Introduction to Machine Learning (67577)

Exercise 3
Classification

Second Semester, 2023

Contents

1 Submission Instructions

Please make sure to follow the general submission instructions available on the course website. In addition, for the following assignment, submit a single `ex3_ID.tar` file containing:

- An `Answers.pdf` file with the answers for all theoretical and practical questions (include plotted graphs *in* the PDF file).
- The following python files (without any directories): `loss_functions.py`, `perceptron.py`, `linear_discriminant_analysis.py`, `gaussian_naive_bayes.py`, `classifiers_evaluation.py`

The `ex3_ID.tar` file must be submitted in the designated Moodle activity prior to the date specified *in the activity*.

- Late submissions will not be accepted and result in a zero mark.
- Plots included as separate files will be considered as not provided.
- Do not forget to answer the Moodle quiz of this assignment.

2 Theoretical Part

2.1 Hard- & Soft-SVM

Based on Lecture 3 and Recitation 5

1. Prove that following Hard-SVM optimization problem is a Quadratic Programming problem:

$$\underset{(\mathbf{w},b)}{\operatorname{argmin}} ||\mathbf{w}||^2 \quad \text{s.t.} \quad \forall i \; y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1 \tag{1}$$

   That is, find matrices Q and A and vectors $\mathbf{a}$ and $\mathbf{d}$ such that the above problem can be written in the following format

$$\underset{\mathbf{v}\in\mathbb{R}^n}{\operatorname{argmin}} \tfrac{1}{2}\mathbf{v}^\top Q\mathbf{v} + \mathbf{a}^\top \mathbf{v} \quad \text{s.t.} \quad A\mathbf{v} \leq \mathbf{d} \tag{2}$$

   Hint: Observe that $||\mathbf{w}||^2 = \mathbf{w}^\top \mathbf{I}\mathbf{w}$

2. Consider the Soft-SVM optimization problem:

$$\underset{\mathbf{w},\{\xi_i\}}{\operatorname{argmin}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m}\sum_i \xi_i \quad \text{s.t.} \quad \forall i \; y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 - \xi_i \; \wedge \; \xi_i \geq 0 \tag{3}$$

   Denote the hinge-loss function as $\ell^{hinge}(a) := \max\{0, 1-a\}$. Show that the Soft-SVM optimization problem is equivalent to the following unconstraint optimization problem:

$$\underset{\mathbf{w},\{\xi_i\}}{\operatorname{argmin}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m}\sum_i \ell^{hinge}\left(y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\right) \tag{4}$$

2.2 Naive Bayes Classifiers

Based on Lecture 3 and Recitation 6. Let $\mathcal{X}$ be a domain set and $\mathcal{Y} \in [K], K \in \mathbb{N}$ the response set and let us assume there exists a joint probability distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ with $f_{\mathcal{D}}$ the joint probability distribution function.

Recall the Bayes Optimal Classifier which predicts the response maximizing the posterior distribution:

$$\hat{y}^{MAP} := \underset{k\in[K]}{\operatorname{argmax}} f_{Y|X=\mathbf{x}}(k) = \underset{k\in[K]}{\operatorname{argmax}} \frac{f_{X|Y=k}(\mathbf{x})\, f_Y(k)}{f_X(\mathbf{x})} \tag{5}$$

Naive Bayes classifiers are a family of classifiers realizing the Bayes Optimal classifier where we assume that all features are *independent*. That is, for $\mathbf{x} \sim \mathcal{P}$ then $f_{X_i,X_j}(x_i,x_j) = f_{X_i}(x_i)\, f_{X_j}(x_j) \quad \forall i,j$.

3. The ***Gaussian*** **Naive Bayes** classifier assumes a multinomial prior and independent feature-wise Gaussian likelihoods:

$$\begin{aligned} y &\sim \text{Multinomial}(\boldsymbol{\pi}) \\ x_j|y=k &\overset{ind.}{\sim} \mathcal{N}\left(\mu_{kj}, \sigma_{kj}^2\right) \end{aligned} \tag{6}$$

   for $\boldsymbol{\pi}$ a probability vector: $\boldsymbol{\pi} \in [0,1]^K, \sum \pi_j = 1$.
   (a) Suppose $x \in \mathbb{R}$ (i.e each sample has a single feature). Given a trainset $\{(x_i,y_i)\}_{i=1}^m$ fit a Gaussian Naive Bayes classifier solving (5) under assumptions (6). Fitting means finding the expressions for the maximum likelihood estimators.
   (b) Suppose $\mathbf{x} \in \mathbb{R}^d$ (i.e each sample has d feature). Given a trainset $S = \{(\mathbf{x}_i,y_i)\}_{i=1}^m$ fit a Gaussian Naive Bayes classifier solving (5) under assumptions (6). You are encouraged to use the results from (3.a).
4. The ***Poisson*** **Naive Bayes** classifier assumes a multinomial prior and independent feature-wise Poisson likelihoods:

$$\begin{aligned} y &\sim \text{Multinomial}(\boldsymbol{\pi}) \\ x_j|y=k &\overset{ind.}{\sim} \text{Poi}\left(\lambda_{kj}\right) \end{aligned} \tag{7}$$

   for $\boldsymbol{\pi}$ a probability vector: $\boldsymbol{\pi} \in [0,1]^K, \sum \pi_j = 1$.
   (a) Suppose $x \in \mathbb{R}$ (i.e each sample has a single feature). Given a trainset $\{(x_i,y_i)\}_{i=1}^m$ fit a Poisson Naive Bayes classifier solving (5) under assumptions (7).
   (b) Suppose $\mathbf{x} \in \mathbb{R}^d$ (i.e each sample has d feature). Given a trainset $S = \{(\mathbf{x}_i,y_i)\}_{i=1}^m$ fit a Poisson Naive Bayes classifier solving (5) under assumptions (7). You are encouraged to use the results from (4.a).

3 Practical Part

In the following part you will implement and compare different classifiers for different data scenarios. Be sure to have pulled the latest version of the GreenGilad/IML.HUJI repository.

3.1 Perceptron Classifier

Based on Lecture 3 and Recitation 5. Complete the following implementations

- Implement the `misclassification_error` function in the `metrics/loss_functions.py` file as described in the function documentation.
- Implement the Perceptron algorithm in the `learners/classifiers/perceptron.py` file as described in the class documentation. In toy implementation use the misclassification error implemented above.

In the `exercises/classifiers_evaluation.py` file, implement the `run_perceptron` function as described in documentation.

- To retrieve the loss at each iteration **do not** change the previously implemented Perceptron class. Instead **specify a callback function** which receives the object and uses it's `loss` function to calculate the loss over the training set. Store these values in an array to be used for plotting.

