Дата последней сборки документа:
23 декабря 2021 г. в 13:45

Заметки по машинному обучению и анализу данных. Том 2

Подвойский А.О.

Здесь приводятся заметки по некоторым вопросам, касающимся машинного обучения, анализа данных, программирования на языках `Python`, `R` и прочим сопряженным вопросам так или иначе, затрагивающим работу с данными.

Краткое содержание

Содержание

1. Приемы работы с библиотеками Gym и Ecole

1.1. Gym

Функция окружения (environment) `step` возвращает четыре значения:

- `observation` (object): это объект, специфичный для окружающей среды и представляющий результат наблюдения за этой средой (например, состояние доски в настольной игре),
- `reward` (float): вознаграждение, полученное за предыдущее действие. Масштаб варьируется в зависимости от среды, но цель всегда в том, чтобы сделать суммарное вознаграждение как можно больше,

- `done` (boolean): флаг завершения эпизода. Многие (но не все) задачи разделены на четко определенные эпизоды, и `done = True` указывает на то, что эпизод завершился (например, мы потеряли последнюю жизнь в игре),
- `info` (dict): диагонстическая информация, полезная для отладки.

Это просто реализация классического цикла «агент – среда». На каждом шаге агент совершает то или иное действие и среда возвращает наблюдения (observation) и вознаграждение (reward).

Процесс запускается вызовом функции `reset()`, которая возвращает первое приближение `observation`.

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
env.close()
```

В этом примере мы отбирали случайные действия из пространства действий среды. Каждая среда поставляется с атрибутами `action_space` и `observation_space`. Эти атрибуты имеют тип `Space` и описывают формат допустимых действий и наблюдений

```
import gym

env = gym.make("CartPole-v0")
print(env.action_space)  # Discrete(2)

