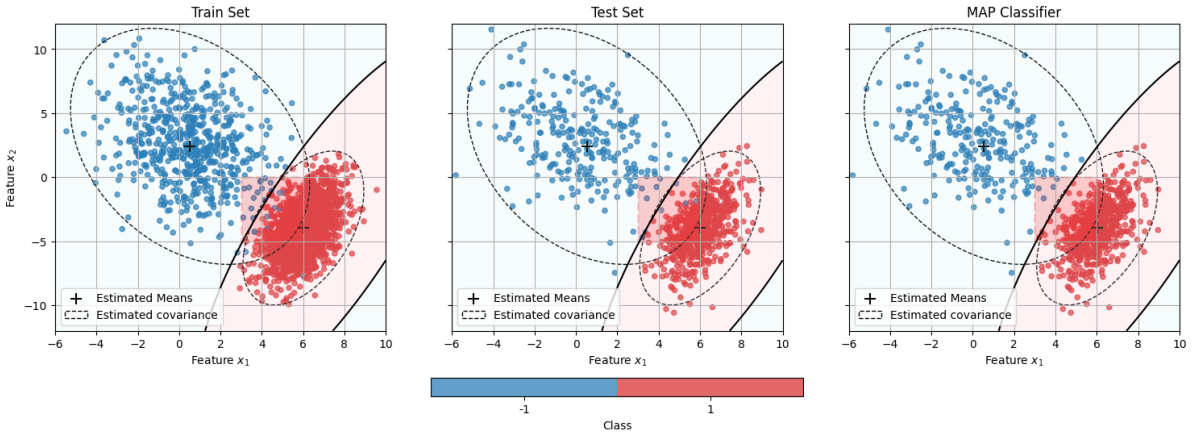
PPGEE2249 Aprendizado de Máquina
Assignment 1
Prof. Daniel Guerreiro e Silva

Thiago Tomás de Paula
November 11, 2025



▶ Question 1

X is a continuous-valued random variable with uniform density in $(-1, +1)$.

▷ Item a

Draw its probability density function (PDF). What is the area under the curve? Justify your answer.

▷ Item b

Draw its cumulative distribution function (CDF).

▷ Item c

Calculate the probability of the event $X \in (-0.2, 0.2)$.

▷ Item d

Calculate the expected value $\mathrm{E}[X]$, the second $\mathrm{E}[X^2]$ and the fourth moment $\mathrm{E}[X^4]$ of the random variable. Calculate its variance $\sigma^2[X]$, as well.

Let $p : \mathbb{R} \to \mathbb{R}$ denote the PDF of X and $c : \mathbb{R} \to \mathbb{R}$ its CDF. Since the random variable is of uniform density in $(-1, +1)$,

$$p(x) = \begin{cases} k, & |x| < 1 \\ 0, & |x| \geq 1 \end{cases} \tag{1}$$

for some constant $k > 0$. By definition, the area under the PDF curve is equal to the probability P of X assuming a value in the associated range (a, b), $P(a < X < b) = \int_a^b p(x)\, dx$. For $a = -\infty$ and $b = \infty$, $P(-\infty < X < \infty) = 1$, i.e., the probability of X assuming *any* value is 100%. We can use this result (in general, an identity) to find the value of k:

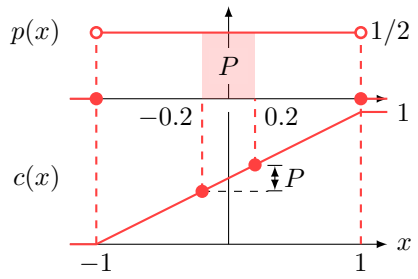
$$P(-\infty < X < \infty) = \int_{-\infty}^{\infty} p(x)\, dx = 1 \implies \int_{|x|\geq 1} p(x)\, dx + \int_{|x|<1} p(x)\, dx = 1 \tag{2}$$

$$\int_{|x|\geq 1} 0\, dx + \int_{|x|<1} k\, dx = 1 \tag{3}$$

$$2k = 1 \quad \therefore \quad k = \frac{1}{2}. \tag{4}$$

With k evaluated, we can find any P, e.g., $P(-0.2 < X < 0.2) = [0.2 - (-0.2)] \cdot (1/2) = 0.2$. A drawing of $p(x)$ and $c(x) = \int_{-\infty}^{x} p(z)\, dz$ is presented in fig. 1.

Figure 1: PDF and CDF of a uniform distribution in $(-1, 1)$.



Next, the variance and some of the moments of p is calculated. By definition, $\mathrm{E}[X^n] = \int_{\mathbb{R}} x^n p(x)\,dx$. For this question's distribution,

$$\mathrm{E}[X^n] = \int_{-1}^{1} x^n \cdot \frac{1}{2}\,dx = \frac{1}{2}\cdot\frac{x^{n+1}}{n+1}\bigg|_{x=-1}^{x=1} = \frac{1+(-1)^n}{2(n+1)} = \begin{cases} 0, & n \text{ odd} \\ 1/(n+1), & n \text{ even} \end{cases} \tag{5}$$

The above expression gives, for example, $\mathrm{E}[X] = 0$, $\mathrm{E}[X^2] = 1/3$, and $\mathrm{E}[X^4] = 1/5$. As for the variance, the definition $\sigma^2[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$ results in $\sigma^2[X] = 1/3$.

