



СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ



Н.А. Обухова, А.А. Мотыко

Методы обработки и анализа медицинских изображений

Методические рекомендации по
практическим работам

СПбГЭТУ «ЛЭТИ», 2021 г.





1 МЕТОДЫ АНАЛИЗА ДАННЫХ В МЕДИЦИНСКИХ ИНТЕЛЛЕКТУАЛЬНЫХ ВИДЕОСИСТЕМАХ

1.1 Классификация объектов с помощью расстояния Махаланобиса

1.1.1 Общие теоретические сведения

Интеллектуальный анализ данных («data mining») - это область знания, включающая в себя различные методы и алгоритмы для поиска и выделения из данных какой-либо целевой информации.

Найденная информация может представлять ценность для пользователя, как материал, на основании которого он самостоятельно может делать какие-либо заключения, или может служить исходными данными для построения и работы интеллектуальной компьютерной системы.

В интеллектуальный анализ данных входят:

- статистические методы;
- методы машинного обучения.

Методы первой группы базируются на теории вероятности и математической статистике, они включают в себя корреляционный, дисперсионный и факторный анализ, байесовские методы, дискриминантный анализ и другие.

Методы машинного обучения основаны теории оптимизации, теории графов, численных методах.

Методы разделяют на

- обучение с учителем;
- обучение без учителя.

Обучение с учителем предполагает, что для задачи, требующей решения, существуют так называемые обучающие примеры - случаи, где входным данным сопоставлен корректный ответ. Существуют алгоритмы «обучения», то есть настройки параметров математических моделей (нейронных сетей, лесов решающих деревьев, опорных векторов и других) с помощью анализа имеющихся примеров. В результате «обученные» математические модели





способны генерировать ответ при подаче на вход нового образца, не встречавшегося при обучении.

Методы обучения с учителем решают, например, задачи классификации и регрессии (частный случай прогнозирования). На этом основаны системы распознавания образов и детектирования объектов.

К обучению без учителя (случай, когда имеется выборка данных, но нет обучающих примеров) относят методы кластеризации и понижения размерности признакового пространства.

Простой и эффективный алгоритм классификации может быть построен на основе меры Махаланобиса, которая определяет расстояние между случайной величиной, описываемой вектором значений, и некоторым распределением случайных величин. Расстояние Махаланобиса обобщает евклидово и фактически позволяет установить сходство между некоторым объектом и выборкой объектов, соответствующих определенному классу.

Расстояние между объектом и выборкой определяются следующим образом.

$$d(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{S}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

где $\mathbf{x} = (x_1, x_2, \dots, x_n)$ - многомерный вектор значений признаков объекта, $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_n)$ - вектор средних значений признаков объектов, составляющих выборку, $\mathbf{S}$ - матрица ковариации признаков.

Квадратная симметричная матрица ковариации $\mathbf{S}$ в данном случае состоит из попарных ковариаций средних значений признаков (элементов $\boldsymbol{\mu}$).

$$\mathbf{S} = \begin{bmatrix} \text{cov}(\mu_1, \mu_1) & \text{cov}(\mu_1, \mu_2) & \dots & \text{cov}(\mu_1, \mu_m) \\ \text{cov}(\mu_2, \mu_1) & \text{cov}(\mu_2, \mu_2) & \dots & \text{cov}(\mu_2, \mu_m) \\ \dots & \dots & \dots & \dots \\ \text{cov}(\mu_m, \mu_1) & \dots & \dots & \text{cov}(\mu_m, \mu_m) \end{bmatrix}$$

Каждый элемент матрицы вычисляется как:

$$\text{cov}(\mu_n, \mu_k) = M((\mu_n - M(\mu_n))(\mu_k - M(\mu_k))),$$

где M - означает математическое ожидание.

В результате в матрице ковариации некоторой выборки (совокупности объектов одного класса, применительно к задаче классификации) на главной





диагонали располагаются дисперсии признаков, а вне главной диагонали стоят значения, показывающие меру линейной зависимости признаков.

Нетрудно заметить, что если S представляет собой единичную матрицу, то расстояние Махаланобиса вырождается в евклидово расстояние. В случае, если матрица ковариации является диагональной, расстояние становится стандартизованным евклидовым.

