Большие данные и машинное обучение

Второй отборочный этап

Задача IV.1. Активация функции активации (100 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Владислав сильно увлёкся тематикой нейронных сетей и даже прочитал 33 главы из учебника по нейронным сетям, но так и не понял, как же они все-таки обучаются. Он решил подойти к проблеме логически и вспомнил, что в 31-й главе было сказано, что любую логическую функцию можно интерполировать нейронной сетью.

Владислав приготовил логическую функцию f, которая задана таблицей истинности. Помогите ему построить нейронную сеть, которая будет интерполировать её.

Формат входных данных

Первая строка содержит натуральное число K ($1 \leqslant K \leqslant 10$) — число входов (аргументов) f. Следующие 2^K строк содержат значения f из таблицы истинности (0 — ложь, 1 — истина). Для уменьшения размера ввода значения аргументов f были опущены, так как их можно восстановить. Для этого условимся, что строки в таблице заданы в возрастающем порядке, где приоритеты аргументов при сравнении также возрастают от первого к последнему.

Например, для $K = 1, 2, 3$ значения функции f будут заданы в следующем виде.

$K = 1$	$K = 2$	$K = 3$
$f(0)$	$f(0,0)$	$f(0,0,0)$
$f(1)$	$f(1,0)$	$f(1,0,0)$
	$f(0,1)$	$f(0,1,0)$
	$f(1,1)$	$f(1,1,0)$
		$f(0,0,1)$
		$f(1,0,1)$
		$f(0,1,1)$
		$f(1,1,1)$

Формат выходных данных

В первой строке выведите целое положительное число D ($1 \leqslant D \leqslant 2$) — число слоёв (преобразований) в вашей сети.

На следующей строке выведите D целых положительных чисел n_i ($1 \leqslant n_i \leqslant 512$ и $n_D = 1$) — число искусственных нейронов на i-м слое. Предполагается, что $n_0 = M$.

Далее выведите описание D слоёв. i-й слой описывается n_i строками, описанием соответствующих искусственных нейронов на i-м слое. Каждый искусственный нейрон описывается строкой, состоящей из n_{i-1} вещественных чисел с плавающей точкой w_j и одного вещественного числа b — описание линейной зависимости текущего нейрона от выходов предыдущего i-го слоя. Линейная зависимость задается по формуле: $Y = \sum w_j \cdot x_j + b$. Предполагается, что после каждого вычисления линейной зависимости к её результату применяется функция ступенчатой активации $a(Y) = \begin{cases} 1 & Y > 0 \\ 0 & Y < 0 \end{cases}$. Обратите внимание, что в нуле данная функция не определена, и если в ходе вычисления вашей сети будет вызвана активация от нуля, вы получите ошибку.

Примеры

Пример №1

Стандартный ввод
2 0 1 0 1

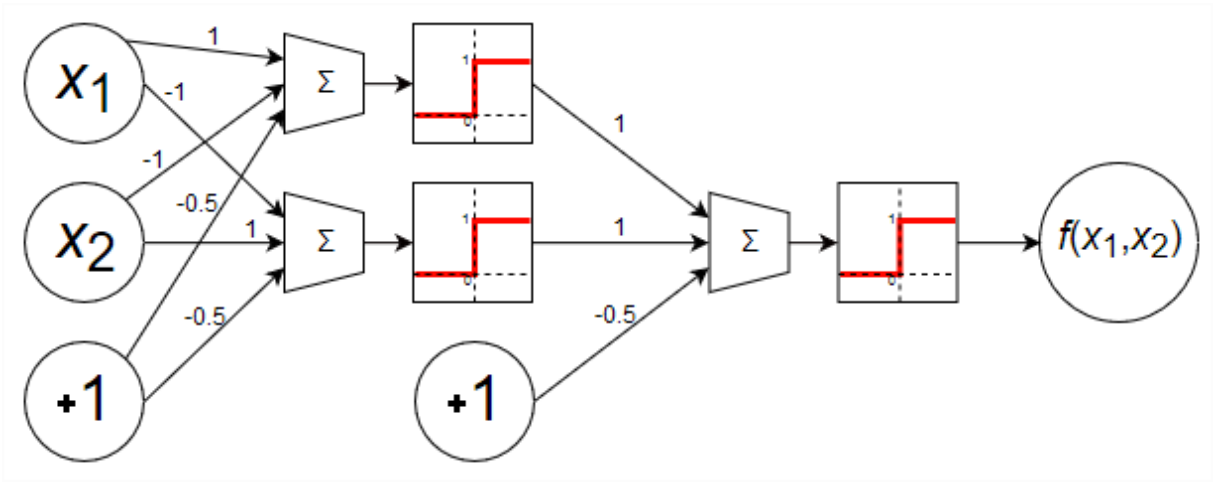
Стандартный вывод
2 2 1 1.0 -1.0 -0.5 1.0 1.0 -1.5 1 1 -0.5