► Question 2

X is a discrete-valued random variable with uniform distribution over the set $\{-2, -1, 0, 1, 2\}$. Draw its probability mass function (PMF) and calculate $\mathrm{E}[X]$ and $\sigma^2[X]$.

Let $p : \mathbb{Z} \to \mathbb{R}$ denote the PMF of X. Since the random variable is uniform on $\{x \in \mathbb{Z} : |x| < 3\}$, we have

$$p(x) = \begin{cases} k, & |x| < 3 \\ 0, & |x| \geq 3 \end{cases} \tag{6}$$

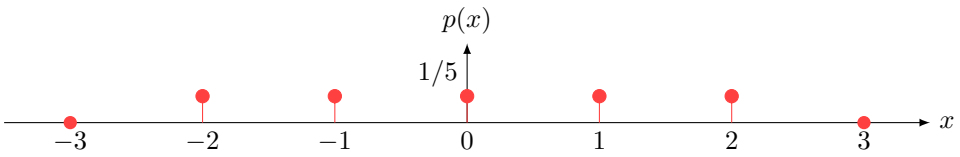
for some constant $k > 0$. Since the PMF must satisfy the identity $\sum_{x\in\mathbb{Z}} p(x) = 1$, we have

$$\sum_{x\in\mathbb{Z}} p(x) = \sum_{|x|\geq 3} p(x) + \sum_{|x|<3} p(x) = 1 \tag{7}$$

$$0 + 5k = 1 \quad \therefore \quad k = \frac{1}{5} \tag{8}$$

The drawing in fig. 2 illustrates the PMF with k evaluated.

Figure 2: PMF of an uniform distribution in $\{-2, -1, 0, 1, 2\}$.



As for $\mathrm{E}[X]$ and $\sigma^2[X]$, we can use the definition for the n-th moment $\mathrm{E}[X^n] = \sum_{x\in\mathbb{Z}} x^n p(x)$ and the definition $\sigma^2[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$:

$$\mathrm{E}[X] = \sum_{|x|<3} \frac{x}{5} = \frac{1}{5}(-2 - 1 + 0 + 1 + 2) = 0 \tag{9}$$

$$\mathrm{E}[X^2] = \sum_{|x|<3} \frac{x^2}{5} = \frac{1}{5}[(-2)^2 + (-1)^2 + 0^2 + 1^2 + 2^2] = 2 \tag{10}$$

$$\sigma^2[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2 = 2 - 0 = 2 \tag{11}$$

▶ Question 3

Consider the normal random variables $X_1 \sim N(-2, 2)$, $X_2 \sim N(1, 4)$, with $\text{cov}(X_1, X_2) = -0.8$. Calculate the joint pdf of $X = (X_1, X_2)^T$.

Let $X_i \sim N(\mu_i, \sigma_i^2)$, $1 \leq i \leq d$ be several univariate normal random variables. The joint PDF $p : \mathbb{R}^{d\times 1} \to \mathbb{R}$ of $\mathbf{x} = [x_1 \; x_2 \; \cdots \; x_d]^\top$ is the multivariate gaussian

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \times \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \tag{12}$$

with $\boldsymbol{\mu} = [\mu_1 \; \mu_2 \; \cdots \; \mu_d]^\top$ and $\Sigma_{ij} = \sigma_{ij} = \text{cov}(X_i, X_j)$ the covariance matrix of $\mathbf{x}$.

The question statement gives

$$\mu_1 = -2, \quad \sigma_1^2 = 2, \quad \mu_2 = 1, \quad \sigma_2^2 = 4, \quad \sigma_{12} = \sigma_{21} = -0.8 \tag{13}$$

which leads to

$$\boldsymbol{\mu} = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \tag{14}$$

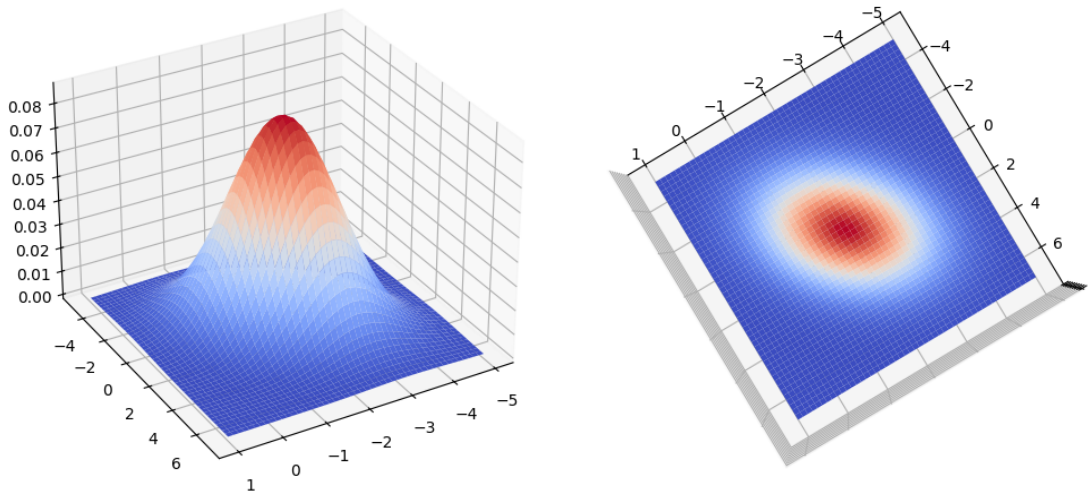
$$\boldsymbol{\Sigma} = \begin{bmatrix} 2 & -0.8 \\ -0.8 & 4 \end{bmatrix}, \quad |\boldsymbol{\Sigma}| = 8 - 0.36 = 7.64, \quad \boldsymbol{\Sigma}^{-1} = \frac{1}{7.64}\begin{bmatrix} 4 & 0.8 \\ 0.8 & 1 \end{bmatrix} \tag{15}$$