1. Fitting and plotting over the `datasets/linearly_seprable.npy` dataset, what can we learn from the plot?
2. Next run the Perceptron algorithm over the `datasets/linearly_inseprable.npy` dataset and plot its loss as a function of the iterations. What is the difference between this plot and to the one in the previous question? How can we explain the difference in terms of the objective and parameter space?

3.2 Bayes Classifiers

Based on Lecture 3 and Recitation 6. Complete the following implementations

- Implement the `accuracy` function in the `metrics/loss_functions.py` file as described in the function documentation.
- Implement the `LDA` classifier in the `learners/classifiers/linear_discriminant_analysis.py` file as described in the class documentation. Use expressions derived in class.
- Implement the `GaussianNaiveBayes` classifier in the `learners/classifiers/gaussian_naive_bayes.py` file as described in the class documentation. Use expressions derived in question 3b of the theoretical part.

Then, implement and answer the following questions:

1. In the `compare_gaussian_classifiers` function, `classifiers_evaluation.py` file, load the `datasets/gaussians1.npy` dataset. Fit both the Gaussian Naive Bayes and LDA algorithms previously implemented. Plot the following:
   - A single figure with two subplots:
     (a) 2D scatter-plot of samples, with marker color indicating Gaussian Naive Bayes *predicted* class and marker shape indicating *true* class.
     (b) 2D scatter-plot of samples, with marker color indicating LDA *predicted* class and marker shape indicating *true* class.
     (c) Provide classifier name and accuracy (over train) in sub-plot title
   - For both subplots add:
     (a) Markers (colored black and shaped as 'X') indicating the center of fitted Gaussians.
     (b) An ellipsis (colored black) centered in Gaussian centers and shape dictated by fitted covariance matrix.
   - Specify dataset name in figure title.

   Explain what can be learned from the plots above regarding the distribution used to sample the data?

2. Repeat the procedure above (while avoiding code repetition) for `datasets/gaussians2.npy`.

What is the difference between the two scenarios? What can be learned regarding the distribution used to sample the data? Which of the two classifiers better matches this dataset and why?

שיר רשקוביץ
209144013

2 Theoretical Part

2.1 Hard- & Soft-SVM

Based on Lecture 3 and Recitation 5

1. Prove that following Hard-SVM optimization problem is a Quadratic Programming problem:

$$\underset{(\mathbf{w},b)}{\operatorname{argmin}} ||\mathbf{w}||^2 \quad \text{s.t.} \quad \forall i \; y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \tag{1}$$

That is, find matrices Q and A and vectors $\mathbf{a}$ and $\mathbf{d}$ such that the above problem can be written in the following format

$$\underset{\mathbf{v}\in\mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\mathbf{v}^\top Q\mathbf{v} + \mathbf{a}^\top \mathbf{v} \quad \text{s.t.} \quad \overset{m\times n}{A}\overset{n\times 1}{\mathbf{v}} \leq \mathbf{d} \tag{2}$$

Hint: Observe that $||\mathbf{w}||^2 = \mathbf{w}^\top \mathbf{I}\mathbf{w}$

1) ראשית, נבחין שהנעלמים שאנחנו רוצים למצוא הם $b\in\mathbb{R}$, $w\in\mathbb{R}^d$. לכן נגדיר $w_0=b$ ונקבל $v=\begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix}\in\mathbb{R}^{d+1}$ הנעלמים בבעיה (כלומר $n=d+1$). כעת נשים לב שב- objective לא נרצה להתייחס ל- $w_0=b$ כשנחפש למזער את $||w||^2$. לכן נגדיר- $Q=2\cdot\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & I_{n-1} & \\ 0 & & \end{pmatrix}\in\mathbb{R}^{n\times n}$ וכן $a=\bar{0}\in\mathbb{R}^n$, ונקבל:

$$\frac{1}{2}v^T Q v + a^t v = \frac{1}{2}(w_0 \cdots w_n)\, 2\cdot\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & I_{n-1} & \\ 0 & & \end{pmatrix}\begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix} + \bar{0}^t v = (w_1 \cdots w_{n-1})\, I_{n-1}\begin{pmatrix} w_1 \\ \vdots \\ w_{n-1} \end{pmatrix} = ||w||^2$$

כנדרש. עבור האילוצים, נשים לב שמתקיים:

$$y_i(\langle w, x_i\rangle + b) \geq 1 \iff -y_i\left(1\cdot w_0 + \sum_{i=1}^{n-1} w_i x_i\right) \leq -1$$

לכן נגדיר: $d=-\bar{1}\in\mathbb{R}^m$ כאשר m זה מספר הדוגמאות בסט האימון, ונגדיר: $A_i=-y_i(1 \; — x_i^t —)$ כאשר A_i זאת השורה ה-i במטריצה $A\in\mathbb{R}^{m\times n}$.

באופן זה נקבל:

$$A\cdot v \leq d \iff \forall i\in[m] \quad A_i\cdot v \leq d_i \iff -y_i(1 \; — x_i^t —)\begin{pmatrix} w_1 \\ \vdots \\ w_{n-1} \end{pmatrix} \leq -1$$

$$\iff -y_i\left(w_0 + \sum_{i=1}^{n-1} x_i w_i\right) \leq -1$$

כנדרש.

2. Consider the Soft-SVM optimization problem:

$$\operatorname*{argmin}_{\mathbf{w},\{\xi_i\}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m}\sum_i \xi_i \quad \text{s.t.} \quad \forall i \; y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 - \xi_i \wedge \xi_i \geq 0 \tag{3}$$

Denote the hinge-loss function as $\ell^{hinge}(a) := \max\{0, 1-a\}$. Show that the Soft-SVM optimization problem is equivalent to the following unconstraint optimization problem:

$$\operatorname*{argmin}_{\mathbf{w},\{\xi_i\}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{m}\sum_i \ell^{hinge}\left(y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\right) \tag{4}$$

(2 עבור $w\in R^d$ כלשהו וסט אימון $S=\{(x_i,y_i)\,|\,x\in R^d, y_i\in R\}$ אם $i\in[m]$ כלשהו מתקיים:

בכדי למזער את $\frac{\lambda}{2}\|w\|^2+\frac{1}{m}\sum_{i=1}^{m}\xi_i$ תחת המגבלות ב-(3) עלינו למזער ככל הניתן את $\frac{\lambda}{2}\|w\|^2$ וכן את $\frac{1}{m}\sum_{i=1}^{m}\xi_i$. כדי למזער את $\frac{1}{m}\sum_{i=1}^{m}\xi_i$ עלינו למזער כל אחד מהגורמים $\xi_1,\dots,\xi_m$ תוך שמירה על המגבלות.

