**Практическая работа №15**

**Линейный и квадратичный дискриминантный анализ**

**ML | Линейный дискриминантный анализ**

* Уровень сложности : [Средний](https://translated.turbopages.org/proxy_u/en-ru.ru.daac0f21-626a85f6-1ab5cb00-74722d776562/https/www.geeksforgeeks.org/medium/)
* Последнее обновление : 10 ноя, 2021

**Линейный дискриминантный анализ** или **Нормальный дискриминантный анализ** или **дискриминантный функциональный анализ** - это метод уменьшения размерности, который обычно используется для задач контролируемой классификации. Он используется для моделирования различий в группах, т.Е. Разделения двух или более классов. Он используется для проецирования объектов в пространстве более высоких измерений в пространство более низких измерений.
Например, у нас есть два класса, и нам нужно эффективно их разделить. Классы могут иметь несколько функций. Использование только одного признака для их классификации может привести к некоторому перекрытию, как показано на рисунке ниже. Таким образом, мы будем продолжать увеличивать количество признаков для правильной классификации.



**Пример:**
Предположим, что у нас есть два набора точек данных, принадлежащих двум разным классам, которые мы хотим классифицировать. Как показано на данном 2D-графике, когда точки данных построены на 2D-плоскости, нет прямой линии, которая могла бы полностью разделить два класса точек данных. Следовательно, в этом случае используется LDA (линейный дискриминантный анализ), который уменьшает 2D-граф в 1D-граф, чтобы максимизировать разделимость между двумя классами.



Здесь линейный дискриминантный анализ использует обе оси (X и Y) для создания новой оси и проецирует данные на новую ось таким образом, чтобы максимизировать разделение двух категорий и, следовательно, уменьшить 2D-график до 1D-графика.

Два критерия используются LDA для создания новой оси:

1. Максимизируйте расстояние между средними значениями двух классов.
2. Минимизируйте вариации внутри каждого класса.



На приведенном выше графике видно, что новая ось (выделенная красным цветом) генерируется и наносится на 2D-график таким образом, что она максимизирует расстояние между средними значениями двух классов и минимизирует вариации внутри каждого класса. Проще говоря, эта вновь сгенерированная ось увеличивает расстояние между точками данных двух классов. После генерации этой новой оси с использованием вышеупомянутых критериев все точки данных классов наносятся на эту новую ось и показаны на рисунке, приведенном ниже.



Но линейный дискриминантный анализ терпит неудачу, когда среднее значение распределений является общим, так как для LDA становится невозможным найти новую ось, которая делает оба класса линейно разделимыми. В таких случаях мы используем нелинейный дискриминантный анализ.

**Математика**

Предположим, что у нас есть два класса и d-мерные выборки, такие как x1, x2 ... xn, где:

* образцы n1 из класса (c1) и n2 из класса (c2).

Если xi является точкой данных, то ее проекция на линию, представленную единичным вектором v, может быть записана как vTxi

Рассмотрим u1 и u2 как средние выборки класса c1 и c2 соответственно до проекции, а u1hat обозначает среднее значение выборок класса после проекции, и оно может быть вычислено по:

Аналогично,

Теперь в LDA нам нужно нормализовать |\widetilde{\mu\_1} -\widetilde{\mu\_2} |. Пусть y\_i = v ^ {T} x\_i - проецируемые выборки, тогда разброс для выборок c1 равен:

Аналогично:



Теперь нам нужно спроецировать наши данные на линию, имеющую направление v, которое максимизирует

Для максимизации приведенного выше уравнения нам нужно найти вектор проекции, который максимизирует разницу средних уменьшений рассеяний обоих классов. Теперь матрица рассеяния s1 и s2 классов c1 и c2:

и s2



Упростив приведенное выше уравнение, получим:

Теперь мы определяем, разброс внутри классов (s w) и разброс b / w классов (s b):

Теперь мы попытаемся упростить числительную часть J (v)

Теперь, чтобы максимизировать приведенное выше уравнение, нам нужно вычислить дифференцирование относительно v



Здесь для максимального значения J (v) мы будем использовать значение, соответствующее наивысшему собственному значению. Это обеспечит нам лучшее решение для LDA.