and, making all substitutions in (12),

$$p(\mathbf{x}) = \frac{1}{2\pi\sqrt{7.36}} \times \exp\left[-\frac{1}{2}\begin{bmatrix} x_1+2 \\ x_2-1 \end{bmatrix}^\top \frac{1}{7.64}\begin{bmatrix} 4 & 0.8 \\ 0.8 & 1 \end{bmatrix}\begin{bmatrix} x_1+2 \\ x_2-1 \end{bmatrix}\right]. \tag{16}$$

fig. 3 shows a plot of the multivariate distribution given by the expression above.

Figure 3: Joint PDF of $X_1 \sim N(-2, 2)$ and $X_2 \sim N(1, 4)$.



► Question 4

Assume you want to design a classifier which receives univariate data as input and must choose between class +1 or class −1. The data associated with class "+1" follow a gaussian density with mean 2 and unit variance. On the other hand, the data associated with class "−1" follow a uniform density between −2 and 2. The prior probabilities are $P(C_{+1}) = 0.6$ and $P(C_{-1}) = 0.4$.

▷ Item a

Apply Bayes' methodology to obtain the optimal classification model and show, using adiagram, the model's architecture (with clear indication of the discriminant function(s)).

▷ Item b

Calculate the model's decision boundary. Analytically calculate the classifier error probability (the integrals can be numerically solved).

▷ Item c

Recalculate the model's decision boundary if the Maximum-Likelihood criterion is adopted, this time. Analytically calculate the classifier error probability (the integrals can be numerically solved).

▷ Item d

Compare the performances of both methodologies.

A classification model is tasked with assigning a value x on the line as belonging to one of the distributions, i.e. classifying it as "+1" or "−1"; an *optimal* classification model will do so while maximizing its chances/minimizing its error.

It is given an input $x \in \mathbb{R}$ and must output a class C_B such that the conditional probability $P(C = C_B|x)$ is the greatest amongst the considered classes. Using Bayes' theorem, we can write it as

$$P(C_B|x) = \frac{p(x|C_B)P(C_B)}{p(x)} \tag{17}$$

with $p(x|C)$, called likelihood, the probability of x happening given that it comes from class C. Since the expression above is valid not only to the optimal choice but for any class, and $p(x)$ is invariant with respect to classes, we have

$$C_B = \underset{C}{\operatorname{argmax}}\, P(C|x) = \underset{C}{\operatorname{argmax}}\, p(x|C)P(C). \tag{18}$$

Since $x \in \mathbb{R}$, the maximization above splits the real line in compact intervals such that x in adjacent intervals result in different C_B. Classification error comes from the true class of a sample not being the class assigned to the associated interval.

For example, let's say two classes H_0 and H_1 have likelihoods such that the classifier assings $x \in R_0$ to H_0 and $x \in R_1$, to H_1. Then, the probability of error is given by the integration of $P(C = H_1|x \in R_0)$ plus the integration of $P(C = H_0|x \in R_1)$, i.e., the proportion of false negatives and positives with respect to H_1:

$$P_{\text{error}} = P(H_1|x \in R_0) + P(H_0|x \in R_1) \tag{19}$$

$$= P(H_1) \int_{R_0} p(x|H_1)\, dx + P(H_0) \int_{R_1} p(x|H_0)\, dx. \tag{20}$$

Had we begun this answer by assuming a classifier maps regions R_0, R_1 to classes, the optimal classifier would minimize P_{error} by picking looking at the values of $P(H_i)p(x|H_i)$ and chossing H_i with biggest result.

Let's finally apply this Bayesian approach to the problem in question. Since there are two classes and one of them is gaussian, it is useful to rewrite the maximization problem in (18) as

$$C_B = \begin{cases} C_{+1}, & g(x) > 0 \\ C_{-1}, & g(x) < 0 \end{cases}, \quad g(x) = g_B(x) = \ln \frac{p(x|C_{+1})P(C_{+1})}{p(x|C_{-1})P(C_{-1})} \tag{21}$$

The function g_B, the discriminant of the classifier, is explicitly given by

$$\begin{aligned} g_B(x) &= \ln \frac{p(x|C_{+1})P(C_{+1})}{p(x|C_{-1})P(C_{-1})} && (22)\\ &= \ln[p(x|C_{+1})] + \ln[P(C_{+1})] - \ln[p(x|C_{-1})] - \ln[P(C_{-1})] && (23)\\ &= \begin{cases} -\frac{1}{2}\ln(2\pi) - \frac{1}{2}(x-2)^2 + \ln(0.6) - \ln(1/4) - \ln(0.4), & |x| \le 2 \\ \infty, & |x| > 2 \end{cases} && (24) \end{aligned}$$

where it was used the fact that $P(C_{+1}) = 0.6$, $P(C_{-1}) = 0.4$, $p(x|C_{+1}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-2)^2}$, and $P(x|C_{-1}) = 1/4$, if $|x| \le 2$, and $P(x|C_{-1}) = 0$, otherwise. From (24), one can deduce that the classifier decision boundary, i.e., the roots of the discriminant, is a single point $\delta_B = 2 - \sqrt{\ln(36/2\pi)}$; $x > \delta_B$ or $x < -2$ are classified as C_{+1}, and $-2 < x < \delta_B$ are classified as C_{-1}.