עבור $i\in[m]$ נפריד למקרים-

- אם מתקיים $y_i\langle w,x_i\rangle>1$, אז $\xi_i=0$ היא הבחירה האופטימלית למזעור שמקיימת את המגבלות.
- אם מתקיים $y_i\langle w,x_i\rangle\leq 1 \longleftrightarrow 0\leq 1-y_i\langle w,x_i\rangle$ אז משום שצריך לשמור על- $\xi_i\geq 1-y_i\langle w,x_i\rangle \leftrightarrow y_i\langle w,x_i\rangle\geq 1-\xi_i$ וכן על $\xi_i\geq 0$, אז הבחירה האופטימלית תחת המגבלות היא $\xi_i=1-y_i\langle w,x_i\rangle$.

מכאן ש- $\xi_i=\ell^{hinge}(y_i\langle w,x_i\rangle)$ $\forall i\in[m]$ ולכן מזעור $\frac{1}{m}\sum_i \ell^{hinge}(y_i\langle w,x_i\rangle)$ שקול למזעור $\frac{1}{m}\sum_{i=1}^{m}\xi_i$ תוך שמירה על האילוצים ב-(3).

מהתרגול:

Claim 4.1 Given a training set $S=\{(\mathbf{x}_i,y_i)\}_{i=1}^m$ and hyperplane $(\mathbf{w},b)$, the Soft-SVM optimization problem (12) is equivalent to

$$\operatorname*{argmin}_{\mathbf{w},b}\left(\lambda\,||\mathbf{w}||^2+L_S^{hinge}((\mathbf{w},b))\right)$$

where $L_S^{hinge}((\mathbf{w},b)) := \frac{1}{m}\sum \ell^{hinge}(y_i\langle \mathbf{x}_i,\mathbf{w}\rangle)$

Proof. Given a specific hyperplane $(\mathbf{w},b)$ consider the minimization over $\xi_1,\dots,\xi_m$. Since we defined the auxiliary variables to be nonnegative, the optimal assignment of ξ_i is

$$\xi_i := \begin{cases} 0 & y_i(\langle \mathbf{x}_i,\mathbf{w}\rangle+b)>1 \\ 1-y_i(\langle \mathbf{x}_i,\mathbf{w}\rangle+b) & otherwise \end{cases}$$

Thus $\xi_i=\ell^{hinge}(y_i(\langle \mathbf{x}_i,\mathbf{w}\rangle+b))$ ■

3. The ***Gaussian* Naive Bayes** classifier assumes a multinomial prior and independent feature-wise Gaussian likelihoods:

$$y \sim \text{Multinomial}(\pi)$$
$$x_j | y = k \overset{ind.}{\sim} \mathcal{N}\left(\mu_{kj}, \sigma_{kj}^2\right) \qquad (6)$$

for π a probability vector: $\pi \in [0,1]^K, \sum \pi_j = 1$.

(a) Suppose $x \in \mathbb{R}$ (i.e each sample has a single feature). Given a trainset $\{(x_i, y_i)\}_{i=1}^m$ fit a Gaussian Naive Bayes classifier solving (5) under assumptions (6). Fitting means finding the expressions for the maximum likelihood estimators.

3. a) נרצה למצוא את- $\hat{\mu}_i^{MLE}, \hat{\sigma}_i^{2\,MLE}, \hat{\pi}_i^{MLE}$ $\forall i \in [K]$. לשם כך קודם כל נמצא את פונקציית הליקליהוד-

הדגימות ב"ת ; מהגדרת התניה $f(x|y) = \frac{f(x\cap y)}{f(y)}$

$$L(\theta | X, Y) = f_\theta(X, Y) = f_\theta\left(\{x_i, y_i\}_{i=1}^m\right) = \prod_{i=1}^m f_{X,Y|\theta}(x_i, y_i) = \prod_{i=1}^m f_{X|Y=y_i}(x_i) \cdot f_{Y|\theta}(y_i) =$$

$$= \prod_{i=1}^m N(x_i | \mu_{y_i}, \sigma_{y_i}^2) \cdot Mult(y_i | \pi) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma_{y_i}^2}} \cdot e^{-\frac{(x_i - \mu_{y_i})^2}{2\sigma_{y_i}^2}} \cdot \pi_{y_i}$$

$x \in \mathbb{R}$ חד מימדי

נרצה למקסם את הליקליהוד. מכיוון ש- log פונקציה מונוטונית, נוכל למקסם את ה- log-likelihood במקום:

$$\log(L(\theta|X,Y)) = \log\left(\prod_{i=1}^m N(x_i|\mu_{y_i}, \sigma_{y_i}^2) \cdot Mult(y_i|\pi)\right) = \sum_{i=1}^m \log(N(x_i|\mu_{y_i}, \sigma_{y_i}^2)) + \log(Mult(y_i|\pi))$$

חוקי לוגריתמים

$$= \sum_{i=1}^m \log(\pi_{y_i}) + \log\left(\frac{1}{\sqrt{2\pi\sigma_{y_i}^2}} \cdot e^{-\frac{(x_i-\mu_{y_i})^2}{2\sigma_{y_i}^2}}\right) = \sum_{i=1}^m \log(\pi_{y_i}) + \log\left(\frac{1}{\sqrt{2\pi\sigma_{y_i}^2}}\right) + \log\left(e^{-\frac{(x_i-\mu_{y_i})^2}{2\sigma_{y_i}^2}}\right) =$$

$\frac{1}{\sqrt{2\pi\sigma^2}} = (2\pi\sigma_i^2)^{-\frac{1}{2}}$, $\log x^m = m \log x$, log,exp הופכיות

$$= \sum_{i=1}^m \log(\pi_{y_i}) - \frac{1}{2}\log(2\pi\sigma_{y_i}^2) - \frac{(x_i - \mu_{y_i})^2}{2\sigma_{y_i}^2} = -\frac{1}{2}m\log(2\pi) + \sum_{i=1}^K \left(n_i \log(\pi_i) - \frac{1}{2} n_i \log\sigma_i^2 - \frac{1}{2\sigma_i^2} \sum_{j: y_j = i} (x_j - \mu_i)^2\right)$$

עד כה סכמנו לפי מספר הדגימה בסט האימון.
כעת נסכום לפי מחלקות $\{1, \dots, K\}$. נסמן $\forall i \in [K] \quad n_i := \sum_{j=1}^m \mathbb{1}_{y_j = i}$
בנוסף נשתמש לפשט עם חוקי לוגריתמים

כדי למצוא את המקסימום נגזור לפי הפרמטרים שנרצה למצוא- $\{\mu_i\}_{i=1}^K, \{\sigma_i^2\}_{i=1}^K, \{\pi_i\}_{i=1}^K$ ונשווה לאפס.

