Improving Softmax Regression on MNIST

Last update: November 9, 2024

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

Stochastic Gradient Descent (SGD)

Error function on training data

$$E_{train}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} e\left(h_{\mathbf{w}}(\mathbf{x}^{(n)}), y^{(n)}\right)$$

Minimize $E_{train}(\mathbf{w})$

Gradient Descent (GD):

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla E_{train}(\mathbf{w})$$

$$\nabla E_{train}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \nabla e\left(h_{\mathbf{w}}(\mathbf{x}^{(n)}), y^{(n)}\right)$$

To take a step, GD needs to go through N training examples

→ Slow when N is large

Minimize $E_{train}(\mathbf{w})$

Stochastic Gradient Descent (SGD): let's use a subset of B $(B \ll N)$ examples — called a *mini-batch* — to estimate $\nabla E_{train}(\mathbf{w})$

Q: How should we choose B examples from N examples?

A: Choose randomly

Q: What is the effect of B on the quality of estimating $\nabla E_{train}(\mathbf{w})$?

A: The larger B, the better quality of estimating $\nabla E_{train}(\mathbf{w})$

Minimize $E_{train}(\mathbf{w})$

SGD (left) vs GD (right)

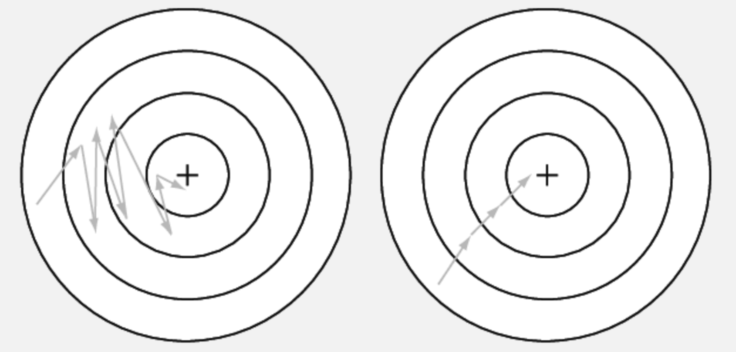


Image source: https://datascience.stackexchange.com/a/94355

Minimize $E_{train}(\mathbf{w})$

Q: So, with a step of SGD, we do it faster than GD, but we pay the price of worse quality. Is it worth it?

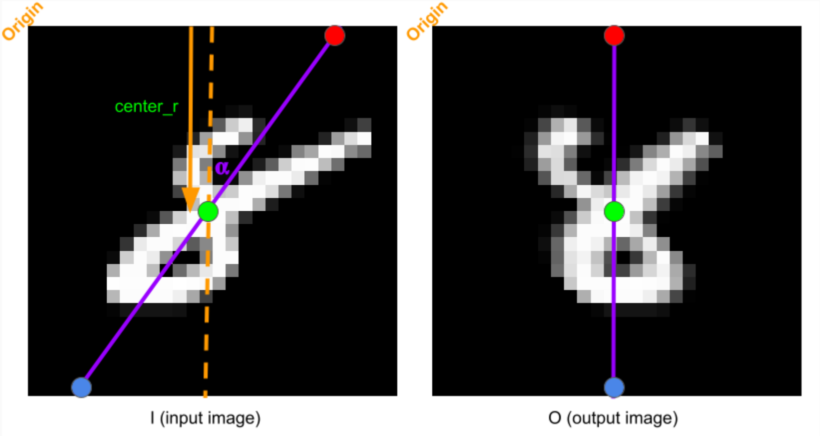
A: Yes. In reality, people often see that: with the same starting point and the same amount of time, SGD runs much more steps than GD and ends up with much smaller E_{train} than GD

Minimize $E_{train}(\mathbf{w})$

Sketch of SGD implementation:

1. Initialize **w**
2. Repeat until termination criteria are satisfied:
    a. Shuffle the order of training examples
    b. For mini-batch $b = 1, ..., \frac{N}{B}$:
    $\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{1}{B} \sum_{n=(b-1)B+1}^{bB} \nabla e\left(h_{\mathbf{w}}(\mathbf{x}^{(n)}), y^{(n)}\right)$

