Image Classification Pipeline

- An image is a big grid of numbers between [0, 255]

  800 x 600 x 3
  ↳ 3 channels of RGB

  3 x 3 x 3 = 3072 numbers in an image

- First classifier: Nearest Neighbor
  - Distance Metric to compare images

    L_1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$ # Manhattan

- K-Nearest Neighbors

  L_2 distance: $d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$ # Euclidean

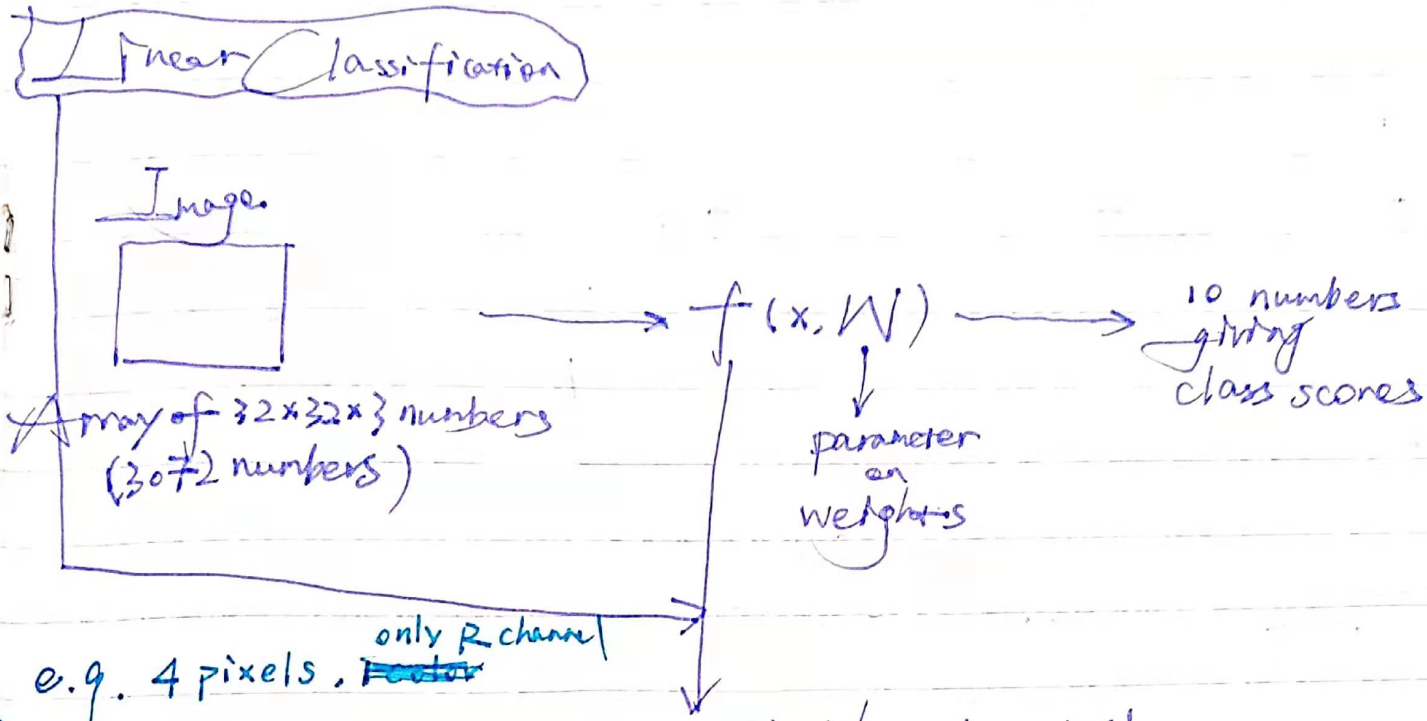
  - Decide on the best value of k and distance
  - Split data into train, validation and test.

    Choose hyperparameters on validation & evaluate

    on test → only once at the end

    Cross-validation

Linear Classification

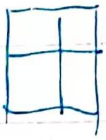


e.g. 4 pixels, ~~Vector~~ only R channel

3 classes

$f(x, W) = Wx + b \Rightarrow 10\times1$

W: 10×3072, x: 3072×1, b: 10×1



$f(x, W) = Wx + b$

W: 3×4, x: 4×1, b: 3×1

- Loss function:

tells how good current ~~classifier~~ is

$$L = \frac{1}{N}\sum_i L_i(f(x_i, W), y_i)$$

actual label (x_i, y_i): x_i is the img & y_i is the integer label

$s = f(x_i, W)$

scores vector

SVM Loss: $L_i = \sum_{j\neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$

正确的score比当前score高，该判断正确

$$= \sum_{j\neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i$$

Regularization

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

prevent model from doing too well on training data

prefers simpler models

L2 Regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 Regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Softmax Classifier (Multinomial Logistic Regression)

$$P(Y=k \mid X=x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \qquad \frac{e^{s_k}}{\sum_j e^{s_j}}$$

$$L_i = -\log P(Y=y_i \mid X=x_i)$$

最大似然估计

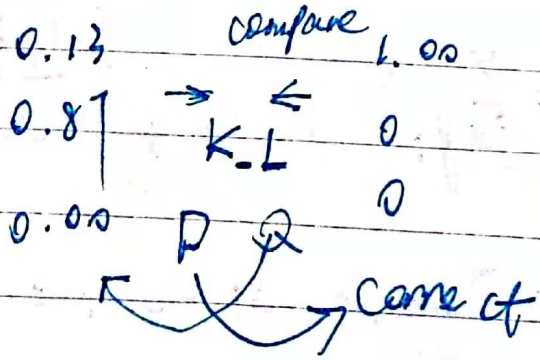
1. ~~相对熵~~ (Kullback-Leibler divergence)

$$D_{KL}(P \| Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$$

2. ~~交叉熵~~ (Cross Entropy) ☆

$$H(P, Q) = H(P) + D_{KL}(P \| Q)$$

1. e.g.



