Московский государственный университет им М.В. Ломоносова
факультет Вычислительной математики и кибернетики

Метод k взвешенных ближайших соседей

Выполнил: Казаринов А. В.
группа 316, кафедра МС
Преподаватель:
Горшенин А. К.

Москва — 2021

Содержание

Введение

Метод k ближайших соседей (KNN) – метрический алгоритм, основанный на гипотезе компактности и понятии метрики. Метод впервые был разработан Эвелином Фиксом и Джозефом Ходжесом в 1951 году. Широко применяется для решения задачи классификации, а также и для задачи регресии. Данная работа посвящена классификации фильмов по жанру методом k взвешенных ближайших соседей. Классификация фильмов по жанру может быть использована например в рекомендательных системах онлайн-кинотеатров. Цель данного реферата - сравнить несколько способов взвешивания соседей и узнать какой подход работает лучше на выбранных данных.

Области применения алгоритма

1) Рекомендательные системы. Задачу можно сформулировать следующим образом: найти что-то похожее, близкое к тому, что нравится пользователю. При такой формулировке KNN является очевидным решением.

2) Поиск документов похожих семантически. Если векторные представления близки друг к другу, то темы документов схожи.

3) Поиск аномалий и выбросов.

4) Задача кредитного скоринга.

Постановка задачи классификации

Сформулируем задачу классификации в общем виде.

$X = \mathbb{R}^n$ – множество объектов, $Y = \{1, \ldots, M\}$ – множество ответов.

$X^l = \{(x_i, y_i),\ i = \overline{1,n}\}$ – обучающая выборка, где $y_i = y(x_i)$.

y– неизвестная функция.

Также задано множество объектов $X^m = \{(x_i),\ i = \overline{1,n}\}$, для которых нам нужно выяснить ответы.

Требуется найти решающую функцию $a : X \to Y$, приближающую y на всем множестве X. Функция a называется классификатором.

Математическое обоснование метода

Гипотеза о компактности.

В задачах классификации предполагаем, что классы образуют компактно локализованные подмножества в пространстве объектов. Это также означает, что граница между классами имеет достаточно простую форму. Или, иначе говоря, "близкие" объекты, как правило, лежат в одном классе. Понятие "близости" формализуется метрикой.

Пусть $x_i = (x_i^1, \ldots, x_i^n)$ - вектор признаков объекта x_i, $i = 1, 2$

Евклидова метрика:

$$\rho(x_1, x_2) = \left(\sum_{j=1}^{n} |x_1^j - x_2^j|^2 \right)^{\frac{1}{2}}$$

метрика Минковского:

$$\rho(x_1, x_2) = \left(\sum_{j=1}^{n} |x_1^j - x_2^j|^p \right)^{\frac{1}{p}}$$

Косинусная метрика:

$$\rho(x_1, x_2) = \frac{(x_1, x_2)}{||x_1|| * ||x_2||}$$

Охватить все способы введения метрик невозможно, но также в метрических классификаторах часто используются манхэттенское расстояние, расстояние Хэмминга, расстояние Махаланобиса, расстояние Жаккара.

Метод k ближайших соседей

Возьмём произвольный объект x множеста X. Для объектов выборки $x_1, \ldots, x_l$ введём новую нумерацию $x^{(1)}, \ldots, x^{(l)}$ в порядке возрастания их расстояния от объекта x:

$$\rho(x, x^{(1)}) \leqslant \rho(x, x^{(2)}) \leqslant \cdots \leqslant \rho(x, x^{(l)}),$$

$y^{(i)}$ – ответ на объекте $x^{(i)}$

Всё готово для определения классификатора a. Задаём его следующей формулой:

$$a(x; X^l) = \arg\max_{y \in Y} \sum_{i=1}^{l} \mathbb{1}_{\{y^{(i)}=y\}} w(i, x),$$

$\mathbb{1}\{y^{(i)} = y\}$ - индикатор,

$w(i, x)$ – весовая функция (вес) объекта $x^{(i)}$. Вес - неотрицательная, невозрастающая по i функция. Она является показателем важности объекта $x^{(i)}$ для последующей классификации объектов.

Для метода одного ближайшего соседа вес задаётся как $w(i, x) = \mathbb{1}_{\{i \leqslant 1\}}$

Обычный способ введения весов для k ближайших соседей: $w(i, x) = \mathbb{1}_{\{i \leqslant k\}}$

Веса для метода ближайших соседей

Есть другие способы введения весов.