print(env.observation_space)  # Box([-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e
    +38], [4.8000002e+00 3.4028235e+38 4.1887903e-01 3.4028235e+38], (4,), float32)
```

Пространство `Descrete` описывает фиксированный диапазон неотрицательных чисел, так что в данном случае допустимыми действиями будет 0 или 1. Пространство `Box` представляет n-мерный ящик, так что в данном случае допустимыми наблюдениями будут 4-мерные массивы.

1.2. Ecole

Полезный ресурс о специальных приемах работы с задачами линейного программирования в частично-целочисленного постановке https://www.gams.com/37/docs/UG_LanguageFeatures.html?search=sos1

Полезный ресурс по математической оптимизации https://scipbook.readthedocs.io/en/latest/

1.2.1. Observations

Класс `ecole.observation.NodeBipartiteObs`: двудольный граф наблюдений для узлов branch-and-bound дерева. Оптимизационная задача представляется в виде гетерогенного двудольного графа. Между переменной и ограничением будет существовать ребро, если переменная присутствует в ограничении с ненулевым коэффициентом.

Метод `reset()` в Ecole принимает в качестве аргумента экземпляр проблемы.

2. Отбор признаков с библиотекой BoostARoota

BoostARoota https://github.com/chasedehan/BoostARoota – алгоритм отобора признаков на базе экстримального градиентного бустинга в реализации XGBoost. Алгоритм требует гораздо меньших затрат времени на выполнение. Перед применением необходимо выполнить дамми-кодирование, поскольку базовая модель работает только с количественными признаками.

Отбор признаков выполняется на обучающем поднаборе данных, поэтому предполагается, что массив меток и массив признаков *обучающие*, а для проверки качества модели отбора признаков есть независимая, *тестовая* выборка. Кроме того, если необходимо выбрать оптимальные значения гиперпараметров модели отбора признаков (например, значения гиперпараметров `cutoff`, `iters` и `delta`), то понадобиться еще *проверочная* выборка.

3. Площадь по ROC-кривой

Построение ROC-кривой происходит следующим образом (рис. 1):

1. Сначала сортируем все наблюдения по убыванию спрогнозированной вероятности положительного класса,
2. Берем единичный квадрат на координатной плоскости. Значения оси абсцисс будут значениями 1 - специфичности (цена деления оси задается значением 1/neg), а значения оси ординат будут значениями чувствительности (цена деления оси задается значением 1/pos). При этом pos — это количество наблюдений положительного класса, а neg — количество наблюдений отрицательного класса,
3. Задаем точку с координатами (0, 0) и для каждого отсортированного наблюдения x:
    - если x принадлежит положительному классу, двигаемся на 1/pos вверх,
    - если x принадлежит отрицательному классу, двигаемся на 1/neg вправо.

Значение вероятности положительного класса, при котором ROC-кривая находится на минимальном расстоянии от верхнего левого угла – точки с координатами (0, 1), дает наибольшую правильность классификации. В данному случае (рис. 2) будет 0.72.

Площадь под ROC-кривой (ROC-AUC) можно интерпретировать как вероятность события, состоящего в том, что классификатор присвоит более высокий ранг (например, вероятность) случайно выбранному экземпляру положительного класса, чем случайно выбранному экземпляру отрицательного класса (если не рассматривать вариант равенства значений рангов).

Замечание

На ROC-кривые не влияет баланс классов (при достаточном объеме выборки) и они могут чрезмерно оптимистично оценивать качество работы алгоритма в случае дисбалансов. Лучше пользоваться гармоническим средним или PR-кривыми

Однако недостаток такой интепретации заключается в том, что мы пренебрегаем часто встречающейся ситуацией равенства вероятностей. Поэтому правильнее будет сказать, что ROC-AUC равен доле пар вида (экземпляр положительного класса, экземпляр отрицательного класса), ко-

Спрогнозированные вероятности положительного класса, отсортированные по убыванию

№	фактический класс	спрогнозированная вероятность положительного класса
20	P	0,92
19	P	0,9
18	P	0,88
12	N	0,85
17	P	0,82
