Báo cáo bài tập lớn

Học máy IT4866

****

Để tài

Sử dụng mạng neuron trong nhận diện chữ số viết tay

***

Sinh viên thực hiện

Vũ Công Luật 20142745

Nguyễn Văn Túc 20145070

Hà Văn Quang 20143578

Võ Anh Tuấn 20144963

Giáo viên hướng dẫn: TS. Thân Quang Khoát

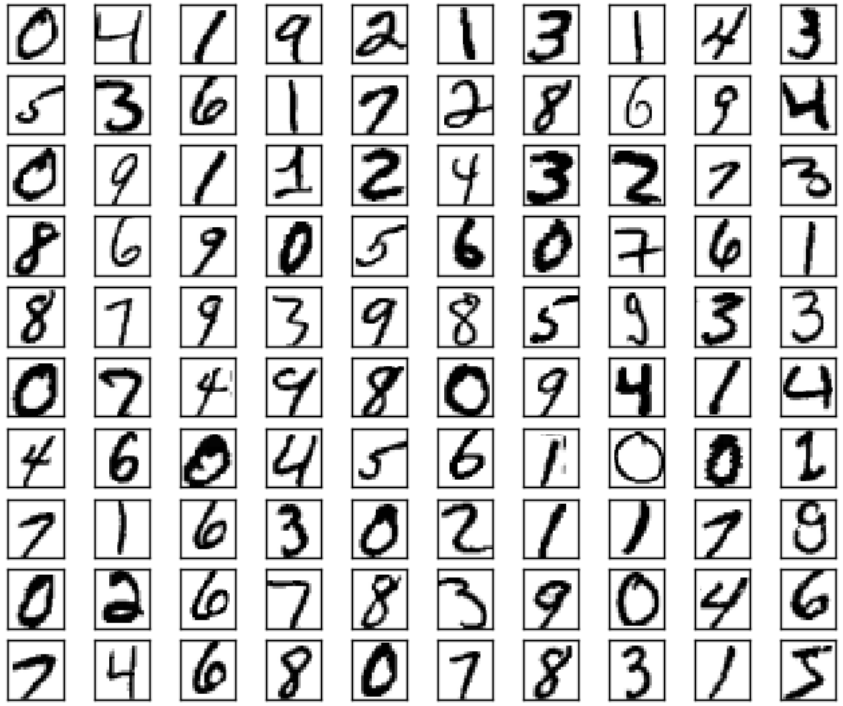
Bài toán thực tế

- Xuất phát từ thực tế một số lượng lớn các văn bản, thủ tục hành chính vẫn phải viết bằng tay. Nảy sinh nhu cầu cần số hóa số lượng văn bản đó thành dữ liệu lưu trữ trên máy tính.
- Nếu sử dụng tới con người để chuyển số văn bản chữ viết tay sang văn bản đánh máy để lưu trên máy tính, sẽ dẫn đến tốn kém chi phí, thời gian và năng xuất công việc sẽ không cao.
- Do đó việc có được một hệ thống tự động nhận diện chữ viết tay mà không cần can thiệp của con người là rất hữu ích, giúp tiết kiệm công sức và tăng hiệu quả công việc.

Bài toán thực tế

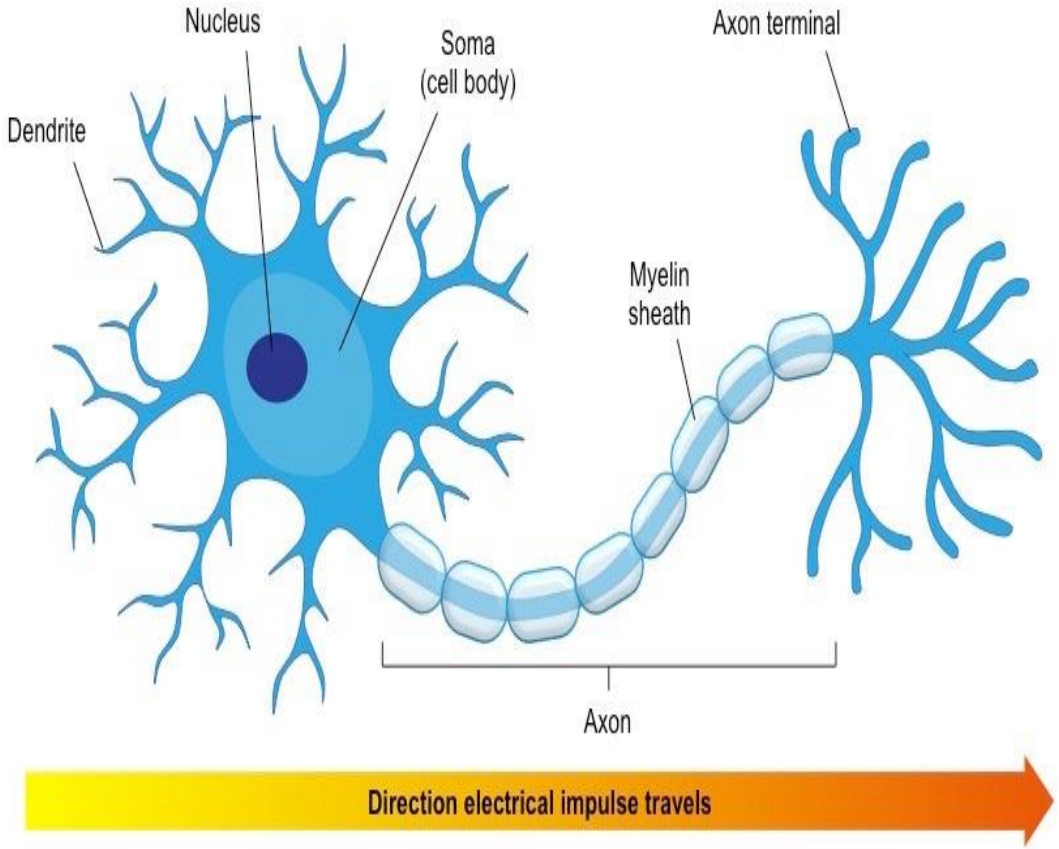
- Một hệ thống nhận diện chữ viết tay trên thực tế được chia thành các lớp nhận diện khác nhau.
- Lớp đầu tiên nhận diện các dòng trong văn bản gốc
- Lớp thứ hai tách ra các từ trong mỗi dòng.
- Lớp tiếp theo tách ra từng kí tự.
- Lớp cuối cùng nhận diện các kí tự tương ứng.
- Trong phạm vi bài tập lớn, hệ thống mà nhóm xây dựng sẽ chỉ giới hạn trong việc nhận diện các kí tự, mà cụ thể hệ thống chỉ nhận diện được các kí tự là chữ số từ 0 tới 9.

Kịch bàn hoạt động



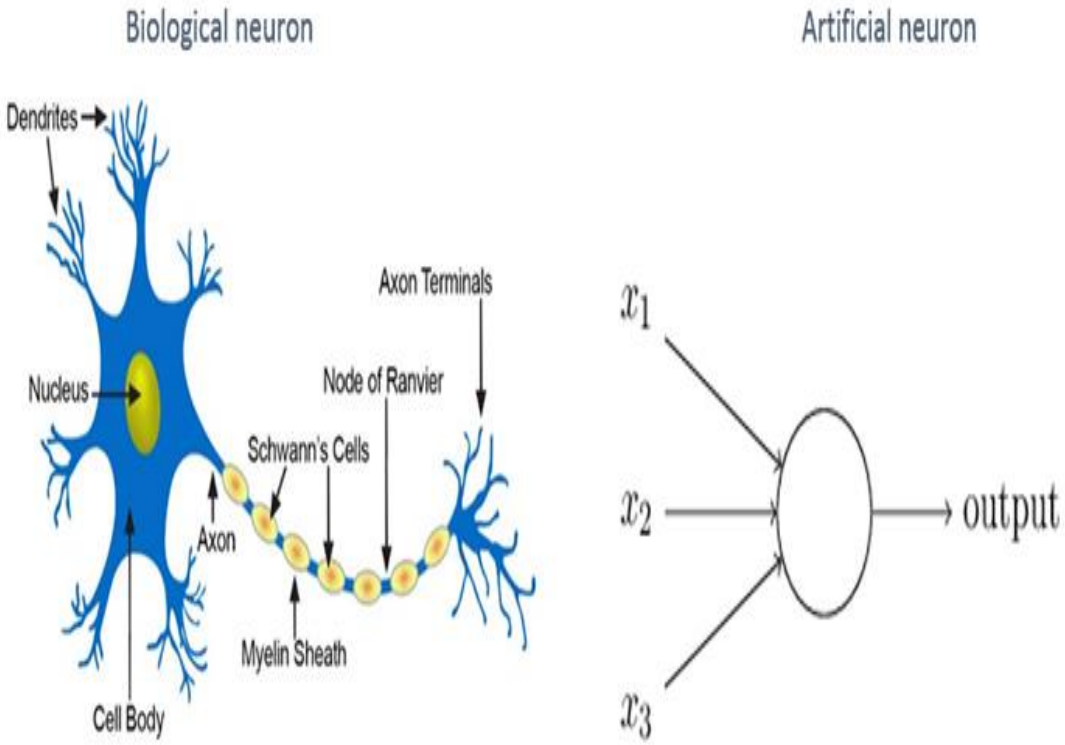
- Hệ thống hoạt động theo 3 giai đoạn
- Giai đoạn thu thập dữ liệu: dữ liệu lấy từ tập MNIST là tập các ảnh 28x28 pixel mỗi ảnh chứa một số có giá trị trong đoạn [0,9]. Tập dữ liệu MNIST là tập đã tiền xử lý sẵn. Chia tập MNIST ra thành 3 tập con (training dataset, validation dataset, test dataset).
- Giai đoạn huấn luyện: với dữ liệu từ tập training data ta tiến hành huấn luyện hệ thống. Chi tiết quá trình huấn luyện sẽ được trình bày cụ thể ở các slide tiếp theo.
- Giai đoạn phán đoán: với mỗi ví dụ trong tập thử nghiệm khi được hệ thống xử lý sẽ được gán nhãn theo các nhãn số có giái trị trong đoạn [0,9] tương ứng.