Отсюда следует вывод, что применение нормы Евклида в случае классификации возможно только тогда, когда все признаки имеют одну размерность и масштаб (что в результате и делает S единичной матрицей). Иначе, вклад признаков, измеряющихся в маленьком масштабе (например, от 0 до 100) будет полностью нивелирован вкладом признаков с большим масштабом (например, от 1000 до 100000).

Чтобы нивелировать негативный эффект от разного масштаба признаков проводят их нормализацию и в результате используют стандартизованное евклидово расстояние. Это расстояние позволяет работать с признаками в разном масштабе, но не учитывает линейную зависимость между ними. Разницу при использовании расстояния Махаланобиса и стандартизованного евклидова поясняет рисунок 4.1.

Белыми кружками обозначены объекты выборки, составляющие класс. Черными - объекты, для которых нужно определить расстояния до класса (центр обозначен черной звездочкой). Объект В будет ближе к классу, чем объект А по стандартизованной евклидовой метрике (пунктирная окружность), но дальше по метрике Махаланобиса (овал со сплошной линией).

Это связано с тем, что использование расстояния Евклида предполагает, что данные имеют изотропное нормальное распределение, а расстояние Махаланобиса допускает анизотропное распределение.



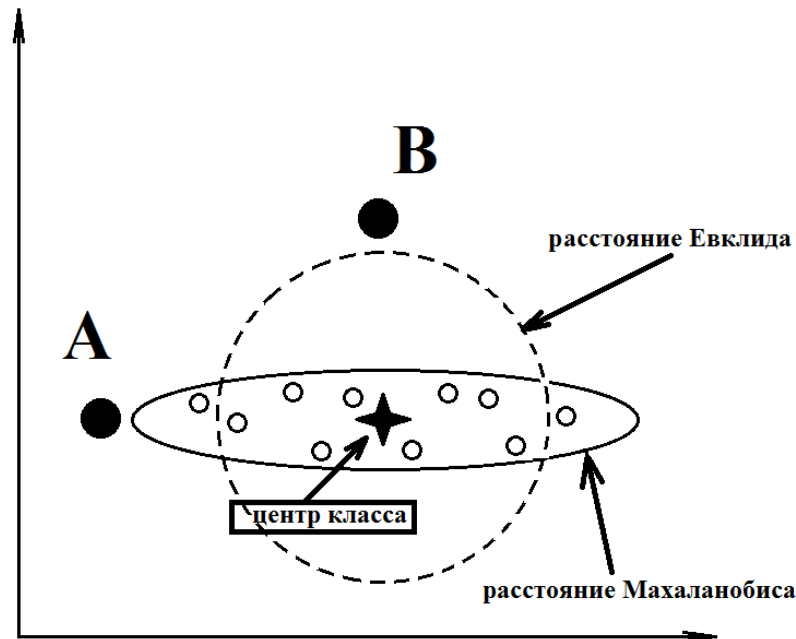


Рис. 1 - Иллюстрация к использованию различных метрик при классификации объектов.

На практике в задачах классификации чаще всего ковариация между признаками объектов не нулевая, то предпочтительнее использовать именно расстояние Махаланобиса.

Таким образом задача классификации объекта x , то есть определение его принадлежности к одному из имеющихся классов, сводится к оценке расстояний $d(x)$ от объекта до каждого из классов и идентификации минимального.

Для проведения расчетов необходимо вычислить матрицы ковариации S и векторы средних значений признаков μ для каждого класса. Это возможно сделать с помощью имеющихся *обучающих примеров* (выборки объектов, значения признаков которых, как и их принадлежность к классам известна). Неизвестные математические ожидания для классов объектов $\mu = (\mu_1, \mu_2, \mu_k, \mu_n)$ заменяются на их оценки:

$$\widehat{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} x_i,$$

где N - число элементов в обучающей выборке для класса, x_i - значение признака, входящего в вектор обучающего примера $x = (x_1, x_2, x_i, x_n)$. Далее вычисляются оценки для ковариационных матриц.

В машинном обучении с учителем популярны следующие методы, использующие метрику Махаланобиса.



