Defining NN Guidelines →

1. Preprocess data →

   A. Normalize

   B. For images, only translation.

2. Choose NN Architecture →

3. Parameter Initialization → Depending on the NN.

4. 1st Step →

   A. Disable regularization.

   B. Check if reasonable loss →

   $$L = \sum_i^{\#C} Y_i \log P_i$$

   2.3026... ∝ log ≈ 2.3 ; Correct for 10 classes.

5. 2nd Step →

   A. Crank up regularization → $1e^3$ ⇒ loss ↑ 3.0685977

6. 3rd Step →

   A. Small piece of data to overfit.

   B. Regularization = 0; num_epoch → loss ≈ 0 ; Accuracy = 100%

7. 4rd Step → Find a learning rate that works

   A. lr = 10^{-6} → Too small ; loss changes.

   B. lr = 10^{-3} → Too high ; explodes.

Define region.

8. Hyper parameter search →

* Implement code to detect explotion and leak-out.

* By now rough idea of lr.

A. Coarse search → 5 epochs.

for count 100

reg = 10^ uniform (-5, 5)

lr = 10^ uniform (-3, -6)

→ do search in log space because of how gradient works.

learning decay = 0.9.

batch size = 100

examples.

Look at Accuracy to define a finer search.

B. Fine search →

reg = 10^ uniform (-5, 5) → uniform (-4, 0)

lr = 10^ uniform (-3, -6) → uniform (-3, -4)

NN Features →

1. Architecture →

   * Deep ; Shallow
   * Regularization
   * Drop out.
   * Weight initialization.
   * Loss function.

2. Hyperparameters →

   * Momentum → $\beta_1 = 0.9$
   * RMSprop → $\beta_2 = 0.999$

   → Adam

   * Learning decay.

X ≡ Input; training set with η_x dimension

$X \in (\eta_x, m) \in \mathbb{R}$ / η_x = dimension of set; m = training set length.

Y ≡ Result of binary classification

$y \in \mathbb{R} \,/\, y \in \{0,1\} \Longrightarrow Y = [y(1) \ldots y(m)]$

$(X, Y) \,/\, X \in \mathbb{R}^{(\eta_x, m)};\ Y \in \mathbb{R}^{(1, m)}$

* Logistic Regression ⟶

$\hat{y} = P(y=1/x)$; probability of $y=1$ for $x \longrightarrow \hat{y}$ estimation.

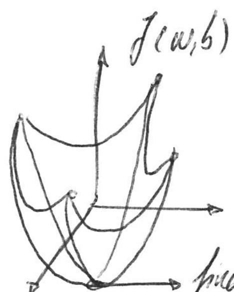
$$\hat{y} = \sigma(W^T \cdot x + b) = \sigma(z) = \frac{1}{1+e^{-z}}$$

* Loss function ⟶ Error function.

$$\mathcal{L}(\hat{y}, y) = -(\log(p(y/x))) = -(y \log \hat{y} + (1-y)\log(1-\hat{y})) = \begin{cases} -\log \hat{y} & y=1 \\ \cancel{(1-y)} \log(1-\hat{y}) & y=0 \end{cases}$$

* Cost function ⟶

$$J(w,b) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}(\hat{y}(i), y(i)) = -\frac{1}{M}\sum_{i=1}^{M} y(i)\log\hat{y}(i) + (1-y(i))\log(1-\hat{y}(i))$$



find w,b that minimizes the cost.

- Idea ⟶ for training set, keep updating w,b value based on $\partial J(w,b) / \partial w \partial b$

$$w = w - \alpha \frac{\partial J(w,b)}{\partial w}$$

$$b = b - \alpha \frac{\partial J(w,b)}{\partial b}$$

* Training set $m=1 \longrightarrow$ 2 features.

x_1
w_1
$x_2 \longrightarrow z = w_1 x_1 + w_2 x_2 + b \longrightarrow a = \sigma(z) \longrightarrow \mathcal{L}(a,y)$
w_2
b

$$\mathcal{L}(a,y) = -[y \log(a) + (1-y)\log(1-a)] \longrightarrow \frac{\partial \mathcal{L}(a,y)}{\partial a} = -\frac{y}{a} + (y-1)\frac{1}{a-1}$$

$$a = \sigma(z) = \frac{1}{1+e^{-z}} \longrightarrow \frac{\partial a}{\partial z} = \frac{e^{-z}}{(1+e^{-z})^2} = a(1-a)$$

$$\frac{\partial \mathcal{L}(a,y)}{\partial z} = a - y$$

- logistic regression for m examples $\longrightarrow$ M training set.

