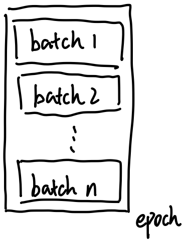
陈翊条 YiqiaoChen 2024/10/20

① Training Phase

I. Data Set



epoch-by-epoch

the entire training process may repeat the epoch thousands or millions of time.

a) Epoch:

one complete presentation of the entire training set during the learning process.

Randomizing the order of training samples from one epoch to the next tends to avoid the oscillation

(and NN may learn the order of the data samples)

b) Sequential Learing

Weight updating is performed after the presentation of each training samples

Advantages

i. Less Memory Requirements:

Sequential learning don't need to accumulate errors $\{\mathcal{E}(n)\}_{n}$ to update like batch learning

ii. Redundancy Utilization:

Each training samples provides a gradient to update the weight, which is batch.size times more frequent than batch learning

iii. Online Stochastic Learning

Disadvantages

i. No Theoretical Convergence:

The convergence derivation is all based on batch learning

ii. Limited Parallelization

iii. Sensitivity to Data Order:

NN might learn specific patterns of data order, rather than the generalized feature of the data itself.

C) Batch learning

weights are update after the presentation of all training samples in a set which is named as batch.

Derivation: $\frac{\partial \mathcal{E}_{av}}{\partial e_j} \cdot \frac{\partial e_j}{\partial w_{ji}}$

dim: 1×1

$$= \left[\boxed{\frac{\partial \mathcal{E}_{av}}{\partial e_j(1)}} \; \frac{\partial \mathcal{E}_{av}}{\partial e_j(2)} \cdots \frac{\partial \mathcal{E}_{av}}{\partial e_j(N)} \right] \begin{bmatrix} \frac{\partial e_j(1)}{\partial w_{ji}} \\ \frac{\partial e_j(2)}{\partial w_{ji}} \\ \vdots \\ \frac{\partial e_j(N)}{\partial w_{ji}} \end{bmatrix}$$

$$= \sum_{i=1}^{N} \frac{\partial \mathcal{E}_{av}}{\partial e_j(i)} \frac{\partial e_j(i)}{\partial w_{ji}}$$

$$= \sum_{i=1}^{N} \frac{e_j(i)}{N} \cdot \frac{\partial e_j(i)}{\partial w_{ji}}$$

batch 1
⋮
batch m
⋮
batch n

in m'th update

$$\mathcal{E}_{av} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{E}(i) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j\in C} e_j^2(i)$$

$$\Delta w_{ji} = -\eta \frac{\partial \mathcal{E}_{av}}{\partial w_{ji}} = -\frac{\eta}{N}\sum_{i=1}^{N} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}}$$

Advantages

i. Accurate Gradients for Convergence:
The gradient used for batch update is the average of the gradients, which is more statistically representative

ii. Parallelization

iii. Improved regularization

Disadvantages

i. Limited Redundancy Utilization

ii. Memory Requirements

☆ iii. Initialization Sensitivity

iv. Stalling in Local Minima

II. Stopping Criteria

GD is considered to have converged when

① $\|\nabla_w \mathcal{E}_{av}\|$ reaches a sufficiently small gradient threshold

② $|\Delta \mathcal{E}_{av}|$ is sufficiently small (0.1% to 1%)

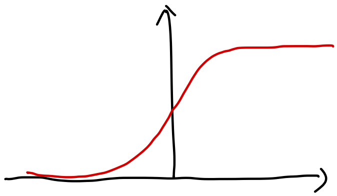
② Training Improvement

I. Stochastic versus batch update

II. Maximizing information content. Use training samples that
1) result in the largest training error
2) are radically different from all those previously used

distinct samples

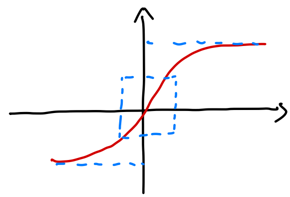
III. Activation function



Logistic function

i. $\varphi(x) = \dfrac{1}{1+e^{-ax}}$, $a>0$

ii. $\varphi'(x) = a\varphi(x)(1-\varphi(x))$



Hyperbolic Tangent

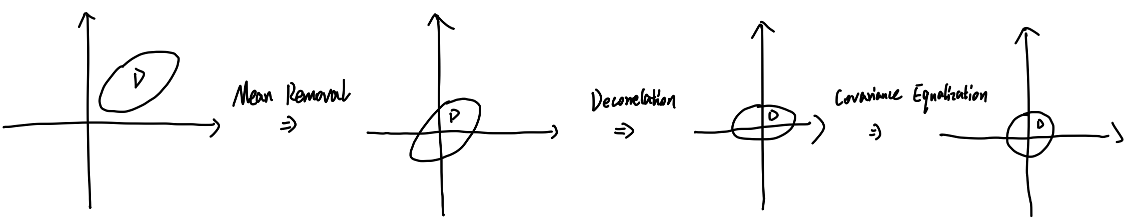
i. $\varphi(x) = a\tanh(bx)$, $(a,b)>0$

ii. $\varphi'(x) = \dfrac{b}{a}(a-\varphi(x))(a+\varphi(x))$

Using an antisymmetric activation function to learn faster

☆ Because the network is more balanced in handling positive and negative inputs to reducing bias, and helping to train data centered around an output of 0.

