ARCHITECTING INTELLIGENCE

Assignment 1

Q1) $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_d x_{id}$

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1d} \\ 1 & x_{21} & x_{22} & \dots & x_{2d} \\ \vdots & \vdots & & & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix}$$

$$\hat{y} = X\beta$$

$$J(\beta) = \frac{1}{2}\|y - X\beta\|^2$$

$$J(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

normal eq: soln obtained by setting $\frac{\partial J(\beta)}{\partial \beta} = 0$.

(a) what regression is doing:

→ gives us a hypothesis function to predict the value of the dependant variable y with regard to the independant variable x, while using parameters β.

why to minimize squared error:

→ squared error represents the difference between the predicted value of y and the actual value of y. minimizing this makes our model as accurate and as close to reality as possible.

(b) $J(\beta) = \frac{1}{2}\|y - X\beta\|^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

$$\frac{\partial J}{\partial \beta} = \sum_{i=1}^{n}\left[0 - \frac{\partial \hat{y}_i}{\partial \beta}\right]$$

$y_{n\times 1}$

$y^T_{1\times n}$.

$$J(\beta) = \frac{1}{2}(y - X\beta)^T \cdot (y - X\beta).$$

$$= y^Ty - y^TX\beta - (X\beta)^T y + (X\beta)^T X\beta.$$

$$= y^Ty - y^TX\beta - \beta^TX^Ty + (X\beta)^T X\beta.$$

$[y_{n\times 1},\ y^Ty \equiv 1\times 1\ ;\ X_{n\times(n+1)},\ \beta_{(n+1)\times 1}]$ → all of these terms become scalars (w dimension 1×1)

$$J(\beta) = y^Ty - 2y^TX\beta + \beta^TX^TX\beta.$$

$\frac{\partial J}{\partial \beta} = -2y^TX$

$$\nabla_\beta J = \begin{bmatrix} \frac{\partial J}{\partial \beta_1} \\ \vdots \\ \frac{\partial J}{\partial \beta_p} \end{bmatrix}$$

OR $J(\beta) = y^Ty - 2\beta^TX^Ty + \beta^TX^TX\beta.$

$$\nabla_\beta J = \frac{1}{2}\left[-2X^Ty + 2X^TX\beta\right]$$

$$= X^TX\beta - X^Ty = 0.$$

$$\therefore \beta = (X^TX)^{-1}X^Ty$$

(c) directly inverting X^TX in the normal equation:

$$\beta = (X^TX)^{-1}X^Ty.$$

is often problematic in high dimensional or large scale settings for both numerical and computational reasons.

Following are the issues:

(i) computational cost (time & memory):

- forming (X^TX): $O(np^2)$.
- inverting (X^TX): $O(p^3)$

⇒ in contrast, one gradient descent step costs only $O(np)$.

(ii) poor scalability to large datasets:

Normal equations require all data at once;
- it must load the entire X into memory.

⇒ Iterative methods allow mini-batch or stochastic gradient descent.

Q.2) (a) core idea behind backpropagation:

⇒ to efficiently compute how much each parameter (weight and bias) is responsible for the final prediction error, and then use this information to update the parameters so that the error is reduced.

Role of chain rules:

The neural network consists of:

- input transferred layer by layer through weighted sums.
- activation functions until final output is produced.
- loss function measuring how far the output is from the true target.

⇒ loss depends on output, output depends on earlier layers, which in turn depend upon weights and biases.

⇒ The chain rule makes it possible to calculate these dependencies systematically

(b) $z_1 = w_1 x + b_1$; $a_1 = \sigma(z_1)$.