$J=0$; $\partial w_1 = 0$; $\partial w_2 = 0$; $\partial b = 0$ $\longrightarrow$ Inizialization values.

for $i=1$ to M:

$z(i) = w^T x(i) + b \longrightarrow (1 \times n_x) \times (n_x \times 1) + (1)$; $\cancel{z(i) = w_1^T x_1(i) + w_2^T x_2(i) + b}$

$a(i) = \sigma(z(i))$ (1×1)

$n_x = 2$; cross cause now general.

$J += -y[y(i) \log a(i) + (1-y(i)) \log(1-a(i))]$ (1×1)

$dz(i) = a(i) - y(i) = a(i)(1-a(i))$ (1×1)

$dw_1 += x_1(i)\, dz(i)$
$dw_2 += x_2(i)\, dz(i)$ $\longrightarrow$ number of features $\longrightarrow$ this will imply $n_x = 2$.

$db += dz(i)$ (1×1)

$\longrightarrow dw(i) += \underset{n_x \times 1}{x(i)}\ \underset{1 \times 1}{dz(i)}$

$J = J/M$; $dw_1 = dw_1/M$; $dw_2 = dw_2/M$; $db = db/M$

- Vectorized logistic regression $\longrightarrow$

$$Z = W^T X + B \longrightarrow (1 \times n_x)(n_x \times m) + (1 \times m) = (1 \times m)$$

$$A = \sigma(Z) \longrightarrow (1 \times m)$$

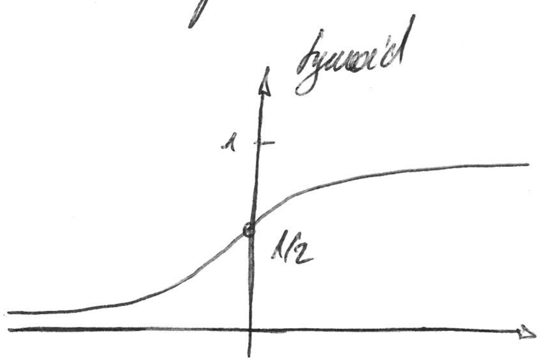
$$dZ = A - Y \longrightarrow (1 \times m)$$

$$dW = \frac{1}{M} X \, dZ^T \longrightarrow (n_x \times M) = (n_x \times m)(1 \times m)^T$$

$$dB = \frac{1}{M} \text{sum}(dZ) \longrightarrow (1, 1)$$

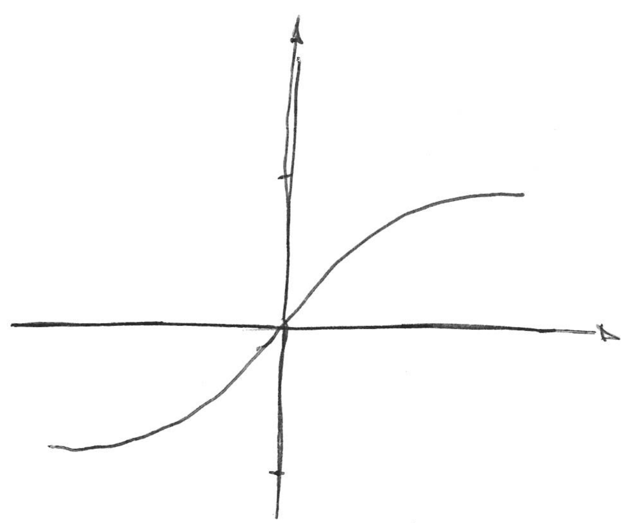
* Shallow Neural Networks $\longrightarrow$

