1 Loss1
2 Loss2
3
4

Loss = $(\hat{y} - y)^2$

$\hat{y}$ → true
y → pred

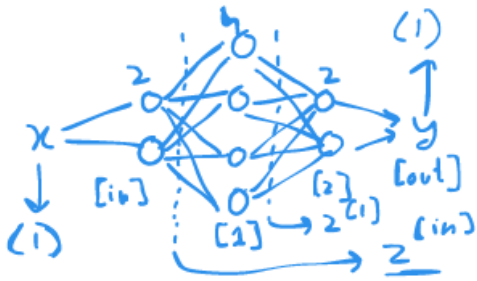
GD:

$$w' = w - \alpha \frac{\partial L}{\partial w}$$

Lecture 6 – Deeplearning 2

$x \longrightarrow \bigcirc \longrightarrow y$ | $y = wx + b$ (slope, intercept)

w → weight
b → bias



$W^{[in]}, W^{[1]}, W^{[2]}, W^{[out]}$
$b^{[in]}, b^{[1]}, b^{[2]}, b^{[out]}$

$W \rightarrow$ (input x no. of nodes)
$b \rightarrow$ (1 x no. of nodes)

$z^{[in]} = W^{[in]T} x + b^{[in]}$ $(w^T x + b)$

$$z^{[in]} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \cdot x + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$z^{[1]} = W^{[1]T} z^{[in]} + b^{[1]}$ ⇒ (4) → $z^{[2]}$ → $z^{[out]}$ → y

$W^{[1]T}$: (2,4) · $z^{[in]}$ + $b^{[1]}$: (4)

$z^{[1]} = W^{[1]T}(W^{[in]T} x + b) + b^{[1]}$ → always linear

Activations → Non-linearity

↓

Approximate (complex Data (for eg. Images, Audio, Robot Joint Data)

$x = \begin{bmatrix} [-1] \\ [-0.4091] \\ [-0.9987] \\ \vdots \\ (1) \end{bmatrix}$ (bx1)

1 Loss
2 Loss
3
4

↓ Dec

Loss = $(\hat{y} - y)^2$ ✓

$\hat{y}$ → true
y → pred

GD:

$$w' = w - \alpha \frac{\partial L}{\partial w}$$ → Goes down

Lecture 6 – Deep learning 2

x → ○ → y

$z = wx + b$

slope, intercept

w → weight
b → bias

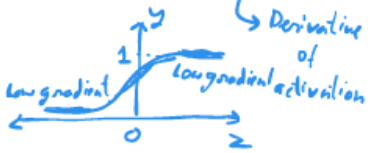
$y = \sigma(z)$

$\frac{\partial L}{\partial w} \rightarrow \frac{\partial L}{\partial y} \left(\frac{\partial y}{\partial z}\right) \frac{\partial z}{\partial w}$

→ Derivative of

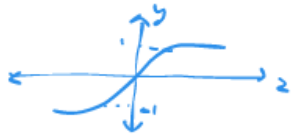
① Sigmoid

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



② Tanh

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



Tanh better sigmoid

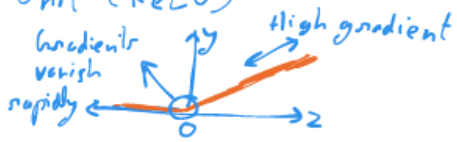
Sigmoid ↙
→ Better for binary

Tanh
→ Better for -ve to +ve range

---

③ Rectified Linear Unit (ReLU)

$y = \max(0, z)$



Tips:

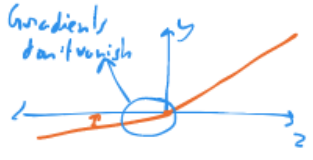
$\sigma(z)$ → Use only output layer

$\tanh(z)$ / ReLU → Use for hidden layer

④ Leaky ReLU:

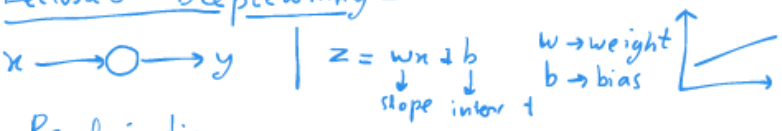
$y = \max(0.01z, z)$

↓ constant



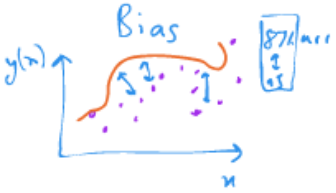
⑤ ... much more → Explore

Lecture 6 – Deeplearning 2



Regularization.