- נגזור את האילוץ $\sum_{i=1}^K \pi_i = 1$. כדי להתחשב בו במציאת המקסימום, נשתמש בכופלי לגראנז'. נגדיר $g(\pi) = \sum_{i=1}^K \pi_i - 1$
ובעזרתה נגדיר $L = \log(L(\theta|X,Y)) - \lambda g(\pi)$. כעת נגזור את L לפי הפרמטרים ונשווה ל- 0:

$$\frac{\partial L}{\partial \pi_i} = \frac{\partial}{\partial \pi_i}\left(n_i \log(\pi_i) - \lambda \pi_i\right) = \frac{n_i}{\pi_i} - \lambda = 0 \implies \pi_i = \frac{n_i}{\lambda}$$

אפשר להתעלם מכל הקבוע ; $\log'(x) = \frac{1}{x}$

כעת נציב באילוץ $\pi_i = \frac{n_i}{\lambda}$ כדי למצוא את λ:

(גורם שותף, הצבה, האילוץ)

$$\frac{m}{\lambda} = \frac{1}{\lambda}\sum_{i=1}^{k} n_i = \sum_{i=1}^{k}\frac{n_i}{\lambda} = \sum_{i=1}^{k}\pi_i = 1 \implies m = \lambda$$

לכן: $\hat{\pi}_i^{MLE} = \frac{n_i}{m}$.

$$\frac{\partial \log(L(\theta|x,y))}{\partial \mu_i} = \frac{\partial}{\partial \mu_i}\left(-\frac{1}{2\sigma_i^2}\sum_{j:y_j=i}(x_j-\mu_i)^2\right) = -\frac{1}{2\sigma_i^2}\sum_{j:y_j=i} -2(x_j-\mu_i) = 0 \implies \frac{1}{\sigma_i^2}\sum_{j:y_j=i}(x_j-\mu_i) = 0 \implies \sum_{j:y_j=i} x_j = n_i \cdot \mu_i$$

$$\Downarrow$$

$$\frac{1}{n_i}\sum_{j:y_j=i} x_j = \mu_i$$

לכן $\hat{\mu}_i^{MLE} = \frac{1}{n_i}\sum_{j:y_j=i} x_j$.

$$\frac{\partial \log(L(\theta|x,y))}{\partial \sigma_i^2} = \frac{\partial}{\partial \sigma_i^2}\left(-\frac{1}{2\sigma_i^2}\sum_{j:y_j=i}(x_j-\mu_i)^2 - \frac{n_i}{2}\log \sigma_i^2\right) = \frac{1}{2(\sigma_i^2)^2}\sum_{j:y_j=i}(x_j-\mu_i)^2 - \frac{n_i}{2\sigma_i^2} = 0 \implies$$

$$\left(\frac{1}{x}\right)' = -\frac{1}{x^2}$$

$$\log'(x) = \frac{1}{x}$$

$$\implies \frac{1}{\sigma_i^2}\sum_{j:y_j=i}(x_j-\mu_i)^2 = n_i \implies \sigma_i^2 = \frac{1}{n_i}\sum_{j:y_j=i}(x_j-\mu_i)^2$$

לכן $\hat{\sigma}_i^{2\,MLE} = \frac{1}{n_i}\sum_{j:y_j=i}(x_j-\mu_i)^2$.

(b) Suppose $\mathbf{x} \in \mathbb{R}^d$ (i.e each sample has d feature). Given a trainset $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ fit a Gaussian Naive Bayes classifier solving (5) under assumptions (6). You are encouraged to use the results from (3.a).

— $y \sim \text{Multinomial}(\pi)$
— $x_j | y = k \overset{ind.}{\sim} \mathcal{N}\left(\mu_{kj}, \sigma_{kj}^2\right)$

b) כעת נרצה למצוא את- $\hat{\mu}_{ij}^{MLE}$, $\hat{\sigma}_{ij}^{2\,MLE}$, $\hat{\pi}_i^{MLE}$ $\forall i \in [k], \forall j \in [d]$ עבור $x \in \mathbb{R}^d$ תוך שימוש במה שמצאנו בסעיף א'.
נתחיל מלמצוא את פונקציית הלייקליהוד-

(הבלתי תלות; $f(x|y) = \frac{f(x \cap y)}{f(y)}$ מהגדרת התניה)

$$L(\theta | X, Y) = f_\theta(X, Y) = f_\theta\left(\{x_i, y_i\}_{i=1}^m\right) = \prod_{i=1}^m f_{X,Y|\theta}(x_i, y_i) = \prod_{i=1}^m f_{Y|\theta}(y_i) \cdot f_{X|Y=y_i}(x_i) =$$

$$= \prod_{i=1}^m f(X_1 = x_{i1}, \dots, X_d = x_{id} | Y = y_i) \cdot \text{Mult}(y_i, \pi) = \prod_{i=1}^m \left(\prod_{j=1}^d f(X_j = x_{ij} | Y = y_i) \right) \text{Mult}(y_i, \pi) =$$

(הנחת הבייס הנאיבית)

$$= \prod_{i=1}^m \left(\prod_{j=1}^d N(x_{ij} | \mu_{y_i j}, \sigma^2_{y_i j}) \right) \cdot \text{Mult}(y_i, \pi)$$

נזכור שנרצה למקסם את הליקליהוד. מכיוון ש-log פונקציה מונוטונית נוכל למקסם את ה-log-likelihood במקום:

$$\log(L(\theta | X, Y)) = \log\left(\prod_{i=1}^m \left(\prod_{j=1}^d N(x_{ij} | \mu_{y_i j}, \sigma^2_{y_i j}) \right) \cdot \text{Mult}(y_i, \pi) \right) =$$

(חוקי לוגריתמים)

$$= \sum_{i=1}^m \left(\sum_{j=1}^d \log(N(x_{ij} | \mu_{y_i j}, \sigma^2_{y_i j})) + \log(\text{Mult}(y_i, \pi)) \right) \overset{*}{=} \sum_{i=1}^k n_i \log(\pi_i) + \sum_{m: y_m = i} \sum_{j=1}^d \log(N(x_{mj} | \mu_{ij}, \sigma_{ij})) \quad \circ$$

נעבור לסכימה לפי מחלקות $\{1, \dots, k\}$.
נסמן $\forall i \in [k] \;\; n_i := \sum_{j=1}^m \mathbb{1}_{y_j = i}$

נשים לב שכאשר נגזור את * לפי π_i $\forall i \in [k]$, נקבל את אותה הנגזרת בדיוק כמו בסעיף א' (מכיוון שהביטויים שקשורים ל-π_i זהים בין סעיף א' לסעיף זה, וכן מתקיים אותו אילוץ, לכן נקבל אותו L). לכן:

$$\hat{\pi}_i^{MLE} = \frac{n_i}{m}$$

בנוסף נשים לב שכאשר נגזור את $\circ$ לפי התוחלות או השונויות, עבור $j \in [d]$ ו- $i \in [k]$, הנגזרת תכיל רק ביטויים שקשורים לפיצ'ר ה-j של דגימות שתיוגן i.
כלומר נקבל נגזרת של log-likelihood עבור $x_j \in \mathbb{R}$ שמתפלג גאוסיאני $N(x_{y_i j} | \mu_{y_i j}, \sigma_{y_i j})$.
אך עבור $j \in [d]$ כזה מצאנו כבר בסעיף בסעיף א' את התוחלת והשונות, ולכן אנחנו יודעים שנקבל עבורו לכל $i \in [k]$:

$$\hat{\sigma}^{2\,MLE}_{ij} = \frac{1}{n_i} \sum_{m: y_m = i} (x_{mj} - \mu_{ij})^2$$

$$\hat{\mu}_{ij}^{MLE} = \frac{1}{n_i} \sum_{m: y_m = i} x_{mj}$$

4. The ***Poisson*** **Naive Bayes** classifier assumes a multinomial prior and independent feature-wise Poisson likelihoods:

$$\begin{aligned} y &\sim \text{Multinomial}(\pi) \\ x_j|y=k &\overset{ind.}{\sim} \text{Poi}(\lambda_{kj}) \end{aligned} \quad (7)$$

for π a probability vector: $\pi \in [0,1]^K, \sum \pi_j = 1$.