Пример №2

Стандартный ввод
2 0 1 1 0

Стандартный вывод
2 2 1 1.0 -1.0 -0.5 -1.0 1.0 -0.5 1 1 -0.5

Примечание: во втором примере в результате получается следующая сеть.



Решение

Представим логическую функцию по таблице истинности в дизъюнктивной нормальной форме при помощи стандартного алгоритма: построим дизъюнкцию конъюнкций комбинаций аргументов, при которых результат функции должен быть равен истине. Далее превратим полученную функцию в нейронную сеть.

Каждая конъюнкция превратится в нейрон на промежуточном слое. Чтобы полученная сеть в итоге состояла из двух слоёв, нейрон должен также учитывать отрицание некоторых аргументов. Для этого соответствующие коэффициенты, на которые будет произведено умножение, должны быть равны минус единице, а остальные плюс единице. Свободный коэффициент должен быть равен минус числу аргументов без отрицания с небольшим сдвигом, чтобы нейрон активировался, только когда истине равны только нужные аргументы. Сдвиг нужен, чтобы сумма произведений не равнялась нулю.

Дизъюнкция превратится в нейрон на выходном слое, все коэффициенты которого равны единице. Свободный коэффициент должен быть равен небольшому отрицательному числу по указанным ранее причинам. Если в таблице истинности будет больше 512 значений истина, то на промежуточном слое будет больше 512 нейронов, что не допустимо по условию задачи. В таком случае требуется строить функции по отрицанию таблицы истинности, а затем применить отрицание к полученной дизъюнкции. Для этого можно инвертировать знаки у всех коэффициентов соответствующего нейрона.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
k = int(input())

f = [0] * (2 ** k)
nums = [[]] * (2 ** k)

for i in range(2 ** k):
    f[i] = int(input())
    b = bin(i)[2:]
    b = '0' * (k - len(b)) + b
```

```
    b = list(b[::-1])
    b = list(map(lambda x: x if x == '1' else '-1', b))
    nums[i] = b

if f.count(1) <= 512:
    obj = 1
else:
    obj = 0

neurons = []
for i in range(len(f)):
    if f[i] == obj:
        neurons.append(nums[i])

