No.

Date

Sahil Sheikh, A20518693, FALL 22

Assignment 3

* Loss

Q.1

$$L1\ (loss) = \sum_{j=1}^{k} \left| \hat{y}_j^{(i)} - y_j^{(i)} \right|$$

- L1 loss also known as least Absolute Derivation

$$L2\ (loss) = \sum_{j=1}^{k} \left(\hat{y}_j^{(i)} - y_j^{(i)} \right)^2$$

- L2 loss are also known as Least Square Error.

Huber (loss) =

$$L_\delta(d) = \begin{cases} \frac{1}{2} d^2 & \text{if } |d| \le \delta \\ \delta\left(d - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases}$$

$$\text{Log-cosh (loss)} = \sum_{j=1}^{k} \log\left(\cosh\left(\hat{y}_j^{(i)} - y_j^{(i)}\right)\right)$$

log-cosh of d

$$\log(\cosh(d)) \approx \begin{cases} d^2/2 & \text{if } d \text{ is small} \\ |d| - \log(2) & \text{otherwise} \end{cases}$$

- How each loss is calculated, defines the difference b/t them.

→ Advantages of each loss:

L1: Not effected by outliners in dataset. It is robust

L2: Always provides one solution, which is stable.

Huber: is a combination of both MSE & MAE, for large loss values migrates weight in linear manner & for small loss maintain quadratic func.

Log-cosh: Reduced sensitivity towards outliners

KOKUYO

No.

Date

	L1 loss	L2 loss	Huber	Log Cosh
-	robust,	- not very robust	- can modify	- double
-	unstable solution,	- stable solution	hyperparameter	differentiable
-	possible multiple	- one solution	- if loss value	- compartively
	solutions		small, maintains	less commutations
-	not effected	- highly effected	quadratic func,	than huber
	by outliners	by outliners	- if loss value	- has fixed scale
			large, maintains	so less
			linear func.	adaptive than
			- for good results,	Huber.
			needs to be	
			optimized &	
			expensive in	
			terms of	
			computations.	

Q.2

Equation for Cross entropy loss :-

$$Li(\theta) = -\sum_{j=1}^{k} y_j^{(i)} \log\left(\hat{y}_j^{(i)}\right),$$

$$\hat{y}_j = P(y = j \mid x^{(i)})$$

- Derivation

likelihood : $L(\theta) = \prod_{i=1}^{m} \prod_{j=1}^{k} P(y = j \mid x^{(i)})^{y_j^{(i)}}$

- We need to maximize log-likelihood, we can do so by minimizing negative log-likelihood

KOKUYO

No.

Date

∴ log-likelihood = $-\log L(\theta)$

$(\ell(\theta))$

$$= -\sum_{i=1}^{m} \sum_{j=1}^{k} y_j^{(i)} \log\left(P(y=j \mid x^{(i)})\right)$$

$$= -\sum_{i=1}^{m} \sum_{j=1}^{k} y_j^{(i)} \log\left(\hat{y}_j^{(i)}\right)$$

∴ sample loss : $L_i(\theta) = \sum_{j=1}^{k} y_j^{(i)} \log\left(\hat{y}_j^{(i)}\right)$

- Worst cross entropy value for random assignment :-

$P(y=j \mid x^{(i)}) = 1/k \rightarrow \log\left(P(y=j \mid x^{(i)}\right) = \log(k)$

∴ $\underline{\underline{L(\theta) = \log(k)}}$

Q.3

Softmax is used as activation in output layer and uses cross-entropy for loss.

∴ Equation of Softmax loss :-

$$L(\theta) = \sum_{i=1}^{k} y_j^{(i)} \log\left(\hat{y}_j^{(i)}\right)$$

Softmax is used for multiclass classification.

Q.4

Equation for Kullback-Liebler :

$$L(\theta) = -\sum_{i=1}^{m} \sum_{j=1}^{k} y_j^{(i)} \log\left(\frac{y_j^{(i)}}{\hat{y}_j^{(i)}}\right)$$

KOKUYO

No.

