1 Module 2: Fundamentals of ML

Contents

1.1 Uniform distributions

- Models situations with just two outcomes (a,b) which are equally probable
- Discrete uniform distribution models n values from low value a to high value b (e.g. a roll of die n=6, a=1, b=6, p(n)=1/6)
  - n = b - a + 1
  - p(n) = $\frac{1}{n}$
  - Expectation = $\frac{a+b}{2}$
  - Variance = $\frac{n^2-1}{12}$
- Continuous uniform distribution (e.g. angle of a clock at any time in the day: a = 0deg, b = 360deg, all probabilities are equally likely therefore $\int_a^b$[PDF of clock values] = 1)
  - p(n) = $\frac{1}{b-a}$
  - Expectation = $\frac{a+b}{2}$
  - Variance = $\frac{(b-a)^2}{12}$

1.2 Sampling uniform distributions in Python

- scipy.stats.uniform(loc = [left edge] , scale = [length])

  e.g. dist. from a=10 to b=15 → loc = 10 and scale = 5 *

- .pdf(loc=) to sample likelihood at a point
  * e.g. .pdf(8) = 0, .pdf(12) = 0.2, .pdf([8,12,15]) = [0,0.2,0.2]
- .mean() * e.g. .mean() = 12.5
- .var() * e.g. = 2.08
- .std() * e.g. = 1.44
- .rvs(size=) : Random variates sampled from dist
  * e.g. .rvs()=any val∈[10,15] or .rvs(size = 100, density = True) to plot a histogram with proper scaling using density

1.3 Gaussian (normal) distributions

- Random (often, measurement) error induces small variations around a mean
- Gaussians are parameterized by μ mean and σ^2 variance
- Central Limit Theorem – for the mean and variance of a random variable $\mu(\bar{X_n})$ and $\sigma^2_{\bar{X_n}}$, the $\lim_{n\to\infty} \mu_{\bar{X_n}} = \mu_X$ (larger n converges to more correct mean) and $\sigma^2_{\bar{X_n}} \to \frac{\sigma^2_X}{n}$ (larger n decreases variance)
- PDF = $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

1.4 Multivariate distributions

- A collection of random variables $X = \begin{bmatrix} X_1 \\ X_2 \\ ... \\ X_n \end{bmatrix}$ with joint distribution $f_x(x_1, x_2, ..., x_n)$ which can be sampled as $X \to \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix}$
- Expectation $E[X] = \begin{bmatrix} E[X_1] \\ E[X_2] \\ ... \\ E[X_n] \end{bmatrix}$
- Law of large numbers $\bar{\mu_n} \to E[X] as n \to \infty$

- Central Limit Theorem remains the same $\bar{X}_n \to N(\mu_x, \sigma_x^2/n)$ as $n \to \infty$
- Variance becomes Covariance Matrix $Covar[X] = \Sigma_X^2 = \begin{bmatrix} \sigma_{1,1}^2 & \sigma_{1,2}^2 & \dots & \sigma_{1,n}^2 \\ \sigma_{2,1}^2 & \sigma_{2,2}^2 & \dots & \sigma_{2,n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n,1}^2 & \sigma_{n,2}^2 & \dots & \sigma_{n,n}^2 \end{bmatrix}$
  - Each σ^2 still generated from $(X - E[X])^2$
  - Entries on the diagonal of the matrix $(\sigma_{1,1}^2, \sigma_{2,2}^2, etc.)$ are the variances of the individual random variables
  - The off-diagonal elements are covariances between variables, meaning $\sigma_{1,2}^2 = \sigma_{2,1}^2$ for variables X_1, X_2
  - Can estimate covariance of a dataset using the sample covariance $\hat{\Sigma_X^2} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{mu})^T$
  - True (population) covariance $\Sigma_X^2 = E[\ (X - E[X])(X - E[X])^T\ ]$vi

1.5 Covariance and correlation matrices in Python

- DataFrame.cov()
- DataFrame.corr(): normalizes to ones on the diagonal so it is easier to see covariances
- sns.pairplot(DataFrame): diagonal plots = correlation scatter plot, off-diagonal plots = histogram

1.6 Correlation, Conditional Probability & Independence

- $Corr[X] = \Sigma_X^2 = \begin{bmatrix} 1 & \rho_{1,2} & \dots & \rho_{1,n} \\ \rho_{2,1} & 1 & \dots & \rho_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{n,1} & \rho_{n,2} & \dots & 1 \end{bmatrix}$ for $\rho_{i,j} = \frac{\sigma_{i,j}^2}{\sigma_i \sigma_j} \in [-1, 1]$
- Lack of correlation does not mean that the data does not have a pattern, given that correlation only evaluates on linearity (uncorrelated, but not independent)

- Conditional probability: predict on one value when the other values are fixed (on a scalar or a range)
  $p_Y(y|X = x) = \frac{p_{X,Y}(x,y)}{p_X(x)}$ gives a PDF over values of y (slice of the joint PDF using X at x)
- Conditional distribution using .histplot(data, x, **hue**)
- Two random variables are independent when knowing the value of one tells us nothing about the value of the other
  $p_Y(y|X = x_1) = p_Y(y|X = x_2) = p_Y(y)$; as such, the joint distribution $p_{X,Y}(x,y) = p_X(x)p_Y(y)$