if f.count(1) == len(f) or f.count(0) == len(f):
    print(1)
    print(1)
    print('0 ' * k, end='')
    if f.count(1) == len(f):
        print('1')
    else:
        print('-1')
else:
    print(2)
    print(len(neurons), 1)
    if obj:
        for neuron in neurons:
            c = neuron.count('1')
            print(' '.join(neuron), str((c * -1) + 0.5))
        print('1 ' * len(neurons) + '-0.5')
    else:
        for neuron in neurons:
            c = neuron.count('1')
            print(' '.join(neuron), str((c * -1) + 0.5))
        print('-1 ' * len(neurons) + str(len(neurons) - 0.5))
```

Задача IV.2. Категоричный профессор (100 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Однажды один профессор дал своему ученику задание, вычислить коэффициент корреляции Пирсона между двумя признаками a и b на множестве объектов. Ученик тут же вычислил результат и показал его профессору. «Неправильно, — возразил профессор, — это задание с подвохом. Первый признак a на самом деле категориальный. Поэтому сперва требуется сделать one-hot-преобразование, а затем уже вычислить среднее взвешенное корреляций между b и новыми признаками, которые были получены после преобразования a».

Вам необходимо выполнить задание профессора.

Формат входных данных

Первая строка содержит два натуральных числа N и K, разделённых пробелами: N ($1 \leqslant N \leqslant 10^5$) — число объектов, K ($1 \leqslant K \leqslant 10^5$) — число значений категории первого признака.

Вторая строка содержит N натуральных чисел, разделённых пробелами: i-е из них a_i ($1 \leqslant a_i \leqslant K$) — значение первого признака i-го объекта.

Третья строка содержит N целых чисел, разделённых пробелами: i-е из них b_i ($|b_i| \leqslant 10^9$) — значение второго признака i-го объекта.

Формат выходных данных

Выведите одно вещественное число с плавающей точкой — коэффициент корреляции Пирсона между a и b. Абсолютная или относительная погрешность ответа не должна превышать 10^{-9}.

Примеры

Пример №1

Стандартный ввод
6 3 1 2 2 3 3 3 1 2 3 4 5 6

Стандартный вывод
0.19203297584037293

Примечание: в примере значение корреляции между первым новым признаком $(1, 0, 0, 0, 0, 0)$ и b равно $-0,654653671$, а его вес равен единице, так как соответствующие значение встретилось только один раз. Значение корреляции между вторым новым признаком $(0, 1, 1, 0, 0, 0)$ и b равно $-0,414039336$, а его вес равен двум. Значение корреляции между третьим новым признаком $(0, 0, 0, 1, 1, 1)$ и b равно $0,878310066$, а его вес равен трём.

Решение

Пусть A_c — это c-я колонка после one-hot преобразование признака A. Тогда по условию задачи требуется вычислить:

$$\sum_{c=1}^{K} \frac{w_c}{N} \frac{\operatorname{cov}(A_c, B)}{\sqrt{\mathbb{D}[A_c]\,\mathbb{D}[B]}}.$$

Вычисление этой формулы в явном виде потребует $\mathcal{O}(KN)$ операций, что не укладывается в заданные ограничение. Для получения эффективного решения требуется вычислять каждое слагаемое этой суммы за $\mathcal{O}(1)$.

Заметим, что $\frac{w_c}{N} = \mathbb{E}[A_c]$ и его можно вычислить за $\mathcal{O}(1)$, если заранее предподсчитать для каждого значения c сколько раз оно встретилось. Это можно сделать эффективно один раз пройдясь по всем объектам за $\mathcal{O}(N)$.

Дисперсия $\mathbb{D}[B]$ вычисляется за $\mathcal{O}(N)$, но она не зависит от c, поэтому её достаточно вычислить только один раз. Теперь научимся вычислять $\mathbb{D}[A_c]$ за $\mathcal{O}(1)$:

$$\mathbb{D}[A_c] = \mathbb{E}\left[(A_c - \mathbb{E}[A_c])^2\right] = \frac{1}{N}\sum_{i=1}^{N}(A_{c,i} - \mathbb{E}[A_c])^2 =$$

$$= \frac{1}{N}\left(w_c\,(1 - \mathbb{E}[A_c])^2 + (N - w_c)\,(0 - \mathbb{E}[A_c])^2\right).$$

Последнее равенство получено из факта, что $A_{c,i}$ всегда равно единице или нулю, а число раз встречи нуля равно N минус числу раз встречи единицы.

$\mathrm{cov}(A_c, B)$ можно быстро вычислить примерно тем же способом:

$$\mathrm{cov}(A_c, B) = \frac{1}{N}\sum_{i=1}^{N}[A_i = c]\cdot B_i.$$

Эту сумму можно вычислить за $\mathcal{O}(N)$ сразу для всех значений c, так как сумма числа встреч каждого значения равна числу объектов.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
from collections import Counter

n, k = [int(i) for i in input().split()]
a = [int(i) for i in input().split()]
b = [int(i) for i in input().split()]

sb = sum(b)
c = Counter(a)
mean_a = [c[i + 1] / n for i in range(k)]
mean_b = sum(b) / n
sp = [0] * k
for i in range(n):
        sp[a[i] - 1] += b[i] - mean_b

sk = 0
for i in range(n):
        sk += (b[i] - mean_b) ** 2
gp = [sk] * k

for i in range(k):
        gp[i] *= c[i + 1] * (1 - c[i + 1] / n)
        gp[i] **= 0.5


s = 0
for i in range(k):
        if gp[i] > 0:
                s += sp[i] / gp[i] * c[i + 1]

print(s / n)
```