16	P	0,79
11	N	0,75
15	P	0,73
14	P	0,72
10	N	0,7
9	N	0,6
8	N	0,59
7	N	0,58
6	N	0,53
13	P	0,52
5	N	0,4
4	N	0,33
3	N	0,32
2	N	0,24
1	N	0,18

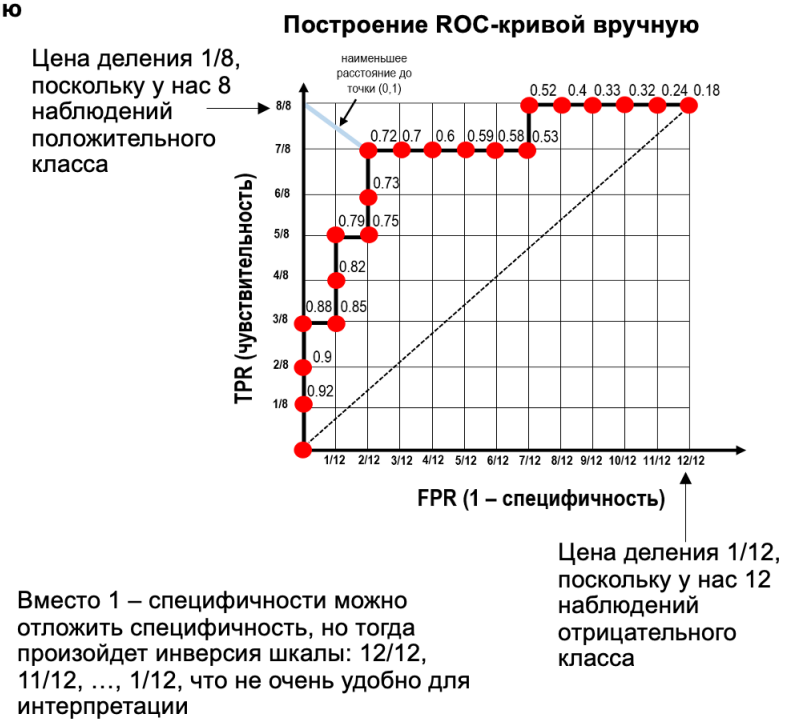

Рис. 1. Построение ROC-кривой

торые алгоритм верно упорядочил в соответствии с формулой

$$\frac{\sum\limits_{i,j=1}^{n_i,n_j} s(x_i,x_j)}{n_i\,n_j}, \quad s(x_i,x_j)=\begin{cases}1, x_i > x_j,\\ 1/2, x_i = x_j,\\ 0, x_i < x_j,\end{cases} \tag{1}$$

где x_i – ответ алгоритма для положительного экземпляра, x_j – ответ алгоритма для отрицательного экземпляра.

По сути числитель дроби представляет собой сумму количеств j-ых наблюдений отрицательного класса, лежащих ниже каждого i-ого наблюдения положительного класса. Каждое такое количество мы берем по каждому i-ому наблюдению положительного класса в последовательности, отсортированной по мере убывания вероятности положительного класса. Знаменатель дроби – это произведение количества наблюдений положительного класса и наблюдений отрицательного класса.

Если говорить более точно, мы берем наблюдение положительного класса под номером 20 и каждый раз образовываем пару с наблюдением отрицательного класса (3), у нас 12 пар, 12 раз наблюдение полжительного класса под номером 20 было проранжировано выше наблюдений отрицательного класса 12, 11, 10 и т.д. Записываем число 12 напротив наблюдения 20.

Разные модели нельзя сравнивать только по ROC-AUC. ROC-AUC оценивает разные классификатор, используя метрику, которая сама зависит от классификатора. То есть ROC-AUC оценивает разные классификаторы, используя разные метрики.

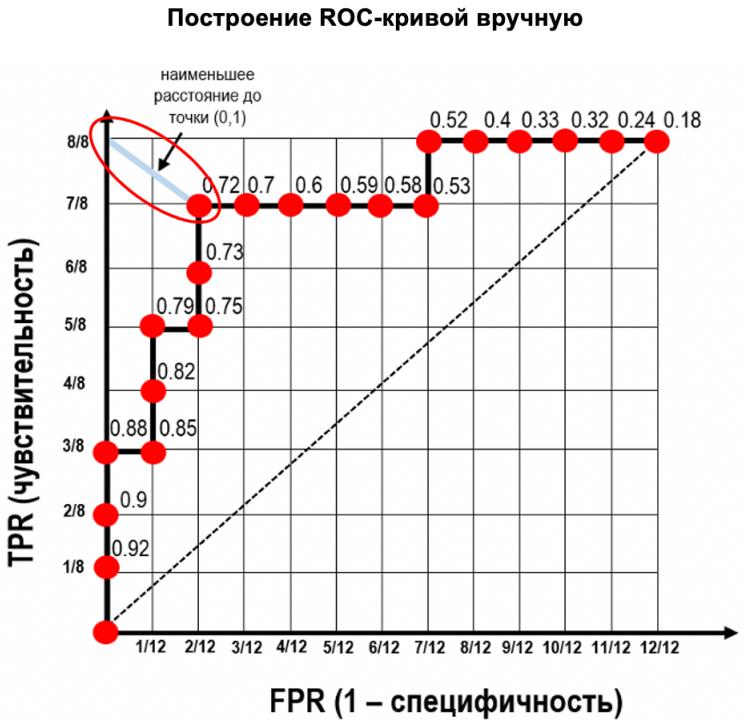

Рис. 2. ROC-кривая. Порог отсечения 0.72

Замечание

Если часть ROC-кривой лежит ниже диагональной линии, а часть – выше, то это означает, что классы не являются линейно-сепарабельными, а при этом используется линейная модель

При одинаковой ROC-AUC у разных моделей (соответственно с разными ROC-кривыми) будет разное распределение стоимостей ошибочной классификации. Проще говоря, мы можем вычислить ROC-AUC для классификатора A и получить 0.7, а затем вычислить ROC-AUC для второго классификатора и снова получить 0.7, но это не обязательно означает, что у них одна и та же эффективность.

Список иллюстраций

Список литературы

1. *Лутц М.* Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
2. *Бизли Д.* Python. Подробный справочник. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с.

Отсортированные спрогнозированные вероятности положительного класса

№	фактический класс	спрогно-зированная вероятность положитель-ного класса	скоринговое правило $S(x_i, x_j) = \begin{cases} 1, x_i > x_j, \\ \frac{1}{2}, x_i = x_j, \\ 0, x_i < x_j \end{cases}$	количество наблюдений отрицательного класса, лежащих ниже соответствующего наблюдения положительного класса	
20	P	0,92	0	12	Считаем количество отрицательных ниже каждого наблюдения положи-тельного класса
19	P	0,9	0	12	
18	P	0,88	0	12	
12	N	0,85	1		
17	P	0,82	0	11	
16	P	0,79	0	11	
11	N	0,75	1		
15	P	0,73	0	10	
14	P	0,72	0	10	
10	N	0,7	1		
9	N	0,6	1		
8	N	0,59	1		
7	N	0,58	1		
6	N	0,53	1		
13	P	0,52	0	5	
5	N	0,4	1		
4	N	0,33	1		
3	N	0,32	1		
2	N	0,24	1		
1	N	0,18	1		

Рис. 3. Расчет ROC-AUC по формуле (1)

Отсортированные спрогнозированные вероятности положительного класса

случай равенства вероятностей

№	фактический класс	спрогно-зированная вероятность положитель-ного класса	скоринговое правило $S(x_i, x_j) = \begin{cases} 1, x_i > x_j, \\ \frac{1}{2}, x_i = x_j, \\ 0, x_i < x_j \end{cases}$	количество наблюдений отрицательного класса, лежащих ниже соответствующего наблюдения положительного класса
20	P	0,92	0	12
19	P	0,9	0	12
18	P	**0,88**	0,5	11,5
12	N	**0,88**		
17	P	0,82	0	11
16	P	0,79	0	11
11	N	0,75	1	
15	P	0,73	0	10
14	P	0,72	0	10
10	N	0,7	1	
9	N	0,6	1	
8	N	0,59	1	
7	N	0,58	1	
6	N	0,53	1	
13	P	0,52	0	5
5	N	0,4	1	
4	N	0,33	1	
3	N	0,32	1	
2	N	0,24	1	
1	N	0,18	1	

Считаем количество отрицательных ниже каждого наблюдения положи-тельного класса

Рис. 4. Расчет ROC-AUC по формуле (1) для случая равных вероятностей принадлежности экземпляра положительному классу