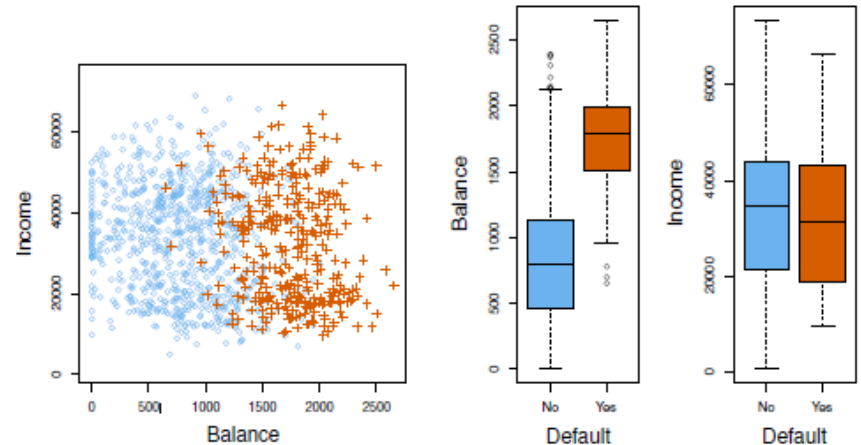


Лекция 10: Задача классификации. Оценка качества моделей

Задача классификации

- Переменная отклика является *категориальной*, например:
 - *Бинарная классификация*: почтовое сообщение может принадлежать одному из следующих классов $Y = (\text{spam}; \text{ham})$
 - *Многоклассовая классификация*: изображение цифры принадлежит одному из классов $Y = \{0, \dots, 9\}$
- Цели:
 - Построить классификатор $a: X \rightarrow Y$, который связывает метку класса с неразмеченным x
 - Понимание роли различных предикторов $x = (x_1, \dots, x_p)$
 - Оценка достижимого **качества** классификации



Задача классификации

- Дано: множество «размеченных» примеров :

- обучающая выборка или тренировочный набор:

$$Z = \{(x_i, y_i)\}_{i=1}^l$$

- $y_i \in Y = \{C_1, \dots, C_k\}$: известный **категориальный** отклик и его множество значений мощности K

- Основные определения:

- Постановка задачи: найти алгоритм (или гипотезу, или модель)

$$a_Z: X \rightarrow Y$$

- Естественная функция потерь – несовпадение прогноза (неудобно - не дифференцируема):

$$L(x, y) = [y \neq a(x)]$$

Дискриминантные функции

- Во многих случаях удобно пользоваться дискриминантными функциями для каждого класса c : $g_c: X \rightarrow G \subset \mathbb{R}$, такими что:

$$a^*(x) = \operatorname{argmax}_c g_c(x)$$

- Частный (и частый) случай *оптимальный байесовский классификатор*:

$$g_c(x) = P(y = c|x), G = [0,1]$$

- Граница между классами i и j :

$$\{x \in X | g_i(x) = g_j(x)\}$$

- Отступ (зазор, марджин) оценивает «гладкое» качество классификации:

$$M(x, y) = g_y(x) - \max_{c \neq y} g_c(x)$$

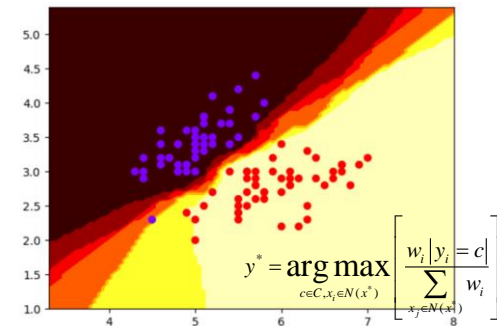
- Линейный классификатор:

$$g_c(x) = \langle w_c, x \rangle, \text{ где вектор } x = [x_1, \dots, x_p, 1]$$

Уже рассмотренные в курсе методы классификации

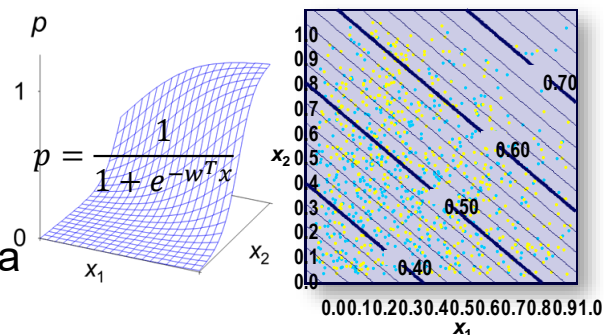
■ KNN (не линейный):

- Взвешенные, kernel и др.
- Прогноз – голосование соседей по окрестности
- Дискр. ф-ция – усредненные голосов
- Обучения нет, вместо него поиск



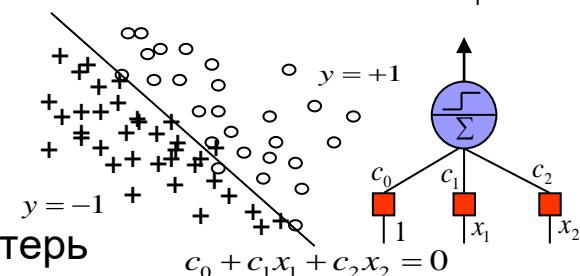
■ Логистическая регрессия (линейная):

- GLM с распределением Бернулли
- Прогноз – вероятность целевого отклика
- Дискр. ф-ция – линейная $w^T x$
- Обучение – методы оптимизации 1 или 2 порядка



