Bayes Classifier with Gaussians, DINO-Features and PCA

- K Classes
- For each image, we can encode it into a $\underline{x} \in \mathbb{R}^d$ feature representation of dimensionality d.
- We have some train data $\{(\underline{x}, y_i)\}_{i=1}^{N}$ for N samples and each one has label $y_i \in \{0, 1, \dots (K-1)\}$
- So, our data matrix will be: $\underline{\underline{X}} = \begin{bmatrix} -\underline{x}_1- \\ -\underline{x}_2- \\ \vdots \\ -\underline{x}_N- \end{bmatrix}_{N \times d}$
- And, our ground truth "vector": $\underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N \times 1}$
- Since classes are mostly defined by names e.g. "airplane", "car", and our $y_i \in \mathbb{R}$, we need a mapping, simply:

$$\text{label\_map} = \{\text{"airplane"}: 0, \text{"auto"}: 1 \dots\}$$

- For our matrix operations, we will use the one-hot-encoding for the labels $\underline{y}$, e.g. if $K = 4$ and $\underline{y} = [1, 0, 3, 2]$ our one hot encoding matrix is: $\underline{\underline{Y}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ (Class 1, Class 2, Class 3, Class 4; Sample 1, Sample 2, Sample 3, Sample 4)
- In general: $\underline{\underline{Y}} \in \mathbb{R}^{N \times K}$
- In our code, in one_hot_encode, given $\underline{y}$ and K (num classes) we create a zeros matrix first: $\underline{\underline{Y}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

$\underline{y} = [1, 0, 3, 2]$ — set at row 0 col 1 to 1; set at row 1 col 0 to 1; set at row 2 col 3 to 1

2) $\underline{\underline{Y}}[[0,1,2,3], [1,0,3,2]] = 1 \rightarrow \underline{\underline{Y}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

"rows" np.arange(N) — $\underline{y}$

- For Bayes Classifier, we are interested in computing the Posterior:

$$P(Y=K \mid X=x_i) = \frac{P(X=x_i \mid Y=K) \cdot P(Y=K)}{P(X=x_i)}$$

$P(Y=K \mid X=x_i)$: Given sample data x_i prob. it is class K → Posterior

$P(X=x_i \mid Y=K)$: Conditional. Given $Y=K$, probability of observing x_i

$P(Y=K)$: prior over class K.

$P(X=x_i)$: prob. of observing x_i

- At the end of the day, for a data x_i, in order to assign it to a class K, we just compute:

$$\hat{K} = \underset{K \in C}{\arg\max}\ P(Y=K \mid X=x_i)$$

$$= \underset{K \in C}{\arg\max}\ \frac{P(X=x_i \mid Y=K)\, P(Y=K)}{P(x_i)}$$

↳ since this does not depend on K we can drop it.

$$\hat{K} = \underset{K \in C}{\arg\max}\ P(X=x_i \mid Y=K)\, P(Y=K)$$

- For the prior of class K, we can estimate it empirically just by counting of all the Dataset samples N, how many belong to class K.

$$P(Y=K) = \frac{N_K}{N}$$

- In our code, we can easily do this by summing over the rows of our one-hot-encoded Y and dividing by N.

$$\text{prior} = \frac{1}{4}[1\ 1\ 1\ 1] = [0.25\ 0.25\ 0.25\ 0.25]$$

$$Y = \begin{matrix} K & 0 & 1 & 2 & 3 \end{matrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{4 \times 4} \quad \Sigma_i$$

- The conditional prob. $p(x=x_i \mid y=K)$: this we can set to a Gaussian for which we need to determine its Mean μ_K and Covariance Σ_K. Given our dataset with labels we can estimate these are:

$$\mu_K = \frac{1}{N_K} \sum_{\substack{i=1 \\ y_i=K}}^{N_K} x_i$$

$$\Sigma_K = \frac{1}{N_K - 1} \sum_{i=1}^{N_K} (x_i - \mu_K)(x_i - \mu_K)^T$$

↳ In Matrix Form:

Mean: $\mu_K = \frac{1}{N_K} \begin{bmatrix} — x_1 — \\ — x_2 — \\ \vdots \\ — x_N — \end{bmatrix} = \frac{1}{N_K} X\,[Y[:,K]==1,:].\text{mean}(0)$ *

↳ "Mask"

* Mean over selected rows.

Mean only over x_i which $Y[i]=K$.

$Y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ ↳ $K=1$ ↳ only sample 0 and 3.

