



# Лекция 9:

## Задача классификации, линейные модели классификации, SVM, методы Байеса

# Задачи классификации

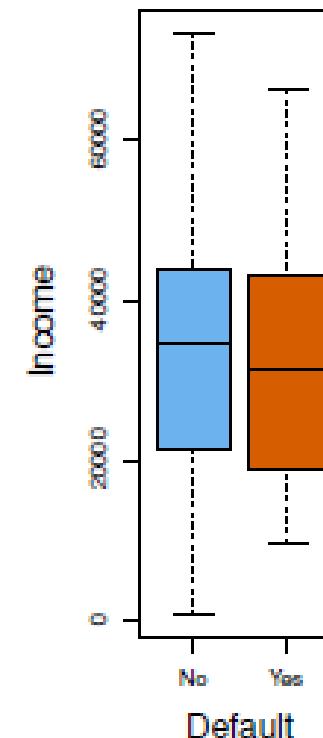
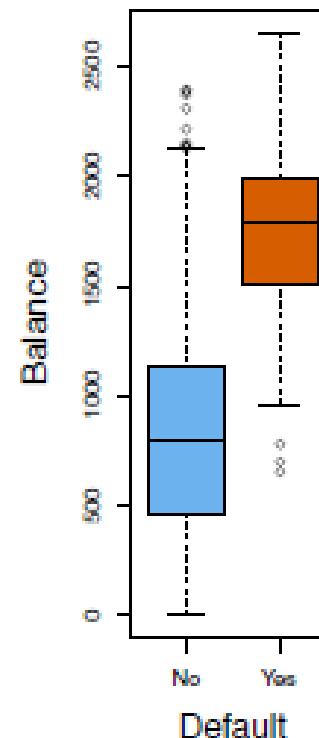
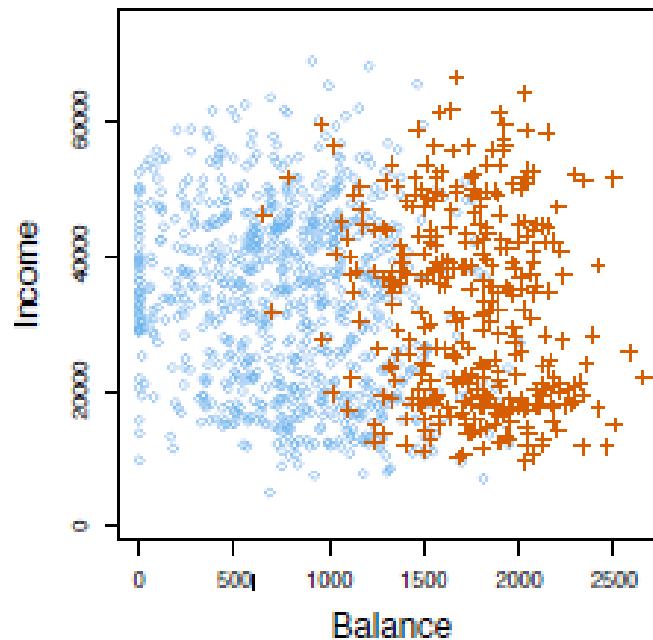
Переменная отклика  $Y$  является *категориальной* – например, почтовое сообщение может принадлежать одному из следующих классов  $C = (\text{spam}; \text{ham})$  ( $\text{ham}$  = обычная почта), изображение цифры может принадлежать одному из классов  $C = \{0, 1, \dots, 9\}$ .

Цель задачи:

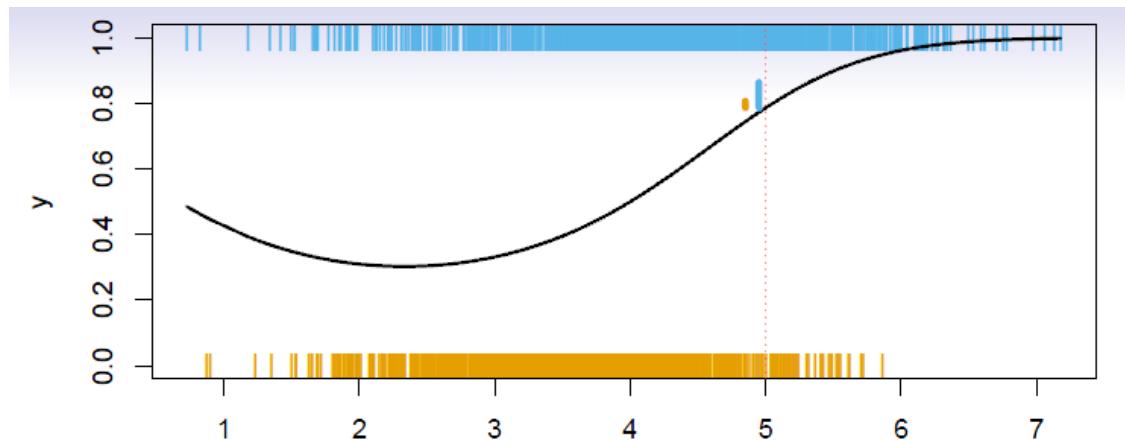
- Построить классификатор  $C(X)$ , который связывает метку класса из множества  $C$  с неразмеченным образцом  $X$ .
- Оценка степени неточности для каждой классификации
- Понимание роли различных предикторов среди

$$X = (X_1, X_2, \dots, X_p).$$

# Пример: Банкротство по кредитным картам



# Задача классификации



Существует ли идеальная  $C(X)$ ? Пусть  $X$  классов в множестве  $C$  пронумерованы как  $1, 2, \dots, K$ . Пусть