Задача IV.3. Большие данные (100 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Недавно Борис изучил модель линейной регрессии на занятиях по статистике. Придя домой, он решил применить свои знания на практике. Борис обнаружил, что набор данных слишком большой для ручного вычисления. Тогда он решил написать программу, которая будет решать задачу при помощи стохастического градиентного спуска с гребневой регуляризацией.

Изначально программа стартует с нулевых коэффициентов линейной регрессии a_j. Затем на каждой итерации:

1. Выбирается номер i очередного объекта x_i, и вычисляется предсказание $\hat{y}_i$ для этого объекта по формуле: $\hat{y}_i = \sum_j x_{i,j} \cdot a_j$.
2. Далее вычисляется производная функции потери суммы квадратов, которая совпадает с ошибкой: $\delta_i = \hat{y}_i - y_i$.
3. Эта производная используется для вычисления новых коэффициентов линейной регрессии: $a_j^{\text{new}} = a_j - \lambda\,(x_{i,j} \cdot \delta_i + \tau \cdot a_j)$, где λ — скорость градиентного спуска, а τ — коэффициент регуляризации.

К сожалению, у Бориса очень слабый компьютер, поэтому ему требуется определить подмножество объектов и признаков, на которых следует обучать алгоритм, а также гиперпараметры λ и τ.

Формат входных данных

Набор данных доступен по ссылке: https://disk.yandex.ru/d/Ehm4Xp2F44ffoQ.

Формат выходных данных

Первая строка решения должна содержать два натуральных числа n и m, разделённых пробелом: n ($1 \leqslant n \leqslant 2500$) — число рассматриваемых объектов для обучения, а m ($1 \leqslant m \leqslant 50$) — число рассматриваемых признаков для обучения.

Следующая строка должна содержать n номеров объектов из тренировочного набора данных в соответствующем порядке рассмотрения в стохастическом градиентном спуске. Объекты нумеруются с единицы.

Следующая строка должна содержать m номеров признаков из тренировочного набора данных, которые будут использоваться для обучения. Признаки нумеруются с единицы.

Последняя строка должна содержать два неотрицательных числа с плавающей точкой, разделённых пробелом: гиперпараметры λ и τ.

Критерии оценивания

Для оценки решения используется коэффициент детерминации.

Решение

Точного решения задача не имеет. Данную задачу можно свести к задаче оптимизации в пространстве бинарных векторов. Помимо этого можно использовать методы выбора признаков и поиска аномалий для нахождения соответствующих подмножеств. Для выбора аномалий можно отсеять часть объектов, на которых модель ошибается больше всего. Так как известно, что используется линейная модель, для выбора признаков можно использовать коэффициент корреляции.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import pandas as pd
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
import numpy as np
from sklearn.model_selection import train_test_split

data = pd.read_csv("train.csv")

data = data.drop(columns = ["f1"])
corr = data.corr()
corr_target = abs(corr["target"])
cols = corr_target.sort_values().tail(51)
data = data[cols.index]

data_train, data_test = train_test_split(data, test_size=0.1)

y_train = data_train["target"]
X_train= data_train.drop(columns=["target"])

y_test = data_test["target"]
X_test= data_test.drop(columns=["target"])

def error_func(y_pred, y):
    return abs(y_pred - y)

class SGD:
  def __init__(self, lr, r, in_d, limit = 10e6):
    self.good_X = []
    self.bad_X = []
    self.lr = lr
    self.r = r
    self.limit = limit
    self.W = np.random.randn(in_d) / np.sqrt(in_d)