■ Линейные гиперплоскости (персептроны):

- Разделяющие гиперплоскости
- Прогноз – знак в уравнении гиперплоскости
- Дискр. ф-ция – расстояние до гиперплоскости
- Обучение – например, SGD, разные функции потерь



Бинарный классификатор

- Два класса $|Y| = 2$
 - например $Y = \{0,1\}$, $Y = \{-, +\}$, $Y = \{no, yes\}$
 - один класс – целевой, или класс «событие», обычно 1, +, yes
 - другой класс - не целевой или «не событие», обычно 0, -, no
- Дискриминантная функция, отступ и модель, примеры:

- Разделяющая гиперплоскость:

$$Y = \{-1, +1\}, \quad g(x) = g_+(x) - g_-(x),$$

$$M(x, y) = yg(x),$$

$$a(x) = \text{sign}(g(x))$$

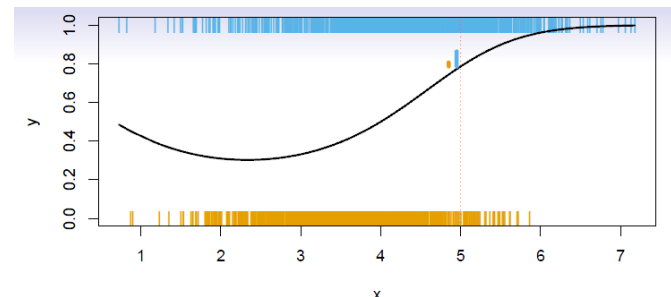
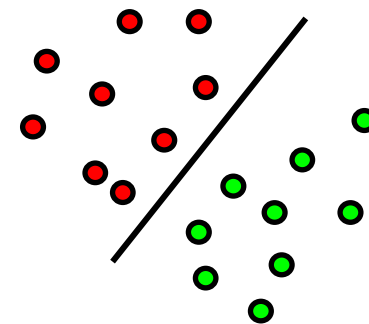
- Вероятностная модель:

$$Y = \{0,1\}, \quad g(x) = p(y = 1|x),$$

$$M(x, y) = p(y = 1|x) - p(y = 0|x) =$$

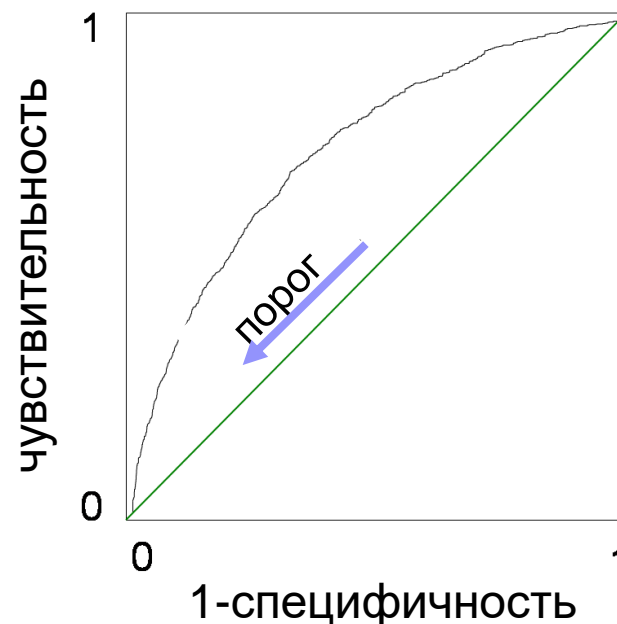
$$= 2p(y = 1|x) - 1,$$

$$a(x) = [M(x)]_+$$



Оценка качества бинарного классификатора

		Прогноз		
		0	1	
Реальность	0	True Negative	False Positive	Всего 0 (не событий)
	1	False Negative	True Positive	Всего 1 (событий)
		Всего прогноз 0 (не событий)	Всего прогноз 1 (событий)	



ЧУВСТВИТЕЛЬНОСТЬ (true positive rate (TPR), hit rate, recall)
 $TPR = R = SE = TP / (TP + FN)$

СПЕЦИФИЧНОСТЬ (true negative rate (TNR))
 $SPC = TN / (FP + TN)$

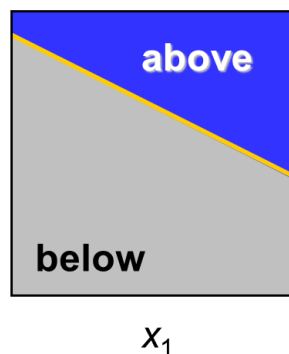
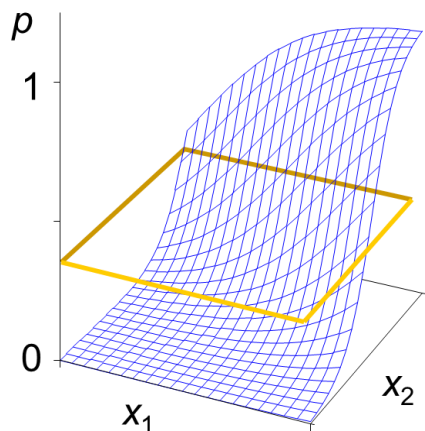
«АККУРАТНОСТЬ»
 $ACC = (TN + TP) / (FP + FN + TP + TN)$

