**BÁO CÁO TUẦN 3**

[0. SVM với các kernel khác nhau 1](#_Toc14763931)

[1. FeedForward Neural Netword 2](#_Toc14763932)

[1.1. Backpropagtion algorithm, SGD, dropout 2](#_Toc14763933)

[1.1.1. Backpropagtion algorithm 2](#_Toc14763934)

[1.1.2. SGD 3](#_Toc14763935)

[1.1.3. Dropout 4](#_Toc14763936)

[1.2. FeedForward Neural Netword 5](#_Toc14763937)

[2. Word2vec, CNN 6](#_Toc14763938)

[2.1. Word2vec 6](#_Toc14763939)

[2.2. CNN 6](#_Toc14763940)

[3. Phân loại sentiment sử dụng CNN 8](#_Toc14763941)

# SVM với các kernel khác nhau

Với LinearSVC, độ chính xác = 88.62%

Với SVC kernel = ‘rbf’, độ chính xác = 87.89%

Với SVC kernel = ‘linear’, độ chính xác = 87.55%

# FeedForward Neural Netword

## Backpropagtion algorithm, SGD, dropout

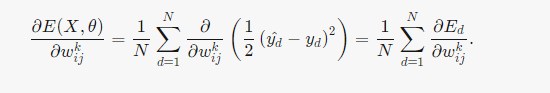
### Backpropagtion algorithm

* Là thuật toán học có giám sát của mạng nơ ron nhân tạo bằng cách sử dụng gradient descent.
* ‘Back’: tiến hành tính toán đạo hàm ngược qua mạng. Đạo hàm của bộ W ở layer cuối được tính đầu tiên và đạo hàm của layer đầu tiên được tính sau cùng.

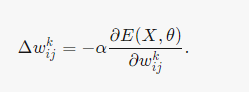
Ảnh có chứa văn bản

Mô tả được tạo tự động

* Mô tả chung cho thuật toán:
* Khởi tạo bộ W, b ngẫu nhiên tại mỗi layer.
* Forward phase (tính đầu ra): với mỗi cặp input-output (xi, yi) và lưu trữ y’i (dự đoán), zjk (tổng các bias), ojk (output) cho mỗi node j tại layer k được tính từ input layer đến output layer.
* Backtrack phase (tính hàm lỗi và đạo hàm): với mỗi cặp input-output (xi, yi) lưu trữ đạo hàm của hàm lỗi Ed cho wijk (kết nối node i ở layer k-1 đến node j ở layer k) được tính từ output layer ngược về input layer.
* Với mỗi cặp input-output, kết hợp các đạo hàm riêng lẻ của hàm lỗi Ed cho wijk để tính tổng đạo hàm.



* Cập nhật bộ W theo learning\_rate và tổng đạo hàm đã tính được trong bước trước đó.



### SGD

* Gradient Descent: Để giải bài toán tìm điểm global optimal của hàm mất mát J(θ) (với θ là tập hợp các tham số của mô hình):
* Dự đoán một điểm khởi tạo θ = θ0
* Cập nhật θđến khi đạt được kết quả chấp nhận được:

θ = θ – η ∇θJ(θ)

với ∇θJ(θ) là đạo hàm của hàm mất mát tại θ.

* SGD – Stochastic Gradient Descent: Sử dụng các mẫu ngẫu nhiên để đánh giá đạo hàm.
* Tại 1 thời điểm, chỉ tính đạo hàm của hàm mất mát dựa trên một (vài) điểm dữ liệu rồi cập nhật θ dựa trên đạo hàm này, lặp lại với từng điểm dữ liệu.

θ = θ – η ∇θJ(θ, xi, yi)

* Mỗi lần duyệt một lượt qua tất cả các điểm trên toàn bộ dữ liệu được gọi là một epoch. Sau mỗi epoch, chúng ta cần xáo trộn thứ tự của các dữ liệu để đảm bảo tính ngẫu nhiên.
* SGD phù hợp với bài toán có lượng dữ liệu lớn và các bài toán yêu cầu mô hình thay đổi liên tục (online learning).