(a) Suppose $x \in \mathbb{R}$ (i.e each sample has a single feature). Given a trainset $\{(x_i, y_i)\}_{i=1}^m$ fit a Poisson Naive Bayes classifier solving (5) under assumptions (7).

(4

(a) נרצה למצוא את- $\hat{\lambda}_i^{MLE}, \hat{\pi}_i^{MLE}$ $\forall i \in [K]$. נמצא את פונקציית הלייקליהוד-

הבלתי תלות ; מהגדרת התניה $f(x|y) = \frac{f(x \cap y)}{f(y)}$

$$L(\theta|X,Y) = f_\theta(X,Y) = f_\theta(\{x_i,y_i\}_{i=1}^m) = \prod_{i=1}^m f_{X,Y|\theta}(x_i,y_i) = \prod_{i=1}^m f_{Y|\theta}(y_i) \cdot f_{X|Y=y_i}(x_i) =$$

$$\prod_{i=1}^m \text{Mult}(y_i|\pi) \cdot \text{Pois}(x_i|\lambda_{y_i}) = \prod_{i=1}^m \pi_{y_i} \cdot \frac{e^{-\lambda_{y_i}} \cdot \lambda_{y_i}^{x_i}}{(x_i)!}$$

$y \sim \text{Mult}(\pi)$

מכיוון ש- $x \in \mathbb{R}$ ההתפלגות חד-מימדית: $X|Y=y_i \sim \text{Poi}(\lambda_{y_i})$

תזכורת: $X \sim \text{Poi}(\lambda) \Rightarrow P(X=k) = \frac{e^{-\lambda} \cdot \lambda^k}{k!}$

נזכור שנרצה למקסם את הלייקליהוד. מכיוון ש- log פונקציה מונוטונית, נוכל למקסם את ה- log-likelihood במקום:

$$\log\left(\prod_{i=1}^m \pi_{y_i} \cdot \frac{e^{-\lambda_{y_i}} \cdot \lambda_{y_i}^{x_i}}{(x_i)!}\right) = \sum_{i=1}^m \log(\pi_{y_i}) - \lambda_{y_i} + x_i \log(\lambda_{y_i}) - \log((x_i)!) =$$

חוקי לוגריתמים, log,exp הופכיות

כעת נסכום לפי מחלקות $\{1,\dots,K\}$.

נסמן $\forall j \in [K]$ $n_j := \sum_{i=1}^m \mathbb{1}_{y_i=j}$

$$\sum_{j=1}^K n_j \log(\pi_j) - n_j \lambda_j + \log(\lambda_j) \sum_{i: y_i=j} x_i - \sum_{i=1}^m \log(x_i)!$$

כדי לצמצם תוך התחשבות באילוץ $\sum_{i=1}^K \pi_i = 1$ נעזר בדומה לשאלה 3 ב- $L = \log(L(\theta|X,Y)) - \lambda g(\pi)$.

כך ש- $g(\pi) = \sum_{i=1}^K \pi_i - 1$ ונגזור את L לפי הפרמטרים שאנו מחפשים שימקסמו ונשווה ל-0:

$$\frac{\partial L}{\partial \pi_j} = \frac{\partial}{\partial \pi_j}\left(n_j \log(\pi_j) - \lambda \pi_j\right) = \frac{n_j}{\pi_j} - \lambda = 0 \Rightarrow \pi_j = \frac{n_j}{\lambda} \Rightarrow \frac{m}{\lambda} = \frac{1}{\lambda}\sum_{j=1}^K n_j = \sum_{j=1}^K \pi_j = 1 \Rightarrow \lambda = m$$

לכן $\hat{\pi}_i^{MLE} = \frac{n_i}{m}$ כמו שקיבלנו בשאלה 3.

$$\frac{\partial L}{\partial \lambda_j} = \frac{\partial}{\partial \lambda_j}\left(-n_j \lambda_j + \sum_{i: y_i=j} \left(x_i \log(\lambda_j)\right)\right) = -n_j + \sum_{i: y_i=j} \frac{x_i}{\lambda_j} = 0 \Rightarrow n_j = \sum_{i: y_i=j} \frac{x_i}{\lambda_j} \Rightarrow \lambda_j = \frac{1}{n_j}\sum_{i: y_i=j} x_i$$

לכן $\hat{\lambda}_j^{MLE} = \frac{1}{n_j}\sum_{i: y_i=j} x_i$ (נזכר שעבור התפלגות פואסונית λ = תוחלת ההתפלגות ולכן זהו משערך הגיוני אינטואיטיבית).

(b) Suppose $\mathbf{x} \in \mathbb{R}^d$ (i.e each sample has d feature). Given a trainset $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ fit a Poisson Naive Bayes classifier solving (5) under assumptions (7). You are encouraged to use the results from (4.a).

(ב) נרצה למצוא את- $\hat{\lambda}_{ij}^{MLE}$, $\hat{\pi}_i^{MLE}$ $\forall i \in [K], \forall j \in [d]$.