Covariance: $\Sigma_K = \frac{1}{N_K - 1} \tilde{X}_K^T \tilde{X}_K$ where

Contains only points belonging to class K. → $\tilde{X}_K \in \mathbb{R}^{N_K \times d}$ is around μ_K centered points/data

$$\tilde{X}_K = X_K - \mu_K$$

* For numerical stability since later in Gaussian Formula we need to invert Σ_K to avoid singular matrix

we add : $\epsilon \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ to Σ_K: $\hat{\Sigma}_K = \Sigma_K + \epsilon \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

- The Multi-Variate Gaussian Formula reads:

$$f_k = \frac{1}{(2\pi)^{d/2} |\underline{\underline{\Sigma}}_k|^{1/2}} \exp\left[-\frac{1}{2}(\underline{x}-\underline{\mu}_k)^T \underline{\underline{\Sigma}}_k^{-1} (\underline{x}-\underline{\mu}_k)\right]$$

↳ This is $p(x|y=k)$

- We remember we wanna: $\underset{k \in C}{\arg\max}\ p(x|y=k)\, p(y=k)$

→ Since $\log(x) > 0$ we can also:

$$\underset{k \in C}{\arg\max}\ \log\ p(x|y=k)\, p(y=k)$$

$$= \underbrace{\log p(x|y=k)}_{\text{Log of Condit. (Gaussian for us)}} + \underbrace{\log(p(y=k))}_{\text{Log of Prior.}}$$

$$= \log \frac{1}{(2\pi)^{d/2} |\underline{\underline{\Sigma}}_k|^{1/2}} \exp\left[-\frac{1}{2}(\underline{x}-\underline{\mu}_k)^T \underline{\underline{\Sigma}}_k^{-1} (\underline{x}-\underline{\mu}_k)\right] + \log(p(y=k))$$

$$= \log\left(\frac{1}{(2\pi)^{d/2} |\underline{\underline{\Sigma}}_k|^{1/2}}\right) - \frac{1}{2}(\underline{x}-\underline{\mu}_k)^T \underline{\underline{\Sigma}}_k^{-1} (\underline{x}-\underline{\mu}_k) + \log(p(y=k))$$

$$= \underbrace{\log\left(\frac{1}{(2\pi)^{d/2}}\right)}_{\text{Const. Term. Same for all } k, \text{ So we can drop it.}} \log\left(|\underline{\underline{\Sigma}}_k|^{-1/2}\right) - \frac{1}{2}(\underline{x}-\underline{\mu}_k)^T \underline{\underline{\Sigma}}_k^{-1} (\underline{x}-\underline{\mu}_k) + \log(p(y=k))$$

$$f_k = \underset{k \in C}{\arg\max} \left[-\frac{1}{2}\log\left(|\underline{\underline{\Sigma}}_k|\right) - \frac{1}{2}(\underline{x}-\underline{\mu}_k)^T \underline{\underline{\Sigma}}_k^{-1} (\underline{x}-\underline{\mu}_k) + \log(p(y=k))\right] \quad [A]$$

↑ determinant of Cov. Matrix

↑ Inverse of Cov. Matrix.

- All in all, after we fit Gaussians, to predict class for a new data point $\underline{x}_i$, we compute for every class K g_K with [A] formula and pick the one with max. value.

$$\underline{g} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} \overset{\text{eg.}}{=} \begin{bmatrix} 2 \\ 4 \\ 7 \\ 1 \end{bmatrix}$$

$$K_{pred} = \arg\max(\underline{g}, 1) = 3$$

Extracting Features for Images:

- Assume we wanna classify images belonging to C classes e.g. "airplane", "car"...
- In older days, we might construct for N images our feature matrix as: $\underline{X} = \begin{bmatrix} -\underline{x}_1- \\ \vdots \\ -\underline{x}_N- \end{bmatrix}$ where $\underline{x}_i$ is the flattened rgb pixel data.
  - ↳ Issues: 1) For 1024 x 1024 RGB images (3 channels) this results in $\underline{x}_i \in \mathbb{R}^{3.14 \text{ Millions}}$ crazy large dimensions.
    - ↳ Cov. Matrix will be 3.14 x 3.14 Mil. Shape ....
- Nowadays, we can use powerful Vision Foundation Models such as DINO to extract Features from our images

→ [DINO] → $\underline{x}_i \in \mathbb{R}^{315}$ Feature

Feature Encoder

- Still, to further compress the feature dimension without loosing too much information, we can apply Principal Component Analysis PCA which maximizes the variance after projecting the data to a lower dimensional space, so we could go from e.g. $\underline{x} \in \mathbb{R}^{315} \longrightarrow \underline{z} \in \mathbb{R}^{50}$. This saves us computations and numerical stability issues when computing $\overline{\Sigma}_{ik}^{-1}$ and $\det(\overline{\Sigma}_{ik})$.