Giải thuật



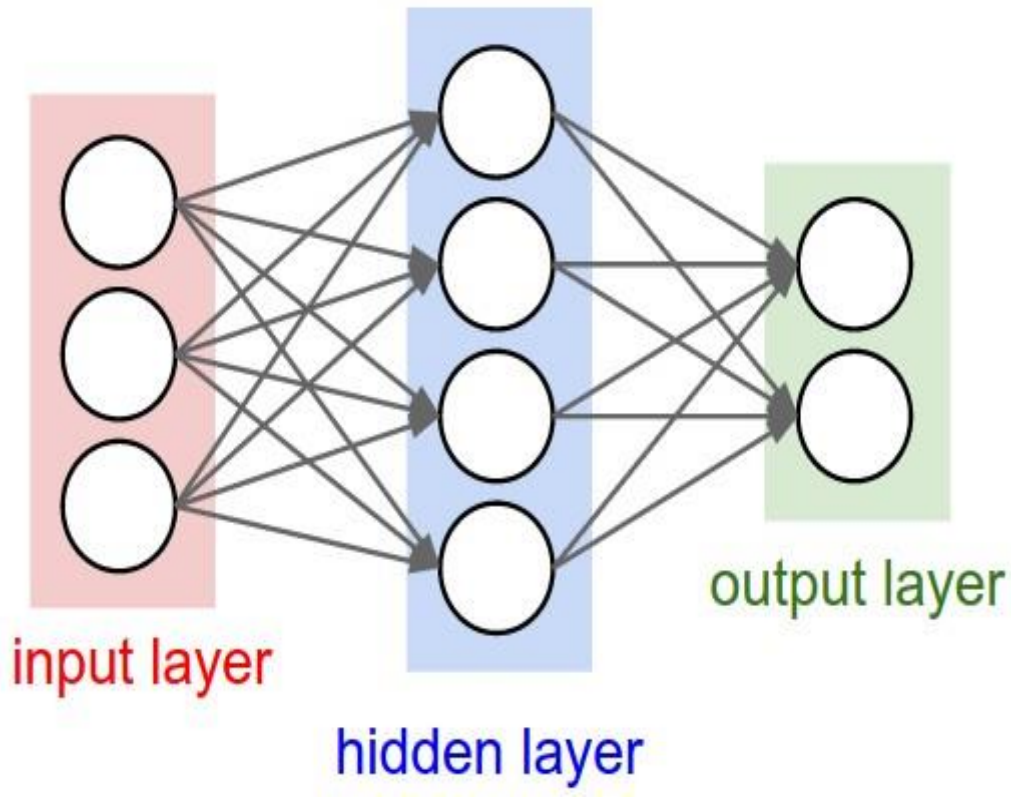
- Neuron là đơn vị cơ bản cấu tạo nên hệ thống thần kinh và là một phần qua trong của bộ não người.
- Cấu tạo của mỗi neuron gồm: một thân chứa nhân, các sợi nhánh, sợi trục.
- Bản thân quá trình học của hệ thần kinh là quá trình điều chỉnh kết nối giữa các neuron thần kinh thông qua các trải nghiệm, một số phản ứng được tăng cường và một số khác bị giảm xuống.

Giải thuật



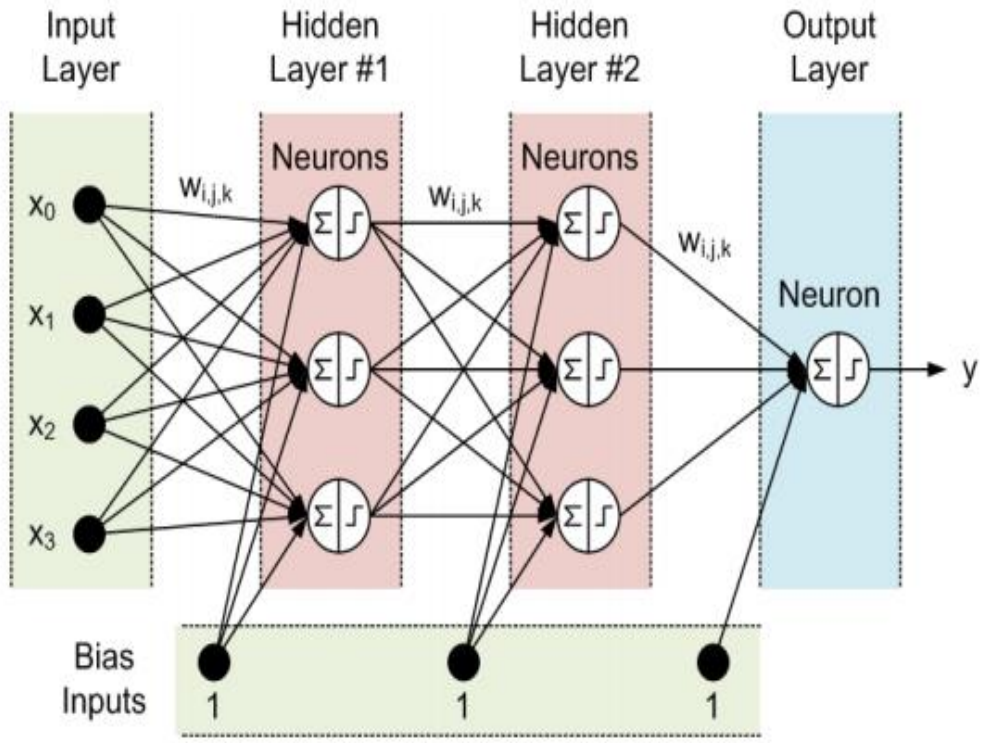
- Tương tự, một neuron nhân tạo cũng là một đơn vị cơ bản cấu tạo lên mạng neuron. Một neoron nhân tạo nhận đầu vào là các x1, x2,.. Và sử dụng một hàm kích hoạt (phổ biến sigmoid, tanh) để tính toán đưa ra một kết quả đầu ra duy nhất.
- Quá trình học của mạng neuron nhân tạo có nét tương đồng đối với neuron thần kinh. Đó là quá trình điều chỉnh các trọng số **w**(weight) và **b**(bias), để đưa ra kết quả đầu ra phù hợp.

Giải thuật



- Kiến trúc của một mạng neuron: mạng neuron thương được chia thành 3 loại lớp.
- Lớp đầu vào (input layer): là lớp đầu tiên của hệ thống, chứa các giá trị thông tin đầu vào. Số lượng neuron bằng với số lượng các thông số của dữ liệu đưa vào.
- Lớp ẩn (Hiddent layer): nằm giữa lớp đầu vào và lớp đầu ra. Số lượng lớp này là tùy chọn, càng nhiều neuron trong lớp ẩn thì mô hình sẽ có năng lực càng cao, tuy nhiên sẽ làm tăng lên khối lượng tính toán.
- Lớp đầu ra (output layer): dùng thông tin từ lớp ẩn để đưa ra kết luận cuối cùng. Số neuron đầu ra sẽ được lựa chọn theo số nhãn lớp cần phân loại.