2. e.g.

Same as 1

...

Summary:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$ Softmax

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$ SVM

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$ full Loss

$$R(W) = \sum_k W_k^2$$

Optimization | find the best W for best loss

1. random search & update W & Loss

2. gradient descent

$$\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

$$W += -\text{step\_size} * \nabla_W$$

Image Features

Build Codebook
- Extract random patches
- Cluster patches to form codebook of visual words

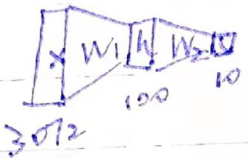
Encode Images

Neural Networks & Backpropagation

2-layer Neural Network : $f = W_2 \max(0, W_1 x)$

$f = W_2 \max(0, W_2 \max(0, W_1 x$

3- _____



$x \in \mathbb{R}^D$

$W_1 \in \mathbb{R}^{H_1 \times D}$

$W_2 \in \mathbb{R}^{H_2 \times H_1}$

$W_3 \in \mathbb{R}^{C \times H_2}$

$\max(0, z)$: activation function

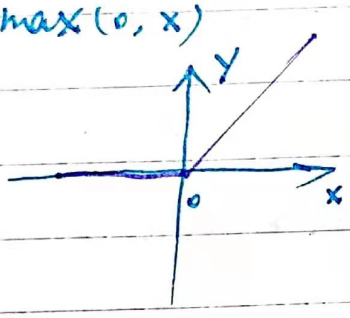
Sigmoid

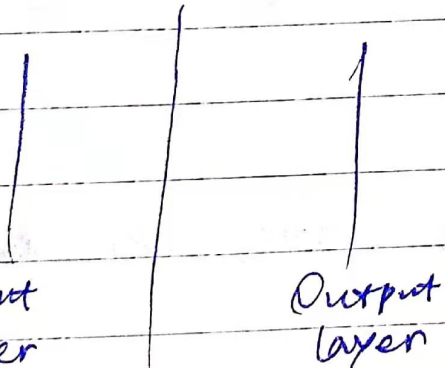
tanh

ReLU ☆ default

ReLU :

$\max(0, x)$





2-layer NN

- Forward pass of a NN :

```
f = #sigmoid
x = np.random.randn(3, 1) # random input
h1 = f(np.dot(W1, x)+b1)
h2 = f(np.dot(W2, h1)+b2)
out = np.dot(W3, h2)+b3
```

x : 3x1

W_1 : 4x3

W_2 : 4x4

W_3 : 1x4

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \lambda R(W_1) + \lambda R(W_2)$$

☆ Backpropagation to compute $\frac{\partial L}{\partial W_1}$ $\frac{\partial L}{\partial W_2}$

e.g. $f = (x+y)z$, $q = x+y$ $\Rightarrow \frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial y}$

↑ upstream gradient, ↓ local gradient

Forward pass: compute output

Backward pass: compute grads

Vector Derivatives:

$x \in \mathbb{R}^N$, $y \in \mathbb{R}^M$

$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M}$

$$\left[\frac{\partial y}{\partial x}\right]_{n,m} = \frac{\partial y_m}{\partial x_n}$$

off-diagonal entries always 0

Jacobian * upstream gradient

$$\frac{dL}{dx} = \frac{dy}{dx} \cdot \frac{dL}{dy}$$

$$\left(\frac{\partial L}{\partial x}\right)_i = \begin{cases} \left(\frac{\partial L}{\partial y}\right)_i & \text{if } x_i > 0 \\ 0 & \text{o.w.} \end{cases}$$

$\begin{bmatrix}1\\2\\3\end{bmatrix}$ $\begin{bmatrix}1\\2\\3\\4\end{bmatrix}$ $\begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix}$

$N=3$ $M=4$ $\mathbb{R}^{3\times 4}$

Jacobian e.g. $f(x,y,z) = (xy+2yz, 2xy^2z)$

$$\begin{bmatrix} y & x+2z & 2y \\ 2yz & 4xyz & 2xy^2 \end{bmatrix}$$

→ $xy+2yz$ 對 x, y, z 分別

→ $2xy^2z$ 分別

No Explicit

- Jacobian : Memory not enough

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & - & - & w_{13} \\ - & - & - & w_{23} \end{bmatrix}$$

$$\frac{\partial L}{\partial X} : \begin{bmatrix} - & - \\ - & - \end{bmatrix} \qquad Y = XW = [\quad]_{2\times 3} \qquad \frac{\partial L}{\partial Y} : [\quad]_{2\times 3}$$

$$\frac{\partial L}{\partial x_{11}} = \frac{\partial L}{\partial Y} \cdot \frac{\partial Y}{\partial x_{11}}$$

dot product to get a scalar

$\frac{\partial Y}{\partial x_{11}}$ → same shape as Y

$\frac{\partial L}{\partial Y}$ → same shape as Y

$$\begin{cases} \frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} W^T \\ \frac{\partial L}{\partial W} = X^T \frac{\partial L}{\partial Y} \end{cases}$$