  def pred(self, X):
    return sum(self.W * X)

  def optimize(self, X, y):
    loss = (y - self.pred(X))**2
```

```
    if loss < self.limit:
      self.W -= self.lr*(X*(self.pred(X) - y) + self.r*self.W)
      self.good_X.append(X)
    else:
      self.bad_X.append(X)
    return loss

def train_model(data, target, model):
  return [model.optimize(data.loc[i], target.loc[i]) for i in data.index]

model = SGD(lr = 0.0000001, r=0.15, in_d = 50, limit=10e5)
losses = train_model(X_train, y_train, model)
train_pred = [model.pred(X_train.loc[i]) for i in X_train.index]
test_pred = [model.pred(X_test.loc[i]) for i in X_test.index]

good_i = [t.name for t in model.good_X]
new_set = X_train.loc[good_i][:2500]
new_target = y_train.loc[good_i][:2500]

f = open("ans.txt", "w")
f.write(str(min(len(new_set), 2500)) + " 50")
f.write("\n")
f.write(" ".join(map(str, new_set.index[:2500] + 1)))
f.write("\n")
f.write(" ".join(X_train.columns).replace("f", ""))
f.write("\n")
# learning rate подобран при длительном исследовании (мы уменьшали его каждый раз,
↪  когда модель плохо обучалась), reg базовый для sklearn
f.write("0.0000001 0.15")
f.close()
```

Задача IV.4. Машинное обучение (100 баллов)

Имя входного файла: стандартный ввод.

Имя выходного файла: стандартный вывод.

Ограничение по времени выполнения программы: 1 с.

Ограничение по памяти: 256 Мбайт.

Условие

Обучите модель на тренировочной части и предскажите значение целевого признака для тестовой.

Формат входных данных

Набор данных доступен по ссылке: https://disk.yandex.ru/d/B2dQ_ChqAHN5lw. Он содержит два `csv` файла: `train` — тренировочная часть, а `test` — тестовая. Признак `target` — является целевым признаком.

Формат выходных данных

Предсказание для тестовой части необходимо загрузить в тестирующую систему. Файл с ответами должен содержать ровно 1000 строк. Каждая строка должна

содержать ровно одну заглавную латинскую букву от A до H. Порядок объектов должен соответствовать порядку объектов из тестового множества. Пример файла `sample_submission` можно найти в архиве с набором данных.

Критерии оценивания

Для оценки решения используется F-мера.

Решение

Точного решения задача не имеет. Заметим, что часть признаков в тестовом наборе данных целиком отсутствует. Такие признаки называются привилегированной информацией. Для решения задачи можно построить две модели. Первая будет предсказывать привилегированную информацию из обычной, а вторая — метку класса из привилегированной.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
import catboost
from sklearn.model_selection import train_test_split
from catboost import CatBoostClassifier
from sklearn.metrics import f1_score
import pandas as pd

train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")

drop = ["f3", "f21", "f13", "f8", "f10", "f12"]
X, y = train_df.drop(["target", *drop], axis=1).fillna(-1), train_df["target"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.01,
↪  random_state=42)
cat_cols = ["f1", "f7", "f9", "f14", "f11"]
BEST = ['f1', 'f2', 'f5', 'f6', 'f7', 'f11', 'f15', 'f17', 'f18', 'f19']

cat_cols = [i for i in cat_cols if i in BEST]

params = {'n_estimators' : 1000,
          'learning_rate': .03,
          'max_depth' : 3,
          "cat_features": cat_cols,
          'auto_class_weights' : 'Balanced',
          'random_state': 42,
          "eval_metric": "TotalF1",
         }

model = CatBoostClassifier(**params)
model.fit(X_train[BEST], y_train, verbose=100, eval_set=(X_test[BEST], y_test))
f1_score(y_test, model.predict(X_test[BEST]), average="weighted")

temp = test_df.fillna(-1)
temp[cat_cols] = temp[cat_cols].astype(str)

pred = model.predict(temp[BEST])

with open("submit.txt", "w") as f:
    f.write("\n".join(pred.flatten()))
```