Giải thuật

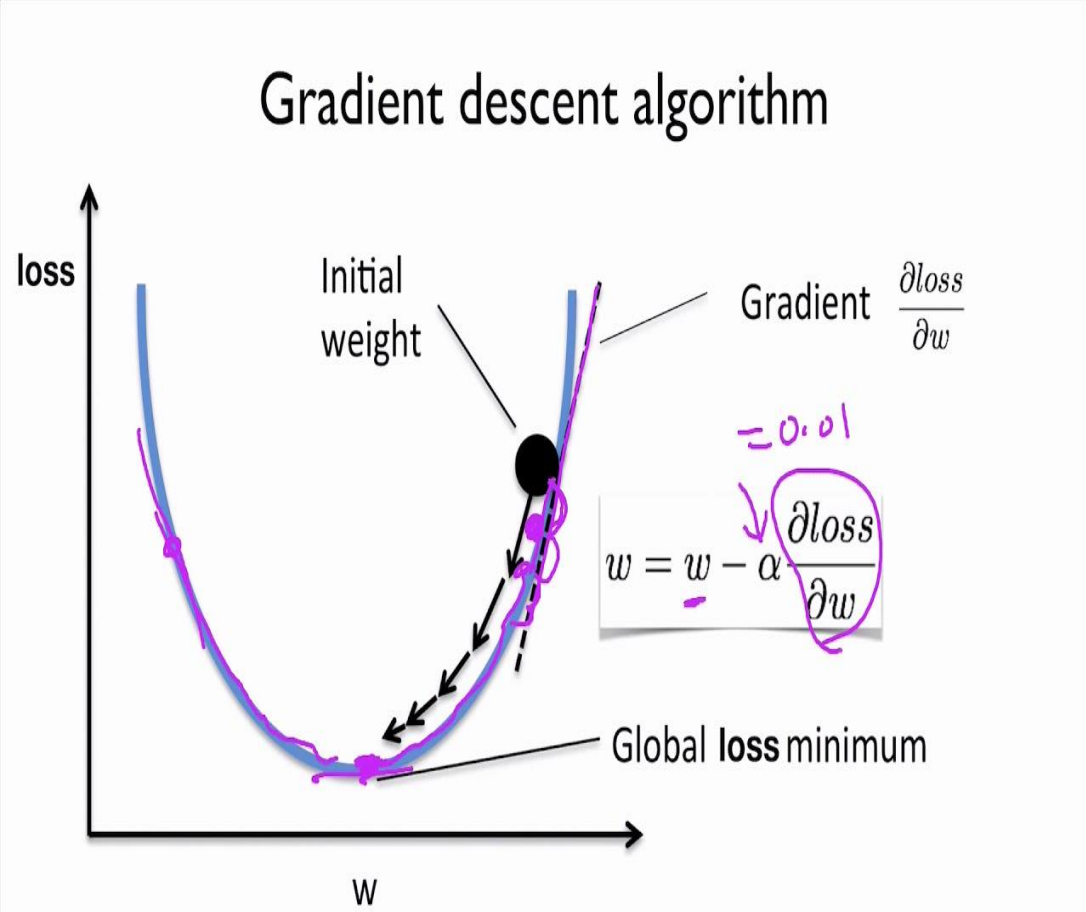


- Lan truyền xuôi (feed forward)
- Đây là giải thuật đơn giản nhất của mạng neuron. Thông tin được lan truyền từ lớp đầu vào, đi qua lớp ẩn và lan truyền tới đầu ra, theo chỉ một chiều và không có chu trình hoặc vòng lặp trong mạng.

$$a_i = activation(\Sigma w_i x_i + b)$$

- Gắn liền với quá trình lan truyền xuôi là hàm kích hoạt (activation function), đối với mỗi neuron trên mỗi tầng cụ thể sẽ nhận dữ liệu đầu vào từ đầu ra của tất cả các neuron ở tầng ngay trước nó và thông qua hàm kích hoạt để tính toán đầu ra cho các neuron tầng tiếp theo. Quá trình này cứ tiếp diễn cho tới khi lan truyền tới tầng đầu ra. Neuron tầng đầu ra cũng sử dụng hàm kích hoạt để đưa ra kết quả cuối cùng.

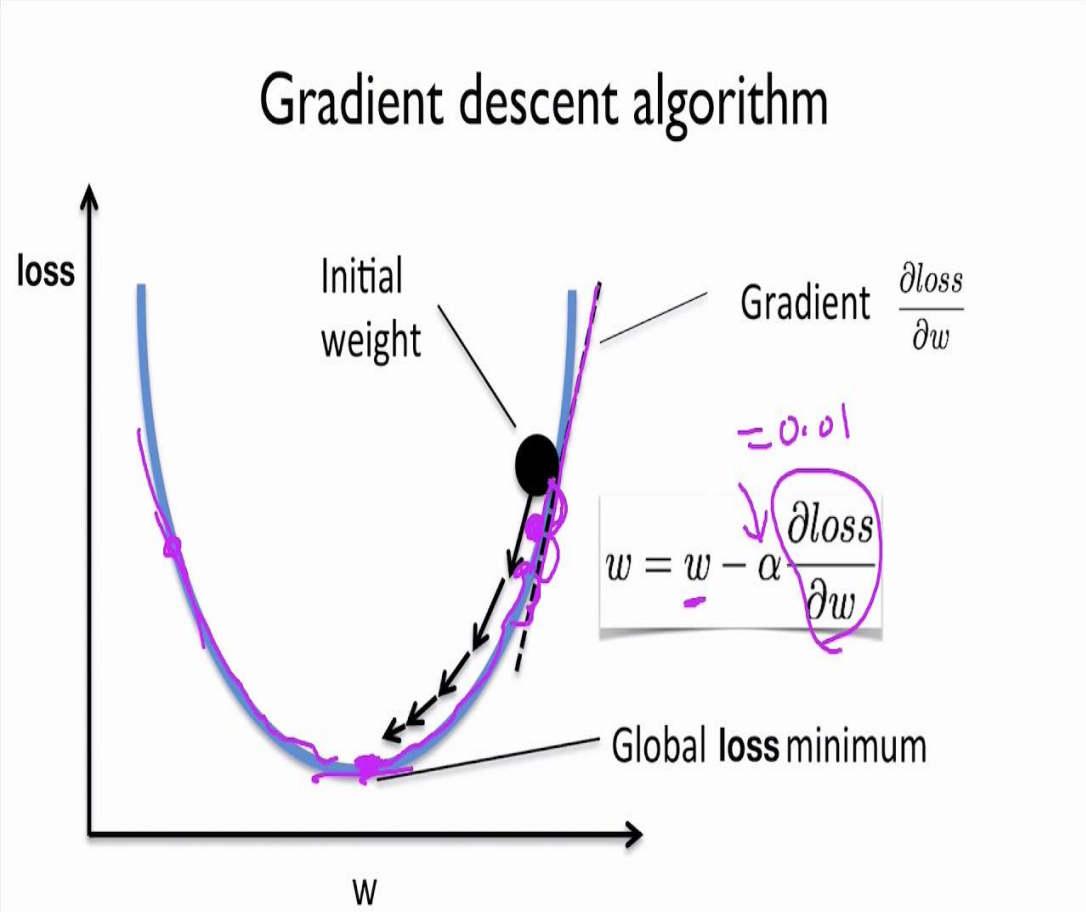
Giải thuật



- Gradient Descent
- Để học được các trọng số **w**(weight) và **b**(bias) giúp cho mạng neuron có thể phán đoán kết quả tương lai tốt nhất tức là sai số đầu ra phải nhỏ nhất cho mỗi quan sát. Tiêu chuẩn thường dùng là hàm lỗi (**loss function**). Hàm lỗi sẽ dùng để so sánh sai số giữa kết quả đầu ra và kết quả thực sự của mỗi ví dụ học.
- Hàm lỗi thường được sử dụng là hàm **Quadratic loss function.**

$$C(w,b) \equiv \frac{1}{2n}\sum_x \|y(x) - a\|^2.$$

Giải thuật



- Mục tiêu của việc học là đi tối ưu hàm lỗi, khi đó các trọng số **w**(weight) và **b**(bias) sẽ được học và sau đó sẽ được sử dụng để mạng neuron phán đoán các kết quả trong tương lai.
- Gradient descent là phương pháp phổ biến được sử dụng để tối ưu hàm lỗi. Gradient descent của hàm lỗi C(w, b) là một vector có hướng.