On $|x| \le 2$, the error probability of this classifier can be calculated using (20):

$$P^B_{\text{error}} = 0.6 \int_{-2}^{\delta_B} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-2)^2}\, dx + 0.4 \int_{\delta_B}^{2} \frac{1}{4}\, dx = 0.1880\ldots \tag{25}$$

As commented, this classifier minimizes the probability of error. If a different discriminant was used, such as Maximum-Likelihood (ML) criterion, error would increase. Indeed, taking

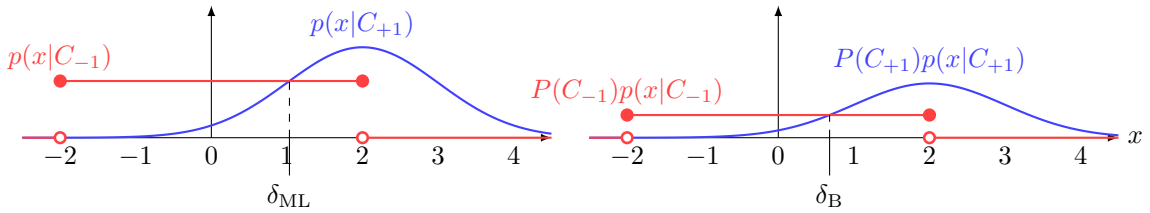
$$g(x) = g_{\text{ML}}(x) = \ln \frac{p(x|C_{+1})}{p(x|C_{-1})} = \begin{cases} -\frac{1}{2}\ln(2\pi) - \frac{1}{2}(x-2)^2 - \ln(1/4), & |x| \le 2 \\ \infty, & |x| > 2 \end{cases} \tag{26}$$

nudges the threshold to $\delta_{\text{ML}} = 2 - \sqrt{\ln(16/2\pi)}$ and result in sligthly higher error probability:

$$P^{\text{ML}}_{\text{error}} = 0.6 \int_{-2}^{\delta_{\text{ML}}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x-2)^2}\, dx + 0.4 \int_{\delta_{\text{ML}}}^{2} \frac{1}{4}\, dx = 0.1967\ldots \tag{27}$$

Figure 4 shows the problem with choosing ML over the Bayesian approach: while both methods try to place a boundary between classes by looking at their likelihoods, ML ignores priors, skewing its perception of data. Had the priors been equal, however, ML would be identical to the optimal classifier. In general, the more variance between classes priors, the worse ML performs.

Figure 4: Class conditional distributions before and after being weighted by their priors.



▶ Question 5

Consider the dataset of 3000 bivariate, labeled samples in `data.csv` file. Firstly, split the dataset into training (70%) and test set (30%).

▷ Item a

Assume the samples labeled with "+1" are drawn by a bivariate gaussian density with parameters μ_{+1}, Σ_{+1}, while the samples labeled with "−1" are drawn by another gaussian density with parameters μ_{-1}, Σ_{-1}. The prior probabilities $P(C_{+1})$ and $P(C_{-1})$ are unknown as well. Estimate the missing parameters (with the training set), present the discriminant functions of the Bayes classifier (MAP criterion) and evaluate its performance (accuracy, precision, recall) over the test set. Comment all your results.

▷ Item b

Calculate the decision boundary and plot it on a graph with the samples of the test set. Comment your results.

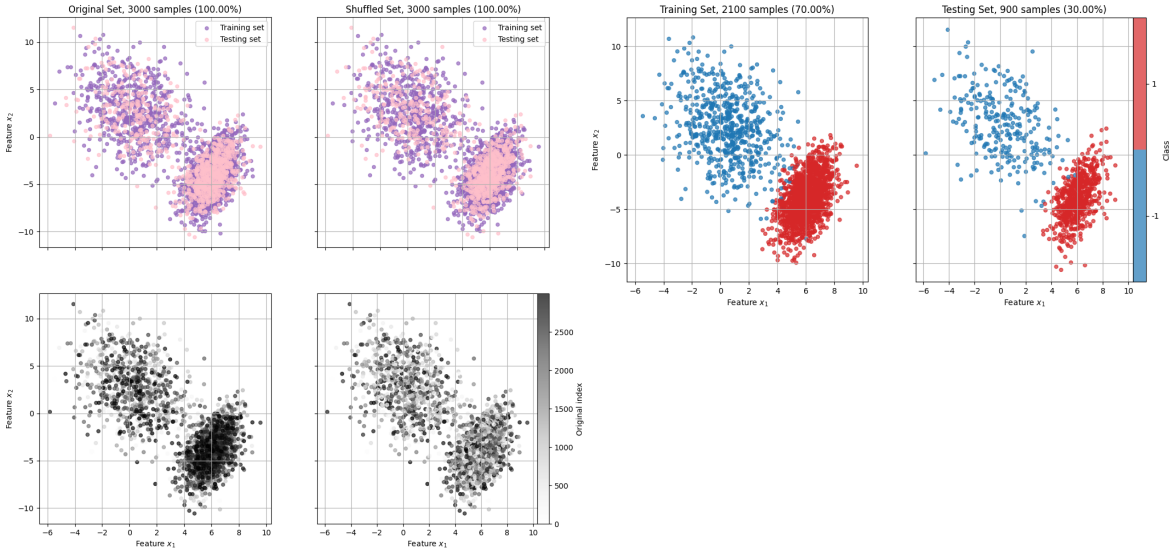
The dataset in `data.csv` consists of three columns, two for the features and one for the label. In the following, the first feature's value is called x_1, and the second, x_2. Since the classifier will be selecting one of two classes for each sample given to it, it is a dichotomizer, and in particular performance will be measured with respect to class +1.

