Метрики качества классификации.

Даулбаев Талгат

7 марта 2018 г.

Задача классификации

$\mathbb{X}$ — множество объектов, $\mathbb{Y}$ — множество признаков.

$\{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$, где $x_i \in \mathbb{X} \subset \mathbb{R}^d$.

Здесь ℓ — количество объектов в выборке, d — количество признаков.

Если задача **бинарной классификации**, то $\mathbb{Y}$ обычно кодируется как $\{+1, -1\}$ или $\{1, 0\}$.

Один класс считается положительным (positive), другой — отрицательным (negative).

Задача классификации

Хотим:

- алгоритм $a(x)$, который каждому новому объекту x ставит в соответствие класс из $\mathbb{Y}$

или

- алгоритм $b(x)$, который каждому новому объект x ставит в соответствие оценку вероятности принадлежности положительному классу.

Если $\mathbb{Y} = \{1, 0\}$, то

$$a(x) = [b(x) > t],$$

где t — это порог, выбираемый нами.

Метрики качества $a(x)$

Чем плоха Accuracy (доля правильных ответов)?

Пусть мы решаем задачу, в которой два несбалансированных класса. Например, здоровые и больные очень редкой болезнью.

Классификатор, который говорит, что все люди здоровы, имеет высокую долю правильных ответов, но не решает задачу.

Матрица ошибок

	$y = +1$	$y = -1$
$a(x) = +1$	True Positive (TP)	False Positive (FP)
$a(x) = -1$	False Negative (FN)	True Negative (TN)

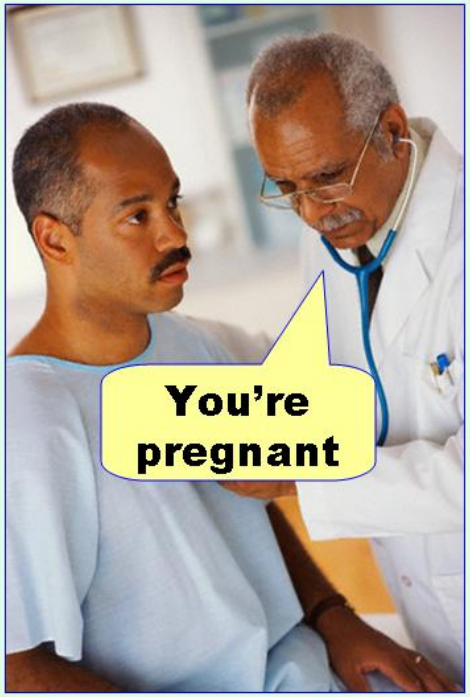
Таблица 1: Матрица ошибок

$$\underbrace{\text{Классификатор ответил верно?}}_{\text{True или False}}$$

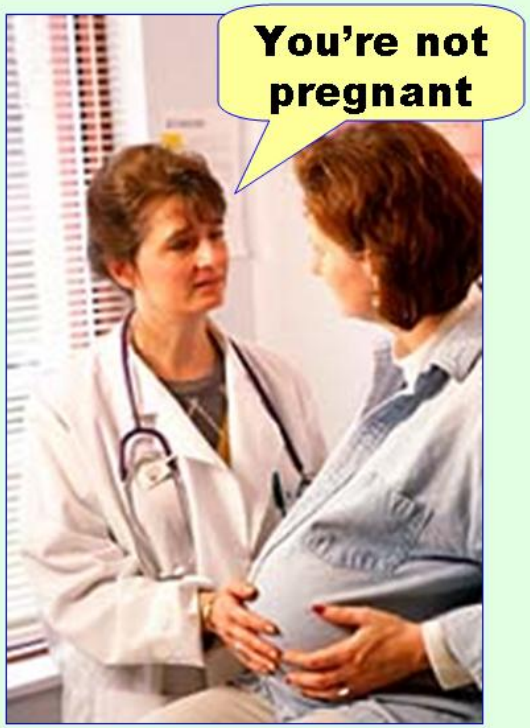
$$\underbrace{\text{К какому классу алгоритм отнёс объект?}}_{\text{Positive или Negative}}$$

Ошибки I и II рода

Type I error
(false positive)



Type II error
(false negative)



Мнемонические правила и смешные картинки часто помогают запоминать материал

Точность и полнота

Со слайдов лекции:

- Точность (precision) — можно ли доверять алгоритму, если $a(x) = +1$
- Полнота (recall) — как много положительных ответов находит $a(x)$

Упражнение

	$y = +1$	$y = -1$
$a(x) = +1$	True Positive (TP)	False Positive (FP)
$a(x) = -1$	False Negative (FN)	True Negative (TN)

Выпишите формулу через элементы матрицы ошибок:

- accuracy = ?
- precision = ?
- recall = ?