F-МЕРА (гарм. среднее)
 $F_b = (1 - b^2) * P * R / (b^2 * (P + R))$

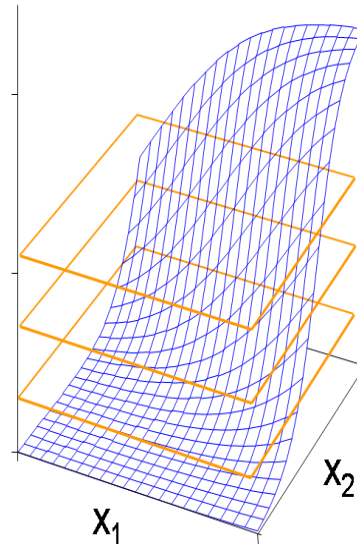
УРОВЕНЬ ОШИБОК
 $ERR = 1 - ACC$

ТОЧНОСТЬ (Precision)
 $P = TP / (TP + FP)$

Оценка бинарного классификатора



x_2 P



	0	1	<u>Se</u>	<u>Sp</u>
0	70	5	.64	.93
1	9	16		
0	66	9	.84	.88
1	4	21		
0	57	18	.96	.76
1	1	24		

Матрица выигрыша-проигрыша:

		решение	
		0	1
Реальность	0	δ_{TN}	δ_{FP}
	1	δ_{FN}	δ_{TP}

Правило Байеса:
Решение 1 если

$$P > \frac{1}{1 + \left(\frac{\delta_{TP} - \delta_{FN}}{\delta_{TN} - \delta_{FP}} \right)}$$

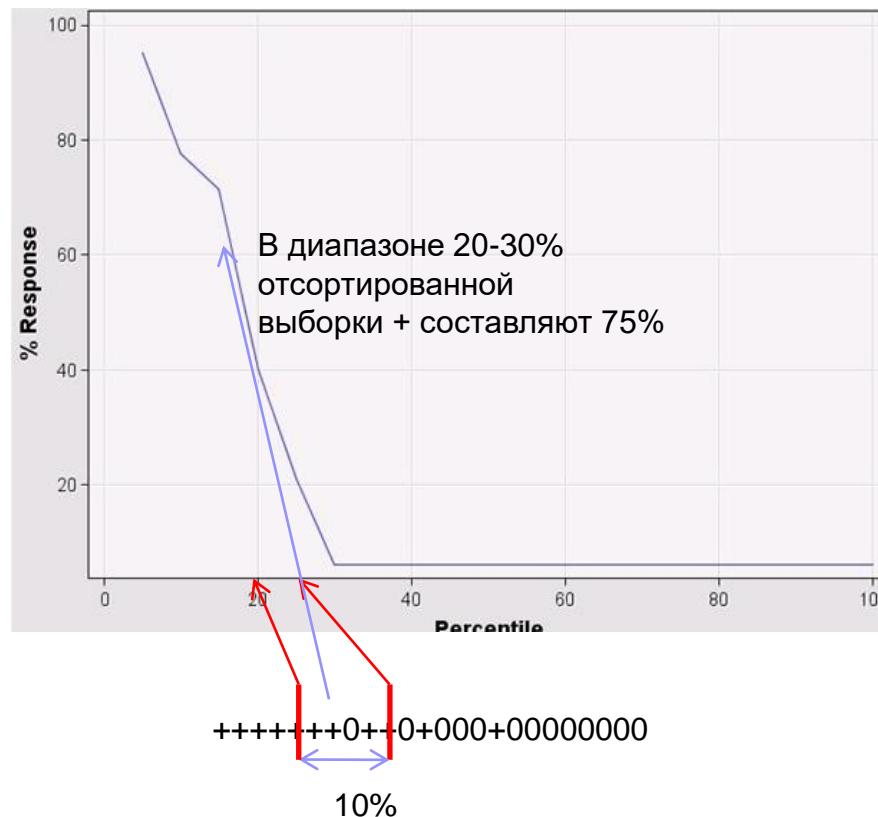
Графические средства сравнения моделей: Response (отклик)

■ Процедура построения:

- Сортируем (например, слева направо) набор по убыванию дискриминантной функции
- Идем порогом отсечения по отсортированному набору (слева направо) с некоторым диапазоном (как правило кратно 5%)
- Для каждого положения диапазона считаем отношение числа положительных примеров к числу всех примеров внутри диапазона
- Ставим точку на графике

■ Агрегированный отклик (cumulative response):