- Линейный дискриминантный анализ
- Квадратичный дискриминантный анализ.

Линейный дискриминантный анализ служит для разделения объектов на два класса. В нем принимается допущение о гомоскедастичности классов (то есть, о равенстве ковариационных матриц двух классов).

$$\mathbf{S}_1 = \mathbf{S}_2 = \mathbf{S}_\Sigma.$$

Классификационное правило элементарно: объект относят к классу, до которого меньше расстояние Махаланобиса.

Квадратичный дискриминантный анализ позволяет проводить многоклассовую классификацию и не требует равенства ковариационных матриц классов.

Объект считается принадлежащим классу k , до которого меньше значение

$$f_k = (\mathbf{x} - \boldsymbol{\mu}_k)^T \mathbf{S}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \ln (\det (\mathbf{S}_k^{-1})).$$

где $\det$ - обозначение определителя матрицы. Данная формула выводится из вероятностной модели, которая моделирует условное распределение данных.

1.1.2 Цель работы

Разработать программу, моделирующую алгоритм классификации объектов с помощью расстояния Махаланобиса. В качестве исходных данных нужно использовать базу данных ирисов Фишера (<http://archive.ics.uci.edu/ml/datasets/Iris>). База содержит образцы для трех различных классов ирисов.

1.1.3 Порядок выполнения

1. Загрузить из текстовых файлов данные в программу, при этом корректно обработать принадлежность к классам и значения признаков.
2. Данные для каждого класса случайным образом разделить на две подгруппы, содержащие 90% и 10% образцов.





3. По группам, содержащим 90% выборок оценить векторы средних значений и матрицы ковариаций признаков.
4. Реализовать процедуру классификации по расстоянию Махаланобиса, обучить классификатор.
5. С помощью тестовой выборки (10% образцов) реализовать процедуру тестирования точности классификатора.
6. Вывести результат.

1.1.4 Результат работы

Компьютерная программа (консольное приложение), реализующая алгоритм классификации объектов с помощью расстояния Махаланобиса.

1.1.5 Контрольные вопросы

1. Сформулируйте основное отличие методов обучения с учителем от методов обучения без учителя.
2. Дайте определение ковариации. Что означает ковариация признаков объекта?
3. Объясните, чему соответствуют элементы матрицы ковариации в методе классификации с помощью расстояния Махаланобиса?
4. Прокомментируйте, как определяется расстояние Махаланобиса.
5. Объясните, как связаны расстояние Махаланобиса и расстояние Евклида.

1.1.6 Дополнительные материалы

Библиотека OpenCV содержит большой инструментарий из области машинного обучения (модуль ml). Это и функции для подготовки данных, и реализации популярных алгоритмов: метода опорных векторов (SVM), лесов решающих деревьев (RDF), байесовского классификатора, нейронных сетей (NN) и других. Существуют также классы для работы с моделями - результатами глубокого обучения.

Полностью готовой реализации дискриминантного анализа в библиотеке нет, данный алгоритм нужно синтезировать самостоятельно. Тем не менее, в OpenCV реализовано большое количество инструментов, которые будут полезны при имплементации алгоритма, в частности есть функция,





вычисляющая расстояние Махаланобиса между векторами и функция для вычисления ковариационной матрицы.

Полезно в целом ознакомиться с возможностями OpenCV при работе с матрицами (не только содержащими изображения). Эти знания помогут при выполнении практических заданий.