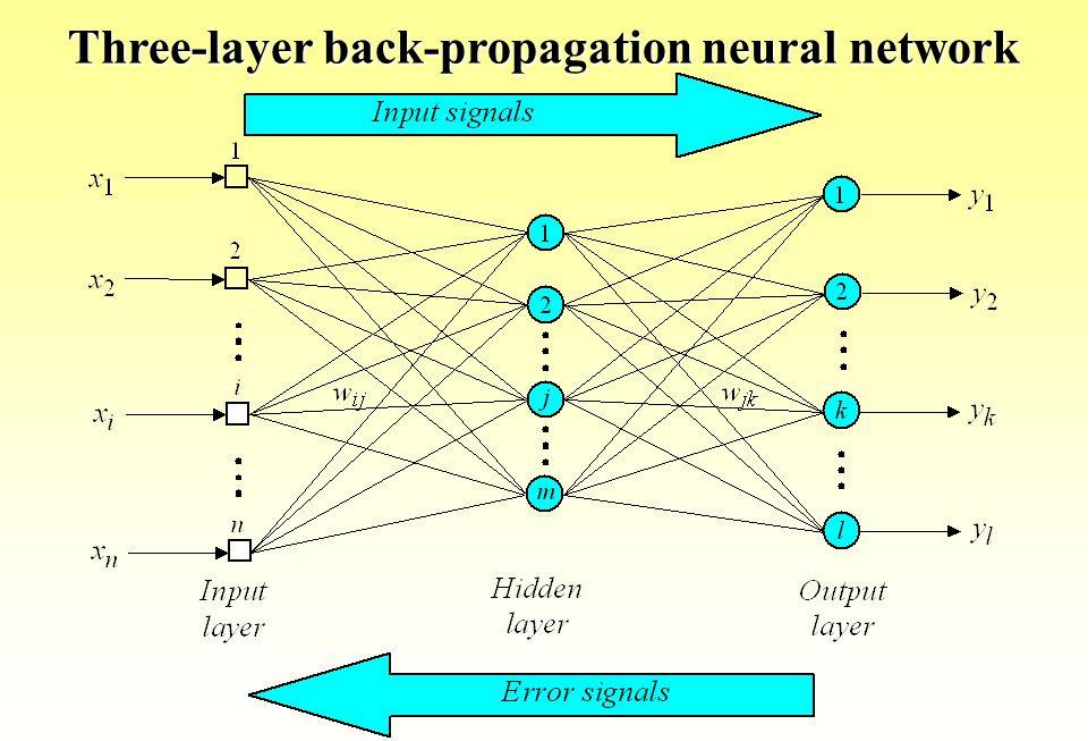
$$\nabla C = (\partial C/\partial w,\ \partial C/\partial b)$$

- Sử dụng Gradient descent để hội tụ hàm lỗi ta lặp đi lặp lại việc cập nhật giá trị các trọng số **w** và **b.**

$$w_k \rightarrow w_k' = w_k - \frac{\eta}{m}\sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \rightarrow b_l' = b_l - \frac{\eta}{m}\sum_j \frac{\partial C_{X_j}}{\partial b_l},$$

Giải thuật



- Lan truyền ngược (back propagation).
- Mục đích của gradient descent là để tính các trọng số **w**(weight), **b**(bias) qua đó làm hội tụ hàm lỗi. Nếu chỉ dựa vào giải tích để tính đạo hàm của hàm lỗi theo các trọng số thì khối lượng tính toán sẽ là rất lớn, chưa kể một số hàm lỗi còn không thể tính được gradient descent. Lan truyền ngược (back propagation) là phương pháp hiệu quả để giải quyết vấn đề trên.
- Ý tưởng: từ một ví dụ học x có nhãn đúng y sau khi cho lan truyền tiến qua mạng ta thu được đầu ra suy diễn h. Để học được các trọng số w và b thì ta phải làm cho sai số giữa y và h là nhỏ nhất. Sai số này người ta gọi là lỗi (error) và được kí hiệu là δ.

Giải thuật

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L).$$

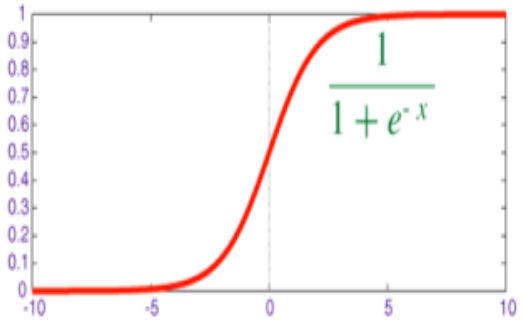
$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l),$$

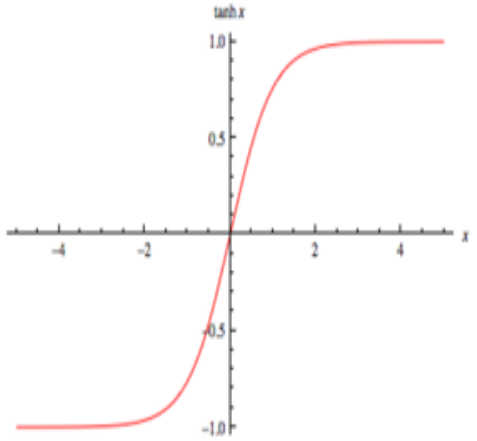
$$\frac{\partial C}{\partial b_j^l} = \delta_j^l.$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l.$$

- Lan truyền ngược (back propagation).
- Lỗi được tính ở mỗi neuron lớp đầu ra bằng công thức đầu tiên. Trong đó C là hàm lỗi, a là kết quả đầu ra suy diễn từ hàm kích hoạt.
- Tại mỗi bước lan truyền ngược, lỗi tầng trước sẽ được tính dựa trên lỗi của tầng sau theo công thức thứ hai.
- Sau khi có được lỗi tại mỗi tầng, ta tiến hành tính gradient descent tại mỗi tầng theo công thức thứ ba và thứ tư.

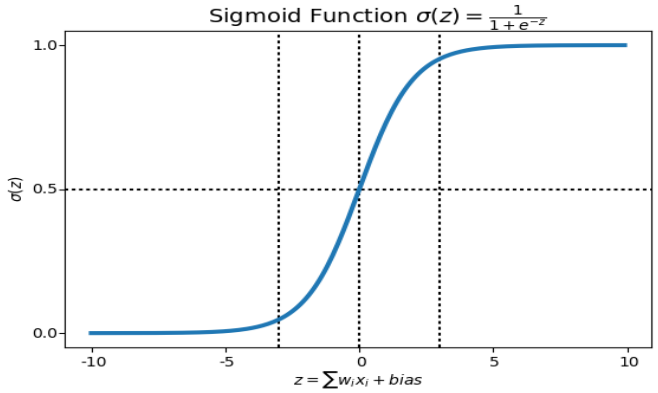
Giải thuật





- Neuron bão hòa.
- Là hiện tượng thường xảy ra khi mỗi neuron đầu ra cho kết quả gần với giá trị lớn nhất hoặc nhỏ nhất trong khi kết quả đúng là ngược lại.
- Các hàm kích hoạt phổ biến hay được sử dụng như hàm sigmoid(), tanh() đồ thị hàm số của các hàm sẽ có độ dốc rất thấp khi đầu ra gần với 0 hoặc 1 đối với hàm sigmoid(), và -1 hoặc 1 đối với hàm tanh() nên giá trị đạo hàm của hàm số theo các biến sẽ nhỏ.
- Dẫn đến tốc độ học **w**(weight) và **b**(bias) sẽ chậm khi sử dụng gradient descent (do phải tính đạo hàm).

Giải thuật



$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2.$$

$$\frac{\partial C}{\partial w} = (a - y)\sigma'(z)x$$

$$\frac{\partial C}{\partial b} = (a - y)\sigma'(z)$$

- Neuron bão hòa.
- Khi sử dụng hàm kích hoạt là sigmoid() và hàm lỗi Quadratic Loss Function. Để tính gradient descent ta gặp phải một vấn đề đó là trong biểu thức tồn tại σ'(z) liên hệ với đồ thị hàm sigmoid(), khi giá trị xấp xỉ 0 hoặc 1 thì đạo hàm của nó sẽ rất nhỏ, dẫn tới mạng học các trọng số sẽ rất chậm.
- Để giải quyết vấn đề này có một phương pháp thay thế hàm lỗi Quadratic thành hàm Cross- Entropy.

Giải thuật

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)],$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z) - y).$$

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_x (\sigma(z) - y).$$

- Neuron bão hòa
- Đạo hàm của Cross Entropy theo **w**(weight) và **b**(bias) loại bỏ được σ’(z) ra khỏi biểu thức, giúp ngăn ngừa hiện tượng neuron bão hòa ở lớp đầu ra.