1) $w(i,x) = \mathbb{1}_{\{i \leqslant k\}} w_i$, где w_i - вес, зависящий от номера объекта $x^{(i)}$

   а) $w_i = \frac{k+1-i}{k}$ – линейно убывающие веса;

   б) $w_i = q^i$ – экспоненциально убывающие веса, $0 < q < 1$

2) $w(i,x) = \rho(x, x^{(i)})$

3) $w(i,x) = K\left(\frac{\rho(x,x^{(i)})}{h}\right)$ – вес, равный ядру с шириной h.

Способы 2-3 задания весов предположительно более точны, потому что в них используется информация о расстоянии между объектами. А из гипотезы компактности следует, что более близкие по расстоянию объекты имеют большую важность при принятии решения к какому классу отнести объект.

Достоинства и недостатки метода

Плюсы:

1) Хорошо подходит для старта в решении задачи (baseline).

2) Простота и высокая интерпретируемость.

3) Алгоритм не чувствителен к выбросам.

Минусы:

1) Проклятие размерности - метод хуже работает на данных с большой размерностью.

2) Метод явно хранит все обучающие объекты, что становится проблемой на сверхбольших выборках.

3) Классификация одного объекта требует числа операций, зависящего линейно от размера обучающей выборки, так как вычисляются расстояния до каждого объекта.

Определение жанра фильма

Описание датасета.

В качестве набора фильмов был взят открытый датасет с портала Kaggle из 85855 фильмов IMDb с такими атрибутами как название, описание, жанр, количество оценок, средняя оценка и т.д. https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset

Для создания меток ответов обучающей выборки будут использоваться жанры фильмов, а для признаков объектов – текстовые столбцы фильмов (название, описание, режиссёр, актёры, название киностудии...).

Постановка: бинарно определить жанр: драматический или не драматический; комедия или не комедия; фантастика или не фантастика.

Векторизация текста производилась методом TfidfVectorizer. Tf означает частоту термина, а tf-idf означает частоту термина, умноженную на обратную частоту документа. Такая схема взвешивания терминов позволяет хорошо решать задачу классификации документов. Если термин встречается в большом числе описаний фильмов, то он менее информативен для определения жанра фильма и наоборот.

В качестве метрики близости использовалась косинусная метрика. Косинусная близость лучше близости по евклидовой метрике, потому что в нашей задаче сонаправленность векторов встречаемости токенов важнее, чем разность их величин. Основным преимуществом косинусного расстояния является то, что данная метрика хорошо работает на разреженных данных таких как текст.

Алгоритм обучения

Разбиение выборки на 75% обучающую и 25% тестовую.

Этап 0:

Выбирается какой вес будет использоваться. Обучение проводится методом кросс валидации на обучающей выборке на количестве соседей от 1 до 10. Для кросс валидации обучающаяся выборка делится на 4 блока. Затем вычисляется среднее качество для каждого метода взвешивания и выбирается метод с наибольшим значением качества.

Этап 1:

Выбирается количество соседей, на котором будет производится обучение. Количество соседей изменяется от 1 до 200. В итоге фиксируется количество соседей, для которого оказалось лучшим значение качества на кросс валидации.

Этап 2 (финальный):

С выбранным методом взвешивания и количеством соседей обучается классификатор на всей обучающей выборке и вычисляется его качество на тестовой выборке.

Оценки качества

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Accuracy - доля правильных ответов алгоритма.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Recall - полнота, демонстрирует способность алгоритма обнаруживать данный класс в целом.

$$recall = \frac{TP}{TP+FN}$$

Для оценки качества будет в основном использоваться полнота (recall), а также accuracy. Recall был выбран по следующей причине: в данной задаче false positive не всегда является ошибкой - например, фильм может иметь жанр драмы как неосновной, но это не указано в столбце жанры, а случай false negative грубая ошибка – в датасете помечено, что фильм драма, но к драматичным алгоритм его не отнёс. Лучше всего такую особенность отражает метрика качества recall.

Результаты

Значения метрик качества: жанр: драма, число соседей: 221, веса: обратно-пропорциональные расстоянию

accuracy: 0.678

recall: 0.860

жанр: комедия, число соседей: 5, веса: экспоненциально убывающие

accuracy: 0.689

recall: 0.581

Качество оказалось не очень высоким, что объясняется сложной структурой признаков. На комедии понизилось качестово recall, это могло быть вызвано меньшим количеством фильмов этого жанра.