Существуют функции, относящиеся к операциям над матрицами: `cv::absdiff()`, `cv::bitwise_and()` и некоторые другие. Одними из самых частых в использовании являются функции для элементарных матричных операций, для которых также реализованы перегруженные операторы, или методы класса. В примере для простоты операции проводятся над единичной матрицей, и матрицей, состоящей из одних единиц. Полезно при реализации примера попробовать и другие операнды.

```
#include "stdafx.h"
#include "opencv2/core/core.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

int main()
{
    //Создадим две матрицы, единичную и состоящую из единиц.
    Mat eye = Mat::eye(3, 3, CV_32F);
    Mat box = Mat::ones(3, 3, CV_32F);

    //Выведем их в консоль
    std::cout << "eye = " << std::endl;
    std::cout << eye << std::endl;
    std::cout << "box = " << std::endl;
    std::cout << box << std::endl;

    //Сложение матриц
    Mat mat_sum;
    add(eye, box, mat_sum);
    std::cout << "mat_sum = " << std::endl;
```





```
std::cout << mat_sum << std::endl;
```

```
//или вариант с перегруженным оператором
```

```
mat_sum = eye + box;
```

```
std::cout << "mat_sum = " << std::endl;
```

```
std::cout << mat_sum << std::endl;
```

```
//Вычитание матриц
```

```
Mat mat_sbtrct;
```

```
subtract(eye, box, mat_sbtrct);
```

```
std::cout << "mat_sbtrct = " << std::endl;
```

```
std::cout << mat_sbtrct << std::endl;
```

```
//или вариант с перегруженным оператором
```

```
mat_sbtrct = eye - box;
```

```
std::cout << "mat_sbtrct = " << std::endl;
```

```
std::cout << mat_sbtrct << std::endl;
```

```
//Умножение матрицы на скаляр
```

```
Mat mat_sc_mat;
```

```
mat_sc_mat = eye*5.0;
```

```
std::cout << "mat_sc_mat = " << std::endl;
```

```
std::cout << mat_sc_mat << std::endl;
```

```
//Поэлементное умножение, или деление
```

```
Mat ew_mult, ew_div;
```

```
eye.mul(box);
```

```
//или
```

```
multiply(eye, box, ew_mult);
```

```
std::cout << "ew_mult = " << std::endl;
```

```
std::cout << ew_mult << std::endl;
```

```
ew_div = eye / box;
```

```
//или
```





```
divide(eye, box, ew_div);
std::cout << "ew_div = " << std::endl;
std::cout << ew_div << std::endl;

//Матричное перемножение
Mat mtx_mult;
mtx_mult = eye*box;
std::cout << "mtx_mult = " << std::endl;
std::cout << mtx_mult << std::endl;

//Транспонирование матриц
Mat mtx_transposed;
transpose(eye, mtx_transposed);
//или
mtx_transposed.t();
std::cout << "mtx_transposed = " << std::endl;
std::cout << mtx_transposed << std::endl;

//Инвертирование матриц
Mat mtx_inv;
invert(eye, mtx_inv);
//или
eye.inv();
std::cout << "mtx_inv = " << std::endl;
std::cout << mtx_inv << std::endl;

waitKey();
return 0;
}
```

Кроме этого, очень полезны функции, вычисляющие различные характеристики матриц. На рис. 2 показан примерный вывод программы.

```
#include "stdafx.h"
#include "opencv2/core/core.hpp"
```





```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

int main()
{
    //Создадим две матрицы, единичную и состоящую из единиц.
    Mat eye = Mat::eye(3, 3, CV_32F);
    Mat box = Mat::ones(3, 3, CV_32F);

    //Выведем их в консоль
    std::cout << "eye = " << std::endl;
    std::cout << eye << std::endl;
    std::cout << "box = " << std::endl;
    std::cout << box << std::endl;

    //Сумма всех элементов
    cv::Scalar sum_ = sum(box);
    std::cout << "sum_ = " << std::endl;
    std::cout << sum_[0] << std::endl;

    //След матрицы (сумма элементов главной диагонали)
    cv::Scalar trace_ = trace(box);
    std::cout << "trace_ = " << std::endl;
    std::cout << trace_[0] << std::endl;

    //Норма
    cv::Scalar norm_ = norm(box);
    std::cout << "norm_ = " << std::endl;
    std::cout << norm_[0] << std::endl;

    //Определитель
    cv::Scalar det_ = determinant(box);
    std::cout << "det_ = " << std::endl;
}
```





```
std::cout << det_[0] << std::endl;
```

```
//Собственные числа и векторы
```

```
Mat eigen_values, eigen_vectors;
```

```
eigen(box, eigen_values, eigen_vectors);
```

```
std::cout << "eigen_values = " << std::endl;
```

```
std::cout << eigen_values << std::endl;
```