$z_2 = w_2 a_1 + b_2$; $a_2 = \sigma(z_2) = \hat{y}$

$\sigma(z) = \frac{1}{1+e^{-z}}$

$L = -[y \log a_2 + (1-y) \log(1-a_2)]$.

$$L = -\left[y \log a_2 + (1-y) \log (1-a_2) \right]$$

- $$\frac{\partial L}{\partial w_2} = -\left[\frac{y}{a_2} \cdot \frac{\partial a_2}{\partial w_2} + \frac{(1-y)}{(1-a_2)} \cdot \frac{\partial a_2}{\partial w_2} \right]$$

$$a_2 = \sigma(z_2) = \sigma(w_2 a_1 + b_2)$$

$$\sigma'(z) = \frac{-1}{(1+e^{-z})^2} \left[-e^{-z} \right] = \frac{e^{-z}}{(1+e^{-z})^2} \cdot dz.$$

$$\frac{\partial a_2}{\partial w_2} = \frac{\partial \sigma(z_2)}{\partial w_2} = \frac{\partial \sigma(z_2)}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2} \rightarrow a_1$$

$$= a_1 \cdot \frac{\partial \sigma(z_2)}{\partial z_2} = \frac{\partial a_2}{\partial w_2}$$

$$\frac{\partial a_2}{\partial w_2} = a_1 \left[\frac{1}{a_2^2} - \frac{1}{a_2^2} \right]$$

$$= a_1 \frac{(a_2^2 - a_2^2)}{a_2^4} = a_1 \left[\frac{1-a_2}{a_2^3} \right]$$

$$\frac{\partial L}{\partial w_2} = + \frac{y a_1 (1-a_2)}{a_2^4} + \frac{(1-y)}{} \cdot \frac{a_1}{a_2^3}$$

- $$\frac{\partial L}{\partial b_2} = -\left[\frac{y}{a_2} \cdot \frac{\partial a_2}{\partial b_2} + \frac{(1-y)}{(1-a_2)} \cdot \frac{\partial a_2}{\partial b_2} \right]$$

$$\frac{\partial a_2}{\partial b_2} = \frac{\partial \sigma(z_2)}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2} = \frac{(1-a_2)}{a_2^3}$$

$$\frac{\partial L}{\partial b_2} = -\frac{y}{a_2^4} (1-a_2) + \frac{(1-y)}{a_2^3}$$

• $\frac{\partial L}{\partial w_1}$ (2). for deriving $\frac{\partial L}{\partial w_1}$, $\frac{\partial L}{\partial b_1}$, we use similar methods.

(c). in gradient descent, for a parameter A,

$A_n = A_{n-1} - \eta \cdot \frac{\partial L}{\partial A}$ L: loss func.
η = learning rate

$w_1 \to w_1 - \eta \frac{\partial L}{\partial w_1}$ $w_2 \to w_2 - \eta \frac{\partial L}{\partial w_2}$

$b_1 \to b_1 - \eta \frac{\partial L}{\partial b_1}$ $b_2 \to b_2 - \eta \frac{\partial L}{\partial w_2}$

⇒ Role of learning rate η:
controls how big each step is.

- If η is too large: updates may overshoot. loss might oscillate or diverge.
- If η is too small: training remains very slow.

⇒ we need a proper learning rate for a steady decrease in loss and a stable convergence

Q3] (b). Simple RNNs struggle w. long term dependencies because gradients become very small or very large when propagated over many time steps

(c) LSTM gates control what info is stored, updated or removed from the cell state.

(d) LSTMs address the vanishing gradient problem by maintaining a nearly constant error flow through the cell state across time steps.

(e) ANNs are best for tasks like image classification, RNNs for short sequence prediction and LSTMs for long sequences tasks such as long language translation.

Q4] (a) ~~The book the~~ In a sentence like "the book that the professor who the students admired wrote was famous." understanding that 'was' refers to the 'book' depends on long-range dependency across many words, and a standard RNN would struggle to model this ~~relatively~~ reliably due to vanishing gradients over long sequences.

(b) An LSTMs memory cell acts like a conveyor belt that carries important info forward, while gates what to keep, update or discard, while allowing relevant context to persist over long sequences. In machine translation, the forget gate will need to be close to 0 when moving from one sentence or clause to another so that outdated context from the previous clause does not interfere with translating the next one correctly.

Q3) (a) An ANN processes each input independently with no memory, while RNN processes sequences by passing info from one step to the next through a hidden state.