Université de Genève

–

Sciences Informatiques

13X005 Intelligence Artificielle

Projet – Regression Logistique / Naive Bayes

Repository Github: 13X005-AI-Project

Gregory Sedykh, Leandre Catogni, Noah Peterschmitt, Noah Munz

Janvier 2024

Contents

Projet – Regression Logistique / Naive Bayes

1 – Introduction & Rappels théoriques

Dans ce document, nous approfondirons des techniques de regression logistique et "Naive Bayes" comme outils d'apprentissage superivisés.

Dans le cadre de l'intelligence artificielle et de l'apprentissage supervisé, la compréhension et la classification précises des données revêtent une importance capitale. Parmi les diverses méthodologies existantes, la Régression Logistique et "Naive Bayes" se distinguent par leur efficacité et leur applicabilité dans de nombreux contextes. Ce document se propose d'étudier ces deux techniques, en mettant l'accent sur leur mise en œuvre pratique, et leur efficacité comparative dans divers scénarios.

1.1 – Régression Logistique

En statistiques, la régression logistique, s'inscrit dans le cadre des modèles de régression pour les variables binaires.

Ce type de modèle vise à expliquer de manière optimale une variable binaire, qui représente la présence ou l'absence d'une caractéristique spécifique, à l'aide d'un ensemble conséquent de données réelles et d'un modèle mathématique.

Autrement dit, il s'agit de relier une variable aléatoire de Bernoulli, généralement notée y, aussi appelé "label" à un vecteur constitué de plusieurs variables aléatoires, $(x_1, \ldots, x_K)$, aussi appelés "features". [4].

La régression logistique s'appuie sur un classifeur linéaire [2] i.e. un classifieur dont la sortie (pour un vecteur de feature $x \in \mathbb{R}^n$) est donnée par:

$$g(x) = f(\langle w, x \rangle + b)$$

où $w \in \mathbb{R}^n$ est le vecteur de poids, $b \in \mathbb{R}$ le biais et $\langle ., . \rangle$ le produit scalair usuel. f est une fonction dite de seuillage qui va séparer nos résultats. Un choix commun pour f est la sigmoide ou la fonction signe [2].

Par exemple, dans notre cas, on suppose le modèle suivant:

$$y_i \sim Bernoulli(p_i), \quad p_i = \sigma(\langle w, x_i \rangle + b), \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

où x_i représente un vecteur (ligne) de K valeurs pour les K features (aussi appelé un *sample*), et y_i la variable aléatoire qui représente le label qui leur est associé.

1.2 – Naive Bayes

"Naive Bayes" se présente comme une méthode de classification probabiliste basée sur le théorème de Bayes, caractérisée par l'adoption d'une hypothèse d'indépendance forte entre les features (attributs), qualifiée de "naïve".
Plus simplement, le classifieur est classifié de "naïf" car il part du principe que chaque feature (attribut) est indépendante des autres et a un poid égal quant à la probabilité qu'un point appartienne à une classe.

Ce model est dit génératif contrairement à la regression logistique étant considéré comme "méthode discriminante" [2] et consiste à modéliser les probabilités conditionnelles $P(X|classe)$ pour chaque classe y et vecteur de features X afin de trouver celle qui maximise cette probabilité.

En d'autres termes, le problème revient à trouver, pour des attributs $X_1, \ldots, X_k$, la classe $\tilde{y}$ telle que:

$$\tilde{y} = \arg\max_{Y \in \mathcal{Y}} \left[P(Y) \prod_{k=1}^{K} P(X_k|Y) \right]$$

Citation Test: [1]

2 – Méthodologie

2.0 – Choix du dataset & outils utilisés

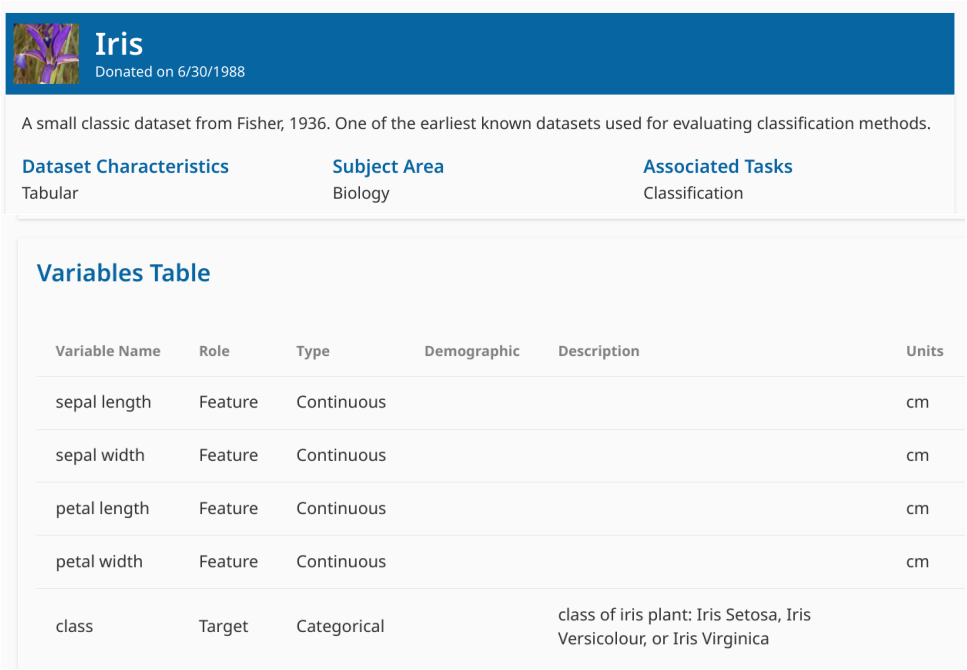
Pour la suite de ce projet les outils suivants ont été utilisés dans chaque parties:

- python
- numpy
- sklearn
- matplotlib
- ucmilrepo

Le package `ucmilrepo` a été utilisé pour charger les données de notre dataset depuis la base de donnée du UC Irvine Machine Learning Repository.

Le dataset que nous avons choisi est le fameux dataset “Iris” [3], un des plus anciens et connus dataset de classification. Il contient 150 observations de 3 espèces différentes d’iris (Iris setosa, Iris virginica et Iris versicolor) avec 4 features (longueur et largeur des sépales et pétales).

Voici un aperçu des points-clés du dataset:



Iris
Donated on 6/30/1988

A small classic dataset from Fisher, 1936. One of the earliest known datasets used for evaluating classification methods.

Dataset Characteristics	Subject Area	Associated Tasks
Tabular	Biology	Classification

Variables Table

Variable Name	Role	Type	Demographic	Description	Units
sepal length	Feature	Continuous			cm
sepal width	Feature	Continuous			cm
petal length	Feature	Continuous			cm
petal width	Feature	Continuous			cm
class	Target	Categorical		class of iris plant: Iris Setosa, Iris Versicolour, or Iris Virginica	

Figure 1: Iris descriptive table

Le label que nous allons prédire sera donc *class*, i.e. l’espèce de l’iris.

2.1 – Gradient Descent

Dans cette section, une implémentation de la "descente en gradient" a été réalisée. La fonction a la signature suivante

```
1  def gradient_descent(df, params: NDArray, alpha: float, num_iters: int) -> NDArray:
```

Elle calcule de manière itérative le(s) paramètre(s) `params` qui minimisent la fonction dont `df` est le gradient avec un "taux de convergence" `alpha`.

La fonction a été testé avec la fonction `scipy.optimize.fmin` [5] de la librairie `scipy` sur la fonction suivante:

$$f(x) = x \cdot \cos(\pi(x+1))$$

avec différents $x_0 \in \{-\pi, 0, \pi\}$ (valeur initiale de `params`, i.e. `NDArray` avec D=0).

Les minimas locaux trouvés par les deux fonctions sont les suivants:

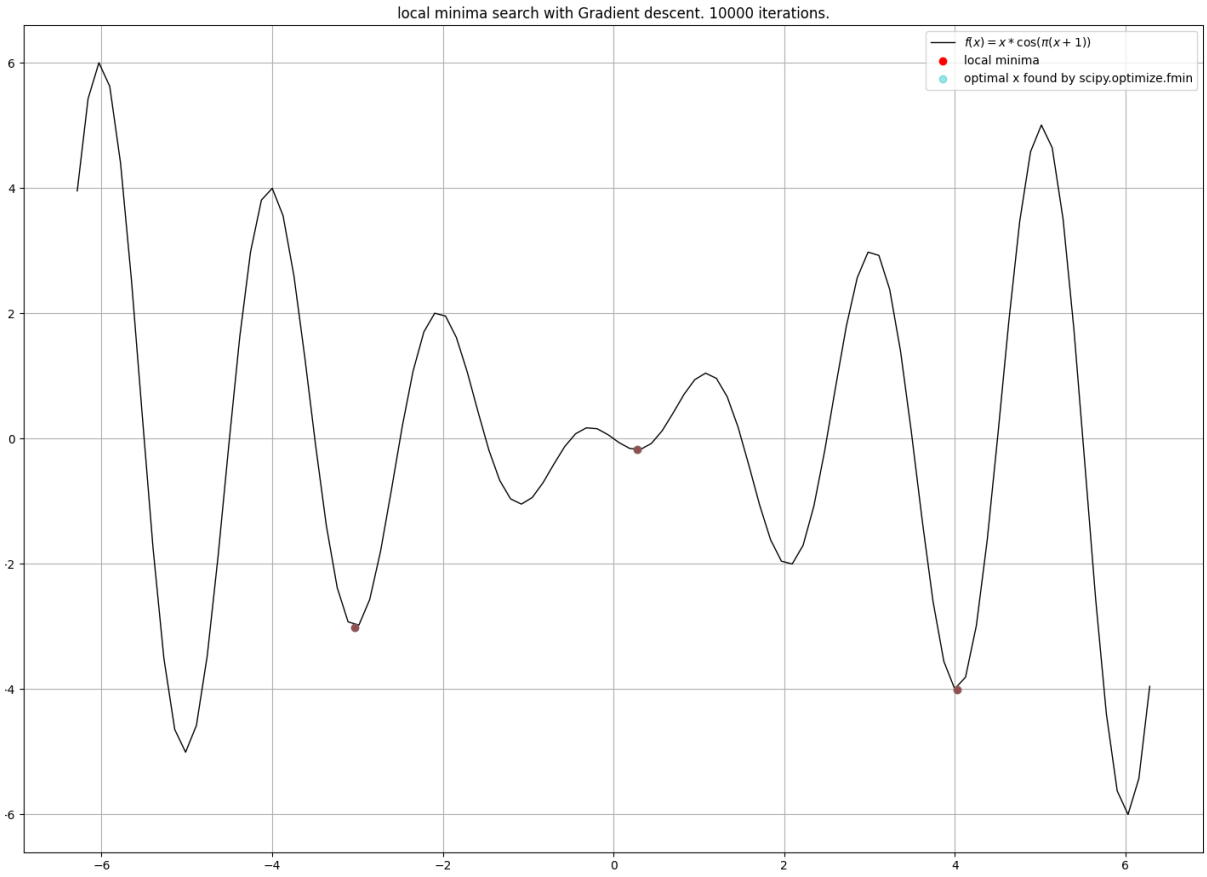


Figure 2: minimas locaux_gradient descent

Ce résultat illustre bien 2 choses: la première est que l'implémentation de la descente en gradient fonctionne correctement puisque pour chaque points trouvé par notre fonction est confondu avec celui trouvé par la fonction de scipy (c'est ce qui donne cette teinte "grise"). La deuxième est que la "qualité" du minima local (i.e. la distance avec le minima globale) dépend fortement de la valeur initiale et ce pour les deux fonctions.

2.2 – Régression Logistique

2.2.1 – Fonction de coût pour la régression logistique

Afin d'entraîner les paramètres de la régression logistique, il faut pouvoir comparer les résultats obtenus par la régression avec les résultats attendus.

Pour cela, on pourrait penser utiliser quelque chose comme la `Mean Squared Error (MSE)`, qui est une moyenne du carré de la différence entre le résultat obtenu par la régression (donné par z) et la valeur estimée y.

La MSE nous donne une estimation de l'erreur moyenne faite entre la fonction approximative f et la valeur attendue y.

L'objectif est donc de minimiser la MSE afin de minimiser l'erreur entre les valeurs estimées et les valeurs attendues.

Ce qui nous donnerait

$$MSE = \frac{1}{n}\sum_{i}^{n}(\sigma(z_i) - y_i)^2$$

avec σ la fonction sigmoïde utilisée pour la régression logistique, définie comme en section 1.1 notre MSE nous donnerait:

$$MSE = \frac{1}{n}\sum_{i}^{n}(\frac{1}{1+e^{-z_i}} - y_i)^2$$

Cependant, nous pouvons remarquer que la fonction $\sigma(z)$ n'est pas linéaire.

En effet, on a $\sigma(z) = \frac{1}{1+e^{-z}}$, ce qui n'est pas une fonction linéaire.

Cela a pour conséquence que la MSE n'est pas convexe.

La descente en gradient ne pourra donc pas fonctionnner correctement, car on pourra trouver des minimum locaux à la place du minimum global, et si on trouve un minimum local, on ne va pas trouver les paramètres optimaux pour la régression logistique.

C'est pourquoi, on utilise plutôt la log loss fonction.

2.3 – Naive Bayes

Dans cette section, une implémentation d'un classifieur linéaire bayesien (naive bayes) a été réalisée. La fonction de prédictition a la signature suivante:

```
1    # TODO
```

et calcule la classe qui maximise la probabilité conditionnelle définie en section 1.2.

Dans cette implémentation, étant données que toutes nos features sont continues, nous avons considéré que *sepal length*, *sepal width*, *petal length* et *petal width* seront représenté comme 4 variables aléatoires $X_0, \cdots, X_3$ suivant 4 lois normales normales de paramètre (μ_k, σ_k).

C'est à dire:

$$X_k \sim \mathcal{N}(\mu_k, \sigma_k) \qquad\qquad k \in [\![0, 3]\!]$$

3 – Résultats

Références

[1] *1.1. Linear Models*. scikit-learn. URL: https://scikit-learn/stable/modules/linear_model.html.
[2] *Classifieur linéaire*. In: *Wikipédia*. URL: https://fr.wikipedia.org/w/index.php?title=Classifieur_lin%C3%A9aire&oldid=189518969 (visited on 01/05/2024).
[3] R. A. Fisher. *Iris*. 1936. DOI: 10.24432/C56C76. URL: https://archive.ics.uci.edu/dataset/53 (visited on 01/07/2024).
[4] *Régression logistique*. In: *Wikipédia*. URL: https://fr.wikipedia.org/w/index.php?title=R%C3%A9gression_logistique&oldid=210479759 (visited on 01/05/2024).
[5] *Scipy.Optimize.Fmin — SciPy v1.11.4 Manual*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin.html#scipy.optimize.fmin.

- TODO: ajouter les autres références des documentations utilisées