Упражнение

	$y = +1$	$y = -1$
$a(x) = +1$	True Positive (TP)	False Positive (FP)
$a(x) = -1$	False Negative (FN)	True Negative (TN)

- accuracy $= \dfrac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$
- precision $= \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$
- recall $= \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$

F1-мера

И точность, и полноту хотим максимизировать.

Из них можно получить единую характеристику, которая называется F1-мера:

$$\text{F1} = \frac{\mathbf{2} \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Пример: метрики качества $a(x)$ в sklearn'е

```
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, \
    recall_score, f1_score

y_pred = np.array([-1, -1, -1, -1, -1, -1,  1,  1, -1, -1,
                   -1, -1, -1, -1, -1, -1, -1, -1,  1, -1])
y_true = np.array([1, -1, -1, 1, -1, -1, 1, -1, -1, 1,
                   -1, 1, 1, 1, -1, -1, 1, -1, 1, 1])

print("Accuracy:", accuracy_score(y_true, y_pred))      # 0.55
print("Precision:", precision_score(y_true, y_pred))    # 0.667
print("Recall:", recall_score(y_true, y_pred))          # 0.2
print("F1:", f1_score(y_true, y_pred))                  # 0.308
```

Матрица ошибок для предыдущего примера

	$y = +1$	$y = -1$
$a(x) = +1$	TP = 2	FP = 1
$a(x) = -1$	FN = 8	TN = 9

Убедитесь, что ответы на предыдущем слайде правильные :)

TPR и FPR

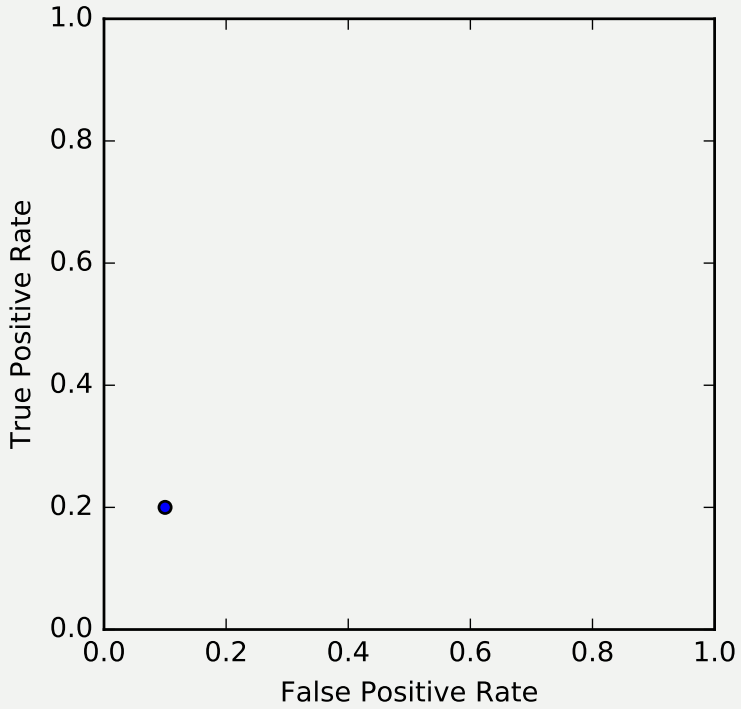
Введём дополнительно две характеристики: True Positive Rate и False Positive Rate.

- $\text{TPR} = \dfrac{\text{TP}}{\text{TP} + \text{FN (число объектов, которые positive)}} =$ recall
- $\text{FPR} = \dfrac{\text{FP}}{\text{FP} + \text{TN (число объектов, которые negative)}}$

Для примера с предыдущих слайдов $\text{TPR} = 0.2$ и $\text{FPR} = 0.1$.

График TPR и FPR

Ничего не мешает нам изобразить алгоритм в виде точки на графике с осями (FPR, TPR).



Каким алгоритмам соответствуют точки $(0, 0)$, $(1, 0)$, $(0, 1)$?