- Диапазон всегда с 0



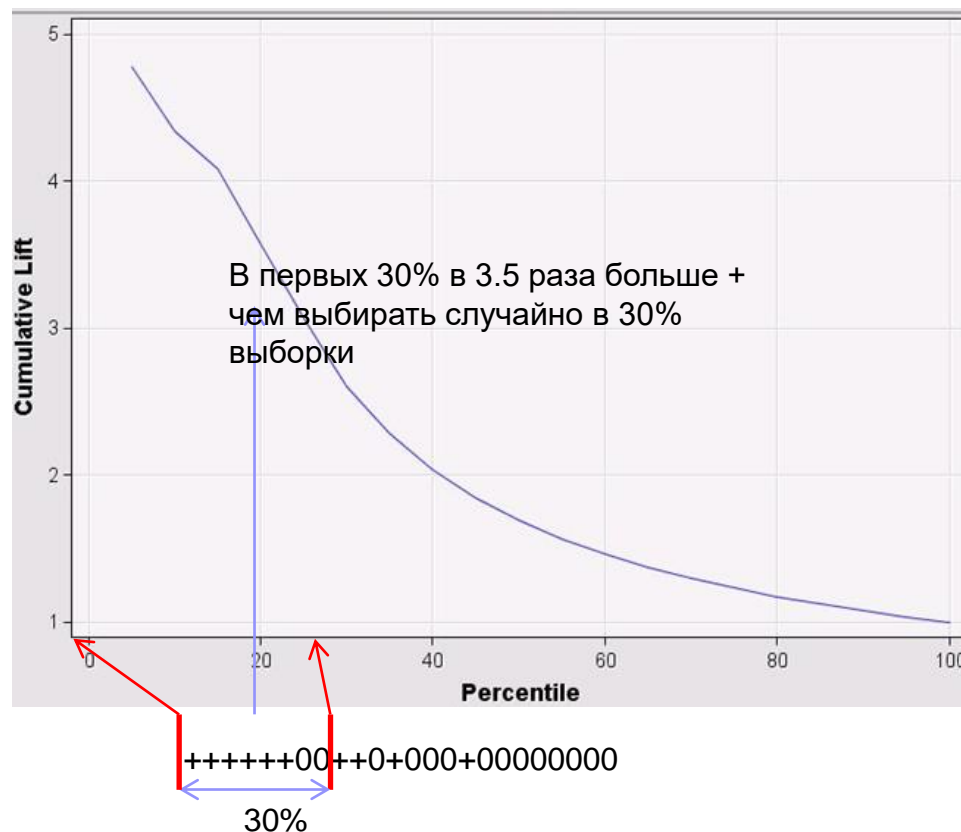
Графические средства сравнения моделей: Lift (подъем)

■ Процедура построения:

- Сортируем (например, слева направо) набор по убыванию дискриминантной функции
- Идем порогом отсечения по отсортированному набору (слева направо) с некоторым диапазоном (как правило кратно 5%)
- Для каждого положения диапазона считаем отношение числа положительных примеров к числу положительных примеров, которые могли бы быть выбраны «случайно» - без модели
- Ставим точку на графике

■ Агрегированный подъем(cumulative lift):

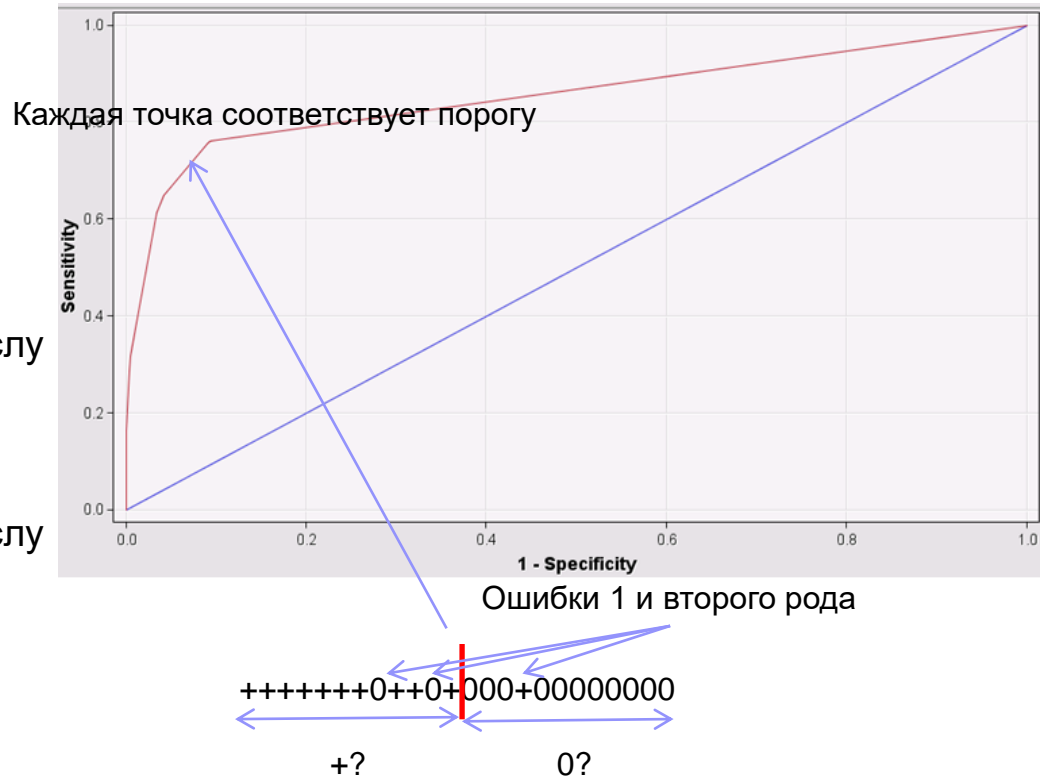
- Диапазон всегда с 0



ROC кривая

■ Процедура построения:

- Сортируем набор по убыванию дискриминантной функции
- Идем порогом отсечения по отсортированному набору
- Для каждого положения порога считаем:
 1. отношение числа положительных примеров «слева» от порога к числу всех положительных примеров – true positive rate
 2. отношение числа отрицательных примеров «слева» от порога к числу всех отрицательных примеров – false positive rate
- Ставим точку на графике



Оценка на основе согласованности всевозможных пар наблюдений (правильной упорядоченности наблюдений в паре), принадлежащих разным классам.