Листинг кода

[6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn import neighbors
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, recall_score
```

Обработка данных

загружаем датасет

[7]:
```
movies0 = pd.read_csv('D:\prac\IMDb movies.csv', delimiter = ',',␣
↪low_memory=False)
```

отбираем столбцы, с которыми будем работать

[8]:
```
movies = movies0[['original_title', 'description', 'actors', 'director',␣
↪'writer', 'production_company', 'genre']]

movies.dropna(axis=0,inplace=True)
```

Добавляем столбцы, в которые будем записывать ответы алгоритма

[9]:
```
movies['drama'] = np.zeros(movies.shape[0])
movies['drama'] = np.where((movies.genre.str.contains('Drama')), 1, movies.
↪drama)
movies['comedy'] = np.zeros(movies.shape[0])
movies['comedy'] = np.where((movies.genre.str.contains('Comedy')), 1, movies.
↪comedy)
```

	original_title	description	actors	director	writer	production_company	genre	drama	comedy
0	Miss Jerry	The adventures of a female reporter in the 1890s.	Blanche Bayliss, William Courtenay, Chauncey D...	Alexander Black	Alexander Black	Alexander Black Photoplays	Romance	0.0	0.0
1	The Story of the Kelly Gang	True story of notorious Australian outlaw Ned ...	Elizabeth Tait, John Tait, Norman Campbell, Be...	Charles Tait	Charles Tait	J. and N. Tait	Biography, Crime, Drama	1.0	0.0
2	Den sorte drøm	Two men of high rank are both wooing the beaut...	Asta Nielsen, Valdemar Psilander, Gunnar Helse...	Urban Gad	Urban Gad, Gebhard Schätzler-Perasini	Fotorama	Drama	1.0	0.0
3	Cleopatra	The fabled queen of Egypt's affair with Roman ...	Helen Gardner, Pearl Sindelar, Miss Fielding, ...	Charles L. Gaskill	Victorien Sardou	Helen Gardner Picture Players	Drama, History	1.0	0.0
4	L'Inferno	Loosely adapted from Dante's Divine Comedy and...	Salvatore Papa, Arturo Pirovano, Giuseppe de L...	Francesco Bertolini, Adolfo Padovan	Dante Alighieri	Milano Film	Adventure, Drama, Fantasy	1.0	0.0
...	...	...	...	...	...	...	...	...	...
85847	Ottam	Set in Trivandrum, the story of Ottam unfolds ...	Nandu Anand, Roshan Ullas, Manikandan R. Achar...	Zam	Rajesh k Narayan	Thomas Thiruvalla Films	Drama	1.0	0.0
85848	Pengalila	An unusual bond between a sixty year old Dalit...	Lal, Akshara Kishor, Iniya, Narain, Renji Pani...	T.V. Chandran	T.V. Chandran	Benzy Productions	Drama	1.0	0.0
85850	Le lion	A psychiatric hospital patient pretends to be ...	Dany Boon, Philippe Katerine, Anne Serra, Samu...	Ludovic Colbeau-Justin	Alexandre Coquelle, Matthieu Le Naour	Monkey Pack Films	Comedy	0.0	1.0
85851	De Beentjes van Sint-Hildegard	A middle-aged veterinary surgeon believes his ...	Herman Finkers, Johanna ter Steege, Leonie ter...	Johan Nijenhuis	Radek Bajgar, Herman Finkers	Johan Nijenhuis & Co	Comedy, Drama	1.0	1.0
85854	La vida sense la Sara Amat	Pep, a 13-year-old boy, is in love with a girl...	Maria Morera Colomer, Biel Rossell Pelfort, Is...	Laura Jou	Coral Cruz, Pep Puig	La Xarxa de Comunicació Local	Drama	1.0	0.0

78384 rows × 9 columns

Жанр драма

[37]:
```
from sklearn.model_selection import train_test_split
X = movies['description'] + ' ' + movies['actors'] + ' ' +␣
 ↪movies['original_title'] + ' ' + movies['director'] + ' ' +␣
 ↪movies['writer'] + ' ' + movies['production_company']
X_train, X_test, y_train, y_test = train_test_split(X, movies['drama'],
                                                    random_state=42,␣
 ↪shuffle=True, test_size=0.25)
X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)

print(f'пример текста\n{X_train[0]}')
print(f'\nразмеры выборок: {len(X_train), len(X_test)}')
```

```
пример текста
Kyu-sik is a seminary student, who one day falls during a church service,
dropping a precious, pope-blessed relic. As punishment, he and his comic␣
 ↪relief
```

```
friend Seon-dal are sent away to a ... Sang-Woo Kwon, Ji-Won Ha, In-kwon Kim,
In-mun Kim, Seon-hwa Kim, Jae-Hyun Cho, Hye-jin Jeon, Hee-soo Kim, Hye-na Kim
Shinbu sueob In-mu Heo In-mu Heo, Eun-kyeong Yun Kihwik Cine

