Gradient descent en kostenfuncties

Wat is N?

f(x) = n^x

N is groter dan 0 en N mag geen 1 zijn, anders is het geen exponentiële functie

Wat is E?

f(x) = e^x

De e-macht is een macht waarvan het grondgetal het getal e is. Het getal e is een wiskundige constante en het grondtal van de natuurlijke logaritme. De waarde van e is 2,71828. Het getal heet Euler, verwijzend naar de ontdekker en wiskundige Leonhard Euler.

Gradient descent

Gradient descent is een optimalisatie-algoritme dat wordt gebruikt in AI om de parameters van een model aan te passen, zodat het model beter past bij de gegeven data. Het werkt door de fout tussen voorspelde en werkelijke waarden te minimaliseren.

Stappen:

1. Initialiseer de parameters van het model.
2. Bereken de fout (kostenfunctie) tussen de voorspelde en werkelijke waarden.
3. Bereken de afgeleide van de foutfunctie met betrekking tot de parameters (gradiënt). 4. Pas de parameters aan in de richting van de negatieve gradiënt om de fout te verminderen.

5. Herhaal de stappen 2-4 totdat een acceptabel minimum is bereikt of een vastgesteld aantal iteraties is bereikt.

Wiskundig voorbeeld:

Laten we de functie (f(x) = e{2x} - 4x) nemen.

Stapsgewijze uitvoering met Gradient Descent:

1. We beginnen met een *willekeurige waarde* voor (x), bijvoorbeeld (x = 1).

2. Vul de x in dus bereken (f(1) = e^{2(1)} - 4(1) = e^2 - 4 ongeveer 1.39).

3. Bereken de afgeleide van (f(x)) = (f'(x) = 2e^{2x} - 4).

4. Bij (x = 1) is de helling (f'(1) = 2e^2 - 4 ongeveer 8.77).

5. We willen het minimum vinden, dus we moeten (x) verlagen. We trekken een klein deel van de helling af van (x), bijvoorbeeld (0.1 keer (2e^2 - 4) ongeveer 0.88).
6. Nu is (x = 1 - 0.88 ongeveer 0.12).
7. We herhalen dit proces totdat we een minimum vinden of een vastgesteld aantal iteraties bereiken.

Deze stappen worden iteratief herhaald totdat we een punt bereiken waar de helling dichtbij nul is, wat aangeeft dat we een minimum hebben gevonden, of totdat we een vastgesteld aantal iteraties hebben bereikt.

Kostenfunctie (MSE):

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta (x^{(i)}) - y^{(i)})^2$$

m	Aantal trainingsvoorbeelden
$x^{(i)}$	De input vector voor het i^e trainingsvoorbeeld
$y^{(i)}$	De daadwerkelijke waarde y behorende bij het i^e trainingsvoorbeeld
θ	De gekozen parameter, oftewel: De gewichten: θ_0 en θ_1
$h_\theta (x^{(i)})$	De voorspelde waarde y van het i^e trainingsvoorbeeld

Voor functie 1:

$$Z(x,y) = x^4 + x^3y^2 + y^5 + 8$$

De eerste-orde partiële afgeleiden zijn:

$$\frac{\partial Z}{\partial x} = 4x^3 + 3x^2y^2$$
$$\frac{\partial Z}{\partial y} = 2x^3y + 5y^4$$

Nu kunnen we de tweede-orde partiële afgeleiden berekenen:

$$\frac{\partial^2 Z}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial Z}{\partial x}\right) = 12x^2 + 6xy^2$$
$$\frac{\partial^2 Z}{\partial y^2} = \frac{\partial}{\partial y}\left(\frac{\partial Z}{\partial y}\right) = 2x^3 + 20y^3$$
$$\frac{\partial^2 Z}{\partial x \partial y} = \frac{\partial}{\partial y}\left(\frac{\partial Z}{\partial x}\right) = 6x^2y$$

Voor functie 2:

$$Z(x,y) = x^3 + x^2y^4 - 2y^5$$

De eerste-orde partiële afgeleiden zijn:

$$\frac{\partial Z}{\partial x} = 3x^2 + 2xy^4$$
$$\frac{\partial Z}{\partial y} = 4x^2y^3 - 10y^4$$

De tweede-orde partiële afgeleiden zijn dan:

$$\frac{\partial^2 Z}{\partial x^2} = 6x + 2y^4$$
$$\frac{\partial^2 Z}{\partial y^2} = 12x^2y^2 - 40y^3$$
$$\frac{\partial^2 Z}{\partial x \partial y} = 8xy^3$$

Voor functie 3:

$$Z(x,y) = -3x^3y^2 + 5$$

De eerste-orde partiële afgeleiden zijn:

$$\frac{\partial Z}{\partial x} = -9x^2y^2$$
$$\frac{\partial Z}{\partial y} = -6x^3y$$

De tweede-orde partiële afgeleiden zijn dan:

$$\frac{\partial^2 Z}{\partial x^2} = -18xy^2$$
$$\frac{\partial^2 Z}{\partial y^2} = -6x^3$$
$$\frac{\partial^2 Z}{\partial x \partial y} = -18x^2y$$

Kostenfuncties

Stap 1: MSE (Mean Squared Error) en Cross Entropy zijn beide kostfuncties die vaak worden gebruikt bij machine learning, met elk hun eigen toepassingsgebied.