(הבלתי תלות; מהגדרת התניה $f(x|y) = \frac{f(x \cap y)}{f(y)}$)

$$L(\theta | X,Y) = f_\theta(X,Y) = f_\theta(\{x_i, y_i\}_{i=1}^m) \overset{\downarrow}{=} \prod_{i=1}^m f_{X,Y|\theta}(x_i, y_i) \overset{\downarrow}{=} \prod_{i=1}^m f_{Y|\theta}(y_i) \cdot f_{X|Y=y_i}(x_i) =$$

$$= \prod_{i=1}^m \pi_{y_i} \cdot \prod_{j=1}^d f(x_j = x_{ij} | Y = y_i) = \prod_{i=1}^m \pi_{y_i} \cdot \prod_{j=1}^d \frac{e^{-\lambda_{y_i j}} \lambda^{x_{ij}}}{(x_j)!}$$

$y \sim Mult(\pi)$; הפיצ'רים של x בת"ל ; $x_j | Y = y_i \sim Pois(\lambda_{y_i j})$

נזכור שנרצה למקסם את הליקליהוד. מכיוון ש-log פונקציה מונוטונית, נוכל למקסם את ה- log-likelihood במקום:

$$\log(L(\theta|X,Y)) = \log\left(\prod_{i=1}^m \pi_{y_i} \cdot \prod_{j=1}^d \frac{e^{-\lambda_{y_i j}} \lambda_{y_i j}^{x_{ij}}}{(x_{ij})!}\right) \underset{\text{חוקי לוגריתמים}}{=} \sum_{i=1}^m \log(\pi_{y_i}) + \log\left(\prod_{j=1}^d \frac{e^{-\lambda_{y_i j}} \lambda^{x_{ij}}}{(x_{ij})!}\right) =$$

$$= \sum_{i=1}^m \log(\pi_{y_i}) + \sum_{j=1}^d \left(-\lambda_{y_i j} + x_{ij} \log(\lambda_{y_i j}) - \log(x_{ij}!)\right) = \sum_{i=1}^K n_i \log(\pi_i) + \sum_{j=1}^d -n_i \lambda_{ij} + \log(\lambda_{ij}) \sum_{t: y_m = i} x_{tj} - \sum_{i=1}^K \sum_{j=1}^d \log(x_{ij}!)$$

כעת, מהפיתוח שהצגנו אליו, ובאופן דומה לטיעון ב- (ג) ב' נקבל:

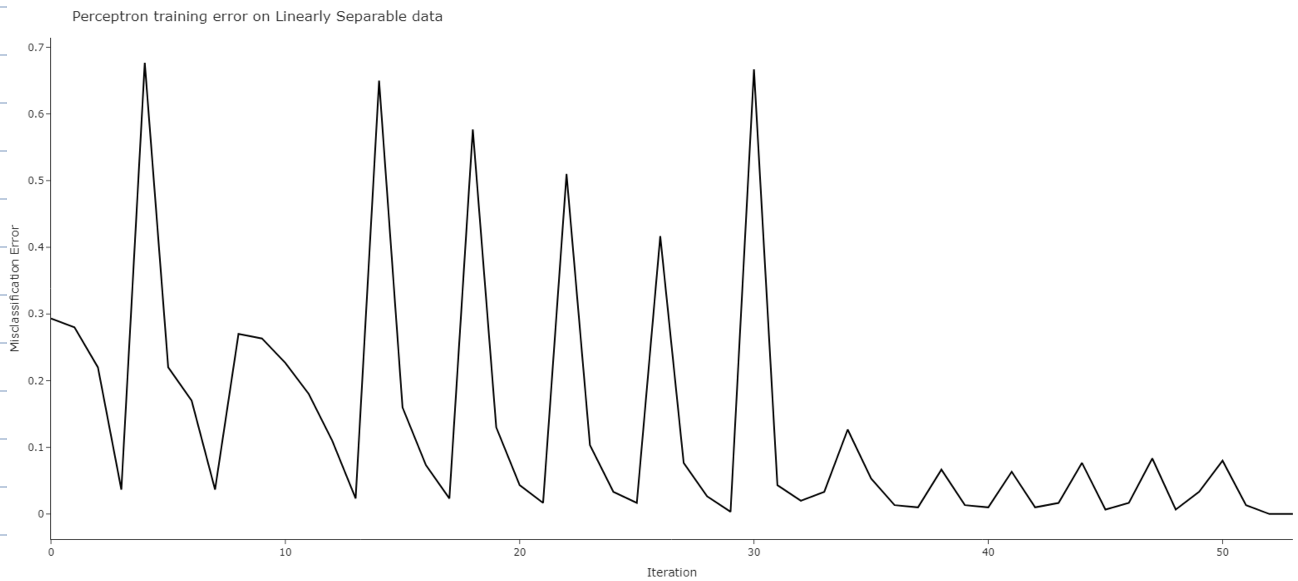
- $\hat{\pi}_i^{MLE} = \frac{n_i}{m}$ (המחוברים בביטוי מסעיף א' שקשורים ב-π_i זהים לסעיף א', מתקיים אותו אילוץ, על כן אין השפעה על תוצאות הגזירה של הביטוי לפני)

- $\hat{\lambda}_{ij}^{MLE} = \frac{1}{n_i} \sum_{m: y_m = i} x_{mj}$ (עבור $j \in [d]$ הגזירה לכל $i \in [K]$ תהיה בדיוק כמו בסעיף א', כי נגזרים רק מחוברים שקשורים לפיצ'ר ה-j. לכן מגיעים בדיוק למקרה בסעיף א' עבור כל פיצ'ר j).

3.1 Perceptron Classifier

1. Fitting and plotting over the `datasets/linearly_seprable.npy` dataset, what can we learn from the plot?
2. Next run the Perceptron algorithm over the `datasets/linearly_inseprable.npy` dataset and plot its loss as a function of the iterations. What is the difference between this plot and to the one in the previous question? How can we explain the difference in terms of the objective and parameter space?

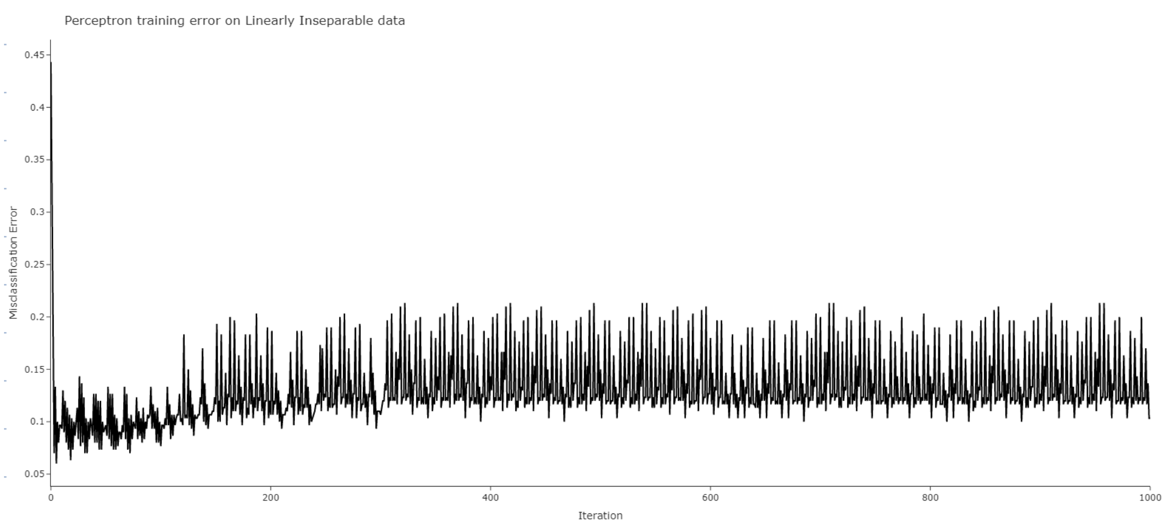
(1



מהגרף ניתן לראות שכאשר האלגוריתם הופעל על מידע שניתן להפרדה לינארית, האלגוריתם עצר לאחר מספר סופי של צעדים (פחות מ-60 איטרציות), לאחר תיקונים שיצרו w שמסווג נכונה את כל הדוגמאות, ואז שגיאת הסיווג שהתקבלה הייתה אפס.

