Homework 1

Phillip Wang (pw1287)

September 2021

1.2

a

1. Feedforward: input is undergoing linear transformation and activation function, $f(Wx + b)$ where f is the activation function, W is the weight matrix, and b is the bias.
2. Computing the loss: use the desired cost function to compute the loss with $\hat{y}$ and y.
3. Zeroing the gradients: set the gradients to zero.
4. Backpropagation: Compute the partial derivative of the loss with respect to the weights and pass on the gradients of the preceding layers.
5. Update parameters: the gradient multiplied by learning rate will be subtracted from the parameters.

b

Assume that $W^{(1)}$ is a m by n matrix, where n is the number of features in $\mathbf{x}$. z_1 and z_2 are column vectors with m rows. Therefore, $W^{(2)}$ is a K by m matrix and z_3 and $\hat{y}$ are column vectors with K rows.
Forward pass for $Linear_1$:

$$z_1 = W^{(1)}x + \mathbf{b}^{(1)}$$

Forward pass for f:

$$z_2 = f(z_1) = \begin{bmatrix} \Phi(W_1^{(1)}x + \mathbf{b}_1^{(1)}) \\ \vdots \\ \Phi(W_m^{(1)}x + \mathbf{b}_m^{(1)}) \end{bmatrix} \text{ where } \Phi(W_i^{(1)}x + \mathbf{b}_i^{(1)}) = \begin{cases} W_i^{(1)}x + \mathbf{b}_i^{(1)} & \text{if } W_i^{(1)}x + \mathbf{b}_i^{(1)} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Forward pass for $Linear_2$ and g:

$$\hat{y} = g(z_3) = z_3 = \begin{bmatrix} \Theta(\mathbf{W}_1^{(1)}x + \mathbf{b}_1^{(1)}) \\ \vdots \\ \Theta(\mathbf{W}_K^{(1)}x + \mathbf{b}_K^{(1)}) \end{bmatrix}$$

$$\text{where } \Theta(W_i^{(1)}x + \mathbf{b}_i^{(1)}) = \begin{cases} \mathbf{W}_i^{(2)}(\mathbf{W}_i^{(1)}x + \mathbf{b}_i^{(1)}) + \mathbf{b}_i^{(2)} & \text{if } \mathbf{W}_i^{(1)}x + \mathbf{b}_i^{(1)} > 0 \\ \mathbf{b}_i^{(2)} & \text{otherwise} \end{cases}$$

c

Since g is an identity function,

$$\frac{\partial l}{\partial z_3} = \frac{\partial l}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3} = \frac{\partial ||\hat{y} - y||^2}{\partial \hat{y}} \overset{(x,y)}{\Longrightarrow} \frac{\partial (\hat{y} - y)^2}{\partial \hat{y}} = 2(\hat{y} - y)$$

Gradients for $W^{(2)}$

$$\frac{\partial l}{\partial \mathbf{W^{(2)}}} = ReLU(W^{(1)}x + b^{(1)})\frac{\partial l}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}$$

Gradients for $\mathbf{b}^{(2)}$

$$\frac{\partial l}{\partial \mathbf{b}^{(2)}} = \frac{\partial l}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}$$

Gradients for $W^{(1)}$

$$\frac{\partial l}{\partial \mathbf{W^{(1)}}} = \mathbf{x}\frac{\partial l}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}\mathbf{W}^{(2)}\frac{\partial z_2}{\partial z_1}$$

Gradients for $b^{(1)}$

$$\frac{\partial l}{\partial \mathbf{b}^{(1)}} = \frac{\partial l}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}\mathbf{W}^{(2)}\frac{\partial z_2}{\partial z_1}$$

d

$$\underset{m\times m}{\frac{\partial z_2}{\partial z_1}} = \begin{cases} 1 & \text{if } z_1 = \mathbf{W}^{(1)}x + b^{(1)} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial \hat{y}}{\partial z_3} = \mathbb{I}_K \text{ Identity matrix with dimension } K$$

$$\underset{1\times K}{\frac{\partial l}{\partial \hat{y}}} = 2(\hat{y} - y)^T$$

1.3

a

(b)

Forward pass for f: $z_2 = \sigma(\mathbf{W}^{(1)} + \mathbf{b}^{(1)})$
Forward pass for $Linear_2$: $z_3 = \mathbf{W}^{(2)}\sigma(\mathbf{W}^{(1)} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}$
Forward pass for g: $\hat{y} = \sigma(\mathbf{W}^{(2)}\sigma(\mathbf{W}^{(1)} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)})$

(c)

$$\frac{\partial l}{\partial \mathbf{W}^{(2)}} = \sigma(\mathbf{W}^{(1)}x + \mathbf{b}^{(1)})\frac{\partial l}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}$$

Other gradients remain unchanged in the representation above.

(d)

$$\frac{\partial z_2}{\partial z_1} = \sigma(z_1)(1 - \sigma(z_1)) = \sigma(\mathbf{W}^{(1)}x + \mathbf{b}^{(1)})(1 - \sigma(\mathbf{W}^{(1)}x + \mathbf{b}^{(1)}))$$

$$\frac{\partial \hat{y}}{\partial z_3} = \sigma(z_3)(1 - \sigma(z_3)) \text{ where } z_3 = \mathbf{W}^{(2)}\sigma(\mathbf{W}^{(1)}x + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}$$

$\frac{\partial l}{\partial \hat{y}}$ is unchanged.

0.1 b

ReLU functions prevents the vanishing gradient problem, which allows the gradient to be passed to a network with more layers (i.e. deeper). Also, it is more computationally efficient, which means convergence of the network is faster.

1.4

a

Argmax function returns the index/location of the maximum in the input vector. Soft(arg)max takes a less extreme approach and assigns

b

Softmax is unstable when the input is really large or really small, which would cause an overflow or underflow problem.

c

No. Two consecutive linear layers are the same as one linear layer.

d

ReLU: Computation is really fast using ReLU, thus converges really quickly. Also, it is less susceptible to the vanishing gradient problem as it passes the gradient intact. However, when the input is zero or negative, the derivative of the function is zero, thus stopping the backpropagation and halts the learning process.

tanh: The output values are zero-centered and bounded between -1 and 1. For input values that are above 1 or below -1, the tanh function brings them closer to the boundaries. However, it suffers from the vanishing gradient problem as the slope of the function is close to 0 when the input is really large or small. Moreover, the function involves the exponential, which is computationally expensive.

Sigmoid: The derivative of the sigmoid function is simple, and its output is bounded between 0 and 1, explanatory in probabilistic problems. However, it also has vanishing gradient problem and is computationally expensive.

LeakyReLU: Compared to ReLU, LeakyReLU prevents the dying ReLU problem as the function still has a positive slope in the negative domain, which enables backpropagation for negative inputs. But we need to tune the slope for LeakyReLU, and as the slope approaches 1, the function gradually loses nonlinearity.

e

Rotation, scaling, translation, reflection. Non linear transformation introduces non-linearity that models the nonlinear relationship between the input and the target variable. Without nonlinearity, the model is simply a linear regression. Linear transformation selects and weighs each input to a different feature space.

f

Generally, the learning rate should increase/decrease in proportion with the increase/decrease of the batch size. This is because the gradients computed using a larger batch are more likely to lead us towards the correct direction, while the gradients produced by a smaller, more variant batch should be cautiously adopted.