размеры выборок: (58788, 19596)
```

Определение функций

[6]:
```
tf_idf = TfidfVectorizer(max_df=0.8, min_df=10, stop_words='english')
```

Функция, считающая качество обучения на кросс валидации.

[7]:
```
def cv_score(X, y, parameters, folds, knn_class):
    ans = []
    scaler = TfidfVectorizer(max_df=0.8, min_df=10, stop_words='english')
    for sc in parameters['scores']:
        ans.append({})
        for n in parameters['n_neighbors']:
            for w in parameters['weights']:
                neigh = knn_class(n_neighbors=n, metric='cosine',␣
 ↪weights=w[1])
                i = parameters['scores'].index(sc)
                ans[i][(n, w[0])] = 0
                for f in folds:
                    scaler.fit(X[f[0]])
                    X_train = scaler.transform(X[f[0]])
                    X_test = scaler.transform(X[f[1]])
                    neigh.fit(X_train, y[f[0]])
                    ans[i][(n, w[0])] += sc(y[f[1]], neigh.predict(X_test))
                ans[i][(n, w[0])] /= len(folds)
    return ans
```

Функция, разбивающая обучающую выборку на блоки (folds) для кросс валидации

[8]:
```
def fold_split(num_objects, num_folds):
    fold_size = num_objects // num_folds
    flag = num_objects % num_folds
    ans = list()
```

```
    for i in range(num_folds):
        x = np.arange(num_objects)
        mask1 = (x >= (i + 1) * fold_size) | (x < i * fold_size)
        mask2 = (x < (i + 1) * fold_size) & (x >= i * fold_size)
        if flag and i == num_folds - 1:
            ans.append((x[x < i * fold_size], x[x >= i * fold_size]))
            break
        ans.append((x[mask1], x[mask2]))
    return ans
```

реализация линейно и экспоненциально убывающих весов

[23]:
```
def linear(distances: np.array)->np.array:
    weights: np.array = np.array(np.full(distances.shape, 0), dtype='float')
    n = distances.shape[1]
    weights[:] = np.arange(1.0, 0, -1 / n)
    return weights