A first step in creating the dichotomizer is readying the data. Pre-processing actions like shuffling samples and normalizing features are common before making the train-test (or train-validation-test) split. Since the features have relatively small ranges and the Bayesian classifier uses data as-is, normalizing is not necessary. As for if shuffling is desirable can be easily determined visually from the data distribution, since the number of features is small. Figure 5a shows how shuffling affects the original data and the 70%-30% train-test split.

Figure 5: Effects of shuffling and resulting data split. Seed: 242104677.

(a) Shuffling effects on data and split.

(b) Train-test split.



The clustering of dark blue points in the original data indicates that label +1 tended to happen more frequently towards the end of the dataset. As such, splitting the data without shuffling would

likely skew the prior of the training portion, biasing the classification process[1]. Therefore, shuffling the samples before the split is adequate. Figure 5b shows the split applied on the shuffled samples to obtain the train and test sets.

For the dichotomizer to work, classes priors and likelihoods must be known. Since they aren't, estimation is required. Given that the likelihoods are gaussian by hypothesis, this means estimating each class' mean and covariance. Let

- $P(C_{+1})$ be the prior of C_{+1};
- $\boldsymbol{\mu}_{+1}$ be the average of C_{+1};
- $\boldsymbol{\Sigma}_{+1}$ be the covariance of C_{+1};
- $P(C_{-1})$ be the prior of C_{-1};
- $\boldsymbol{\mu}_{-1}$ be the average of C_{-1};
- $\boldsymbol{\Sigma}_{-1}$ be the covariance of C_{-1};

then, using the convention that a hat indicates an estimator,

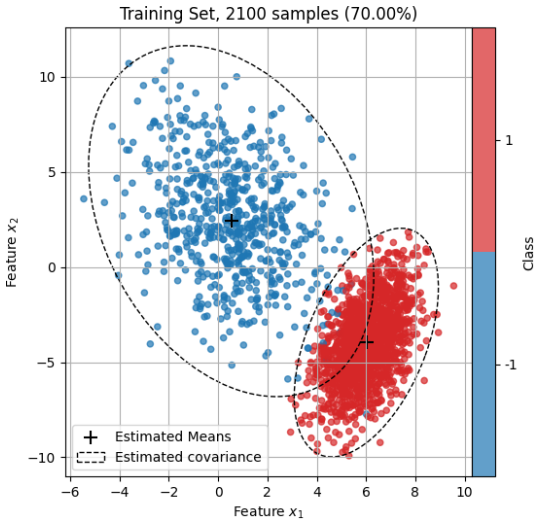
$$\widehat{P}(C_i) = \frac{N_i}{N_{+1} + N_{-1}} \tag{28}$$

$$\widehat{\boldsymbol{\mu}}_i = \frac{1}{N_i} \sum_{n=1}^{N_i} \mathbf{x}_i^n \tag{29}$$

$$\widehat{\boldsymbol{\Sigma}}_i = \frac{1}{N_i - 1} \sum_{n=1}^{N_i} (\mathbf{x}_i^n - \widehat{\boldsymbol{\mu}}_i)(\mathbf{x}_i^n - \widehat{\boldsymbol{\mu}}_i)^\top \tag{30}$$

where i indexes a class, N_i is the number of instances of class i in the training set, and $\mathbf{x}_i^n \in \mathbb{R}^{2\times 1}$ is the n-th sample of class i in the training set. Applying eqs. (28)-(30) on the training data, the following values were obtained. Figure 6 shows what the estimators look like on top of the training data (ellipses encompass three standard deviations).

Figure 6: Average and covariance estimators fit on training data.