### Dropout

* Trong quá trình training, nhiều hidden unit bị tắt ngẫu nhiên và mô hình được huấn luyện trên các bộ tham số còn lại. Trong quá trình test, toàn bộ các unit sẽ được sử dụng. Việc này giúp cho mô hình tránh được overfitting và cũng được coi giống với kỹ thuật ensemble trong các hệ thống machine learning khác. Với mỗi cách tắt các unit, ta có một mô hình khác nhau. Với nhiều tổ hợp unit bị tắt khác nhau, ta thu được nhiều mô hình. Việc kết hợp ở cuối cùng được coi như sự kết hợp của nhiều mô hình (và vì vậy, nó giống với ensemble learning).

Ảnh có chứa bản đồ, văn bản

Mô tả được tạo tự động

* Dropout rate: tham số dropout, nằm trong khoảng [0,1]. Trong đó dropout rate = 1 tương đương không có dropout, dropout rate = 0 có nghĩa không có output từ layer. Dropout rate cho các lớp ẩn thường nằm trong khoảng 0,5 – 0,8, cho input layer thường lớn hơn, chẳng hạn 0.8.

## FeedForward Neural Netword

Ảnh có chứa đối tượng

Mô tả được tạo tự động

* Implement FeedForward với số lượng hidden layer cùng số unit trên mỗi layer tự chọn, có thể lựa chọn dropout rate.
* Với ví dụ đơn giản 150 điểm có 2 features phân ra 3 lớp:
* Với 1 hidden layer có 100 units: ~92%-97%
* Với 2 hidden layer có 100 units và 50 unit: ~98.5%
* Với ví dụ phân loại văn bản với dataset cho SVM tuần 2:

>11k features, 26000 datapoint, 13 class:

* Với 1 hidden layer có 100 units, dropout rate = 0.2: 85.83%
* Với 2 hidden layer có 1000 và 500 units, dropout rate = 0.2:
* Việc chọn số hidden layer và số unit trên mỗi hidden layer tùy thuộc vào kích thước bộ dữ liệu, số hidden layer và số unit càng nhiều không đồng nghĩa thuật toán cho ra kết quả chính xác hơn.

# Word2vec, CNN

## Word2vec

* Thể hiện mối quan hệ, tính tương đồng giữa các từ.
* Là 1 trong 2 model Word Embedding phổ biến cùng Glove.
* Có 2 mô hình Word2vec được áp dụng: Skip-gram, CBOW
* Skip-gram: Tìm content dựa trên 1 từ cho trước.

Phù hợp bộ dữ liệu nhỏ, thể hiện tốt với các từ hiếm.

* CBOW (Continuous Bag of Words): sử dụng content để dự đoán mục tiêu.

Nhanh hơn skip-gram, chính xác hơn với những từ xuất hiện thường xuyên.

Ảnh có chứa bản đồ, văn bản

Mô tả được tạo tự động

* 1. CNN *-* Convolutional Neural Network (Mạng nơ-ron tích chập)
* Cấu trúc mạng CNN:

Một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation để tính đầu vào cho các lớp tiếp theo.

Trong feedforward neural network thì mỗi neural đầu vào (input node) cho mỗi neural đầu ra trong các lớp tiếp theo. Còn trong mô hình CNNs thì layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng cục bộ của neuron trước đó.

Ảnh có chứa văn bản

Mô tả được tạo tự động

* CNN có 02 phần chính: Lớp trích lọc đặc trưng (Conv, Relu và Pool) và Lớp phân loại (FC và softmax).
* Convolutional Layer (lớp tích chập): trích chọn các đặc trưng của đầu vào. Gồm một tập các feature map và mỗi feature map này là một bản scan của input ban đầu, nhưng được trích xuất ra các feature/đặc tính cụ thể.

+ Kernel/Filter/Feature detect/Sliding window: 1 ma trận có kích thước < kích thước ma trận đầu vào.

+ Nhân từng phần tử bên trong kernel với ma trận đầu vào. Kết quả được một ma trận gọi là Convoled feature/Activation Map/Feature Map.

* Relu Layer: áp dụng một mức ngưỡng.

Áp dụng các activation function lên đầu ra của Conv Layer. Layer này không thay đổi kích thước của ma trận và không có thêm bất kì tham số nào.

