Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2025-26

Class:	AI	Semester:	VII
Course Code:		Course Name:	DL

Name of Student:	BARI ANKIT VINOD
Roll No. :	61
Assignment No.:	2
Title of Assignment:	
Date of Submission:	
Date of Correction:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Completeness	5	
Demonstrated Knowledge	3	
Legibility	2	
Total	10	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Completeness	5	3-4	1-2
Demonstrated Knowledge	3	2	1
Legibility	2	1	0

Checked by

Name of Faculty : Raunak joshi

Signature :

Date :

Assignment No. - 2

1) Compare / contrast the following optimization functions: gradient descent, stochastic gradient desent, SGD with momentum, adagrad, RMS & adam.

(i) Gradient descent (GD) - Compute the gradient of the loss wrt. parameters using the entire dataset, then update parameters in that direction.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

Pros : stable, moves in the true direction of steepest descent.

Cons : very slow for large dataset, high memory cost.

Use cases : small datasets or convex problem.

(ii) Stochastic gradient descent (SGD) -

How it works : updates parameters using one sample at a time.

Pros : much faster, scalable, allows online learning.

Cons : updates are noisy, convergence fluctuates around the optimum.

Use case : deep learning, large-scale dataset.

(iii) SGD with momentum -

How it works : Adds a 'velocity' term to smooth updates instead of using only the current gradient, it accumulates past gradient

$$v_t = \beta v_{t-1} + \eta \nabla_\theta J(\theta)$$

- Pros : Reduces oscillation, speed up convergence.

(iv) Adagrad -

- How it works : uses an adaptive learning rate, scale learning rate by the inverse of the square root of the sum of all past

$$\theta = \theta - \frac{\eta}{\sqrt{G_t + \varepsilon}} \nabla_\theta J(\theta)$$

- Use cases : NLP, sparse data problems.

(v) RMS Prop -

- fixes adgrad's 'shrinkles learning rate' problem by using an exponentially decaying average of past squared gradients instead of all pasts gradient.

$$\theta = \theta - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

- works well in non-stationary problems, stable learning rate.

(vi) Adam - $\theta = \theta - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$

Q. 2) Compare / contrast the following activation fn, linear, sigmoid, tanh, Relu, leaky Relu and softmax.

→ (i) linear activation -

- formula - $f(x) = x$
- Range - $(-\infty, +\infty)$
- Simple, no vanishing gradient.

(ii) Sigmoid (logistic) -

- formula - $f(x) = 1/1 + e^{-x}$
- Range : $(0, 1)$
- smooth, interpretability as probability.

(iii) Tanh (Hyperbolic Tangent) -

- formula - $f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
- Range - $(-1, 1)$
- Still suffers from vanishing gradient at large $|x|$.

(iv) ReLU (Rectified linear unit) -

- formula - $f(x) - \max(0, x)$
- Range - $[0, +\infty]$
- Pros - fast, simple; reduces vanishing gradient, sparse activation.
- Most modern deep neural network

(v) Leaky ReLU -

- formula - $f(x) = x$ if $x > 0$, else αx (with $\alpha = 0.01$)
- Range - $(-\infty, +\infty)$
- fixes dying ReLU by allowing small negative slope.
- CNN's, deep networks when ReLU struggles.

(vi) Softmax -

- formula - $f(x_i) = e^{x_i} / \Sigma_j e^{x_j}$ (for vector input)
- Range - $[0, 1]$ with sum $= 1$
- Pros - Converts to outliers, can saturate.
- Cons - sensitive to outliers, can saturates.
- Use cases - output layer for multi-class classification.