# Практическая работа № 10. Данные “Digit Recognizer”

Этот пример относится к области распознавания образов. Задача состоит в распознавании рукописных цифр. В компьютерной памяти изображения могут быть представлены в форме числовых матриц. Широко известные данные *MNIST* (“Modified National Institute of Standards and Technology”) используются в качестве классического примера для тестирования различных алгоритмов машинного обучения.

Следует отметить, что данные *Credit* и *MNIST* являются принципиально разными.

Во-первых, данные *Credit* являются несбалансированными и бинарными: метка принимает только два значения 0 или 1. В случае данных *MNIST* мы имеет дело с относительно сбалансированными данными, в которых метка принимает 10 различных значений: цифры от 0 до 9.

Во-вторых, число признаков в данных *Credit* очень мало (равно 10), что делает неэффективным использование таких моделей как случайный лес или метод опорных векторов. В случае данных *MNIST* такой проблемы не возникает, поскольку количество признаков достаточно велико и составляет 784.

Каждое наблюдение (или изображение) из базы данных *MNIST* представляет собой квадратную матрицу с размерностью 28. Каждое значение матрицы принимает целочисленные значения от 0 до 255. Большее значение означает большую яркость цвета.

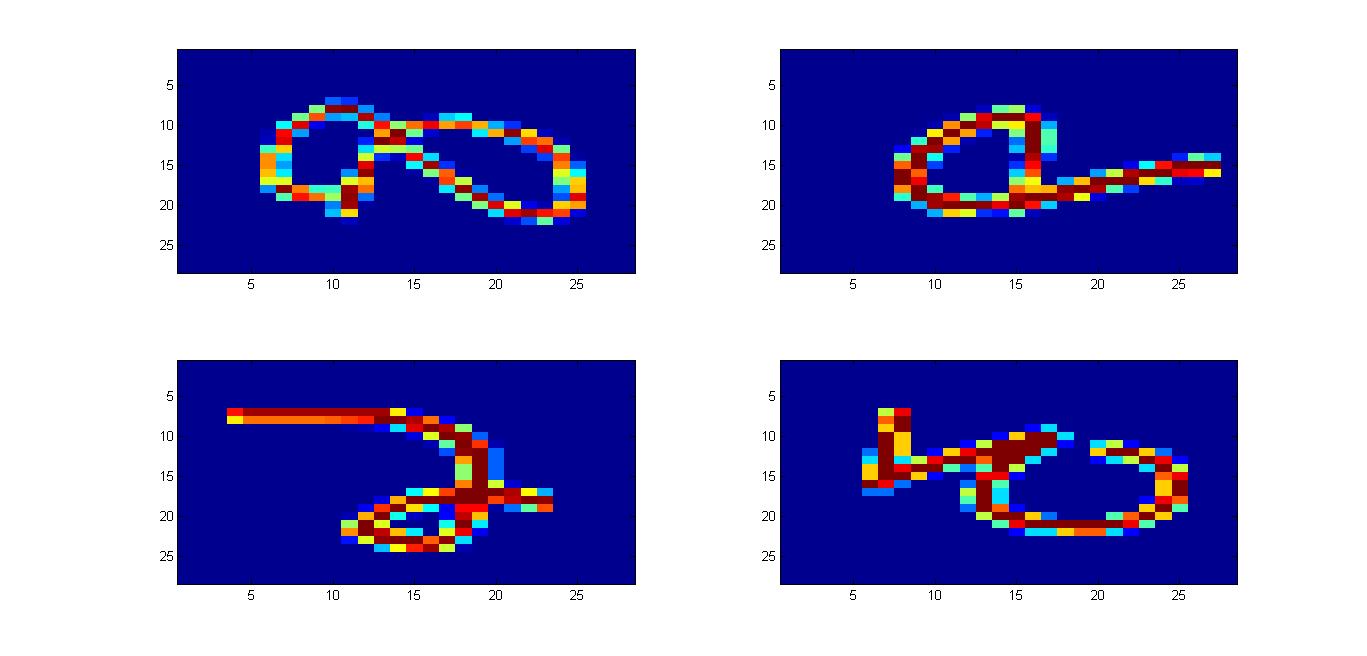


Рис. 9. Изображения рукописных цифр из базы данных *MNIST*

Оценочная метрика– точность классификации, т.е. доля изображений, которые были правильно классифицированы. Например, результат 0.97 означает то, что правильно распознаны все, кроме 3% изображений.

