CNN

(C ⇒ (No of filter)

⇒ Convolution layer (Conv2D), ⇒ filter(Fy), Stride(ΔY)

$$Z^{(L)} = \left(1 + \frac{H^{(L-1)} - Fy}{\Delta Y}\right) \times \left(1 + \frac{W^{(L-1)} - Fx}{\Delta X}\right) \times C^{(L)}$$

⇒ Maxpooling ⇒ Kernel

$$\left(H/K, W/K, C\right)$$

⊙ Cnn layers

Conv2D (filter, kernel, activation, input)

Maxpooling (pool-size)

Dense (Size, activation) ⟶ Could be Relu, Softmax

⊙ Probability

$$p(A|B) = \frac{p(X=A, Y=B)}{p(B)} = \frac{\#(X=A, Y=B)}{\#(Y=b)}$$

$$p(A=x) = \frac{\#(A=x)}{N}$$

⇒ Derivative

$$\frac{d}{dx}\cos(x) = -\sin(x)$$

$$\frac{d}{dx}\sin(x) = \cos(x)$$

DNN

$$A = xW + \vec{b}^{T}$$

In general

$$A^{(i)} = A^{(i-1)} \cdot W^{(i)} + \vec{b}^{(i)T}$$

entropy

⊙ Binary classification

$$C = \begin{pmatrix} t_n & f_p \\ f_n & t_p \end{pmatrix} \begin{matrix} t=1 \\ t=2 \end{matrix} \quad \begin{matrix} y=1 & y=2 \end{matrix}$$

true negative: $t_n = \#(y=1, t=1)$

false positive: $f_p = \#(y=2, t=1)$

true positive: $t_p = \#(y=2, t=2)$

false negative: $f_n = \#(y=1, t=2)$

$\Rightarrow$ negatives: $\#(t=1)$

$\Rightarrow$ positives: $\#(t=2)$

$\Rightarrow$ negative rate:

$$P_n = p(t=1) = n/N$$

$\Rightarrow$ positive rate:

$$P_p = p(t=2) = n/N$$

true negative rate (tnr)

$$t_{nr} = p(y=1 \mid t=1)$$

false positive rate (fpr)

$$f_{pr} = p(y=2 \mid t=1)$$

true positive rate (tpr)

$$t_{pr} = p(p=2 \mid t=2)$$

False negative rate (fnr)

$$f_{nr} = p(y=1 \mid t=2)$$

$$t_{nr} + f_{pr} = 1$$

$$f_{nr} + t_{pr} = 1$$

$\Rightarrow$ Error (E)

$$E = f_{nr} * P_p + f_{pr} * P_n$$

$\Rightarrow$ Sensitivity / recall = tpr (true positive rate)

$\Rightarrow$ Specificity / selectivity = tnr (True ~~False~~ negative rate)

$\Rightarrow$ Precision: $p = \dfrac{t_p}{f_p + t_p} = \dfrac{t_{pr}}{f_{pr} \cdot \frac{P_n}{P_p} + t_{pr}}$

$$\Rightarrow \text{Softmax} = \frac{\exp(x_i)}{\sum_k \exp(x_k)}$$

(S_i)

↳ Partial derivatives: $\frac{\partial S_i}{\partial x_j} = \delta_{ij} S_i - S_i S_j$

$\Rightarrow$ Entropy

$$L = -\sum_k t_k \cdot \log y_k$$

$$\Rightarrow \text{MSE} = \frac{\sum_i^n (\text{pred} - \text{Actual})^2}{N}$$

$\Rightarrow$ Derivation (define rule, decomposition, Applying rule)

Chain Rule = $h(f(g(x)))$

$= h'(f(g(x))) \cdot f'(g(x)) \cdot g'(x)$

Multi. Rule = $f(x) \cdot g(x)$

$= f'(x) \cdot g(x) + g'(x) \cdot f'(x)$

$\Rightarrow$ Gradient

$$\vec{x}_{(i+1)} = \vec{x}_{(i)} \pm \varepsilon \cdot \vec{\Delta} L \Big|_{\vec{x}(i)}$$

Numpy

⇒ generating nD array ⇒ np. random. uniform (low, high, shape)

Mat

⇒ np. linspace (low, high, ~~space~~ Count)

DNN