Date

Consider :

$$KL(P||q) = \sum_{i=1}^{m} P(x_i) \log\left(\frac{P(x_i)}{q(x_i)}\right)$$

$$= \underbrace{\sum_{i=1}^{m} P(x_i) \log(P(x_i))}_{\text{-entropy}} - \underbrace{\sum_{i=1}^{m} P(x_i) \log(q(x_i))}_{\text{cross entropy}}$$

we known $P(x_i) = y^{(i)}$, $q(x_i) = \hat{y}^{(i)}$

if $y^{(i)}$ does not change, KL will be equivalent to cross entropy.

In other words if class label doesn't change, Kullback-liebler is same as cross entropy.

Q.5 Hinge loss equation:-

$$L_i(\theta) = \max(0, 1 - y^{(i)} \hat{y}^{(i)}) \quad \text{or}$$

$$L_i(\theta) = \sum_{j \neq t} \max(0, \hat{y}_j^{(i)} - \hat{y}_t^{(i)} + 1)$$

Square hinge loss equation:-

$$L_i(\theta) = 1/2 \sum_{j=1} \max(0, \hat{y}_j^{(i)} - y_t^{(i)} + 1)^2$$

Worst value for hinge loss before learning. when $\hat{y}_j^{(i)} \approx 0$

$$\underline{L(\theta) = (k-1) \times 1 = k-1}$$

The fundamental idea behind the worst hinge loss if as follows:-

(1) Only correct classification are not make.

(2) The classes are seperated far apart.

KOKUYO

No.

Date

Q.6

	x^1	x^2	x^3
$\hat{y}_1$	(0.5)	1.2	1.4
$\hat{y}_2$	0.4	(0.8)	-0.4
$\hat{y}_3$	0.3	-0.6	(2.7)
y	1	2	3

$L_1 = \max(0, 0.4 - 0.5 + 1) + \max(0, 0.3 - 0.5 + 1) = 0.9 + 0.8 = \underline{1.7}$

$L_2 = \max(0, 1.2 - 0.8 + 1) + \max(0, -0.6 - 0.8 + 1) = 1.4 + 0 = \underline{1.4}$

$L_3 = \max(0, 1.4 - 2.7 + 1) + \max(0, -0.4 - 2.7 + 1) = 0 + 0 = \underline{0}$

	x^1	x^2	x^3
$\hat{y}_1$	(0.5)	0.4	0.3
$\hat{y}_2$	1.2	(0.8)	-0.6
$\hat{y}_3$	1.4	-0.4	(2.7)
y	1	2	3

$L_1 = \max(0, 1.2 - 0.5 + 1) + \max(0, 1.4 - 0.5 + 1) = 1.7 + 1.9 = \underline{3.6}$

$L_2 = \max(0, 0.4 - 0.8 + 1) + \max(0, -0.4 - 0.8 + 1) = 0.6 + 0 = \underline{0.6}$

$L_3 = \max(0, 0.3 - 2.7 + 1) + \max(0, -0.6 - 2.7 + 1) = 0 + 0 = \underline{0}$

KOKUYO

No.

Date

Q.7

The purpose of adding regularization term to the loss function is to obtain an solution with smaller θ, this will make it stable and allow it to generalize better.

- L_1 regularization penalizes the sum of absolute values of weights whereas L_2 regularization penalizes sum of square of weights.
- The solution of L_1 regularization is sparse whereas the solution of L_2 regularization is not sparse.
- L_1 regularization can perform feature selection as it is built in whereas L_2 regularization cannot perform feature selection
- L_1 regularization is robust toward outliners, where as L_2 is not.

- The regularization term coefficient is a hyperparameter and usually its set to zero when training for first time and then changed according to the result.