Deslanting digit image

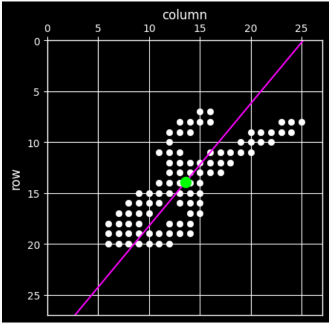


Q: O[r, c] = I[r, **?**]

Hint: **?** can be computed from c, center_r - r, tan α

A: **?** = c + (center_r - r) $\times$ tan α

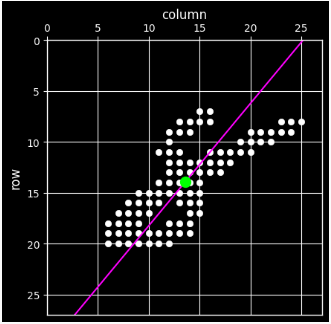
How to find the green point (center_r) and the violet line ($\tan\alpha$)?



A data point (a white point) represents row index and column index of a pixel corresponding to pen stroke

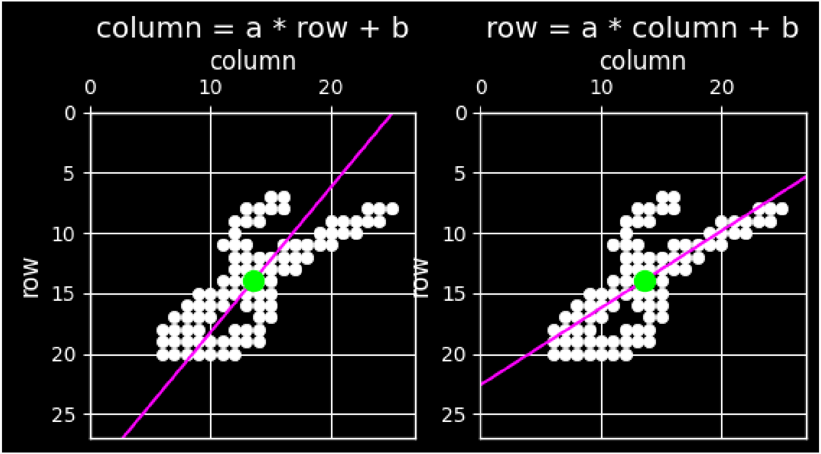
The green point is simply the average of all data points!

How to find the green point (center_r) and the violet line (tan α)?

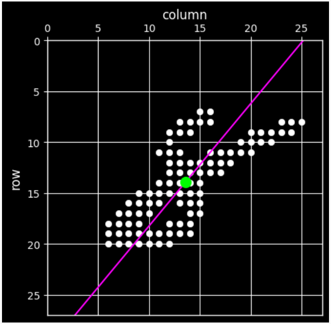


A data point (a white point) represents row index and column index of a pixel corresponding to pen stroke

The violet line is the best-fit line for all data points!
Q: Which is the form of this line: (1) row = a × column + b, or (2) column = a × row + b?


column = a * row + b
row = a * column + b
column
column
0
10
20
0
10
20
0
5
10
15
20
25
0
5
10
15
20
25
row
row

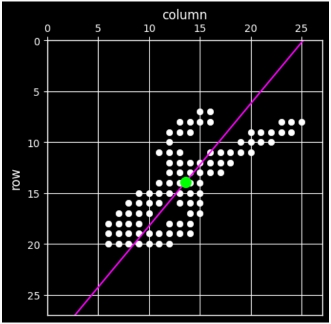
How to find the green point (center_r) and the violet line (tan α)?



A data point (a white point) represents row index and column index of a pixel corresponding to pen stroke

The violet line is the best-fit line for all data points!
Q: Which is the form of this line: (1) row = a $\times$ column + b, or (2) column = a $\times$ row + b?
A: (2)

How to find the green point (center_r) and the violet line (tan α)?



A data point (a white point) represents row index and column index of a pixel corresponding to pen stroke

The violet line is the best-fit line for all data points!
Q: Assume we have found the best-fit line “column = a $\times$ row + b”. What is the formula to compute tan α from a?
A: tan α = -a