def my_exp(distances: np.array)->np.array:
    weights: np.array = np.array(np.full(distances.shape, 0), dtype='float')
    n = distances.shape[1]
    weights[:] = np.geomspace(1, (1/2) ** (n - 1), n)
    return weights
```

Выбор веса (этап 0)

Первый набор параметров

[11]:
```
par0 = {
    'n_neighbors': [i for i in range(1, 11)],
    'weights': [('uniform', 'uniform'), ('linear', linear), ('my_exp',␣
 ↪my_exp), ('distance', 'distance')],
    'scores': [recall_score]
}
```

[12]:
```
folds0 = fold_split(len(y_train), 4)
score_train0 = cv_score(X_train, y_train, par0, folds0, neighbors.
 ↪KNeighborsClassifier)
```

Посчитаем, где в среднем качество лучше

[15]:
```
n = 10
score_weights = {'uniform': 0, 'linear': 0, 'my_exp': 0, 'distance': 0}
for i in score_train0[0].keys():
    score_weights[i[1]] += score_train0[0][i] / n
score_weights
```

[15]:
```
{'uniform': 0.6434384348115793,
 'linear': 0.6953085178688904,
 'my_exp': 0.6605275499731177,
 'distance': 0.7065966666916362}
```

качество в среднем лучше на весах по расстоянию

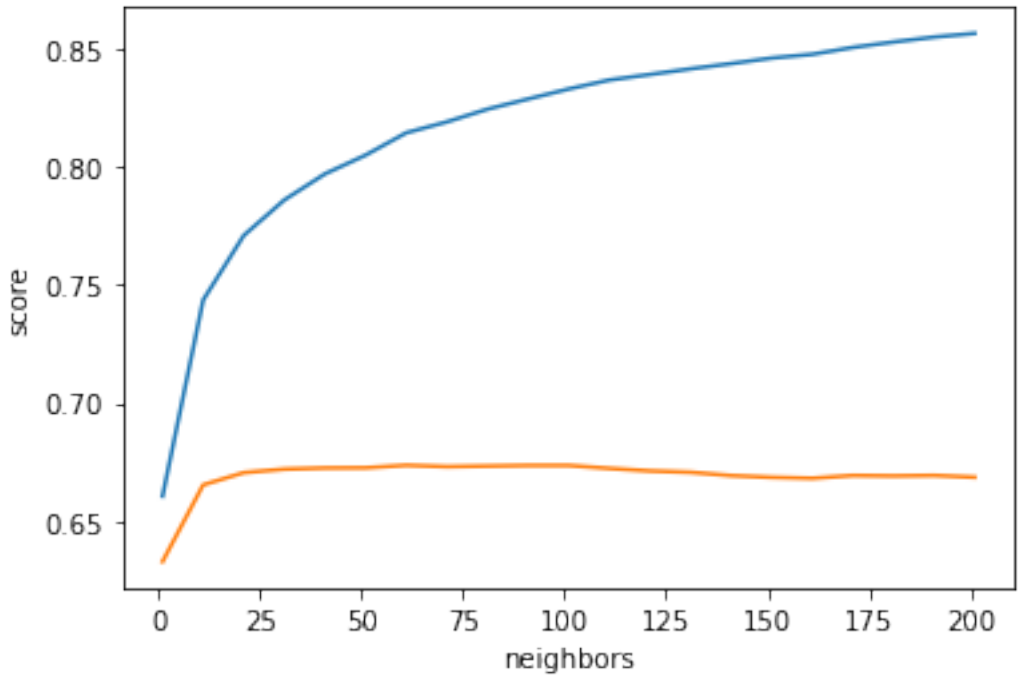
Выбор количества соседей (этап 1)

[21]:
```
tf_idf = TfidfVectorizer(max_df=0.8, min_df=10, stop_words='english')
par1 = {
    'n_neighbors': [i for i in range(1, 200, 10)],
    'weights': [('distance', 'distance')],
    'scores': [recall_score, accuracy_score]
}
```

[22]:
```
folds1 = fold_split(len(y_train), 4)
score_train1 = cv_score(X_train, y_train, par1, folds1, neighbors.
 ↪KNeighborsClassifier)
```

[23]:
```
plt.plot(par1['n_neighbors'], list(score_train1[0].values()))
plt.plot(par1['n_neighbors'], list(score_train1[1].values()))
plt.xlabel("neighbors")
plt.ylabel("score")
```

[23]:
```
Text(0, 0.5, 'score')
```



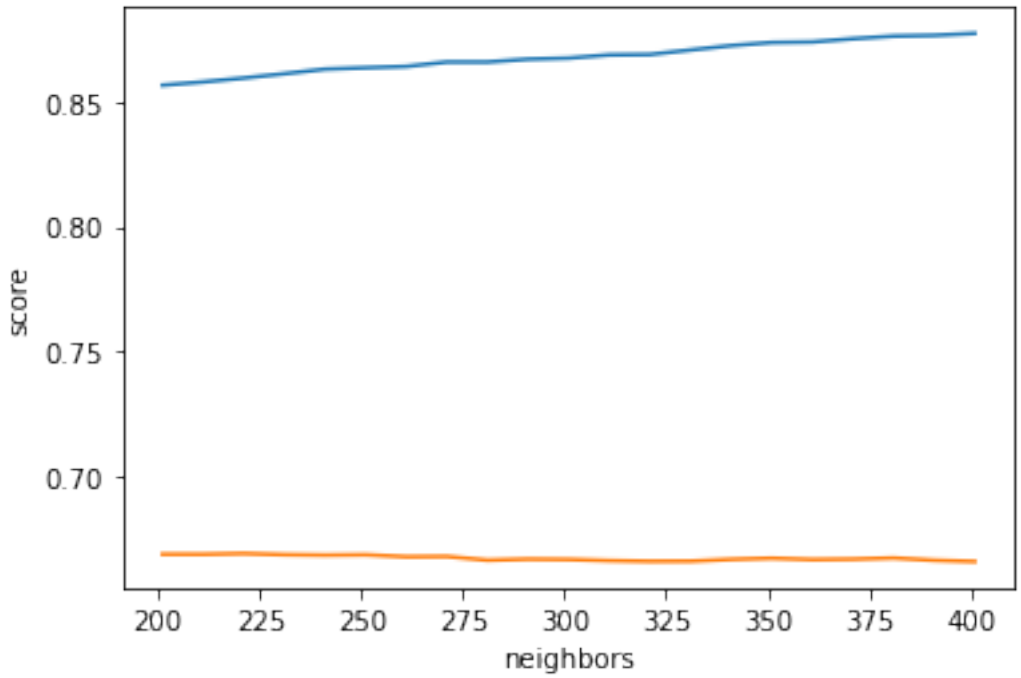
посмотрим на всякий случай изменение качества и на дальше: на 200-400 соседях

[16]:
```
tf_idf = TfidfVectorizer(max_df=0.8, min_df=10, stop_words='english')
par1_2 = {
    'n_neighbors': [i for i in range(201, 402, 10)],
    'weights': [('distance', 'distance')],
    'scores': [recall_score, accuracy_score]
}
```

[17]:
```
folds1 = fold_split(len(y_train), 4)
score_train1_2 = cv_score(X_train, y_train, par1_2, folds1, neighbors.
 ↪KNeighborsClassifier)
```

[18]:
```
plt.plot(par1_2['n_neighbors'], list(score_train1_2[0].values()))
plt.plot(par1_2['n_neighbors'], list(score_train1_2[1].values()))
plt.xlabel("neighbors")
plt.ylabel("score")
```

[18]:
```
Text(0, 0.5, 'score')
```



recall немного растёт, accuracy не меняется

[25]:
```
for k, v in score_train1_2[1].items():
    if v == max(score_train1_2[1].values()):
        best_par = k
best_par
```

[25]:
```
(221, 'distance')
```

Дополнительно

Посмотрим как качестов менялось на ещё большем числе соседей

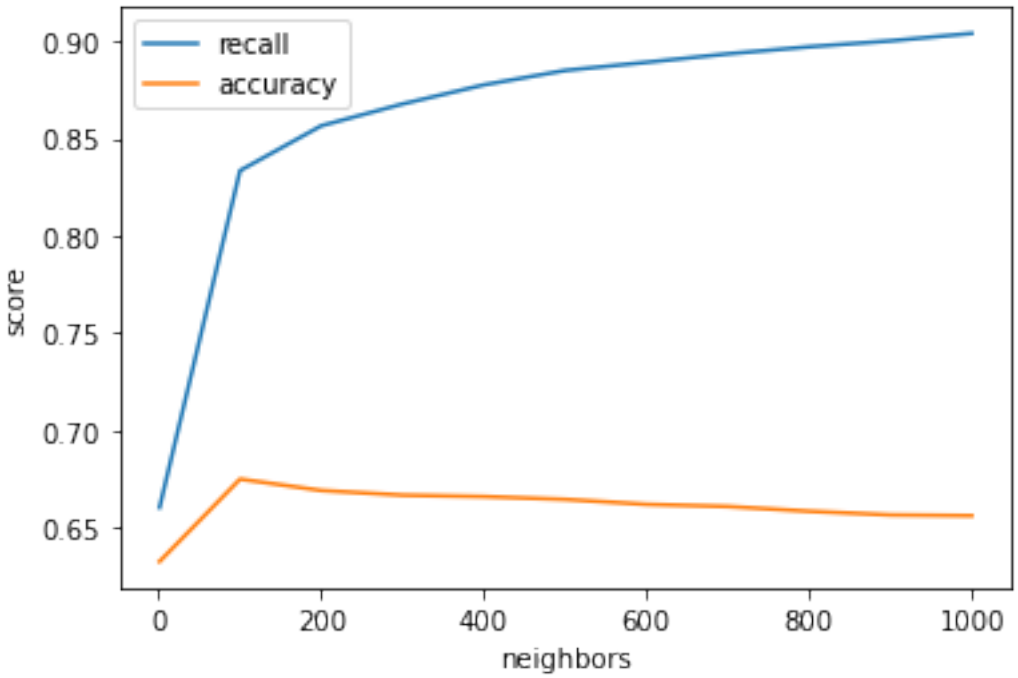
[ ]:
```
tf_idf = TfidfVectorizer(max_df=0.8, min_df=10, stop_words='english')
par1_3 = {
    'n_neighbors': [i for i in range(1, 1002, 100)],
    'weights': [('distance', 'distance')],
    'scores': [recall_score, accuracy_score]
}
```

[ ]:
```
```