Giải thuật

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l),$$

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}.$$

$$z = \sum_j w_j x_j + b$$

- Neuron bão hòa
- Tuy nhiên sử dụng Cross Entropy chỉ giúp mô hình giải quyết được vấn đề ở neuron lớp đầu ra, mà lại gần như không có tác dụng đối với neuron ở các lớp ẩn (hidden layer). Lý do là bởi vì trong quá trình học mạng neuron cần phải thực hiện bước lan truyền ngược lại lỗi để tính gradient descent ở các tầng trước. Tại mỗi lớp lỗi được tính từ lỗi của tầng sau liền kề với nó theo công thức thứ nhất.
- Công thức tính lỗi ở mỗi lớp ẩn trong mạng neuron trên tồn tại σ'(z) nên không thể tránh được neuron ở các tầng ần gặp phải hiện tượng bão hòa, khi hàm sigmoid() có giá trị gần với 0 hoặc 1.

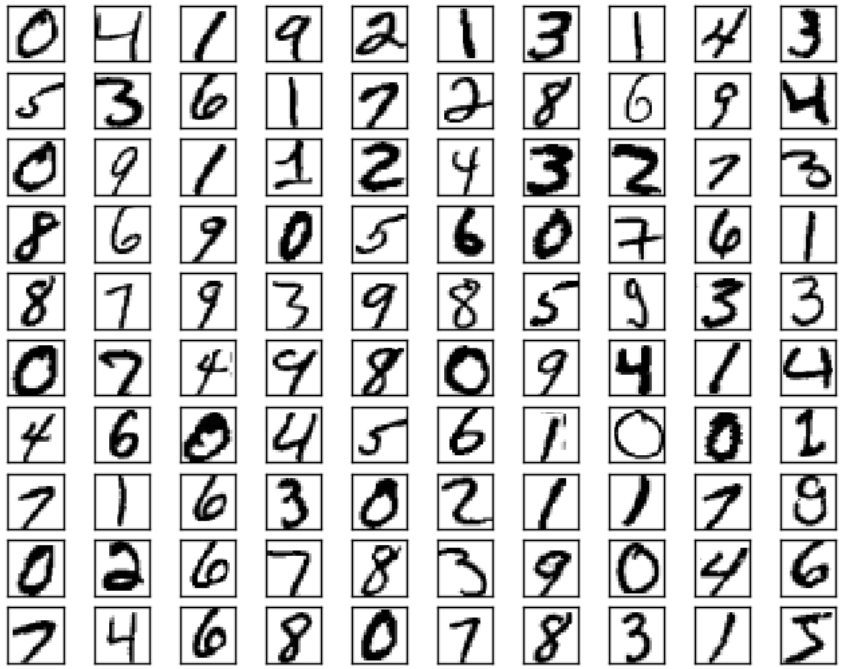
Giải thuật

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}.$$

$$Z = \sum w_j x_j + b$$

- Neuron bão hòa
- Tuy nhiên có một kĩ thuật có thể khắc phục trong trường hợp này. Khi ta nhìn vào hàm sigmoid().
- Ta sẽ thấy hàm nhận các giá trị là 0 hoặc 1 khi Z tiến dần tới âm vô cùng hoặc dương vô cùng, nên để hàm sigmoid() không có giá trị gần với 0 hoặc 1 ta chỉ cần làm cho giá trị của z không quá lớn cũng như không quá bé bằng cách khởi tạo cho các trọng số **w**(weight) các giá trị nhỏ (tiệm cận 0) sẽ giúp cho giá trị của z gần 0, đồng thời sẽ ngăn cho hàm sigmoid() gần 0 hoặc 1.

Triển khai



- Thu thập, biểu diễn dữ liệu
- Dữ liệu sử dụng trong bài tập lớn được nhóm thu thập từ tập dữ liệu mẫu MNIST dữ liệu đã qua tiền xử lý từ trước, bao gồm tập các ảnh 28x28 pixel, và được nhóm biểu diễn thành vector 784 chiều phục vụ cho quá trình huấn luyện mô hình. Từ tập MNIST nhóm chia tập ra làm 3 tập riêng biệt, tập dữ liệu huấn luyện (training dataset) bao gồm 50000 ví dụ, tập dữ liệu tối ưu (validation dataset) bao gồm 10000 ví dụ và tập dữ liệu kiểm thử (test dataset) bao gồm 10000 ví dụ.

Triển khai

```
Epoch 0 training complete
Cost on training data: 0.494966730024
Accuracy on training data: 46855 / 50000
Cost on evaluation data: 0.788382277102
Accuracy on evaluation data: 9401 / 10000

Epoch 1 training complete
Cost on training data: 0.462973365805
Accuracy on training data: 47308 / 50000
Cost on evaluation data: 0.860826651815
Accuracy on evaluation data: 9452 / 10000

Epoch 2 training complete
Cost on training data: 0.468049581399
Accuracy on training data: 47318 / 50000
Cost on evaluation data: 0.917299045798
Accuracy on evaluation data: 9439 / 10000
```

- Giai đoạn huấn luyện.
- Nhóm sử dụng mạng neuron 3 lớp: một lớp đầu vào (input layer) có 784 neuron, một lớp ẩn (hidden layer) có 30 neuron, một lớp đầu ra (output layer) có 10 neuron.
- Ban đầu mạng neuron mới khởi tạo, các trọng số **w**(weight) và **b**(bias) sẽ được khởi tạo mặc định theo phân bố xác suất Gaussian với trung bình là 0 và độ lệch chuẩn là 1. Riêng đối với mỗi **w**(weight) trên mỗi neuron sau khi sử dụng Gaussian giá trị thu được sẽ được chia cho căn bậc hai của số lượng các **w**(weight) trên mỗi tầng (giúp cho giá trị của **w** lúc khởi tạo gần với 0).

Triển khai

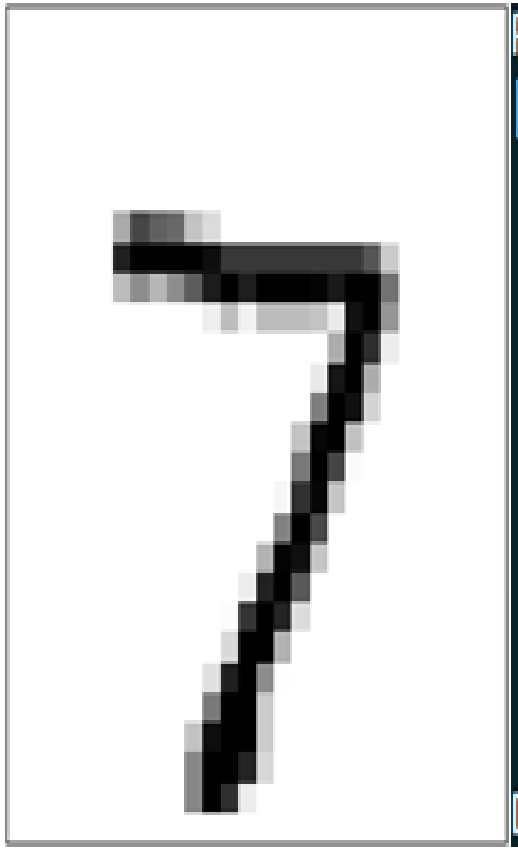
```
Epoch 0 training complete
Cost on training data: 0.494966730024
Accuracy on training data: 46855 / 50000
Cost on evaluation data: 0.788382277102
Accuracy on evaluation data: 9401 / 10000

Epoch 1 training complete
Cost on training data: 0.462973365805
Accuracy on training data: 47308 / 50000
Cost on evaluation data: 0.860826651815
Accuracy on evaluation data: 9452 / 10000

Epoch 2 training complete
Cost on training data: 0.468049581399
Accuracy on training data: 47318 / 50000
Cost on evaluation data: 0.917299045798
Accuracy on evaluation data: 9439 / 10000
```

- Giai đoạn huấn luyện
- Các giá trị của các tham số khác bao gồm:
- Epoch: 30
- Mini-batch size: 10
- Learning rate: 0.5
- Hệ số lamda: 5.0
- Thay vì huấn luyện mô hình trên toàn bộ tập học cùng một lúc, nhóm sử dụng kĩ thuật mini-batch với kích thước là 10 ví dụ học cho 1 quá trình huấn luyện, đối với một epoch tập dữ liệu học bao gồm 50000 ví dụ sẽ được đảo lộn ngẫu nhiên và chia làm nhiều mini-batch với kích thước 10 ví dụ một để cho vào huấn luyện, quá trình lặp lại từ epoch 1 tới epoch 30