AUC – мера согласованности

- Дано:

- бинарная задача классификации в постановке $Y = \{-1, +1\}$
- $x_{(1)}, \dots, x_{(l)}$ - упорядочены по $g(x_{(i)}) \leq g(x_{(i+1)})$

- Поскольку:

$$TPR_k = \frac{1}{l_+} \sum_{i=k}^l [y_{(i)} = +1], FPR_k = \frac{1}{l_-} \sum_{i=k}^l [y_{(i)} = -1]$$

- Получаем AUC:

- вероятность правильно упорядочить пары наблюдений из разных классов:

$$\begin{aligned} AUC &= \sum_{k=1}^{l-1} \frac{TPR_{k+1} - TPR_k}{2} (FPR_k - FPR_{k+1}) = \\ &= \frac{1}{l_- l_+} \sum_{k=1}^{l-1} \sum_{i=k+1}^l [y_{(i)} = +1][y_{(k)} = -1] = \frac{1}{l_- l_+} \sum_{k < i} [y_{(i)} > y_{(k)}] \end{aligned}$$

- не все однозначно с «ничьими», обычно их берут с весом 0.5, но есть разные подходы (в том числе 0 – считать ничьи несогласованными, 1 – считать ничьи согласованными)

Максимизация AUC напрямую с помощью линейной модели

- Дано:

- бинарная задача классификации в постановке $Y = \{-1, +1\}$
- Модель классификации в классе $a(x, w) = \text{sign}(g(x, w))$

- Поскольку AUC – доля правильно упорядоченных пар из разных классов, то

$$AUC(w) = \frac{1}{l_- l_+} \sum_{i,j} [y_i < y_j] (g(x_i, w) - g(x_j, w)) \rightarrow \max_w$$

- Эмпирический риск (функция потерь ранжирования):

$$1 - AUC(w) \leq Q(w) = \frac{1}{l_- l_+} \sum_{i,j: y_i < y_j} \underbrace{L(g(x_i, w) - g(x_j, w))}_{\text{Отступ для пары } x_i, x_j: M(x_i, x_j, w)} \rightarrow \min_w$$

Отступ для пары x_i, x_j :

$$M(x_i, x_j, w)$$

- Алгоритм – например, SGD

Логистическая функция потерь для оценки качества вероятностных моделей

- AUC инвариантна к монотонному преобразованию отклика – не лучший вариант для вероятностных классификаторов
- Поэтому используют логарифмическое правдоподобие для распределения Бернулли

- Если $Y = \{0,1\}$, $Z = \{(x_i, y_i)\}_{i=1}^l$, $p(x) = a(x) = \operatorname{argmax}_{y \in Y} P(y|x)$:

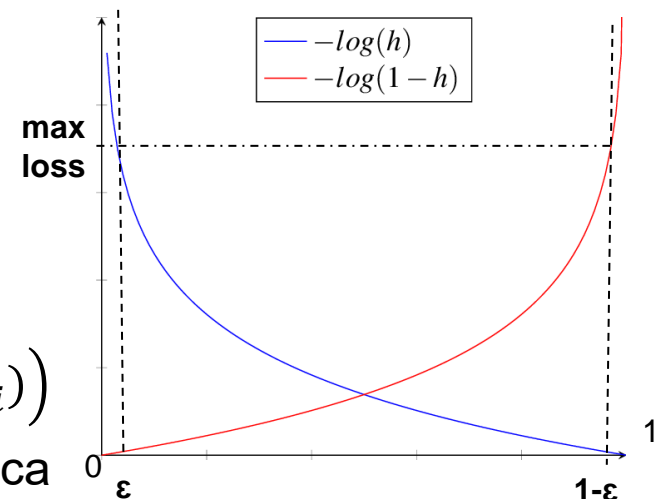
$$\text{logloss}(Z, p(x)) = -\frac{1}{l} \sum_{i=1}^l [y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))]$$

- Ключевой недостаток:
 - ☐ чувствительность к «грубым» ошибкам
 - ☐ неограниченный рост потерь
 - ☐ делают «пороги толерантности»

- Многоклассовое (C классов) обобщение:

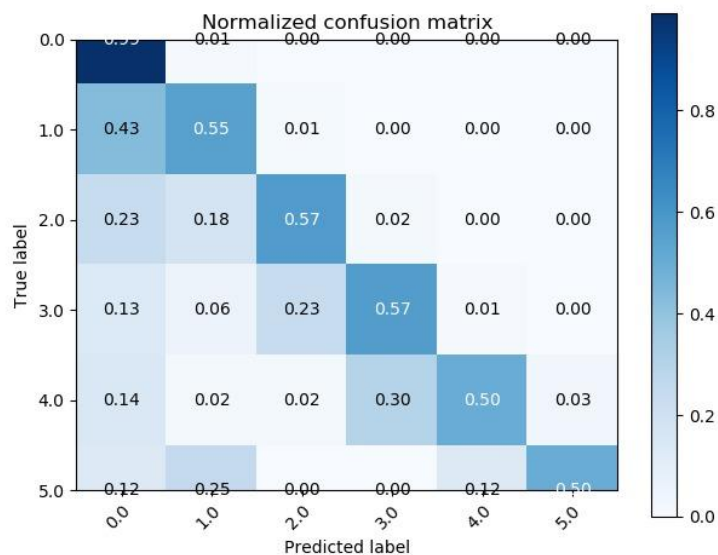
$$\text{logloss}(Z, p(x)) = -\frac{1}{lC} \sum_{i=1}^l \sum_{j=1}^C y_{ij} \log(p_j(x_i))$$