- Activation functions $\longrightarrow$

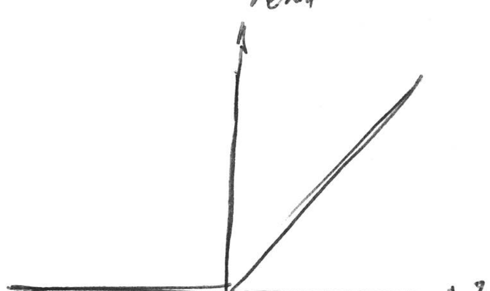


$$\sigma(z) = \frac{1}{1+e^{-z}} \; ; \; \sigma'(z) = \sigma(z)(1-\sigma(z))$$

tanh z

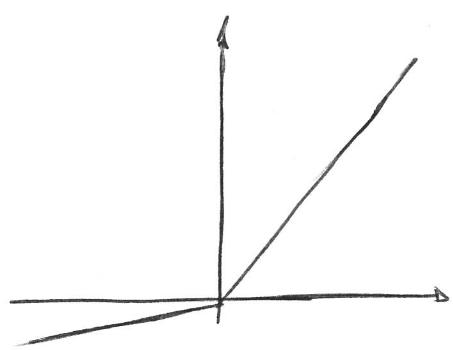


ReLU



$$g(z) = \tanh(z) = \frac{e^{z}+e^{-z}}{e^{z}+e^{-z}} \; ; \; g'(z) = 1 - \tanh^2(z)$$

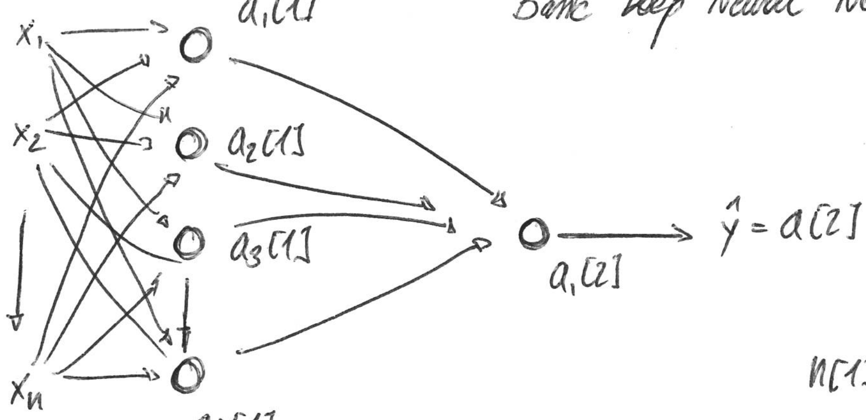
$$g(z) = \max(0, z) \; ; \; g'(z) = \begin{cases} 0 & z<0 \\ 1 & z>0 \\ \text{undef} & z=0 \end{cases}$$



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

$$g'(z) = \begin{cases} 0.01 & z<0 \\ 1 & z>0 \\ \text{undef} & z=0 \end{cases}$$

Basic Deep Neural Networks



$n[1] = I$

$n[2] = 1$

layer [1] = hidden layer

i activation functions.

$$z_i[1] = W_i^T X + B \longrightarrow (1 \times m) = (n_x \times 1)^T (n_x \times m) + (1 \times m)$$

$$A_i[1] = g(z_i[1]) \longrightarrow (1 \times m) = (1 \times m)$$

$$A[1] = g(Z[1]) \longrightarrow (I \times m) = (I \times m)$$

$$Z[2] = W[2]\, A[1] + B[2] \longrightarrow (1 \times m) = (1 \times I)(I \times m) + (1 \times m)$$

$$A[2] = \sigma(Z[2]) \longrightarrow (1 \times m)$$

$$dZ[2] = A[2] - Y \longrightarrow (1 \times m)$$

$$dW[2] = \frac{1}{M} dZ[2] \cdot A[1]^T \longrightarrow (1 \times I) = (1 \times m)(I \times m)^T$$

$$dB[2] = \frac{1}{M} \sum_{i=1}^{M} dZ[2]$$

$$dZ[1] = W[2]^T dZ[2] * g[1]'(Z[1]) \longrightarrow (I \times m) = (1 \times I)^T (1 \times m) * (I \times m)$$

$$dW[1] = dZ[1] \cdot X^T \longrightarrow (n_x \times I) = (I \times m)(n_x \times m)^T$$