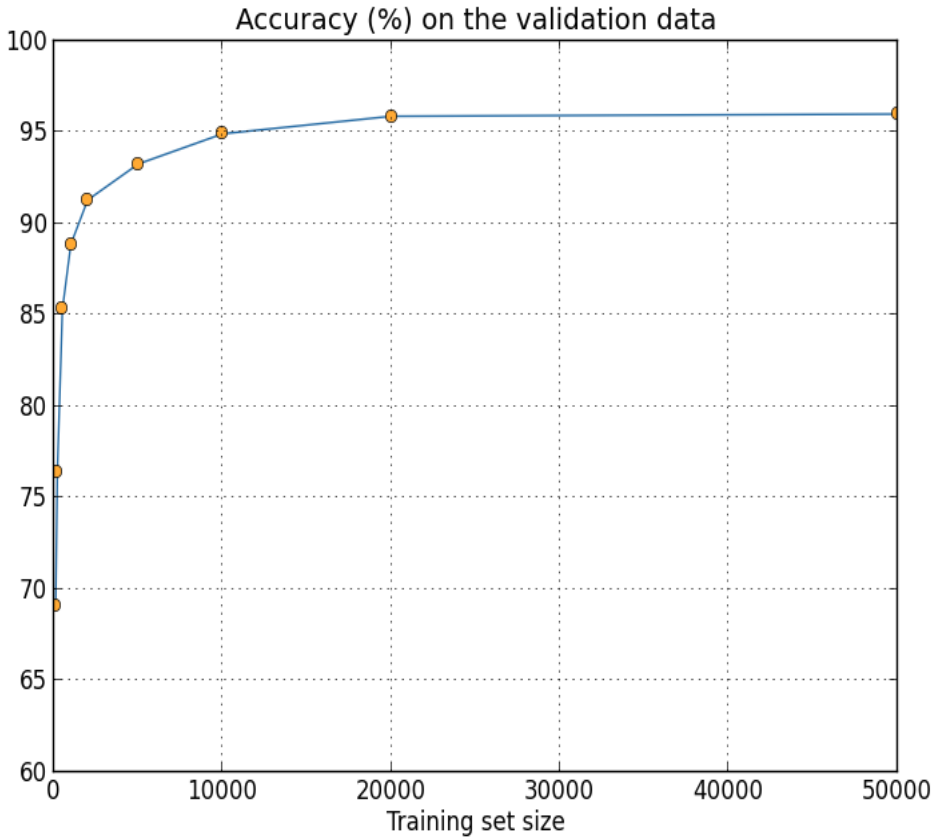
Triển khai



```
Result:
[[  2.23470614e-04]
 [  5.16511752e-05]
 [  7.91049564e-04]
 [  4.78903787e-03]
 [  1.88259604e-06]
 [  5.89510052e-05]
 [  4.64546384e-09]
 [  9.99977173e-01]
 [  6.79965856e-07]
 [  1.21596430e-05]]
Digit:  7
```

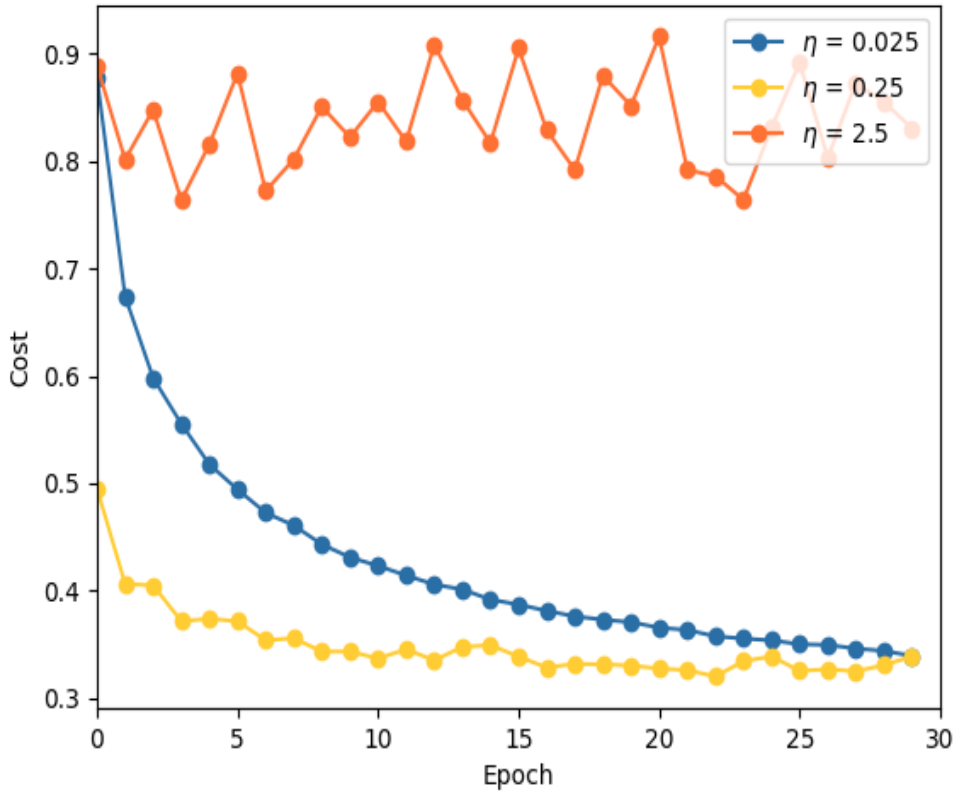
- Giai đoạn phán đoán suy diễn
- Đối với mỗi ví dụ từ tập kiểm thử sau khi đi qua mô hình sẽ cho ra kết quả là một vector 10 chiều, mỗi chiều sẽ có giá trị nằm trọng đoạn [0,1], ví dụ được phân vào nhãn tương ứng với chiều của vector cho giá trị lớn nhất.

Kết quả



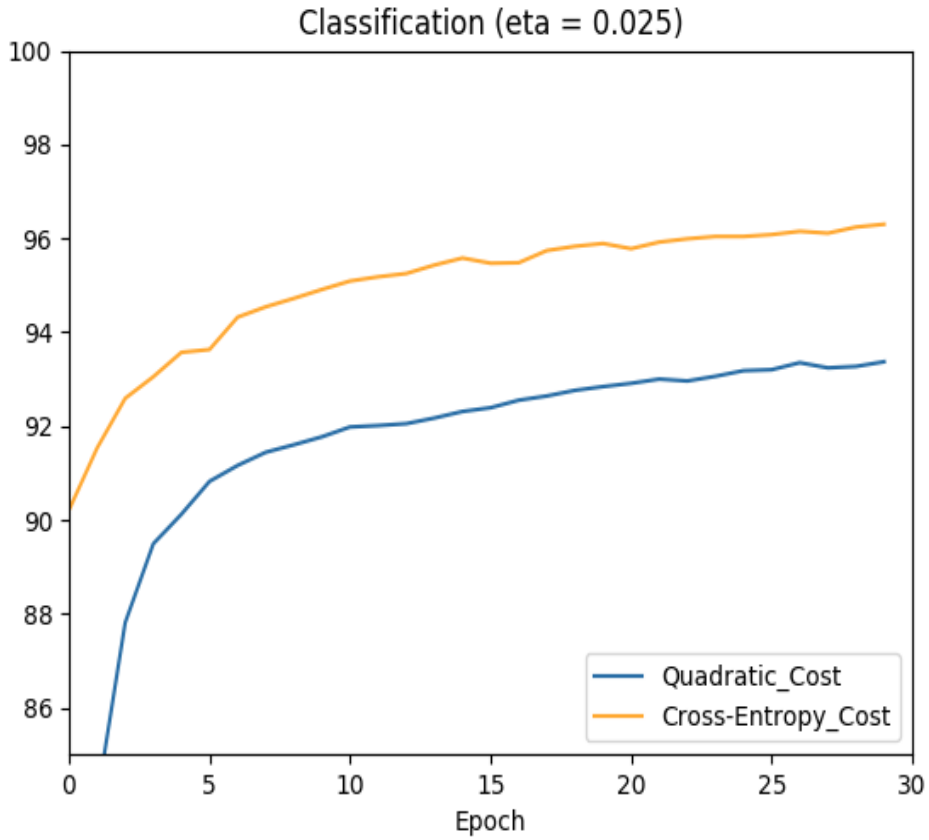
- Độ chính xác mà hệ thống đạt được trên các khối lượng tập học khác nhau.
- Khối lượng tập học càng lớn độ chính xác mà hệ thống đạt được càng cao.
- Đối với tập học có khối lượng 50000 độ chính xác của hệ thống lên tới trên 95%.
- Lưu ý: độ chính xác của hệ thống được đánh giá trên tập tối ưu (validation dataset)