Q.8 $L(\theta) = 1/m \sum_{i=1}^{m} Li(f(x_j^{(i)}, \theta), y^{(i)} + \lambda R(\theta)$

$\frac{\partial L}{\partial \theta} = \frac{dL}{d\theta} 1/m \sum_{i=1}^{m} Li(f(x_j^{(i)}, \theta), y^{(i)} + \lambda \frac{\partial}{\partial \theta} R(\theta)$

for L_2, $R(\theta) = \sum_i \theta_i^2 \therefore \frac{\partial}{\partial \theta} R(\theta) = 2\theta$, it adds $2\lambda\theta$ to gradient

for L_1, $R(\theta) = \sum_i |\theta_i| \therefore \frac{\partial}{\partial \theta} R(\theta) = \text{sign}\theta$, it adds $\lambda \text{sign}\theta$ to gradient.

KOKUYO

No.

Date

Q.9

Kernel regularization:

This regularization introduces a weight decay.
(to reduce weights (w))

Bias regularization:

This regularization is used reduce the bias (b)

Activity regularization:

This regularization introduces a decay on the predicted values.
(to reduce $\hat{y}$)

* Optimization.

Q.1 As discussed in the lecture, back propogation is easier to compute compared to direct numerical computation of gradient.
Back propogation makes use of symbolic derivatives in python.
Direct numerical computation of gradient is a slow and can be used for verification purposes.

Q.2 SGD: $\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \nabla L_j(\theta^{(i)})$
In SGD, the gradient is updated after a batch of training.
In GD, the gradient is updated after training the entire dataset.

KOKUYO

No.

Date

SGD is the one which converges faster as gradient is updated after training of batch of a data due to which weights are updated more frequently when compared to GD.

Q. 3

The trade offs in selecting batch size for SGD are as follows :-

(i) If the batch size is large, it reduces the variance of SGD but the gradient tends to converge sharply.

(ii) If the batch size is small, the weights are updated frequently but gradient may get stuck in local solution (local minima).

The four main problems of SGD are :-

1] What should be learning rate?
2] What happens if loss is more sensitive to some parameters (too fast in one direction and slower in others)
3] How to avoid getting stuck at local minimum or saddle points.
4] Mini batch gradient estimates are noisy.

KOKUYO

No.

Date

Q.4

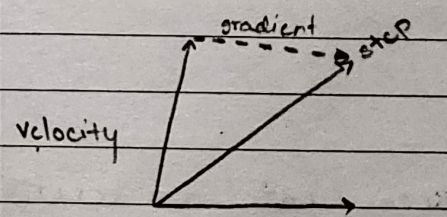
These are the ways, SGD with momentum addresses the following problems:-

(i) poor conditioning (high sensitivity in some direction)
It is handled by smothening out by averaging with previous gradient.

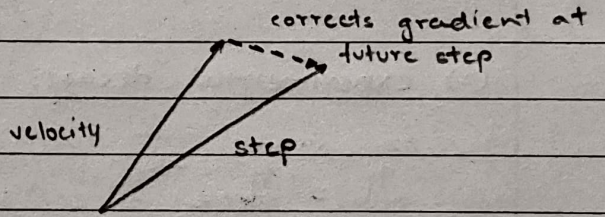
(ii) minimum / saddle points
There exsist a velocity vector, even at local minimum / saddle points to have an opportunity to get out of them.

(iii) Noisy gradient:
This is handled by smoothening out by moving averagly.

Q.5



SGD + momentum



NAG

-NAG is more accurate compared to simple momentum.
-The reason for this is, NAG corrects the location gradient at future step.

KOKUYO

No.

Date

$$\tilde{\theta}^{(i)} \leftarrow \theta^{(i)} + \beta v^{(i)}$$

$$\Rightarrow \theta^{(i+1)} - \beta v^{(i+1)} \leftarrow \tilde{\theta}^{(i)} - \beta v^{(i)} + v^{(i+1)}$$

$$\Rightarrow \tilde{\theta}^{(i+1)} \leftarrow \tilde{\theta}^{(i)} - \beta v^{(i)} + (1+\beta) v^{(i+1)}$$

$$\Rightarrow \tilde{\theta}^{(i+1)} \leftarrow \tilde{\theta}^{(i)} + v^{(i+1)} + \underbrace{\beta\left(v^{(i+1)} - v^{(i)}\right)}$$

This is the accelerated part of NAG.