y_{ij} - бинарный OneHotEncoding метки класса

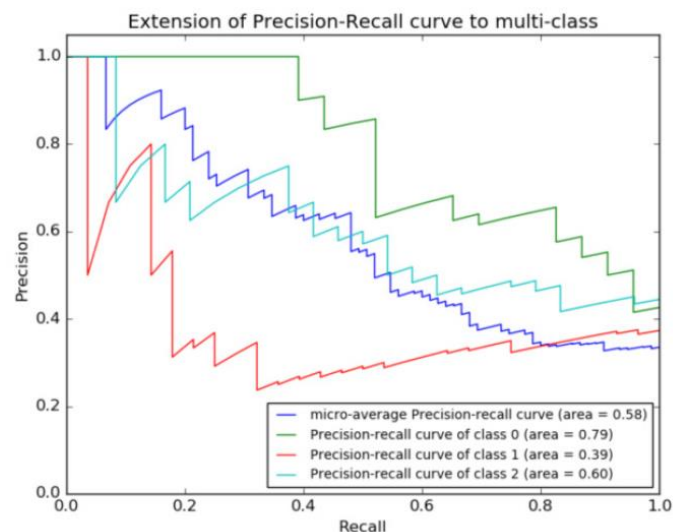


Многоклассовый случай

Многоклассовая
матрица ошибок



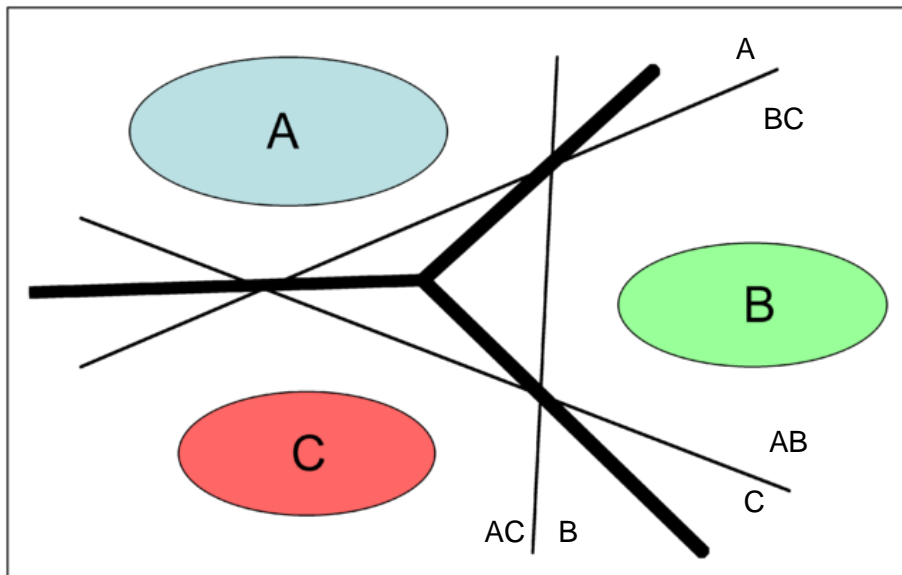
«Бинарные» оценки
графики по каждому классу



	бинарный случай	макроусреднение	микроусреднение
Точность	$\frac{TP}{\hat{P}}$	$\frac{1}{C} \sum_{c=1}^C \frac{TP_c}{\hat{P}_c}$	$\frac{\sum_{c=1}^C TP_c}{\sum_{c=1}^C \hat{P}_c}$
Полнота	$\frac{TP}{P}$	$\frac{1}{C} \sum_{c=1}^C \frac{TP_c}{P_c}$	$\frac{\sum_{c=1}^C TP_c}{\sum_{c=1}^C P_c}$

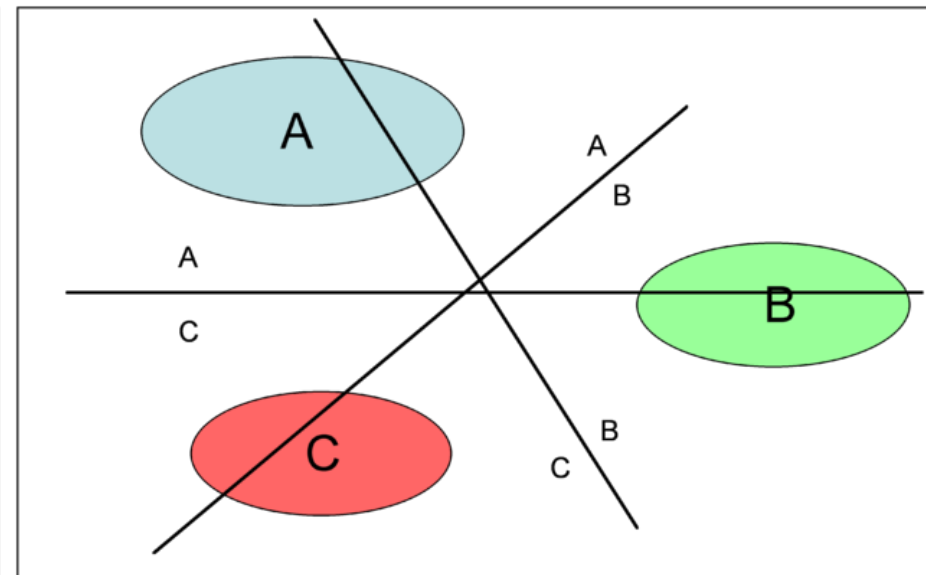
Декомпозиция много-классовой задачи в ансамбль бинарных

Каждый против всех



k бинарных моделей (границ)