$$dB[1] = dZ[1]$$

* Justification gradient descent →

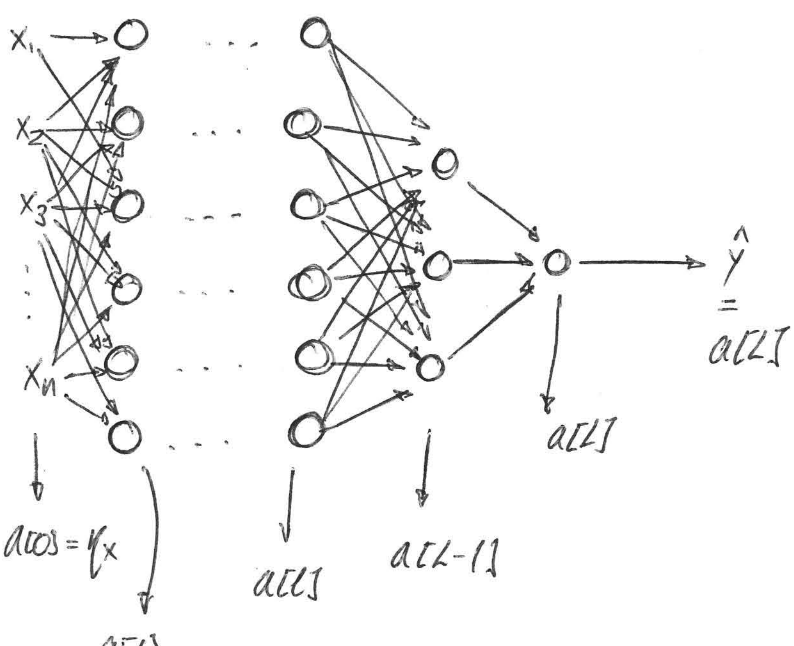
$$\frac{\partial J(a[2],y)}{\partial w_j[1]} = \frac{1}{M}\sum_{i=1}^{M}\frac{\partial L(a[2],y)}{\partial w_j[1]} =$$

$$\frac{1}{M}\sum_{i=1}^{M}\frac{\partial L(a[2],y)}{\partial a[2]}\cdot\frac{\partial a[2]}{\partial z[2]}\cdot\frac{\partial z[2]}{\partial a_i[1]}\cdot\frac{\partial a_i[2]}{\partial z_i[1]}\cdot\frac{\partial z_i[1]}{\partial w_j[1]} =$$

$$\frac{1}{M}\sum_{i=1}^{M}(a[2]-y)\left[w_j[2]\cdot g_j'(z_j[1])\right]\cdot X$$

$$dW[1] = \begin{array}{c} \xrightarrow{n_x} \\ \begin{bmatrix} x_1 w_1[2]\cdot g_1'(z_1[1]) & \dots & x_i w_i[2]\, g_i'(z_i[1]) \\ & & \\ & & \end{bmatrix} \end{array} \quad j\downarrow$$

- Deep L-layer Neural Network →



* Forward Propagation →

$$z[l] = W[l]\cdot A[l-1] + B[l]$$

$$a[l] = g[l](z[l])$$

$$W \in \mathbb{R}(n[l], n[l-1]);\ B \in \mathbb{R}(n[l], m)$$

$$A \in \mathbb{R}(n[l], m)$$

* Backward propagation →

$$dz[l] = dA[l] * g[l](z[l])$$

$$dA[l-1] = W[l]^T dz[l]$$

$$dW[l] = \frac{1}{M} dz[l]\, A[l-1]^T$$

$$dB[l] = \frac{1}{M}\sum_{i=1}^{M} dz[l]$$

$$X \longrightarrow \quad Z[1] = W[1]A[0] + B[1] \quad \xrightarrow{A[1]} \cdots \xrightarrow{A[l-1]} \quad Z[l] = W[l]A[l-1] + B[l] \quad \xrightarrow{A[l]}$$

$$A[1] = g[1](Z[1]) \qquad\qquad A[l] = g[l](Z[l])$$

$$\Big\downarrow\, A[0],\ W[1] \qquad\qquad \Big\downarrow\, A[l-1],\ W[l]$$

$$dZ[1] = dA[1] * g'[1](Z[1]) \qquad\qquad dZ[l] = dA[l] * g'[l](Z[l])$$

$$dW[1] = \frac{1}{m}\, dZ[1]\, A[0]^T \xleftarrow{dA[1]} \cdots \xleftarrow{dA[l-1]} \quad dW[l] = \frac{1}{m}\, dZ[l]\, A[l-1]^T \xleftarrow{dA[l]}$$

$$dB[1] = \frac{1}{m} \sum_{i=1}^{m} dZ[1] \qquad\qquad dB[l] = \frac{1}{m} \sum_{i=1}^{m} dZ[l]$$

$$\cancel{dA[0] = W[0]^T dZ[1]} \qquad\qquad dA[l-1] = W[l]^T dZ[l]$$

$$\Big\downarrow \qquad\qquad \Big\downarrow$$

$$dW[1],\ dB[1] \qquad\qquad dW[l],\ dB[l]$$

$$\cdots \xrightarrow{A[L-1]} \quad Z[L] = W[L]A[L-1] + B[L] \longrightarrow A[L] = \hat{Y}$$