Kết quả



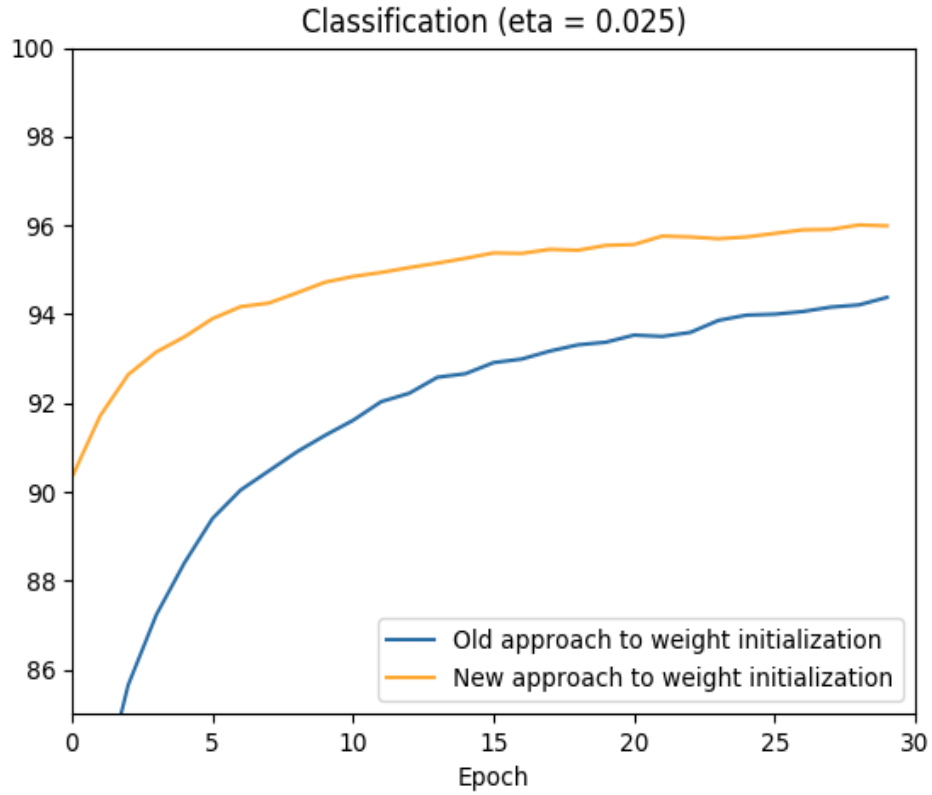
- Tốc độ học của mô hình trên các tỉ lệ học khác nhau.
- Tỷ lệ học khác nhau ảnh hưởng tới tốc độ học của mô hình. Nếu tỷ lệ học quá nhỏ mô hình sẽ học chậm, tỷ lệ học cao mô hình học nhanh nhưng nếu quá cao sẽ dẫn đến trường hợp **gradient descent** không thể chạm được tới cực tiểu toàn cục, vì vậy giá trị hàm lỗi sẽ không giảm mà chỉ giao động quanh một giá trị nào đó.
- Khi η = 2.5 tỉ lệ học quá lớn, hàm lỗi của mô hình chỉ giao động quanh giá trị 0.8 mà không giảm được.
- η = 0.025 giá trị này là quá nhỏ khiến cho mô hình học chậm.
- η = 0.25 là giá trị thích hợp nhất.

Kết quả



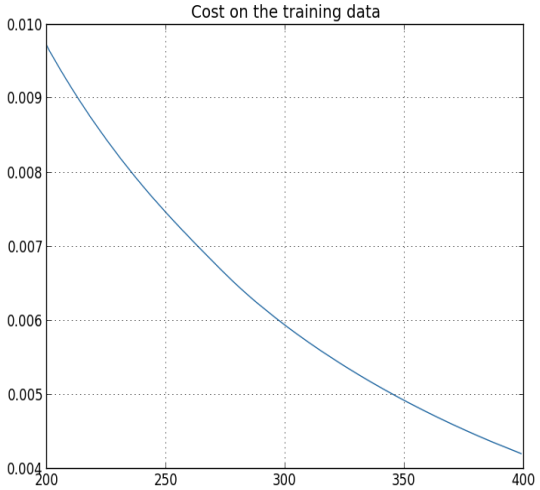
- Tốc độ học của mô hình trên các hàm lỗi
- Hàm lỗi sử dụng để huấn luyện cũng ảnh hưởng một phần không nhỏ tới tộc độ học của mô hình.
- Khi sử dụng hàm Cross-Entropy sẽ cho tốc độ học nhanh hơn hàm Quadratic vì nó tránh được hiện tượng neuron ở lớp đầu ra bị bão hòa.

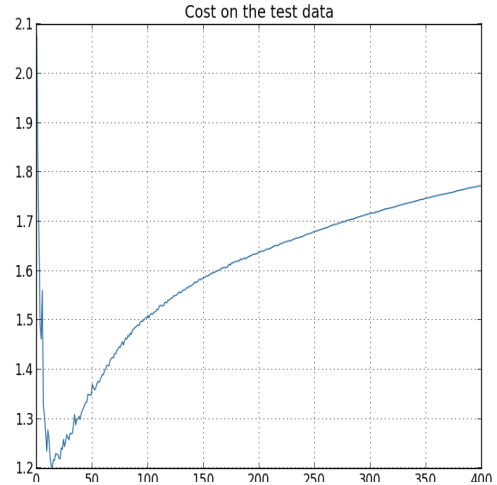
Kết quả

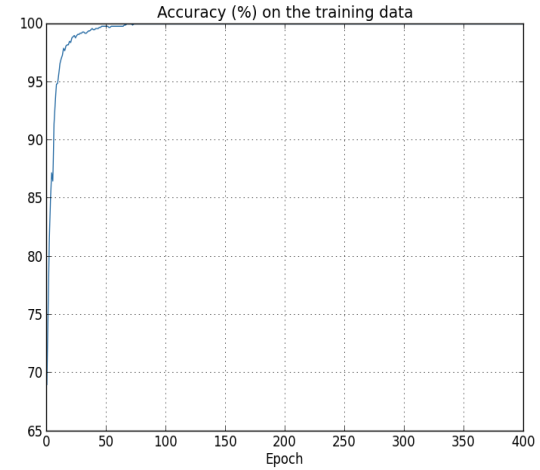


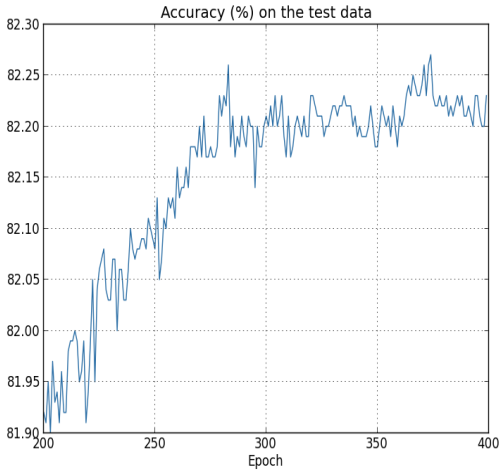
- Kỹ thuật khởi tạo trọng số **w**(weight)
- Khi sử dụng kĩ thuật khởi tạo giá trị **w** ban đầu cho mạng neuron giá trị nhỏ (gần với giá trị 0, cả âm hoặc dương). Mô hình sẽ tránh được hiện tượng neuron bão hòa xảy ra đối với các neuron ở tầng ẩn.
- Mô hình khi sử dụng kĩ thuật khởi tạo giá trị với trọng số **w**(weight) nhỏ (đường màu vàng) cho tốc độ học nhanh hơn so với việc sử dụng khởi tạo trọng số **w** theo phân phối chuẩn Gaussian.
- Khởi tạo **w** của mỗi neuron trên mỗi tầng cụ thể bằng cách khởi tạo giá trị theo phân phối chuẩn Gaussian, giá trị thu được chia cho số lượng neuron trên tầng đó.