בנוסף, אציין שניתן ללמוד מהגרף, מאחר ולא קיבלנו גרף מונוטוני יורד, שצעדי עדכון מסוימים של הפרספטרון דווקא מגדילים את השגיאה, וזאת מכיוון שתיקון יחיד לכיוון דוגמאת אחת שלא סווגה נכון לא מבטיח ישר שכעת הדוגמה תסווג נכון, ולפעמים נדרשים מספר צעדי עדכון. דבר נוסף שעלול לקרות, זה שתיקון w לעבר דוגמה i הופך את הסיווג של דוגמה אחרת ללא נכון. אולם, משום שאנחנו במקרה הריאליזבילי לבסוף נמצא w שמתאים כך שהישר המאונך לו מפריד באופן הרצוי את דוגמאות האימון.

(2



מגרף זה, שמציג את תוצאות ההרצה עבור מידע שלא ניתן להפרדה לינארית, ניתן ללמוד שהאלגוריתם כל פעם מנסה לתקן את w ומעדכנו, אך ממשיך לסווג לא נכון חלק מהדוגמאות בכל פעם. עד שלבסוף לאחר 1,000 איטרציות שבו האלגוריתם נכשל הוא נעצר באופן מלאכותי (כשהגדרנו ל- max_iter=1,000).

ההבדלים בין הגרף ב-(1) וב-(2), הם שבראשון יש לאחר מספר סופי של צעדים מציאה של w שמסווג נכון את כל הדוגמאות, לעומת הגרף ב-(2) שלא מצליח את המשימה גם אחרי ניסיונות חוזרים ונשנים.

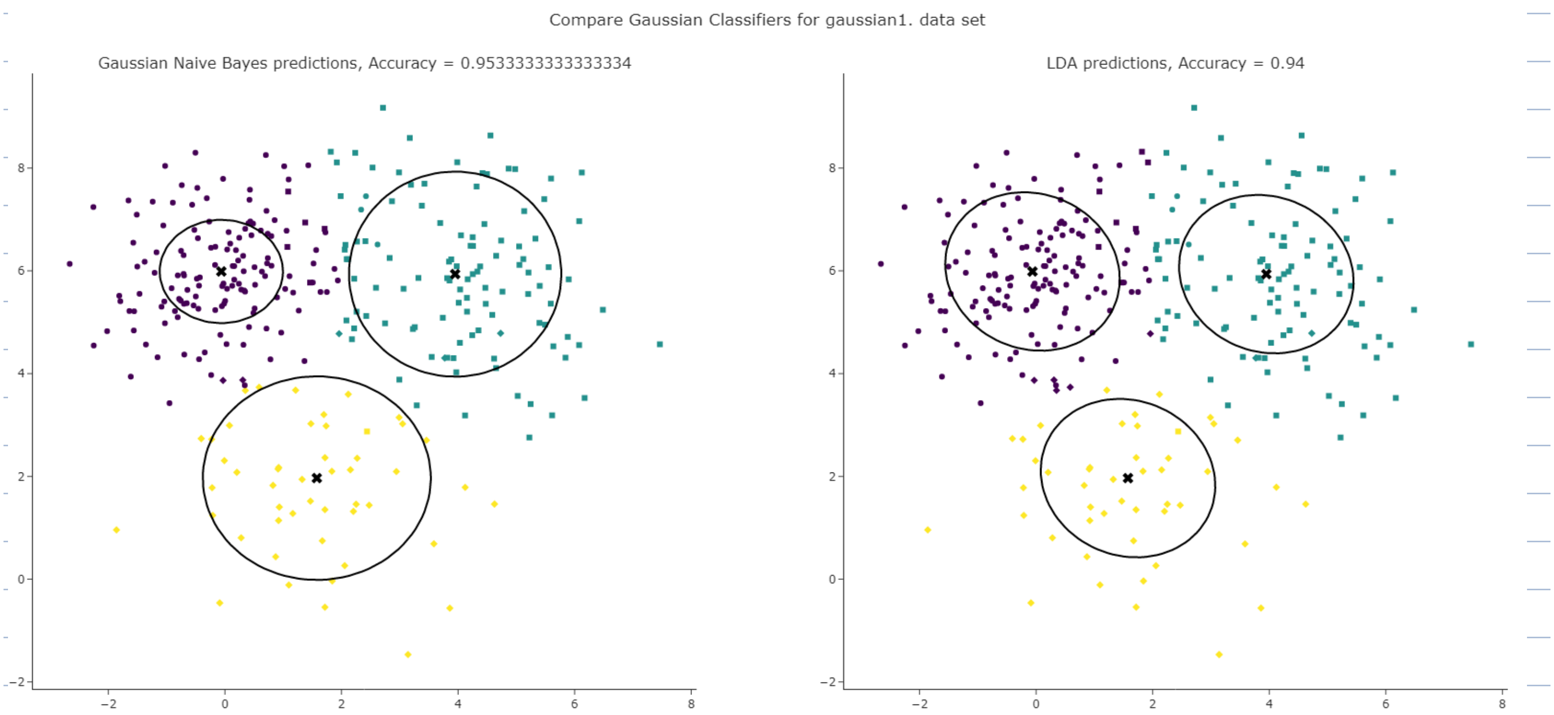
הסיבה להבדל נעוצה בעובדה שאלגוריתם הפרספטרון מתכנס לפתרון נכון רק במקרה הריאליזבילי, ובמקרה הלא ריאליזבילי מאחר שלא קיים על-מישור מפריד (w,b) כך ש- $y_i\langle x_i,w\rangle>0$ לכל $i\in[m]$ אז לא משנה כמה עדכונים ותיקונים נעשה ל-w, עדיין תמיד יהיו דוגמאות שלא מתויגות נכון, ולעולם לא "נמצא" w שיסווג נכון את כל הדוגמאות.