$$A[L] = g[L](Z[L])$$

$$\Big\downarrow\, A[L-1],\ W[L]$$

$$dZ[L] = dA[L] * g'[L](Z[L])$$

$$\cdots \xleftarrow{dA[L-1]} \quad dW[L] = \frac{1}{m}\, dZ[L]\, A[L-1]^T \longleftarrow dA[L] = -\frac{Y}{A[L]} + \frac{(1-Y)}{(1-A[L])}$$

$$dB[L] = \frac{1}{m} \sum_{i=1}^{m} dZ[L]$$

$$dA[L-1] = W[L]^T dZ[L]$$

Backpropagation notes

$$A = g(Z) \; ; \quad A, Z \in \mathbb{R}^{(n_A, m)}$$

$$Z = W A\_prev \; ; \quad A_{prev} \in \mathbb{R}^{(n_x, m)} \; ; \quad W \in \mathbb{R}^{(n_A, n_x)}$$

$$\frac{\partial Z_{kl}}{\partial W_{ij}} = \begin{vmatrix} 0 & \text{if } k \neq i \\ A\_prev_{l,j} & \text{if } k = i \end{vmatrix} = A\_prev_{lj} \; ; \quad \frac{\partial A_{kl}}{\partial Z_{ij}} = g'(Z_{kl})\, \delta_{k,i}\, \delta_{l,j}$$

$$\frac{\partial A_{kl}}{\partial W_{ij}} = \sum_{e=1}^{n_A} \sum_{f=1}^{m} \frac{\partial A_{kl}}{\partial Z_{ef}} \cdot \frac{\partial Z_{ef}}{\partial W_{ij}} = \sum_{e=1}^{n_A} \sum_{f=1}^{m} g'(Z_{kl}) \cdot \delta_{k,e} \cdot \delta_{l,f} \cdot A_{prev\,fj}\; \delta_{e,i}$$

$$= g'(Z_{kl})\, A\_prev_{l,j}\, \delta_{k,i}$$

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = \sum_{e=1}^{n_A} \sum_{f=1}^{M} \frac{\partial \mathcal{L}}{\partial A_{kl}} \frac{\partial A_{kl}}{\partial W_{ij}} = \sum_{k=1}^{n_A} \sum_{e=1}^{M} \frac{\partial \mathcal{L}}{\partial A_{kl}}\, g'(Z_{kl}) \cdot A\_prev_{l,j}\; \delta_{k,i} =$$

$$= \sum_{l=1}^{m} \frac{\partial \mathcal{L}}{\partial A_{il}} \cdot g'(Z_{il}) \cdot A_{l,j} = \left(\frac{\partial \mathcal{L}}{\partial A_{i1}} \cdot g'(Z_{i1}) \cdot A_{1j} + \dots + \frac{\partial \mathcal{L}}{\partial A_{in_A}}\, g'(Z_{in_A})\, A_{n_A j} \right)$$

* Vectorized implementation ⟶

$$dW = (dAL * g'(Z)) \cdot A_{prev}^{T}$$

element wise multiplication.

dot product

$$\frac{\partial Z_{ke}}{\partial A_{prev_{ij}}} = W_{ki} \, \delta_{e,j} \quad ; \quad \frac{\partial A_{ke}}{\partial A_{prev_{ij}}} = \sum_{e=1}^{n_A} \sum_{f=1}^{m} \frac{\partial A_{ke}}{\partial Z_{ef}} \cdot \frac{\partial Z_{ef}}{\partial A_{prev_{ij}}} = g'(Z_{ke}) \cdot W_{k,i} \cdot \delta_{e,j}$$

$$\frac{\partial L}{\partial A_{prev_{ij}}} = \sum_{e=1}^{n_A} \sum_{f=1}^{m} \frac{\partial L}{\partial A_{ef}} \cdot \frac{\partial A_{ef}}{\partial A_{prev_{ij}}} = \sum_{e=1}^{n_A} \sum_{f=1}^{m} \frac{\partial L}{\partial A_{ef}} \cdot g(Z_{ef}) \, W_{e,i} \, \delta_{f,j}$$

$$= \sum_{e=1}^{n_A} \frac{\partial L}{\partial A_{ej}} \, g'(Z_{ej}) \, W_{e,i}$$

* Vectorized implementation. $\longrightarrow$

$$dA_{prev} = W^T \cdot \left(dAL * g'(Z) \right)$$