Kết quả



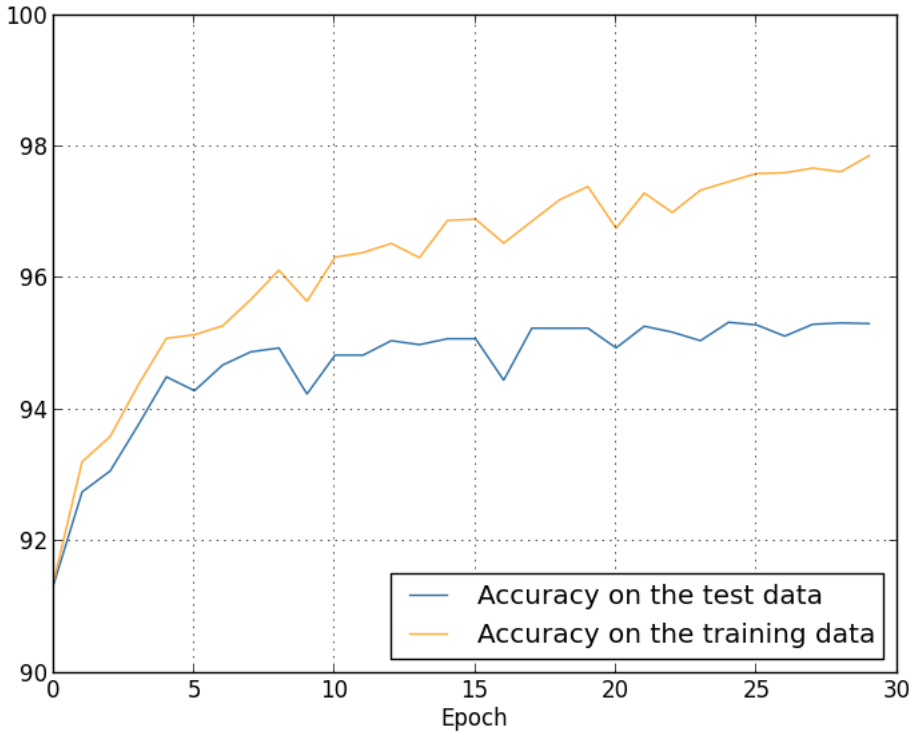






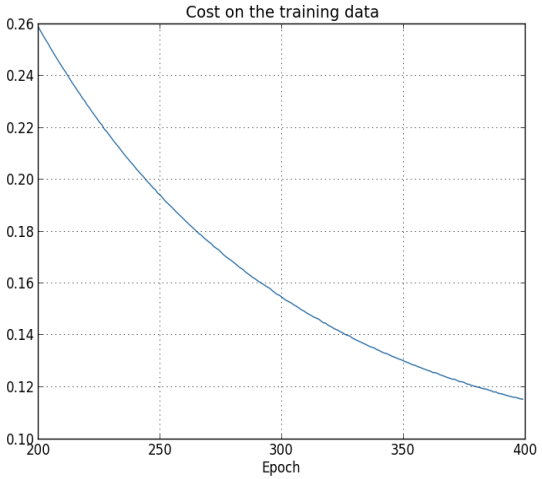
- Overfitting
- Khi mô hình không sử dụng chuẩn hóa, với số lượng neuron trong tầng ẩn lên tới hàng trăm neuron mô hình bắt đầu gặp phải vấn đề overfiting.
- Các tham số sử dụng để huấn luyện mô hình khi gặp phải vấn đề overfitting.
- Epoch: 30
- Mini-batch size: 10
- Learning rate 0.5
- Số tầng ẩn: 1
- Số lượng neuron ở tầng ẩn: 150

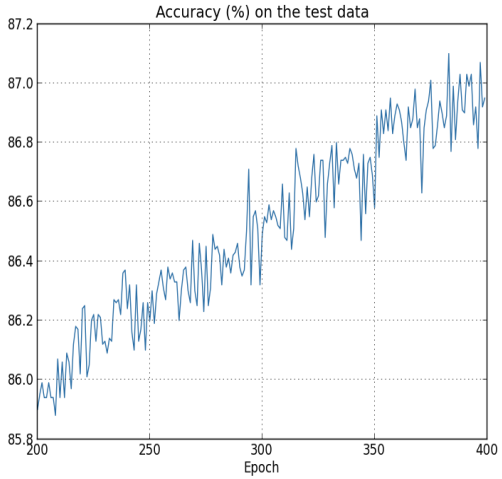
Kết quả

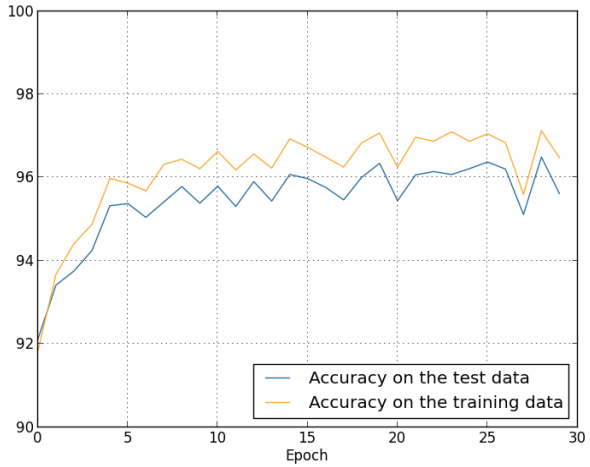


- Overfitting
- Độ chính xác của mô hình đạt được trên tập học và tập thử nghiệm có sự chệnh lệch lớn.
- Khi mô hình bị overfiting, sẽ dẫn đến hiện tượng mô hình cho kết quả phán đoán đạt độ chính xác cao trên tập học, nhưng trên tập thử nghiệm lại cho kết quả không tốt.

Kết quả







- Chuẩn hóa (Regularization)
- Khi sử dụng các kĩ thuật để chuẩn hóa mô hình, ta có thể ngăn ngừa được hiện tượng mô hình bị overfitting. Bằng cách thêm vào hàm lỗi một đại lượng phát R.

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2,$$

- Hệ số lamda nh.
- Kết quả thực nghiệm cho ta thấy mô hình đã giảm được đáng kể sự chệnh lệch độ chính xác đạt được giữa tập dữ liệu huấn luyện và tập dữ liệu kiểm thử. Và khi đó ta nói mô hình đã tránh được hiện tượng overfitting.

Kết quả

	Precision	Recall	F1
Nhãn 0	0,96492986	0,982653061	0,973710819
Nhãn 1	0,967297762	0,99030837	0,978667828
Nhãn 2	0,94581749	0,964147287	0,954894434
Nhãn 3	0,950048972	0,96039604	0,955194485
Nhãn 4	0,937438905	0,976578411	0,956608479
Nhãn 5	0,978873239	0,934977578	0,956422018
Nhãn 6	0,955942623	0,973903967	0,96483971
Nhãn 7	0,958538993	0,944552529	0,951494366
Nhãn 8	0,952723535	0,95174538	0,952234206
Nhãn 9	0,978494624	0,901883053	0,938628159

- Đánh giá mô hình trên các độ đo khác.
- Precision: là độ đo đánh giá chất lượng của một dự đoán đối với một nhãn cụ thể. Công thức tính: ***Precision = TP/(TP + FP).***
- Recall: là độ đo đánh giá khả năng tìm thấy một nhãn trong dữ liệu được tính bởi công thức: ***Recall = TP/(TP + FN).***
- F1: Trung bình điều hòa, công thức tính:

  F1 = 2*precision*recall / (precision + recall)
- *Chú thích:*
- *TP: Số lượng khẳng định đúng*
- *FP*: Số lượng khẳng định sai.
- *FN*: Số lượng phủ định sai.

Kết luận

- Khó khăn
- Các thành viên trong nhóm chưa có đủ kiến thức, kinh nghiệm xây dựng một hệ thống học máy.
- Yếu tố thời gian, với thời gian một kì học chỉ vỏn vẹn trong chưa đầy 5 tháng, nhóm phải dành trải cho nhiều bài tập lớn của các môn học khác không cho phép nhóm có đủ thời gian để đầu tư tìm hiểu, tự tay xây dựng và hoàn thiện hết tất cả các pha trong một hệ thống học máy đầy đủ.

- Hạn chế
- Hệ thống vẫn chưa tự thu thập và tiền xử lý được dữ liệu thực mà vẫn còn phải sử dụng dữ liệu mẫu có sẵn đã qua tiền xử lý từ trước.
- Hệ thống mới chỉ gán nhãn được các kí tự là chữ số viết tay, vẫn chưa gán nhãn được hết tất cả các kí tự viết tay.
- Hướng phát triển
- Từ hệ thống mà nhóm xây dựng, nhóm có thể từ đó phát triển hệ thống, khắc phục các hạn chế đã nêu trên
- Ngoài ra nhóm sẽ áp dụng các kĩ thuật của Deep Learning (sử dụng mạng neuron ConvNet) để cái tiến hệ thống, nâng cao độ chính xác.


The End