Q. 6

The possible strategies for learning rate decay are as follows :-

(1) step decay:

every k iteration, $\eta \leftarrow \eta/2$

(2) experimental decay:

$\eta = \eta\, e^{-k/t}$, k = decay rate, t = iteration index

(3) fractional decay

$\eta = \eta_0 / (1 + k \cdot t)$

KOKUYO

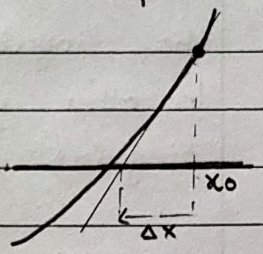
No.

Date

Q.7

Computing learning rate using Newton's method.

The learning rate for different features may very. This means loss may be more sensitive some features.



for newton's method,

we need 'x' such that $f(x) = 0$

- we start with x_0 & update Δx in

$f(x_0 + \Delta x) = 0$

Acc. to talor series expansion

$$f(x_0 + \Delta x) = f(x_0) + \Delta x f'(x_0) + \dots = 0$$

$$\Rightarrow f(x_0) + \Delta x f'(x_0) = 0$$

$$\Rightarrow \Delta x = -\frac{f(x_0)}{f'(x_0)}$$

but loss : $\nabla J(\theta) = 0 \equiv f(x)$

$$\therefore f(x_0 + \Delta x) = f(x_0) + \Delta x^T \nabla f(x_0) + \dots = 0$$

$$\Rightarrow f(x_0) + \Delta x^T \nabla f(x_0) = 0$$

replacing $f(x_0)$ with $\nabla J(\theta_0)$:

$$\Rightarrow \nabla J(\theta_0) + \nabla(\nabla J(\theta_0))\Delta\theta = 0$$

$\Rightarrow H = \nabla(\nabla J(\theta_0))$ .... hession matrix

$$\therefore H = \begin{bmatrix} \partial J^2/\partial\theta_1\partial\theta_2 & \dots & \partial J^2/\partial\theta_1\partial\theta_n \\ \vdots & & \\ \partial J^2/\partial\theta_n\partial\theta_1 & \dots & \partial J^2/\partial\theta_n\partial\theta_n \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$\Rightarrow \nabla J(\theta_0) + H\Delta\theta = 0$$

$$\Rightarrow \Delta\theta = -H^{-1}.\nabla J(\theta_0)$$

$\therefore$ Learning rate acc. to newton's method :-

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - H^{-1}\nabla J(\theta_0)$$

KOKUYO

No.

Date

Hessian matrix is a matrix of second order partial derivatives.

Q.8

Conditional numbers defines the sensitivity using the singular values of Hessian matrix.
In other words it the ratio between largest singular value and smallest singular value.

$$\therefore \text{conditional number} = \frac{SV_1}{SV_2}$$

where SV_1 : largest singular value,
SV_2 : smallest singular value.

The particular case of poor conditioning would be when the conditional number is a high which will make the matrix more sensitive to errors.

Q.9

The computation of Hessian matrix is expensive and could contain a noise.
In AdaGrad, the Hessian matrix is replaced by :-

$$B^{(i)} = \text{diag}\left(\sum_{j=1}^{i} \nabla J(\theta^{(j)}) \, \nabla J(\theta^{(j)})^T \right)^{1/2}$$

which would result in :

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \, B^{(i)^{-1}} \nabla J(\theta^{(i)})$$

KOKUYO

No.