**Расширения для LDA:**

1. **Квадратичный дискриминантный анализ (QDA):** каждый класс использует свою собственную оценку дисперсии (или ковариации при наличии нескольких входных переменных).
2. **Гибкий дискриминантный анализ (FDA):** где используются нелинейные комбинации входных данных, такие как сплайны.
3. **Регуляризованный дискриминантный анализ (RDA):** вводит регуляризацию в оценку дисперсии (фактически ковариации), смягчая влияние различных переменных на LDA.

**Реализация**

* В этой реализации мы выполним линейный дискриминантный анализ с использованием библиотеки Scikit-learn для набора данных Iris.

**Python3**

|  |
| --- |
| # necessary importimport numpy as npimport pandas as pdimport matplotlib.pyplot as pltimport sklearnfrom sklearn.preprocessing import StandardScaler, LabelEncoderfrom sklearn.model\_selection import train\_test\_splitfrom sklearn.discriminant\_analysis import LinearDiscriminantAnalysisfrom sklearn.ensemble import RandomForestClassifierfrom sklearn.metrics import accuracy\_score, confusion\_matrix  # read dataset from URLurl = "[https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data](https://translated.turbopages.org/proxy_u/en-ru.ru.daac0f21-626a85f6-1ab5cb00-74722d776562/https/archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data)"cls = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']dataset = pd.read\_csv(url, names=cls)  # divide the dataset into class and target variableX = dataset.iloc[:, 0:4].valuesy = dataset.iloc[:, 4].values  # Preprocess the dataset and divide into train and testsc = StandardScaler()X = sc.fit\_transform(X)le = LabelEncoder()y = le.fit\_transform(y)X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)  # apply Linear Discriminant Analysislda = LinearDiscriminantAnalysis(n\_components=2)X\_train = lda.fit\_transform(X\_train, y\_train)X\_test = lda.transform(X\_test)  # plot the scatterplotplt.scatter(    X\_train[:,0],X\_train[:,1],c=y\_train,cmap='rainbow',  alpha=0.7,edgecolors='b')  # classify using random forest classifierclassifier = RandomForestClassifier(max\_depth=2, random\_state=0)classifier.fit(X\_train, y\_train)y\_pred = classifier.predict(X\_test)  # print the accuracy and confusion matrixprint('Accuracy : ' + str(accuracy\_score(y\_test, y\_pred)))conf\_m = confusion\_matrix(y\_test, y\_pred)print(conf\_m) |



*LDA 2 - переменный график*

Точность: 0.9

[[10 0 0]

 [ 0 9 3]

 [ 0 0 8]]

**Приложения:**

1. **Распознавание лиц:** В области компьютерного зрения распознавание лиц - очень популярное приложение, в котором каждое лицо представлено очень большим количеством значений пикселей. Линейный дискриминантный анализ (LDA) используется здесь для уменьшения количества признаков до более управляемого числа перед процессом классификации. Каждое из генерируемых новых измерений представляет собой линейную комбинацию значений пикселей, которые образуют шаблон. Линейные комбинации, полученные с использованием линейного дискриминанта Фишера, называются гранями Фишера.
2. **Медицина:** В этой области линейный дискриминантный анализ (LDA) используется для классификации болезненного состояния пациента как легкого, умеренного или тяжелого на основе различных параметров пациента и лечения, которое он проходит. Это помогает врачам ускорить или снизить темпы лечения.
3. **Идентификация клиентов:** Предположим, мы хотим определить тип клиентов, которые с наибольшей вероятностью купят конкретный продукт в торговом центре. Выполняя простой опрос вопросов и ответов, мы можем собрать все особенности клиентов. Здесь линейный дискриминантный анализ поможет нам определить и выбрать функции, которые могут описать характеристики группы клиентов, которые с наибольшей вероятностью купят этот конкретный продукт в торговом центре.