With the prior and likelihoods at hand, defining a discriminant is straightforward. Of the common formulations, the is chosen the log-ratio of the likelihood-prior products, since there are only two classes

[1]The discussion on the classifier's results present at the end of this answer shows that using the original data leads to one more misclassification than using the shuffled data.

and they're both gaussian:

$$g(\mathbf{x}) = \log \frac{P(C_{+1}|\mathbf{x})}{P(C_{-1}|\mathbf{x})} \tag{31}$$

$$= \log \frac{p(\mathbf{x}|C_{+1})}{p(\mathbf{x}|C_{-1})} \tag{32}$$

$$= -\frac{1}{2}(|\widehat{\boldsymbol{\Sigma}}_{+1}| - |\widehat{\boldsymbol{\Sigma}}_{-1}|) - \frac{1}{2}(\mathbf{x} - \widehat{\boldsymbol{\mu}}_{+1})^\top \widehat{\boldsymbol{\Sigma}}_{+1}^{-1}(\mathbf{x} - \widehat{\boldsymbol{\mu}}_{+1}) + \frac{1}{2}(\mathbf{x} - \widehat{\boldsymbol{\mu}}_{-1})^\top \widehat{\boldsymbol{\Sigma}}_{-1}^{-1}(\mathbf{x} - \widehat{\boldsymbol{\mu}}_{-1}) + \log \frac{\widehat{P}(C_{+1})}{\widehat{P}(C_{-1})} \tag{33}$$

Class $+1$ is assigned if $g(\mathbf{x}) > 0$, and -1 otherwise. The dichotomizer thus operates as indicated in the flowchart of fig. 7; there, $\mathbf{x}$ is assumed to be a sample from the test set.

Figure 7: Flowchart of the multivariate MAP classifier.

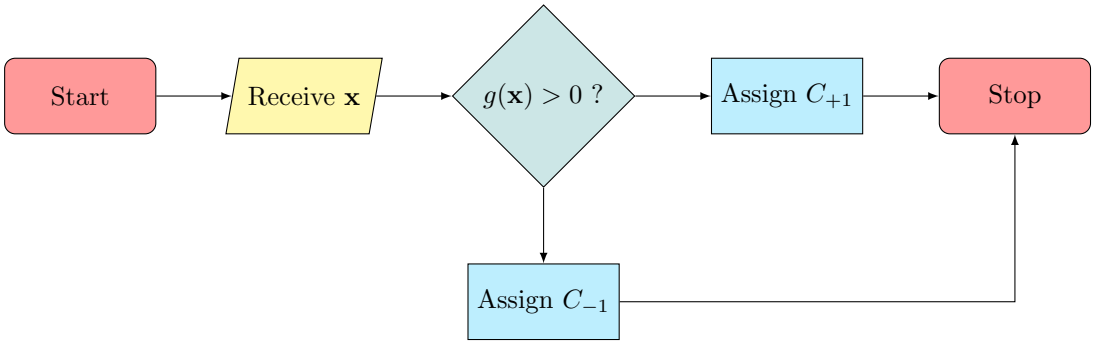


Figure 8 shows the results obtained in testing. The red rectangle highlights the area in feature space where all misclassifications occurred. These are all false positives (i.e., C_{+1} was assigned to sample of class C_{-1}). Remarkably, there were no false negatives, i.e., all C_{-1} assignments were true.

Figure 8: Classifications using test data from shuffled dataset.

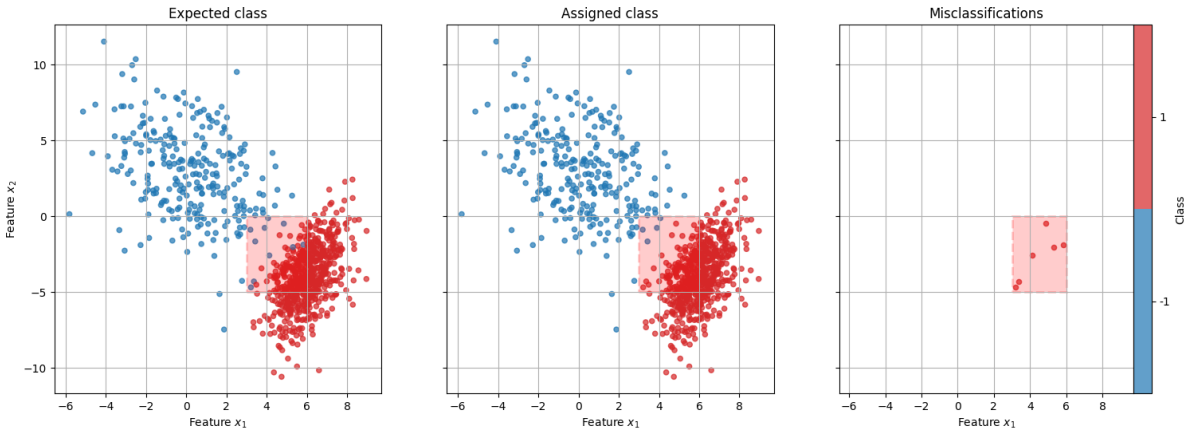


Table 1 exhibits three performance metrics for the classifier, precision, accuracy, and recall, obtained using two different datasets: the original one, and the shuffled one. As expected from the discussion at the beginning of this answer, shuffling improves results (in this case, the precision and the accuracy), albeit only slightly.