זה מתקשר לכך שמה שאלגוריתם הפרספטרון מנסה לפתור היא הבעיה-

$$\begin{aligned} &\text{minimize} && 0 \\ &\text{subject to} && y_i \cdot \langle \mathbf{x}_i, \mathbf{w} \rangle \geq 0 \quad i = 1, \ldots, m \end{aligned}$$

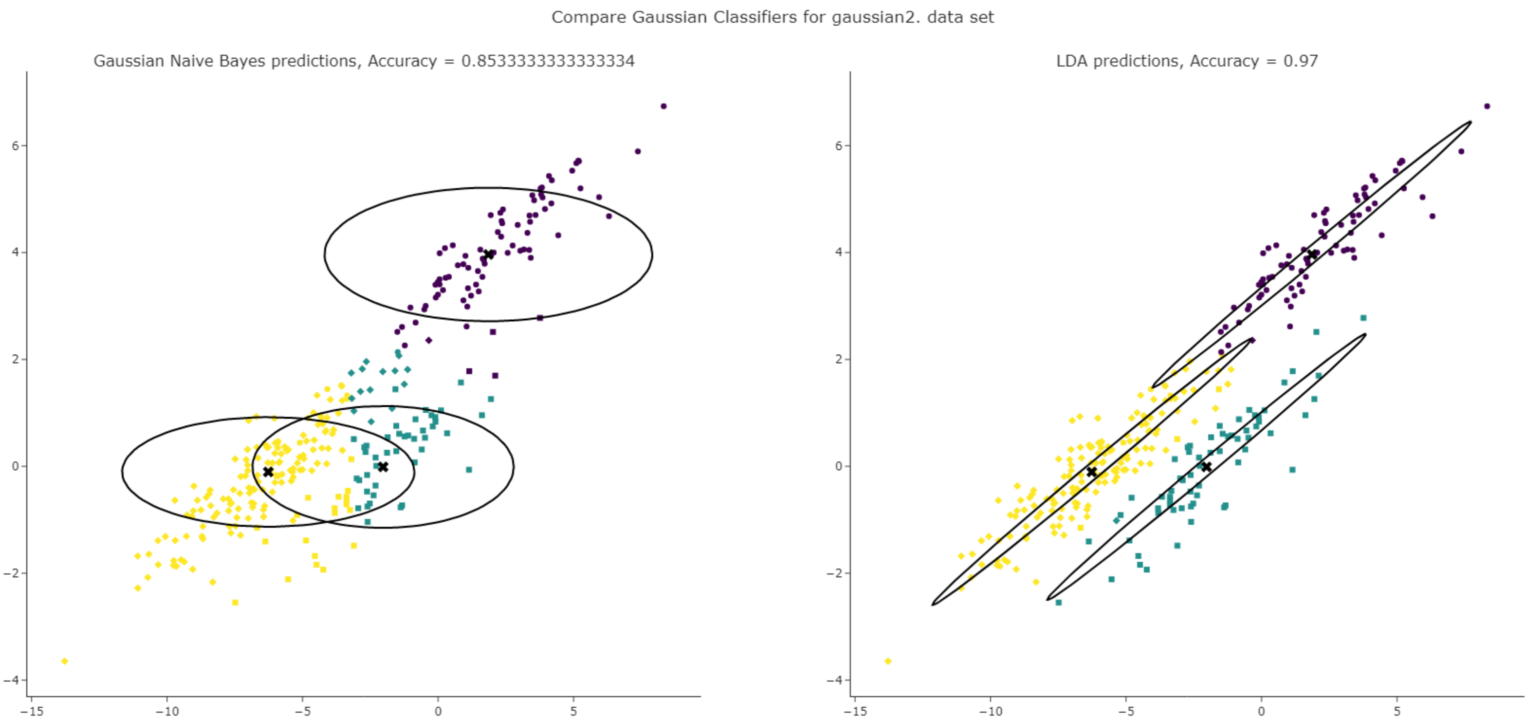
פה ה- objective הוא "ריק", ויש אילוצים מחייבים שרק אם הם מתקיימים "נמצא" פתרון (לעומת soft SVM למשל שם יש סבלנות לטעויות ומתקבל פתרון כלשהו אופטימלי עבור המקרה הלא-ריאליזבילי).

ו)



Explain what can be learned from the plots above regarding the distribution used to sample the data?

ניתן ללמוד שהדגימות בדאטה שהתקבל הגיעו מ-3 התפלגויות גאוסיאניות דו-מימדיות שונות, עם תוחלת שונה (ניתן לראות לפי הנקודות השונות של ה-X-ים במרחב), ועם covariance שנראה יחסית דומה - פיזור דומה בציר ה-x ובציר ה-y. ונראה ש- $cov(x,y)=cov(y,x)=0$ *). נשים לב * שלא נראה שקיימת תלות בין הפיצ'רים של ההתפלגויות הגאוסיאניות (נראה שאין קשר מיוחד בין ערכי ה-x וה-y: לפעמים כש-x גדול אז y גדול ולעיתים הוא דווקא קטן, הפיזור נראה רנדומלי, כלומר נראה ש- $cov(x,y)=0$ עבור כל אחת מההתפלגויות).
נבחין ששני המסווגים סיווגו באופן מוצלח יחסית, ובערך באותה רמת הצלחה (עם דיוק גבוה, כך שבערך 95% מהדגימות סווגו נכון). נציין שזוהי תוצאה הגיונית, מכיוון שההנחה של LDA היא שוויון Covariance בין ההתפלגויות הגאוסיאניות השונות מהן הדגימות מגיעות, וההנחה של Gaussian Naive Bayes היא שהפיצ'רים ב"ת. מאחר ונראה שהדאטה סט הנ"ל מקיים את ההנחות הללו - מובן מדוע קיבלנו ששני המסווגים ביצעו היטב את ההפרדה.



What is the difference between the two scenarios? What can be learned regarding the distribution used to sample the data? Which of the two classifiers better matches this dataset and why?

- נבחין שהפעם המסווג הנאיבי השיג דיוק נמוך יותר, וסיווג פחות טוב את הדגימות, בעוד שה-LDA הצליח גם הפעם לסווג כמעט את כל הדגימות - 97%.
- נראה שה-Covariance של שלושת ההתפלגויות מהן הגיעו הדגימות דומה (שונות יחסית שווה גם בציר ה-x וגם בציר ה-y, $cov(x,y) = cov(y,x) > 0$).
- נראה שהדגימות מגיעות מ-3 התפלגויות גאוסיאניות דו-מימדיות (לפי הגרפים), אך הפעם נראה שקיימת תלות לינארית בין הפיצ'רים של כל אחת מההתפלגויות (נראה שהנקודות "יושבות" על פונקציה מהצורה $y = mx + b$ עבור $m, b \in R$ כלשהם). כלומר, $cov(x,y) \neq 0$. אם כן, הדאטה-סט מפר את ההנחה של Gaussian Naive Bayes, ועל כן הגיוני שהוא השיג דיוק מועט יותר מה-LDA, שהניח - שוויון covariance של המחלקות.
על כן מסווג ה-LDA יותר מתאים לדאטה-סט הנ"ל (מאחר והוא עונה על ההנחות שלו בעוד שהוא לא עונה על ההנחות של GNB).