```
folds1 = fold_split(len(y_train), 4)
score_train1_3 = cv_score(X_train, y_train, par1_3, folds1, neighbors.
 ↪KNeighborsClassifier)
```

[13]:
```
plt.plot(par1_3['n_neighbors'], list(score_train1_3[0].values()),␣
 ↪label='recall')
plt.plot(par1_3['n_neighbors'], list(score_train1_3[1].values()),␣
 ↪label='accuracy')
plt.xlabel("neighbors")
plt.ylabel("score")
plt.legend()
```

[13]:
```
<matplotlib.legend.Legend at 0x13d881982b0>
```



Accuracy стал понемногу падать

Обучение на лучших параметрах

[39]:
```
neigh = neighbors.KNeighborsClassifier(n_neighbors=best_par[0],
                                       metric='cosine', weights=best_par[1])
scaler = TfidfVectorizer(max_df=0.8, min_df=10, stop_words='english')
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
neigh.fit(X_train, y_train)
score_test1_acc = accuracy_score(y_test, neigh.predict(X_test))
score_test1_rec = recall_score(y_test, neigh.predict(X_test))
score_test1_acc, score_test1_rec
```

[39]:
```
(0.6777913859971423, 0.8603077493511309)
```

Демонстрация на конкретных фильмах

Возьмём 20 фильмов с наибольшим числом отзывов и посмотрим сами результат

[40]:
```
mv = movies0.sort_values('votes')
mv = mv.tail(20)
demo_np = mv['description'] + ' ' + mv['actors'] + ' ' +␣
 ↪mv['original_title'] + ' ' + mv['director'] + ' ' + mv['writer'] + ' ' +␣
 ↪mv['production_company']
demo_np = np.array(demo_np)
```

[41]:
```
demo_train = scaler.transform(demo_np)
mv['drama'] = neigh.predict(demo_train)
mv[['original_title','genre', 'drama']]
```

	original_title	genre	drama
43404	Inglourious Basterds	Adventure, Drama, War	1.0
26257	The Silence of the Lambs	Crime, Drama, Thriller	0.0
50773	The Avengers	Action, Adventure, Sci-Fi	0.0
43935	Batman Begins	Action, Adventure	0.0
34440	Gladiator	Action, Adventure, Drama	1.0
62660	Django Unchained	Drama, Western	0.0
29287	Se7en	Crime, Drama, Mystery	1.0
50294	Interstellar	Adventure, Drama, Sci-Fi	1.0
34128	The Lord of the Rings: The Two Towers	Action, Adventure, Drama	1.0
57234	The Dark Knight Rises	Action, Adventure	0.0
15528	The Godfather	Crime, Drama	1.0
34127	The Lord of the Rings: The Return of the King	Action, Adventure, Drama	1.0
31279	The Lord of the Rings: The Fellowship of the Ring	Action, Adventure, Drama	1.0
32229	The Matrix	Action, Sci-Fi	1.0
28066	Forrest Gump	Drama, Romance	1.0
28381	Pulp Fiction	Crime, Drama	1.0
32487	Fight Club	Drama	1.0
57475	Inception	Action, Adventure, Sci-Fi	0.0
48078	The Dark Knight	Action, Crime, Drama	0.0
28453	The Shawshank Redemption	Drama	1.0

Ошибки есть, но в целом классификация верная

Жанр комедия

[28]:
```
from sklearn.model_selection import train_test_split
X = movies['description'] + ' ' + movies['actors'] + ' ' +␣
 ↪movies['original_title'] + ' ' + movies['director'] + ' ' +␣
 ↪movies['writer'] + ' ' + movies['production_company']
X_train, X_test, y_train, y_test = train_test_split(X, movies['comedy'],
                                                    random_state=123,␣
 ↪shuffle=True, test_size=0.25)
X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)