Каждый против каждого



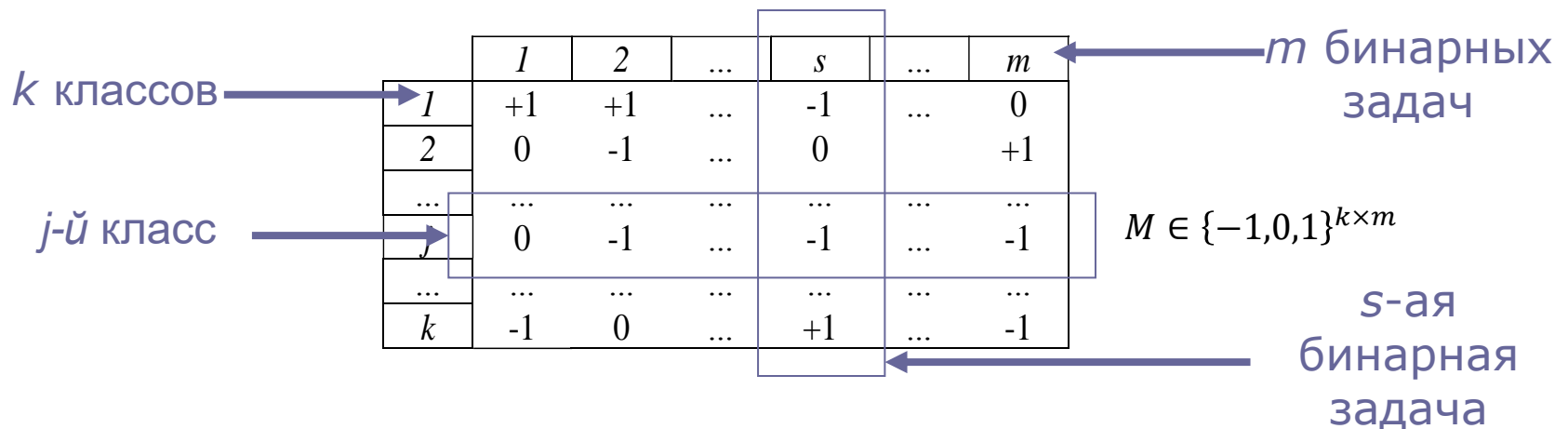
$k(k-1)/2$ бинарных моделей (границ)

Много-классовые задачи: Error Correcting Output Coding (ECOC)

- Идея из теории информации и телекоммуникаций:
 - В телекоммуникациях: использовать избыточные коды для коррекции ошибок при передачи данных по «зашумленному» каналу
 - В машинном обучении: использовать избыточное число бинарных моделей (кодируется множество классов в супер-классы = группы) для повышения точности классификации, т.е. отклик избыточно кодируется
- Три этапа в ECOC:
 - Coding (кодирование): составление кодовой матрицы (coding matrix) и на ее основе обучающих выборок для бинарных задач
 - Learning (обучение): строятся бинарные модели
 - Decoding (декодирование): прогнозируется отклик (метка класса) на основе индивидуальных прогнозов бинарных классификаторов и кодовой матрицы.

Кодирование в ЕСОС

- Исходная задача с k классами конвертируется в m бинарных подзадач с помощью кодовой матрицы



- Каждый j -й класс имеет кодовое слово, соответствующее строке в матрице M
- Каждая s -я бинарная задача имеет 3 типа классов :
 - “positive”: $I_s^+ = \{y | y \in Y \wedge M(y, s) = +1\}$
 - “negative”: $I_s^- = \{y | y \in Y \wedge M(y, s) = -1\}$
 - “ignored”: $I_s^0 = \{y | y \in Y \wedge M(y, s) = 0\}$

Кодирование в ЕСОС

- “Разреженный” ЕСОС – общий случай:

- “Плотный” ЕСОС – матрица без 0

- “Каждый против всех”:

	1	2	$k-1$	k
1	+1	-1	-1	-1	-1	-1
2	-1	+1	-1	-1	-1	-1
...	-1	-1	+1	-1	-1	-1
...	-1	-1	-1	+1	-1	-1
$k-1$	-1	-1	-1	-1	+1	-1
k	-1	-1	-1	-1	-1	+1

“Каждый против каждого”:

	1	2	...	$k-1$	k	$k+1$...	$\binom{k}{2}$
1	+1	+1	...	+1	0	0	...	0
2	-1	0	...	0	+1	+1	...	0
...	0	-1	...	0	-1	0	...	0
...	0	0	...	0	0	-1	...	0
...	0	0	...	0	0	0	...	+1
k	0	0	...	-1	0	0	...	-1

- Методы кодирования:

- *Алгебраическая теория кодирования* (коды Хэмминга, например)
 - *Задаче-зависимое кодирование*: группы задает эксперт или ищутся на основе матрицы ошибок в “Каждый против всех” или “Каждый против каждого”
 - **Случайные коды**: случайные длинные «хорошо разделимые»

Обучение в ЕСОС

- m бинарных задач решаются независимо:

- s -й бинарный классификатор отделяет s -ые “положительные” примеры от s -х “отрицательных”, так что s -й тренировочный набор:

$$Z_s = \{(x_i, M(y_i, s)) | (x_i, y_i) \in Z \wedge (y_i \in I_s^- \vee y_i \in I_s^+)\} \in X \times \{-1, +1\}$$

- Строится бинарные классификаторы $a_s: X \rightarrow Y_{bin}$ и получаем m моделей бинарной классификации (m гипотез)
 $a_1(x), \dots, a_m(x)$