Предварительно необходимо установить и запустить следующие пакеты:

|  |
| --- |
| install.packages("caret")  install.packages("kernlab")  install.packages("randomForest")  library(caret)  library(kernlab)  library(randomForest)  library (pROC) |

Загрузка данных:

|  |
| --- |
| train<-read.table ("F:/train.csv", header=TRUE, sep=',')  #Нормализация (приведение к диапазону [0,1])  train[,-1] <- train[,-1] / 255  inTrain = data.frame(y=train[,1], train[,-1])  inTrain$y <- as.factor(inTrain$y)  #Разделение на обучающую и тестовую выборки 60/40  trainIndex = createDataPartition(inTrain$y, p = 0.60,list=FALSE)  training = inTrain[trainIndex,]  cv = inTrain[-trainIndex,] |

### Метод опорных векторов

Метод опорных векторов *SVM* (Support Vector Machine) — один из наиболее универсальных методов классификации, отличающийся быстродействием и высоким качеством результатов классификации (особенно в тех случаях, когда число признаков сопоставимо или превышает число наблюдений). Забегая вперед, скажем, что с использованием *SVM* была получена наилучшая среди всех использованных алгоритмов качество классификации.

Модель опорных векторов относится к разряду разделяющих, но не аппроксимирующих алгоритмов. Нелинейная модель *SVM* является обобщением классической линейной модели, в которой решение ищется как линейная функция признаков. Целью является минимизация суммы квадратов регрессионных коэффициентов:



при следующем условии



где



Идея подхода *SVM* формулируется следующим образом: Используя метод Лагранжа, мы трансформируем рассмотрение задачи из многомерного пространства исходных признаков во вторичное пространство коэффициентов Лагранжа. Следует отметить, что по определению количество коэффициентов Лагранжа в точности совпадает с количеством наблюдений. Одним из основных гиперпараметров *SVM* является ядерная квадратная матрица или ядро (kernel):



где  - скалярное произведение векторов признаков.

Нормализация исходных данных –снижение разброса чисел, путём деления каждого элемента на максимум по столбцу. Такая нормализация очень существенна для вычисления ядерной функции. Отметим, что если выполнять алгоритм без нормализующей составляющей, то возникает та же проблема, что и была в задаче *Credit* при использовании пакета *nnet* к несбалансированным данным – алгоритм не производит классификацию, и все результаты *auc* в этом случае равны 0.5.

Обучение модели:

|  |
| --- |
| fit <- train(y ~ ., data = head(training, 1000), method = 'svmRadial', tuneGrid = data.frame(sigma=0.0107249, C=1)) |

Проверка обучения:

|  |
| --- |
| results <- predict(fit, newdata = head(cv, 1000))  confusionMatrix(results, head(cv$y, 1000)) |

Результаты метода *SVM*

|  |
| --- |
| Confusion Matrix and Statistics  Reference  Prediction 0 1 2 3 4 5 6 7 8 9  0 105 0 0 1 0 1 0 0 2 0  1 0 112 3 0 1 0 0 4 2 1  2 0 0 89 4 1 0 1 1 0 1  3 0 0 1 77 0 5 0 0 2 0  4 0 0 1 0 96 0 2 1 1 4  5 0 1 0 5 0 88 0 0 3 2  6 1 0 1 0 0 1 100 0 0 0  7 0 0 3 2 1 0 0 84 0 6  8 0 0 5 4 3 2 1 0 76 0  9 0 0 0 1 6 0 0 1 2 82  Overall Statistics    Accuracy : 0.909  95% CI : (0.8894, 0.9261)  No Information Rate : 0.113  P-Value [Acc > NIR] : < 2.2e-16    Kappa : 0.8988  Mcnemar's Test P-Value : NA  Statistics by Class:  Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  Sensitivity 0.9906 0.9912 0.8641 0.8191 0.8889 0.9072  Specificity 0.9955 0.9876 0.9911 0.9912 0.9899 0.9878  Pos Pred Value 0.9633 0.9106 0.9175 0.9059 0.9143 0.8889  Neg Pred Value 0.9989 0.9989 0.9845 0.9814 0.9866 0.9900  Prevalence 0.1060 0.1130 0.1030 0.0940 0.1080 0.0970  Detection Rate 0.1050 0.1120 0.0890 0.0770 0.0960 0.0880  Detection Prevalence 0.1090 0.1230 0.0970 0.0850 0.1050 0.0990  Balanced Accuracy 0.9930 0.9894 0.9276 0.9052 0.9394 0.9475  Class: 6 Class: 7 Class: 8 Class: 9  Sensitivity 0.9615 0.9231 0.8636 0.8542  Specificity 0.9967 0.9868 0.9836 0.9889  Pos Pred Value 0.9709 0.8750 0.8352 0.8913  Neg Pred Value 0.9955 0.9923 0.9868 0.9846  Prevalence 0.1040 0.0910 0.0880 0.0960  Detection Rate 0.1000 0.0840 0.0760 0.0820  Detection Prevalence 0.1030 0.0960 0.0910 0.0920  Balanced Accuracy 0.9791 0.9549 0.9236 0.9216 |

### Задание

1) В задаче распознавания с использованием SVM выполнить обучение модели с различным размером обучающей выборки. (500, 1000, 2000, 5000).

Размер обучающей выборки – data = head(training, 1000) – 1000

Сделать выводы о влиянии величины выборки на качество классификации.

2) Применить различные значения параметра sigma

### Содержание отчёта

1. Полученные результаты по каждому пункту.

2. Выводы о проделанной работе.