print(f'пример текста\n{X_train[0]}')
```

```
print(f'\nразмеры выборок: {len(X_train), len(X_test)}')
```

```
пример текста
While on holiday in Rhodes, Athenian war hero Darios becomes involved in two
different plots to overthrow the tyrannical king, one from Rhodian patriots␣
 ↪and
the other from sinister Phoenician agents. Rory Calhoun, Lea Massari, Georges
Marchal, Conrado San Martín, Ángel Aranda, Mabel Karr, Mimmo Palmara, Roberto
Camardiel, Alfio Caltabiano, George Rigaud, Yann Larvor, Carlo Tamberlani,␣
 ↪Félix
Fernández, Ignazio Dolce, Antonio Casas Il colosso di Rodi Sergio Leone Ennio␣
 ↪De
Concini, Sergio Leone Cine-Produzioni Associate

размеры выборок: (58788, 19596)
```

Выбор веса (этап 0)

[29]:
```
par0 = {
    'n_neighbors': [i for i in range(1, 11)],
    'weights': [('uniform', 'uniform'), ('linear', linear), ('my_exp',␣
 ↪my_exp), ('distance', 'distance')],
    'scores': [recall_score]
}
```

[ ]:
```
folds0 = fold_split(len(y_train), 4)
score_train0 = cv_score(X_train, y_train, par0, folds0, neighbors.
 ↪KNeighborsClassifier)
```

[19]:
```
n = 10
score_weights = {'uniform': 0, 'linear': 0, 'my_exp': 0, 'distance': 0}
for i in score_train0[0].keys():
    score_weights[i[1]] += score_train0[0][i] / n
score_weights
```

[19]:
```
{'uniform': 0.4381890393976558,
 'linear': 0.5140844925609015,
 'my_exp': 0.5629390075229098,
```