Bayes accuracy → Human level 95% → detecting cat



→ Not enough parameters
→ Not long enough training

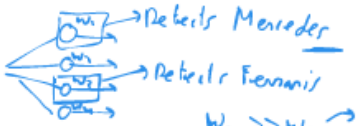
→ Get more unique data
→ Regularization

↳ Fixes the variance

(1) L2 Regularization

$\sum_{i=1}^{m} \frac{1}{m} L(\hat{y}_i, y_i)$ ⇒ loss

m = no. of samples



$w_3 >> w_1$ → Model overfits to detecting Ferraris

↳ Learn abt derivation

$$\sum_{i=1}^{m} \frac{1}{m} L(\hat{y}_i, y) + \frac{\lambda}{2m} \|w\|^2$$

λ ⇒ Scalar

$[w_1\ w_2\ w_3, w_4]$ ⇓ L2 norm $\sqrt{\sum_i^n (w_i)^2}$

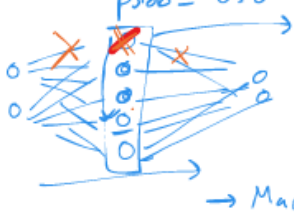
Regularization term

↳ Weighted squared sum of weights
↳ Added in loss
↳ ∴ Try to reduce very high values of weights

---

(2) Dropout

Prob = 0.8 → 4 random nodes → keep 80% of nodes



→ Literally remove some nodes while forward function
↳ In every loop a node disappear.

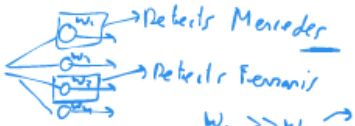
→ Magnitude of gradient decreased

→ Not enough parameters → Get more unique data
→ Not long enough training → Regularization

↳ Fixes the variance

① L2 Regularization

$\sum_{i=1}^{m} \frac{1}{m} L(\hat{y}_i, y)$ ⇒ loss m = no. of samples



$w_3 >> w_1$ → Model overfits to detecting Femanis

↳ Learn abt derivation

$$\sum_{i=1}^{m} \frac{1}{m} L(\hat{y}_i, y) + \frac{\lambda}{2m} \|w\|^2$$

λ ⇒ Scalar

$[w_1\ w_2\ w_3\ w_4]$ ⇓ L2 norm $\sqrt{\sum_{i}^{n} (w_i)^2}$

Regularization term

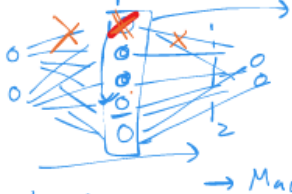
↳ Weighted squared sum of weights
↳ Added in loss
↳ ∴ Try to reduce very high values of weights

---

② Dropout

↳ random nodes
↳ keep 80% of nodes

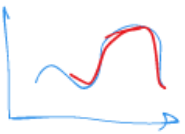
prob = 0.8 → Literally remove some nodes while forward function
↳ In every loop a node disappears.

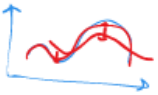


→ Magnitude of gradient decrease
$a(2)$ → will not have 1 nodes' weigh
↳ activation

'a' activate
$[w_1\ w_2\ w_3\ w_4] \begin{bmatrix} w_1 \\ i \end{bmatrix}$

→ $\frac{a(2)}{0.8 \Rightarrow \text{prob}}$ → Helps gradient reduces descent





V. IMP ⇒ Computer Vision
↳ Generalizability