KDG Code Understanding

After the network is trained,

set of all classes.

for c in C:

    $\mu_p = []$
    $\sigma_p = []$
    $X = x[c]$
    polytopes ← get polytopes.

get polytopes function

- get the weights & biases of the trained network.

$$z_i = W_i a_{i-1} + b_i$$

preactivation

- For each layer except for the last layer ⇒

$$\mathbb{1}[z_i > 0] \quad \text{(similar to ReLU)} \qquad a_i = \mathrm{ReLU}(z_i)$$

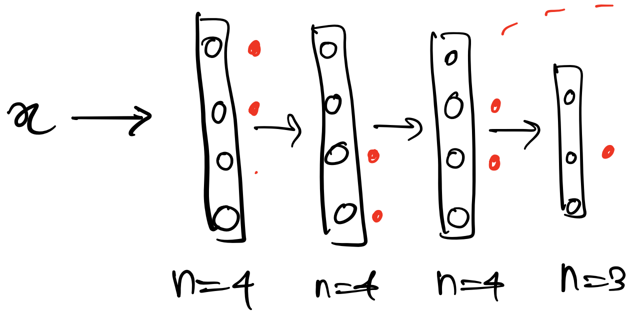
$$\begin{bmatrix} -1 \\ -2 \\ 0.5 \\ 1.2 \end{bmatrix} \xrightarrow{\mathbb{1}} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

- For the last layer.

$$\mathbb{1}[z_i > 0.5]$$

thresholding a probability given by sigmoid/softmax layer.

$$\begin{bmatrix} 0.8 \\ 0.9 \\ 0.1 \\ 0.2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$



polytope memberships

[1100 0011 0110 010]

Polytope obs →

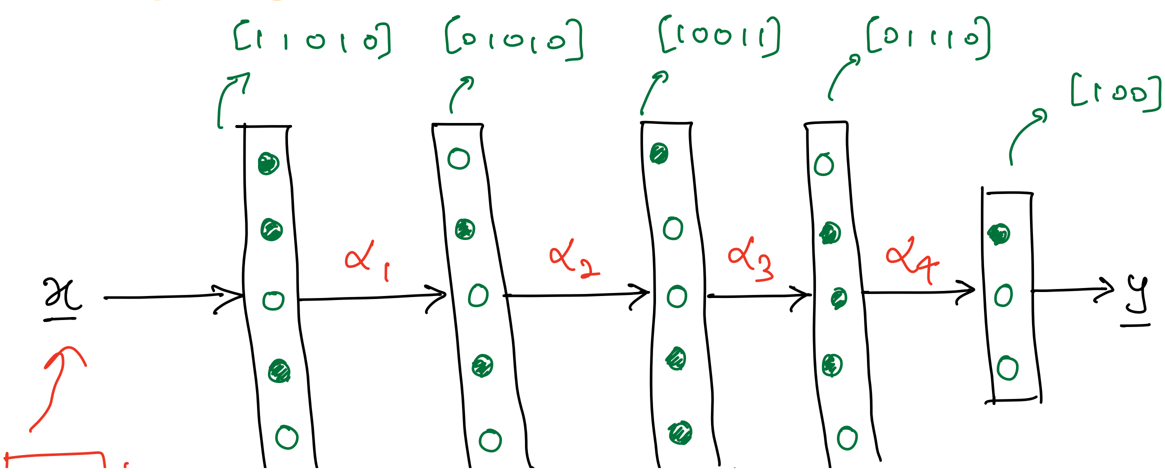
$$\begin{array}{c} \# \text{ neurons} \\ \downarrow \end{array} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

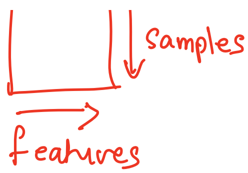
$$\xrightarrow{\hspace{3em}} \quad \# \text{ layers}$$

polytope memberships →

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 2^0 \\ 2^1 \\ 2^2 \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ 6 \\ 2 \end{bmatrix}$$