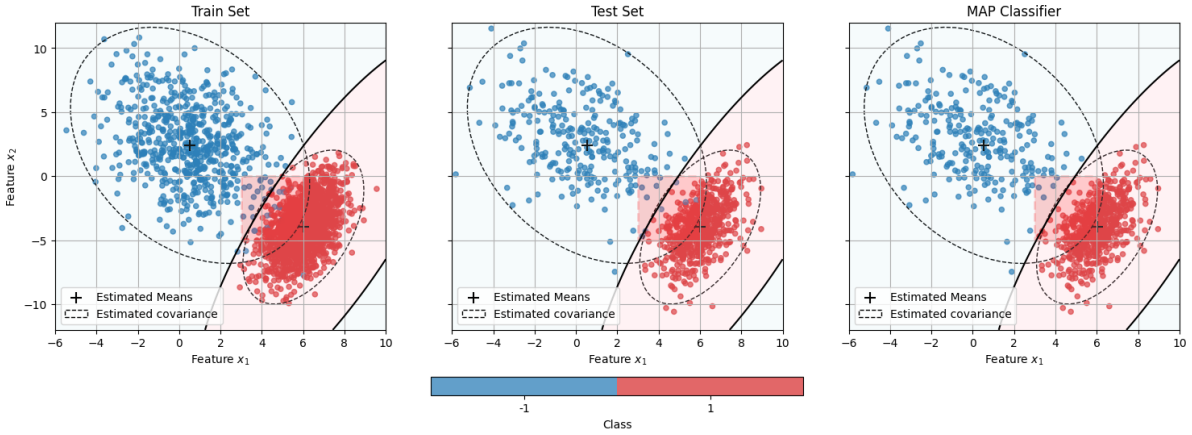
Table 1: Precision, accuracy and recall of the classifier with and without shuffling samples.

Dataset	Precision	Accuracy	Recall
	$\frac{TP+TN}{TP+TN+FP+FN}$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$
Original	99.22%	98.89%	100.00%
Shuffled	99.33%	99.05%	100.00%

TP: True Positives, TN: True Negatives,
FP: False Positives, FN: False Negatives

Looking at the roots of g in fig. 9, the reason behind the misclassifications becomes clear: some points from class -1 had high enough deviation in the direction to cross the classification boundary.

Figure 9: Boundary of the MAP classifier.



Since these points fall well within class' +1 cluster, it would be difficult for any classifier to correctly handle them, particularly classifiers with convex, smooth boundaries. However, given the shapes and orientations of the train distributions, the boundary observed above seems reasonable. Moreover, the classifier showed good performance results, proving itself useful during the test. While whether the classifier would still present good results with further data is unknown, it is fair to say that, should bad results appear, one should consider flaws in the sampling process (noise, distributions different than the ones obsreved before, etc) as much as flaws on the classifier design (e.g., change criterion, change parameter estimators, chage hypothesis on the shape of likelihoods).

► Question 6

Do your own implementation and train a multivariate linear regression model for a given problem (suggestions: Kaggle, UCI Machine Learning Repository). You may use a validation set to train models with different subsets of features and select the best one. Then, use a test set to report and comment the final model results.

Multivariate linear regression (MLR) attempts to predict (model) an outcome $r \in \mathbb{R}$ using a series of features x_i, $1 \leq i \leq d$, in some way correlated with r so that

$$w_0 + w_1 x_1 + \cdots + w_d x_d = r, \tag{34}$$

where w_i, $0 \leq i \leq d$, are entries of a weight vector $\mathbf{w} \in \mathbb{R}^{(d+1)\times 1}$. Given that r often comes from measured real-life phenomena, regression rarely gives a perfect fit of data. Note that eq. (34) can be expressed as the dot product between the weight vector $\mathbf{w}$ and the sample $\mathbf{x} = [1,\ x_1,\ x_2,\ \ldots, x_d]$.

If there are t instances of r, so that $\mathbf{r} = [r_1,\ r_2,\ \ldots, r_t]^\top$, then MLR uses t instances of $\mathbf{x}$, assemblying a matrix $\mathbf{X} = [\mathbf{x}_1,\ \mathbf{x}_2,\ \ldots,\ \mathbf{x}_t]^\top$ so that

$$\mathbf{X}\mathbf{w} = \mathbf{r}. \tag{35}$$

In these terms, it is possible to show that the weight vector that minimizes the square error of the regression is given by

$$\mathbf{w}_{\text{LS}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{r}. \tag{36}$$

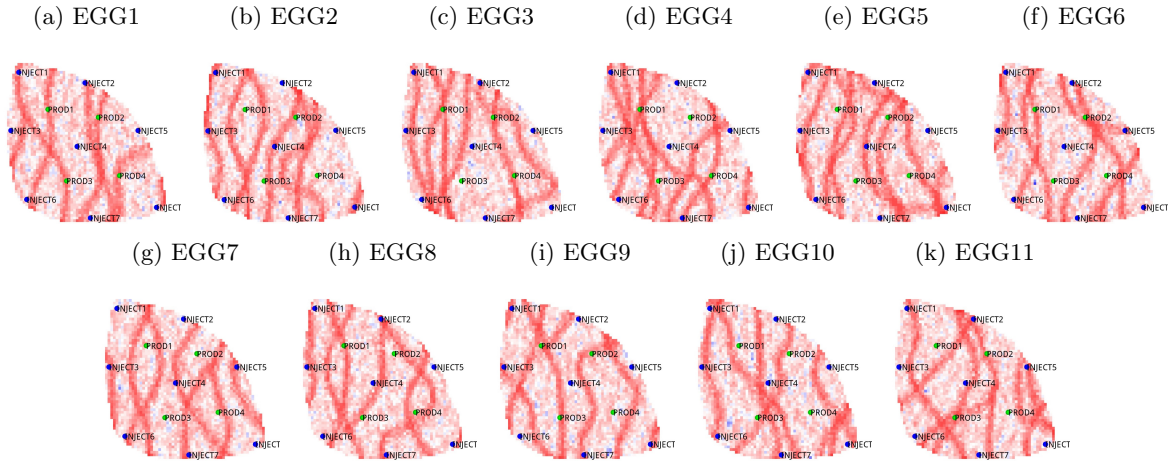
Put simply, MLR finds the best linear combination between a set of features and an outcome. Thus, the weaker the linear relationship, the poorer the regression. As to what may weaken the linearity, there are two main culprits: noise on $\mathbf{r}$, and non-representative data in $\mathbf{X}$.