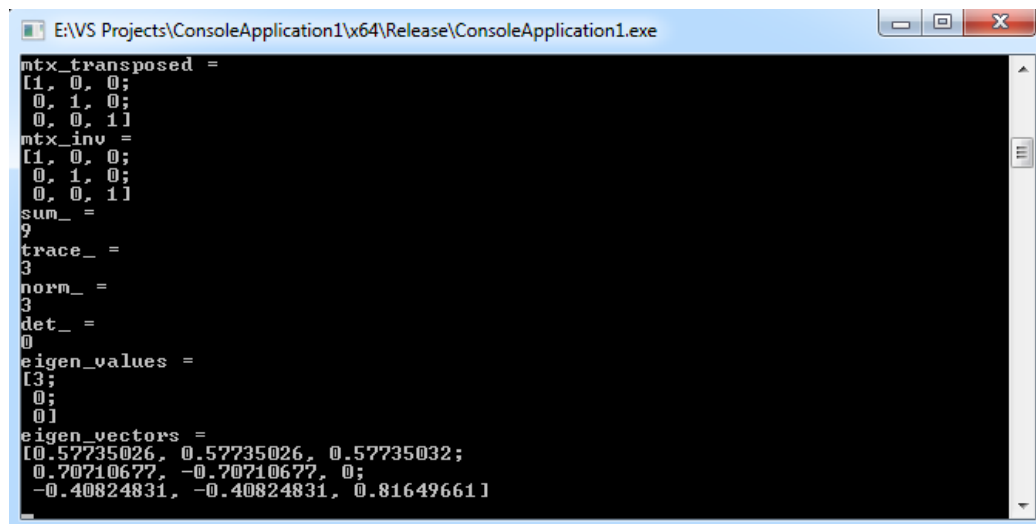
```
std::cout << "eigen_vectors = " << std::endl;
```

```
std::cout << eigen_vectors << std::endl;
```

```
waitKey();
```

```
return 0;
```

```
}
```



```
E:\VS Projects\ConsoleApplication1\x64\Release\ConsoleApplication1.exe
mtx_transposed =
[1. 0. 0.;
 0. 1. 0.;
 0. 0. 1.]
mtx_inv =
[1. 0. 0.;
 0. 1. 0.;
 0. 0. 1.]
sum_ =
9
trace_ =
3
norm_ =
3
det_ =
0
eigen_values =
[3;
 0;
 0]
eigen_vectors =
[0.57735026, 0.57735026, 0.57735032;
 0.70710677, -0.70710677, 0;
 -0.40824831, -0.40824831, 0.81649661]
```

Рис. 2 - Консольный вывод программы-примера

Помимо перечисленных, OpenCV имеет еще целый ряд функций для работы с матрицами. Стоит упомянуть сравнение матриц, спектральные преобразования, проецирования, разложения. Для выполнения практического задания будут полезны:

```
//Вычисление ковариационной матрицы
```

```
Mat covar_, mean_;
```

```
calcCovarMatrix(box, covar_, mean_, COVAR_ROWS);
```

```
std::cout << "covar_ = " << std::endl;
```

```
std::cout << covar_ << std::endl;
```

```
//Вычисление расстояния Махаланобиса
```





//Создаем единичную матрицу, две строки которой якобы содержат векторы признаков двух объектов

```
Mat mtx_samples = Mat::ones(2, 2, CV_64F);
```

```
Mat covar2_, mean_2;
```

//Вычисляем и инвертируем ковариационную матрицу

```
calcCovarMatrix(mtx_samples, covar2_, mean_2, COVAR_ROWS);
```

```
invert(covar2_, covar2_);
```

//Вычисляем расстояние Махаланобиса

```
Mat r1 = mtx_samples.row(0);
```

```
Mat r2 = mtx_samples.row(1);
```

```
double dist = Mahalanobis(r1, r2, covar2_);
```

```
std::cout << "dist = " << std::endl;
```

```
std::cout << dist << std::endl;
```

При работе с функциями OpenCV, содержащими флаги (как, например, `cv::calcCovarMatrix`) следует сверяться с руководством для того, чтобы в зависимости от ситуации выставить соответствующие значения. Это обеспечит корректную работу и желаемый результат.