Date

where $B^{(i)} = \begin{bmatrix} \sqrt{\sum \left(\frac{\partial J}{\partial \theta_1}\right)^2} & & \\ & \ddots & \\ & & \sqrt{\sum \left(\frac{\partial J}{\partial \theta_n}\right)^2} \end{bmatrix}$

$\Rightarrow B^{(i)^{-1}} = \begin{bmatrix} 1/\sqrt{\sum \left(\frac{\partial J}{\partial \theta_1}\right)^2} & & \\ & \ddots & \\ & & 1/\sqrt{\sum \left(\frac{\partial J}{\partial \theta_n}\right)^2} \end{bmatrix}$

Q.10

The main problem with AdaGrad is, we normalize by elementwise sum of square gradient due to which the step size will get smaller as the iteration progresses.

To counter this RMSProp makes use of a decay factor while adding gradients to the gradient sum.

Q.11

The Adam algorithm combines momentum with RSMProp in the following manner:-

(i) Incoperating velocity with first moment:

$$m_1^{(i+1)} = \beta_1 \cdot m^{(i)} + (1-\beta_1) \nabla L(\theta^{(i)})$$

KOKUYO

No.

Date

(2) Incoperating elementwise step size scale with second moment:

$$m_2^{(i+1)} = \beta_2 \cdot m_2^{(i)} + (1-\beta_2)\left(\nabla L(\theta^{(i)}) \odot \nabla L(\theta^{(i)})\right)$$

(3) We end up with :-

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta\, m_1^{(i+1)} \odot \frac{1}{\sqrt{m_2^{(i+1)} + \epsilon}}$$

$$m_1^{(0)} = m_2^{(0)} = 0$$

As the moments are initialized to zero, so when dividing by the second moment we will end up with a large step.

A bias correction term is used, which is dividing the moment by a number depending on the iteration number.

$$\tilde{m}_1^{(i+1)} = \frac{m_1^{(i)}}{1-\beta_1^i}$$

$$\tilde{m}_2^{(i+1)} = \frac{m_2^{(i)}}{1-\beta_2^i}$$

These are unbiased estimates of m_1 & m_2.

KOKUYO

No.

Date

Q. 12

In line search, instead of a fixed step size, the optimal step size is calculated for a given direction.

- Given direction:

$$u = \nabla f(x)$$

- Best step size:

$$\eta^* = \text{argmin}_\eta \, f(x + \eta u)$$

- Gradient descent:

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta^* \nabla f(\theta^{(i)})$$

In bracketing we search for minimum point.

Given bracket $[a, b, c]$:

$$x = \frac{b+c}{2}$$

if $f(x) \leq f(b) \Rightarrow [b, x, c]$

if $f(x) > f(b) \Rightarrow [a, b, x]$

We will repeat the process with smaller and smaller bracket until the bracket is small enough.

An alternate to line search would be Coordinate descent. $\{u^{(i)}\} = \{e^{(i)}\}$

Coordinate descent in more accurate compared to line search but is more expensive in computation.

KOKUYO

No.

Date

Q. 13

The quasi-newton approximates Hessian inverse using gradient evaluation.

This done as inverting the Hessian is expensive. as Hessian has n^2 elements and inverting them has $O(n^3)$ complexity.

* BFGS update

$$\Delta\theta^{(i)} = \theta^{(i)} - \theta^{(i-1)}$$

$$\Delta J = \nabla J(\theta^{(i)}) - \nabla J(\theta^{(i-1)})$$

$$H^{(i)} = H^{(i-1)} + \frac{\Delta J \Delta J^T}{\Delta J^T \Delta\theta} - \frac{H^{(i-1)} \Delta\theta \Delta\theta^T H^{(i-1)}}{\Delta\theta^T H^{(i-1)} \Delta\theta}$$

inverse update:

$$H^{(i)^{-1}} = \left(I - \frac{\Delta\theta \Delta J^T}{\Delta J^T \Delta\theta}\right)\left(H^{(i-1)}\right)^{-1}\left(I - \frac{\Delta J \Delta\theta^T}{\Delta J^T \Delta\theta}\right) + \frac{\Delta\theta \Delta\theta^T}{\Delta J^T \Delta\theta}$$

The inverse update cost $O(n^2)$.