* Pool Layer: làm giảm chiều không gian của đầu ra và giảm độ phức tạp tính toán của model, giúp kiểm soát hiện tượng overffiting. Loại pooling thường gặp nhất là max pooling (lấy giá trị lớn nhất trong một pooling window).
* FC(Fully\_Connected) Layer: Tại lớp mạng này, mỗi một nơ-ron của layer này sẽ liên kết tới mọi nơ-ron của lớp khác. Để đưa mảng từ các layer trước vào mạng này, buộc phải dàn phẳng mảng ra thành 1 vector thay vì là mảng nhiều chiều như trước. Tại layer cuối cùng sẽ sử dụng softmax để phân loại đối tượng dựa vào vector đặc trưng đã được tính toán của các lớp trước đó.

# Phân loại sentiment sử dụng CNN

* Sử dụng bộ dữ liệu imdb
* Vector hóa dữ liệu: Sử dụng word2vec, thu được
* Tập train, test: 25000 điểm dữ liệu, mỗi dữ liệu được biểu diễn dưới 1 mảng gồm 400 từ và 50 features.
* Vocabulary Size: 88585 từ
* Phân loại thành 2 lớp: Neg và Pos
* Các bước thực hiện chương trình:
* Conv Layer: sử dụng n filter ki\* ki (input: h\*w) và bias bi 1\*1,

với patch = 0, strike = 1, thu được n vector Ci có chiều h’i \* w’i

(h’ = h-ki+1, w’ = w-ki+1).

* Pooling Layer:

Áp dụng Activation function lên output của ConvLayer, đầu vào của Pool Layer vẫn là n ma trận h’i \* w’i. Output của lớp này là n ma trận S có chiều bé hơn ma trận đầu vào.

*Max1DPooling*

*AveragePooling*

* FC Layer: Output của Pool Layer là n ma trận S làm phẳng bởi hàm F thành vecto f - input của FC Layer. Coi mỗi phần tử trong input vector này là 1 node. Thực hiện nhân input vector này với ma trận trọng số W, b0 thu được lớp mới có 2 node. Áp dụng Activation function ở lớp này thu được y’ dự đoán.
* Activation function

+ Sử dụng ReLU sau ConvLayer

+ Sử dụng Sigmoid ở layer cuối cùng

Ảnh có chứa văn bản, bản đồ

Mô tả được tạo tự động

Chương trình CNN tự implement

* Backpropagation[[1]](#footnote-2)[[2]](#footnote-3)[[3]](#footnote-4)[[4]](#footnote-5)

Coi là kí hiệu cho activation function ở mỗi bước

+ Loss function L =

+ Đạo hàm của W, b0:

: k = 1

: k = 0 nếu i = j,

+ Đạo hàm của ki, bi:

=

()

= \* Irot180

= =

(

=

Với Max Pooling,

= max(C\_slide)

Với Average Pooling,

= average(\_slide)

+ Update W, b0, k, b:

W += -lr\*dW

b0 += -lr\*db0

ki += -lr\*dki

bi += -lr\*dbi

* Kết quả:
* Sử dụng thư viện Keras với 2 Conv Layer (số filter ở mỗi lớp lần lượt là 20, 8), 2 Pooling Layer, 10 epochs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Ave/Ave | Ave/Max | Max/Ave | Max/Max |
| 10 epochs, adam | 87.28 | 86.97 | 87.47 | 87.1 |
| 10 epochs, sgd | 86.68 | 70.0 | 73.06 | 80.11 |

* Tự implement sử dụng numpy:

+ 1 Conv Layer với 4 filter, 1 MaxPooling Layer, 5 epochs

* kết quả đạt 56,04% trên tập test, 56,46% trên tập train.

+ 1 Conv Layer với 4 filter, 1 AveragePooling Layer, 5 epochs

* kết quả đạt

1. Derivation of Backpropagation in Convolutional Neural Network (CNN) <https://pdfs.semanticscholar.org/5d79/11c93ddcb34cac088d99bd0cae9124e5dcd1.pdf> [↑](#footnote-ref-2)
2. <https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/> [↑](#footnote-ref-3)
3. Convolutional Neural Networks backpropagation: from intuition to derivation

   <https://grzegorzgwardys.wordpress.com/2016/04/22/8/> [↑](#footnote-ref-4)
4. <https://datascience-enthusiast.com/DL/Convolution_model_Step_by_Stepv2.html> [↑](#footnote-ref-5)