```
 'distance': 0.5093544166360856}
```

Экспоненциально взвешенные веса оказались лучше, но их качество не улучшается с ростом числа соседей.

Произведём обучение на большем числе соседей, взвешивая по расстоянию (предположительно это улучшит качество)

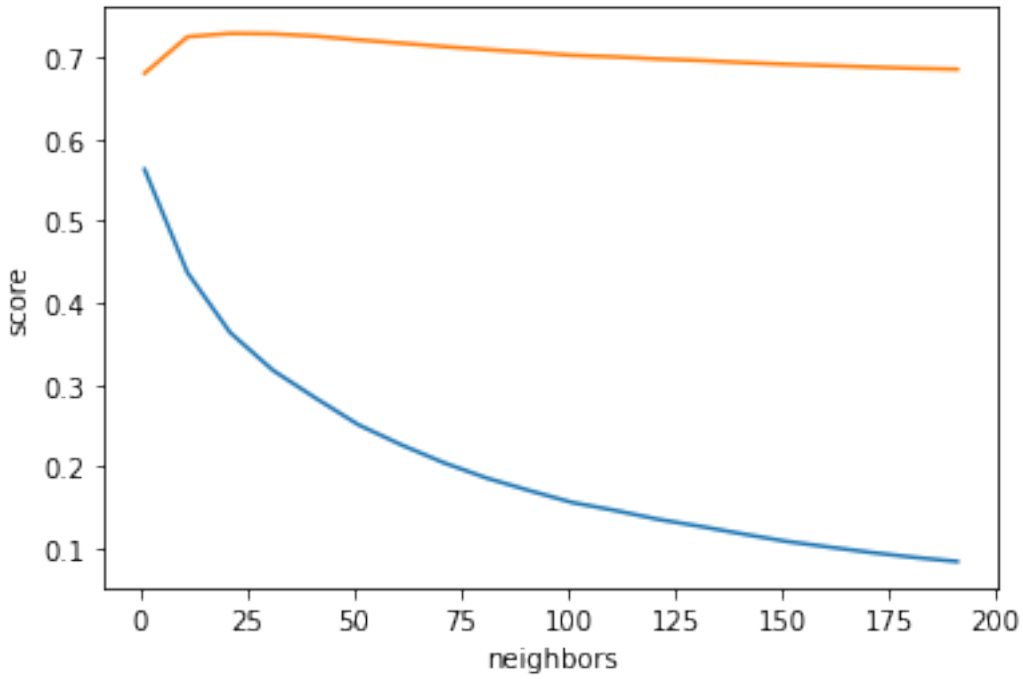
Выбор количества соседей (этап 1)

[13]:
```
tf_idf = TfidfVectorizer(max_df=0.8, min_df=10, stop_words='english')
par1 = {
    'n_neighbors': [i for i in range(1, 200, 10)],
    'weights': [('distance', 'distance')],
    'scores': [recall_score, accuracy_score]
}
```

[17]:
```
folds1 = fold_split(len(y_train), 4)
score_train1 = cv_score(X_train, y_train, par1, folds1, neighbors.
 ↪KNeighborsClassifier)
```

[18]:
```
plt.plot(par1['n_neighbors'], list(score_train1[0].values()))
plt.plot(par1['n_neighbors'], list(score_train1[1].values()))
plt.xlabel("neighbors")
plt.ylabel("score")
```

[18]:
```
Text(0, 0.5, 'score')
```



Предположение не оправдалось. Увеличение числа соседей при взвешивании по расстоянию только снизило качество, остановимся на экспоненциальных в случае комедии.

Обучение на лучших параметрах

[25]:
```
neigh = neighbors.KNeighborsClassifier(n_neighbors=5,
                                       metric='cosine', weights=my_exp)
scaler = TfidfVectorizer(max_df=0.8, min_df=10, stop_words='english')
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
neigh.fit(X_train, y_train)
score_test1_acc = accuracy_score(y_test, neigh.predict(X_test))
score_test1_rec = recall_score(y_test, neigh.predict(X_test))
score_test1_acc, score_test1_rec
```

[25]:
```
(0.6887119820371504, 0.5811213838353713)
```

[33]:
```
movies['drama'].value_counts()
```

[33]:
```
1.0    43326
0.0    35058
Name: drama, dtype: int64
```

[34]:
```
movies['comedy'].value_counts()
```

[34]:
```
0.0    51877
1.0    26507
Name: comedy, dtype: int64
```

Класс комедийных фильмов в два раза меньше некомедийных. Это могло повлечь снижение качества recall

Список литературы

[1] Trevor Hastie, Robert Tibshirani, Jerome Friedman. "The elements of statistical learning: data mining, inference and prediction"— New York: Springer, 2017

[2] Журавлев Ю. И., Рязанов В. В., Сенько О. В. «Распознавание». Математические методы. Программная система. Практические применения. — М.: Фазис, 2006.

[3] Курс лекций по машинному обучению К.В. Воронцов http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_%28курс_лекций%2C_К.В.Воронцов%29