The BfGS algorithm cost $O(n^2)$ whereas the quasi-Newton costs $O(n^3)$. This makes BfGS more efficient.

BFGS is comparitively more accurate than Adam but requires more computation time and resource.

KOKUYO

No

Date

* Regularization

Q.1

Multiplying each coefficient by $P \in [0,1]$. As in iterations progress weights that are not reinforced decay to zero. This is equivalent to add regularization term to loss function.
Alternatively, applying decay to unit outputs.

Q.2

Early stopping stops the training when validation error starts increasing or when training error stops decreasing. This prevents overfitting the model as optimal losses are obtained with less epochs.

These are the strategies to reuse validation data:-

(1) Retrain on all data using the number of iterations determined from validation
(2) Continue training from previous weights with entire data while validation loss is bigger than training loss.

Q.3

Data augmentation can be by using the following techniques :-
1) Augment in feature or data domain
2) Augment by interpolating between examples or by adding noise.

KOKUYO

No.

Date

3) Augment by transforming data eg. crop, rotate, scale, change intensity in images, etc.

Data augmentation means adding synthetic data to increase variability in training which would result in better generalization.
This can help to prevent over-fitting of the model.

Q.4

- Dropout is unit in fully connected layer with probability of (1-p) where 'p' is a hyperparameter.
- As the name suggests, dropout drops nodes randomly.
- The removed / dropped nodes are reinstated with original weights in subsequent stage.

Advantages of Dropout are :-

- Reduce node interactions (co-adaption)
- Reduce overfitting
- increase training speed
- reduce dependancy on single node.
- distribute features across multiple nodes

One major disadvantage of dropout is :-
- longer time to train due to dropout as not all cells are available at each step.

KOKUYO

No.

Date

Q.5

The expected values of all combinations of dropped out networks can be approximated efficiently during testing by :-

Multiplying the output of each node by P is equivalent to computing expected value for 2nd dropped out networks.

$$\hat{y} = E_D[f(x, D)] = \int P(D) f(x, D) dD$$

Q.6

Batch normalization during training :

(applied to activation x over a mini-batch)

Input : Values of x over a mini-batch : $B = \{x_{1,\dots,m}\}$;

Parameters to be learned : γ, β

Output : $\{ y_i = BN_{\gamma,\beta}(x_i) \}$

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B) \qquad \text{// mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Batch normalization adds randomness into training which helps to reduce overfitting.

During testing : use average normalization values computed during training.

KOKUYO

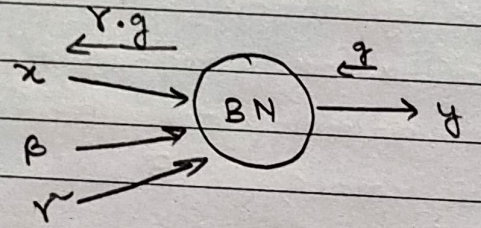
Q.7

The purpose of scale and shift parameters is to control the amount of normalization to be performed.

The parameter : γ_j , β_j.

when $\gamma_j = \sigma_j$ and $\beta_j = \mu_j$ causes normalization to be cancelled :-

$$\rightarrow z_j^{(i)} = \sigma_j \hat{z}_j^{(i)} + \mu_j = \sigma_j \frac{z_j^{(i)} - \mu_j}{\sigma_j} + \mu_j = z_j^{(i)}$$



$$y = \gamma x + \beta$$

$$\frac{\partial y}{\partial x} = \gamma$$

The good initial values for scale and shift parameters are as follows:-

$$\gamma_j = 1 \quad , \beta_j = 0.$$

Q.8

Ensemble classifiers can train multiple independent models. They use majority vote or average during testing. This helps to reduce overfitting.

No.

Date

The strategies for producing ensemble classifiers are as follows:-

i) Change data

ii) Change parameters

iii) Record multiple snapshorts of the model during training. (vary learning rate).

KOKUYO