Метрики качества $b(x)$: AUC-ROC

ROC-кривая

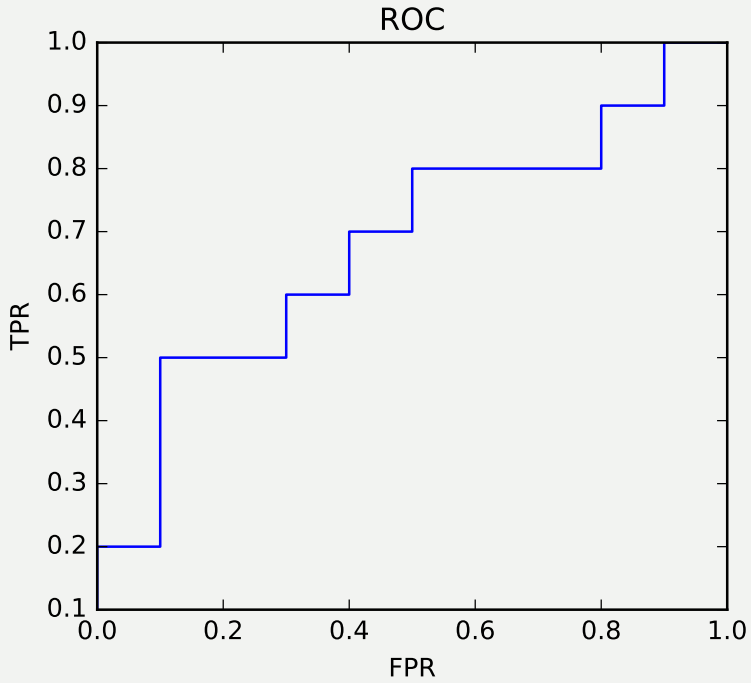
Пусть мы обучили алгоритм, вызвали метод predict_proba и получили вектор y_proba:

```
y_proba = np.array([0.3, 0.33, 0.1, 0.55, 0.37, 0.35, 0.8,
          0.7, 0.52, 0.54, 0.505, 0.505, 0.6, 0.38,
          0.39, 0.53, 0.34, 0.36, 0.9, 0.51])
y_true = np.array([1, -1, -1, 1, -1, -1, 1, -1, -1, 1,
                   -1, 1, 1, 1, -1, -1, 1, -1, 1, 1])
```

Бинаризуем по всем возможным порогам, отметим точки на графике и соединим — получится ROC-кривая.

Строим ROC-кривую на компьютере

```
from sklearn.metrics import roc_curve
fpr, tpr, threshold = roc_curve(y_true, y_proba, pos_label=1)
plt.plot(fpr, tpr)
plt.title('ROC')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.gca().set_aspect('equal', adjustable='box')
```



AUC-ROC

Площадь под ROC-кривой — метрика качества алгоритма $b(x)$.

AUC-ROC совсем плохого алгоритма $\approx \mathbf{0.5}$, у лучшего — $\mathbf{1}$.

```
from sklearn.metrics import roc_auc_score
print('AUC-ROC:', roc_auc_score(y_true, y_pred))
```

У нашего алгоритма 0.55, то есть он не очень хорош.

Строим ROC-кривую своими руками

1. Сортируем ответы по значению вероятности:

```
y_proba = np.array([0.9, 0.8, 0.7, 0.6, 0.55, 0.54,
        0.53, 0.52, 0.51, 0.505, 0.4, 0.39, 0.38,
        0.37, 0.36, 0.35, 0.34, 0.33, 0.3, 0.1])
y_true = np.array([1, 1, -1, 1, 1, 1, -1, -1, 1, -1,
        1, -1, 1, -1, -1, -1, 1, -1, 1, -1])
```

2. Стартуем в точке $(0, 0)$. Идём по массиву y_true слева-направо:

- Видим $+1$ — «идём вверх» на $\dfrac{1}{\text{число positive объектов}}$
- Видим -1 — «идём вправо» на $\dfrac{1}{\text{число negative объектов}}$
- (Особый случай: одинаковые значения y_proba и разные классы — идём «по диагонали»)

Метрики качества $b(x)$: Log Loss

Log Loss

Логарифмической функцией потерь называется

$$-\sum_{i=1}^{\ell}\Big([y_i=+1]\log b(x_i)+[y_i=-1]\log\left(1-b(x_i)\right)\Big).$$

Чем она меньше, тем лучше алгоритм.

Кстати, минимизация Log Loss по w при

$$b(x)=\frac{1}{1+\exp\{-\langle w,x\rangle\}}$$

даст в точности логистическую регрессию.