"_Get_Polytopes" Function ⇒





$$a_i = \sigma(W_i a_{i-1} + b_i)$$
$$a_i^T = \sigma(a_{i-1}^T W_i^T + b_i^T)$$

$$\alpha_i = \sigma(\alpha_{i-1} \omega_i + \beta_i)$$

polytope memberships $= \{ [11010], [01010], \ldots, [100] \}$

polytope observations $= \{ 11010 \mid 01010 \mid \ldots \mid 100 \}$

$$\left.\text{polytope memberships}\right\} = \left[11010\ 01010\ \underbrace{\ldots}_{n \text{ digits}}\ 100 \right] \begin{bmatrix} 2^0 \\ 2^1 \\ 2^2 \\ \vdots \\ 2^{n-1} \end{bmatrix}$$

returned by the get polytopes function for each data sample.

* if two data samples have similar neural network activations, they belong to the same polytope.

Gaussian Mixture Modelling

* A probabilistic model that assumes all data points are generated from a finite number of gaussian distributions with unknown params.

$$p(\theta) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\mu_i, \Sigma_i)$$

"fit" function ⇒

* takes in the dataset $(X, y) \in \mathbb{R}^{n \times d} \times \mathcal{Y}^k$
where $k = \{1, \ldots, K\}$

* Fits a neural network to (X, y)

* For each class label k,
  - $P_k \leftarrow$ _get_polytopes(X_k) ← samples with label k
    ↳ an array with length(X_k) contains the polytope ID of each data sample of X_k
  - for each polytope ID ⇒
    - obtain the indices of all the other data samples of X_k that has the same polytope ID
      (Hence obtain the data samples of X_k that belongs to the same polytope)
      neglect the polytopes with just 1 sample.
    - Using the data samples of class k

that belongs to the polytope, fit a gaussian mixture model & obtain it's means / covariance matrices (params).

"_compute_pdf" function ⇒

* given a class label and a polytope ID, compute the PDF of the respective polytope group using the gaussian mixture parameters of that polytope.

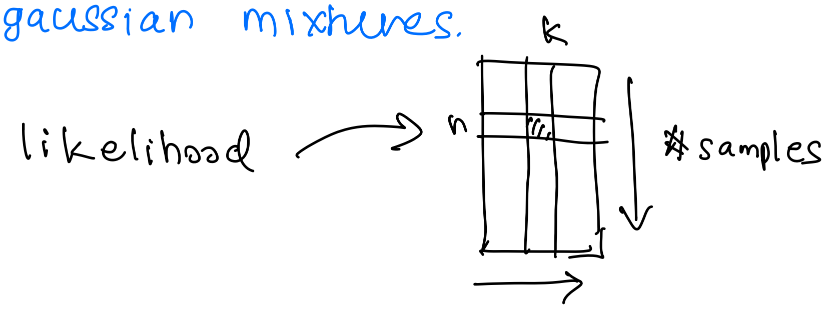
multivariate gaussian distribution

* likelihood = the PDF evaluated at a given data sample point x

according to the code

"predict_proba" function ⇒

* given the test data, obtain the posterior distribution using the stored polytopic gaussian mixtures.



classes.

$$\left.\begin{array}{c}\text{likelihood of kth class}\\ \text{at the nth sample}\end{array}\right\} = \sum_{i=1}^{|P_k|} G(x; \mu_i, \Sigma_i)$$

$$\left.\begin{array}{c}\text{probability of kth class}\\ \text{at the nth sample}\\ P(k \mid x)\end{array}\right\} = \frac{\sum_{i=1}^{|P_k|} G(x; \mu_i, \Sigma_i)}{\sum_{k=1}^{K} \sum_{i=1}^{|P_k|} G(x; \mu_i, \Sigma_i)}$$

∴ No need for the test data to pass through the trained neural network.

"prediction" function ⇒

$$\hat{k} = \arg\max_{k} P(k \mid x) \longrightarrow \text{predicted class.}$$