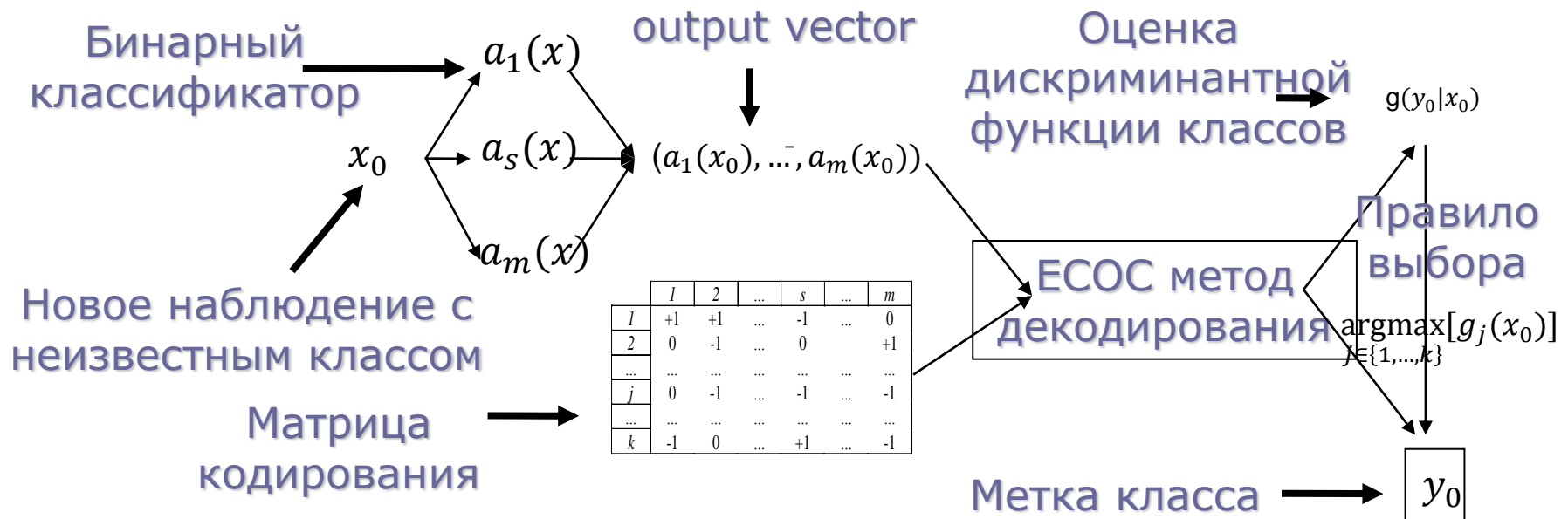
- Типы бинарного отклика:

- Булевый (hard-level): $Y_{hard} = \{-1, +1\}$
- Вещественный (soft-level): $Y_{soft} = [-1, 1]$
- Вероятностный:

$$Y_{prob} = \{-r_s(x) = P(a_s(x) \in I_s^+ | a_s(x) \in I_s^+ \cup I_s^-)\}$$

Декодирование в ЕСОС

■ Процесс прогнозирования:



- Применить все бинарные классификаторы, получить вектор откликов длины m
- Применить к нему выбранный метод декодирования и получить прогноз

Декодирование в ЕСОС

- На основе расстояний:

- Поиск ближайшего к вектору откликов кодового слова

Выходной вектор
прогнозов

$$\bar{a}(x_0) = (0, 1, \dots, -1)$$

	1	2	...	s	...	l
1	+1	+1	...	-1	...	0
2	0	-1	...	0	...	+1
...
j	0	-1	...	-1	...	-1
...
k	-1	0	...	+1	...	-1

Ближайший
код

$$y_0 = j$$

Предсказанный
класс

- Используются разные метрики:

- Хэмминга (hard-level): $d_H(\bar{a}(x), M(y)) = \sum_{s=1}^m [1 - \text{sgn}(M(y, s)a_s(x))]$
- Минковского (probabilistic): $d_{L1}(\bar{r}(x), M(y)) = \sum_{s=1}^m |M(y, s) - r_s(x)|$
- На основе функции потерь:

$$Loss(\bar{a}(x), M(y)) = \sum_{s=1}^m loss(M(y, s)a_s(x))$$

- Вероятностные (например, на основе модели попарных сравнений Бредли-Терри) и др.

Пример

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.multiclass import OneVsOneClassifier
from sklearn.multiclass import OutputCodeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.inspection import DecisionBoundaryDisplay
```

```
X, y = load_iris(return_X_y=True)
X = X[:, :2]
X.shape, y.shape
```

```
((150, 2), (150,))
```

```
ovr = OneVsRestClassifier(LogisticRegression()).fit(X, y)
ovo = OneVsOneClassifier(LogisticRegression()).fit(X, y)
ocs = OutputCodeClassifier(LogisticRegression(), code_size=5).fit(X, y)
print(ocs.code_book_)
```

```
[[ 1. -1. -1.  1. -1. -1.  1. -1. -1. -1.  1. -1.  1.  1.  1.]
 [ 1. -1. -1.  1. -1. -1.  1. -1. -1. -1.  1. -1. -1. -1.  1.]
 [-1.  1. -1.  1. -1. -1.  1.  1. -1. -1. -1.  1.  1. -1. -1.]]
```

```
for estimator in [ovr, ovo, ocs]:
    DecisionBoundaryDisplay.from_estimator(estimator, X, cmap="Pastel1")
    plt.scatter(*X.T, c=y, cmap="Set1")
    plt.title(type(estimator).__name__)
```