Validation is a process that attempts to remove bias introduced in $\mathbf{w}_{\text{LS}}$ from using a poor or inadequate set of features $\mathbf{X}$ in the regression. In it, several subsets of features $\mathbf{X}_s$, $1 \leq s \leq S$, are used to generate one weight vector $\mathbf{w}^s_{\text{LS}}$ each through eq. (36); then, through eq. (35), each vector is applied on a *validation set* $\mathbf{X}_V$. The vector with smallest validation error associated is chosen to proceed to testing, i.e., $\mathbf{X}_T \mathbf{w}^*$ is compared to $\mathbf{r}_T$, with $\mathbf{w}^* = \underset{\mathbf{w}^s_{\text{LS}}}{\text{argmin}} \|\mathbf{X}_V \mathbf{w}^s_{\text{LS}} - \mathbf{r}_V\|_2$. Here, $\mathbf{r}_V$ and $\mathbf{r}_T$ are the outcomes observed in the validation and test sets, respectively. If we see the generation of weight vectors using sets $\mathbf{X}_s$ as training the regressor, then validation tries to reduce test error by introducing new data into the process, i.e., data different than that of training.

The application chosen for the implementation of MLR with validation is the prediction of oil production rate in a well given the oil and water production rates (m^3 per day) in the remaining producing wells. The case study considered is the Egg model, a two-fase (oil and water) synthetic reservoir commonly used in prototyping reservoir control strategies. It has over 18,000 active 60×60×7 m cells, 4 producing wells, 8 injector wells, and 100 different realizations.

In the context of reservoir engineering, a realization is a numerical model of a reservoir; differences in the configurations of each realization (e.g., porosity and permeability at each cell) serve to capture the uncertainties of the reference reservoir, and for that reason they are often used in forecasting. See fig. 10 for a view of the realizations used in this implementation (11 of the original 100). The targeted well is labeled "PROD1". Producer wells are indicated in green, and injectors, in blue. Given the arrangement of wells, oil is pushed towards producers as the injectors fill the reservoir with water.

Figure 10: Permeabilities of the Egg model realizations used, log scale. The redder the cell, the greater its permeability.



In further detail, $\mathbf{r}$ will correspond to the oil production rate at a given well in a realization of reference, and features x_1, x_2, etc., will be the oil and water production rates at the remaining wells. Of the $S = 10$ remaining realizations (the original 11 minus the one where $\mathbf{r}$ comes from), one weight vector will be extracted using (36), and then validated. The resulting selection of the validation will then be tested.

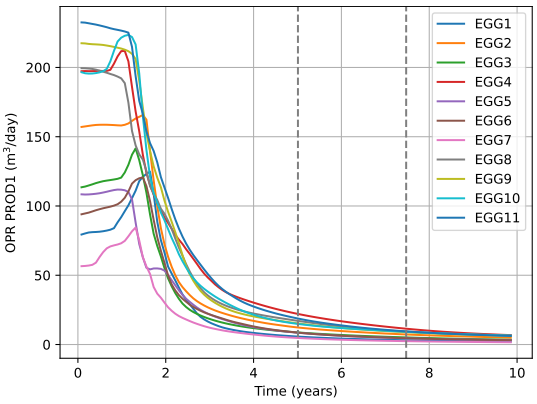
Since $\mathbf{r}$ is a time series, samples can't be shuffled. However, as we wish to evaluate the influence of the other producing wells on PROD1, and weights scale with their associated attributes, features will be normalized (i.e., centered at zero and divided by their standard deviations) and w_0 will be fixed at 0[2].

Figure 11 shows the oil production rate at PROD1 for all 11 realizations used. Times series consist of 120 measurements taken every 30 days, generating nearly 10 years of reservoir life. Training starts at year 0 and ends at year 5 (50% split); validation starts at year 5 and ends at year 7.5 (25% split); testing starts at year 7.5 and covers the remainder of reservoir life (25% split).

Reservoirs have complex dynamics that discourage the use of linear regressors, but MLR's simplicity and ease of interpretability makes it a promising tool for gaining insight on well-on-well interference.

Figure 12 shows the regressions obtained using the validation step considering different realizations as reference (the well, though, is always PROD1). Target realizations are highlighted in blue, and best regressions, in red. Interestingly, realizations with poorer training results occasionally manage to have good or even the best validation results, while good training results do not necessarily guarantee good validation results.

Figure 11: Oil production rate at PROD1 of realizations used.



[2]Mathematically, this is assuming whatever portion of r can't be explained by the features is noise.

Figure 12: Best regressions and weights across different references. Normalized units.