IV. Normalizing the inputs



1) Mean Removal (Subtract the mean)

i. Makes the input of the activation function in the range close to 0, where the gradient is larger.

a. Accelerate the convergence rate of gradient descent

b. Improve numerical stability to avoid gradient vanishing

ii. Eliminate bias and let the NN focus on changes in the data.

2) Decorrelation (PCA)

i. Equalizes weight learning speed

ii. Prevent overfitting by reduce the data dimensionality, decrease the number of parameter.

3) Covariance Equalization

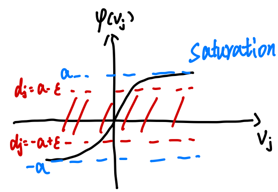
i. Normalize feature scales to ensure the weights are updated in a equalize magnitude, accelerating model convergence

ii. Make the feature space more uniform.

☆ V. Weight initialization

we want v_j falls within the red range to achieve a better gradient

⇒



$$v_j = \sum_{i=1}^{m} w_{ji} y_i$$

Considering v_j, y_i, w_{ji} as random variables, we have:

$$\mu_v = \mathbb{E}[v_j] = \mathbb{E}\left[\sum_{i=1}^{m} w_{ji} y_i\right]$$

Assumption 1:

$\mu_y = \mathbb{E}[y_i] = 0$

$\sigma_y^2 = \mathbb{E}[(y_i - \mu_y)^2]$

$= \mathbb{E}[y_i^2] = 1$

$$= \sum_{i=1}^{m} \mathbb{E}[w_{ji}]\,\mathbb{E}[y_i]$$

$$= 0$$

$$\sigma_v^2 = \mathbb{E}[(v_j - \mu_v)^2]$$

$$= \mathbb{E}[v_j^2]$$

$$= \mathbb{E}\left[\sum_{i=1}^{m} \sum_{k=1}^{m} w_{ji} w_{jk} y_i y_k\right]$$

$$= \sum_{i=1}^{m} \sum_{k=1}^{m} \mathbb{E}[w_{ji} w_{jk}] \, \underline{\mathbb{E}[y_i y_k]}$$

Assumption 2: Decorrelation step

$$\mathbb{E}[y_i y_k] = \begin{cases} 1, & k = i \\ 0, & k \neq i \end{cases}$$

$$= \sum_{i=1}^{m} \mathbb{E}[w_{ji}^2]$$

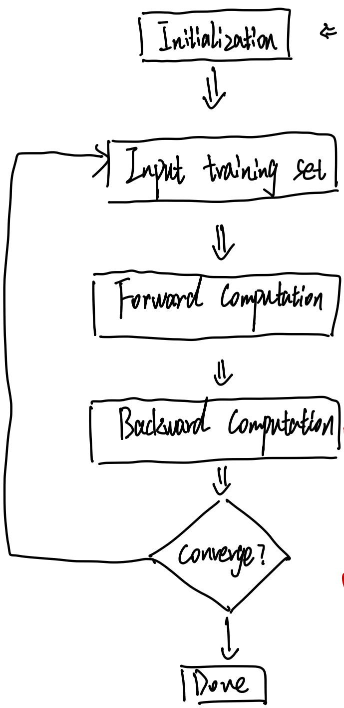
$$= m \sigma_w^2$$

if $\sigma_v = 1$, we have $\sigma_w^2 = \frac{1}{m}$

thus we randomly initialize weights with $\mu_w = 0$ and $\sigma_w^2 = \frac{1}{m}$.

reciprocal of the number of weight connection

③ Training workflow



$\Leftarrow w = \begin{cases} \mu_w = 0 \\ \sigma_w^2 = \frac{1}{m} \end{cases}$ (Guaranteed gradient)

(shuffle the epoch at each time)

① $\delta_j^{(l)}(n) = \begin{cases} e_j^{(L)}(n)\, \varphi_j'(v_j^{(L)}(n)) & \text{output } j \\ \varphi_j'(v_j^{(l)}(n)) \sum_k \delta_k^{(l+1)}(n)\, w_{kj}^{(l+1)}(n) & \text{hidden } j \end{cases}$

② Adjust weights:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha \Delta w_{ji}^{(l)}(n-1) + \eta \delta_j^{(l)}(n) y_i^{(l-1)}(n)$$