MSE:

- Mathematisch gezien berekent MSE het gemiddelde van de kwadratische verschillen tussen de voorspelde waarden en de werkelijke waarden van de doelvariabele.
- Het wordt gebruikt bij regressieproblemen, waarbij de doelstelling is om een continue waarde te voorspellen, zoals prijzen, temperaturen, enz.
- MSE wordt gevoelig voor outliers, omdat het kwadratische verschillen gebruikt.

Cross Entropy:

- Cross Entropy wordt vaak gebruikt bij classificatieproblemen, waarbij de doelstelling is om een discrete klasse te voorspellen.
- Het meet de discrepantie tussen de waarschijnlijkheidsverdeling van de voorspelde klasse en de werkelijke klasse.
- Het wordt vooral gebruikt bij softmax classificatie, waar de output wordt geïnterpreteerd als de kansverdeling over een aantal klassen.

Stap 2:

MSE toepassen:

De formule voor de kostfunctie MSE is gegeven door:

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Waarbij $h_\theta(x) = \theta_1 x$.

Voorbeeldberekening voor $\theta_1 = 1.5$ met trainingsvoorbeelden $x = 1, x = 2, x = 3$:

h_θ voor $\theta_1 = 1.5$:

$$h_\theta(1) = 1.5 \times 1 = 1.5$$
$$h_\theta(2) = 1.5 \times 2 = 3$$
$$h_\theta(3) = 1.5 \times 3 = 4.5$$

$J(\theta_1)$ als $\theta_1 = 1.5$:

$$J(\theta_1) = \frac{1}{3}[(1.5 - y_1)^2 + (3 - y_2)^2 + (4.5 - y_3)^2]$$

Stap 3:

Softmax berekenen:

Softmax is een activatiefunctie die wordt gebruikt in neurale netwerken voor multiklassenclassificatie. Het converteert de uitvoer van de laatste laag van een netwerk naar een kansverdeling over verschillende klassen.

De formule voor softmax is:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Voorbeeldberekening voor $x_1 = 1.4, x_2 = 2.5, x_3 = 0.7$:

$$\text{softmax}(x_1) = \frac{e^{1.4}}{e^{1.4} + e^{2.5} + e^{0.7}}$$
$$\text{softmax}(x_2) = \frac{e^{2.5}}{e^{1.4} + e^{2.5} + e^{0.7}}$$
$$\text{softmax}(x_3) = \frac{e^{0.7}}{e^{1.4} + e^{2.5} + e^{0.7}}$$

cross entropy

De eerste afbeelding is van een kat, en onze voorspelde kans voor kat is 0.8 (en dus 0.2 voor hond).
De tweede afbeelding is van een hond, en onze voorspelde kans voor hond is 0.6 (en dus 0.4 voor kat).
De derde afbeelding is van een kat, en onze voorspelde kans voor kat is 0.9 (en dus 0.1 voor hond).
Laten we zeggen dat de werkelijke labels voor deze afbeeldingen respectievelijk zijn: kat, hond, kat.

De formule voor cross entropy voor dit binair classificatieprobleem is:

$$J(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Waarbij:

- N het aantal voorbeelden is,
- y_i de ware klasse is (1 voor kat, 0 voor hond),
- $\hat{y}_i$ de voorspelde kans op klasse 1 (kat) is.

Nu passen we deze formule toe op onze voorbeelden:

Voorbeeld 1:

- $y_1 = 1$ (kat)
- $\hat{y}_1 = 0.8$ (voorspelde kans voor kat)

$$J_1 = -[1 \cdot \log(0.8) + (1 - 1) \cdot \log(1 - 0.8)] = -\log(0.8)$$

Voorbeeld 2:

- $y_2 = 0$ (hond)
- $\hat{y}_2 = 0.4$ (voorspelde kans voor kat)

$$J_2 = -[0 \cdot \log(0.4) + (1 - 0) \cdot \log(1 - 0.4)] = -\log(0.6)$$

Voorbeeld 3:

- $y_3 = 1$ (kat)
- $\hat{y}_3 = 0.9$ (voorspelde kans voor kat)

$$J_3 = -[1 \cdot \log(0.9) + (1 - 1) \cdot \log(1 - 0.9)] = -\log(0.9)$$

Nu kunnen we de gemiddelde cross entropy over alle voorbeelden berekenen:

$$J_{gemiddeld} = \frac{J_1 + J_2 + J_3}{3}$$

Dit geeft ons de gemiddelde cross entropy voor deze drie voorbeelden.