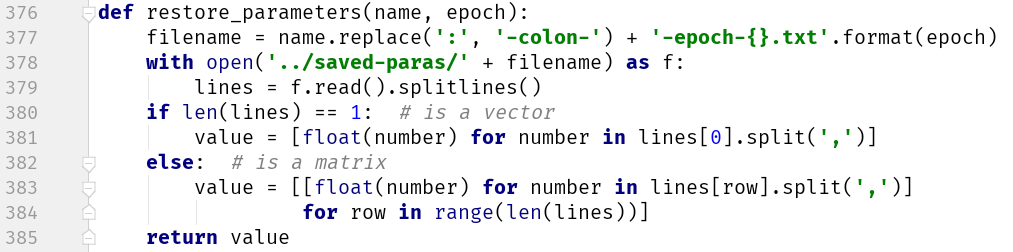
**Nội dung file**

****

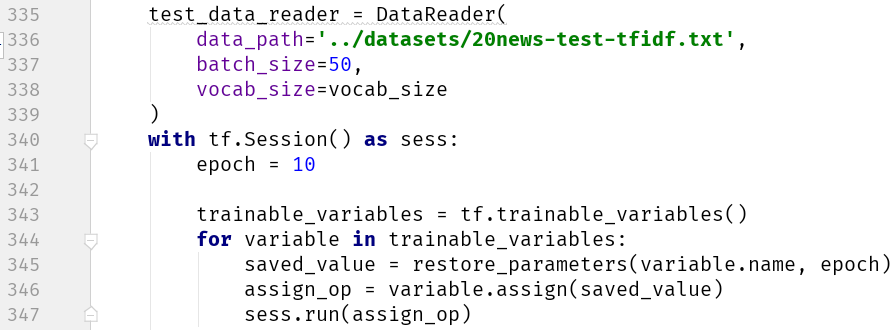
**Khôi phục các tham số đã lưu:**

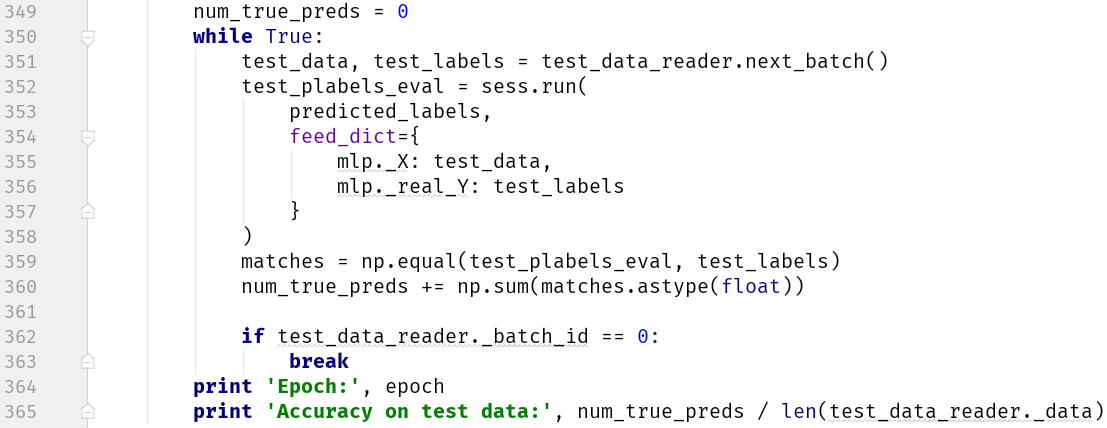
****

**Hàm restore\_parameters**

****

**Đánh giá model trên test data:**

****

****

# Tài liệu tham khảo

[**http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html**](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)

[**http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html**](http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html)

[Tensorflow là gì? 10 tài liệu học tensoflow đầy đủ nhất | TopDev](https://topdev.vn/blog/tensorflow-la-gi/)

[Machine Learning cơ bản (machinelearningcoban.com)](https://machinelearningcoban.com/2017/02/24/mlp/)