$$p_k(x) = \Pr(Y = k | X = x), \quad k = 1, 2, \dots, K.$$

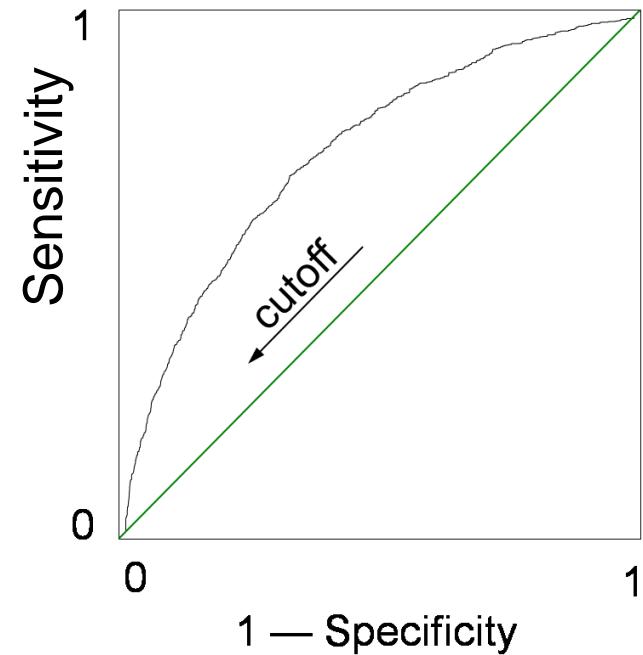
Это условные вероятности классов в точке  $x$ . Тогда оптимальный классификатор Байеса в точке  $x$ :

$$C(x) = j \text{ if } p_j(x) = \max\{p_1(x), p_2(x), \dots, p_K(x)\}$$

Качество классификации:  $\text{Err}_{\text{Te}} = \text{Ave}_{i \in \text{Te}} I[y_i \neq \hat{C}(x_i)]$

# Оценка модели

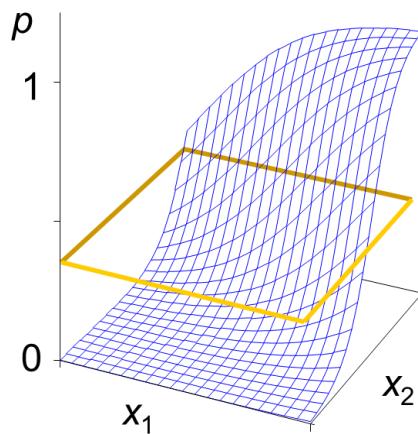
		Predicted Class	
		0	1
Actual Class	0	True Negative	False Positive
	1	False Negative	True Positive
Predicted Negative		Predicted Positive	



**SENSITIVITY** (true positive rate (TPR), hit rate, recall)  
 $TPR = TP / (TP+FN)$

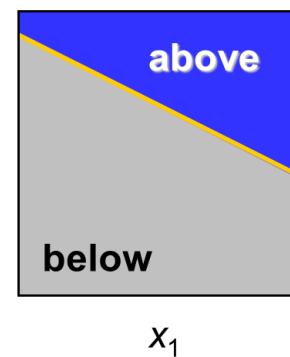
**SPECIFICITY** (SPC) (true negative rate (TNR))  
 $SPC = TN / (FP + TN)$

# Оценка модели



Матрица выигрыша-проигрыша:

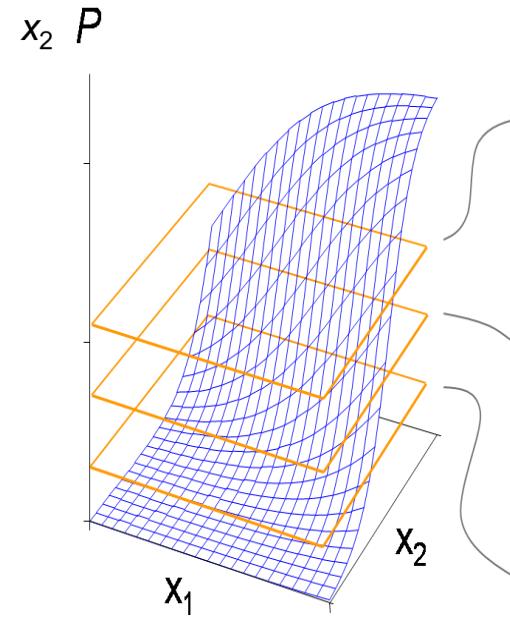
Decision			
		0	1
Actual Class	0	$\delta_{TN}$	$\delta_{FP}$
	1	$\delta_{FN}$	$\delta_{TP}$



Bayes Rule:

Decision 1 if

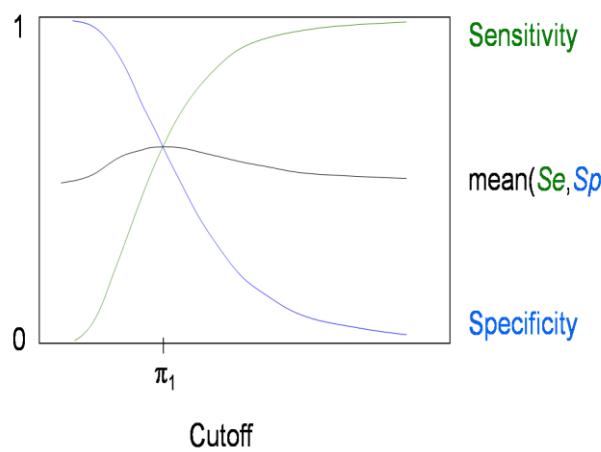
$$P > \frac{1}{1 + \left( \frac{\delta_{TP} - \delta_{FN}}{\delta_{TN} - \delta_{FP}} \right)}$$



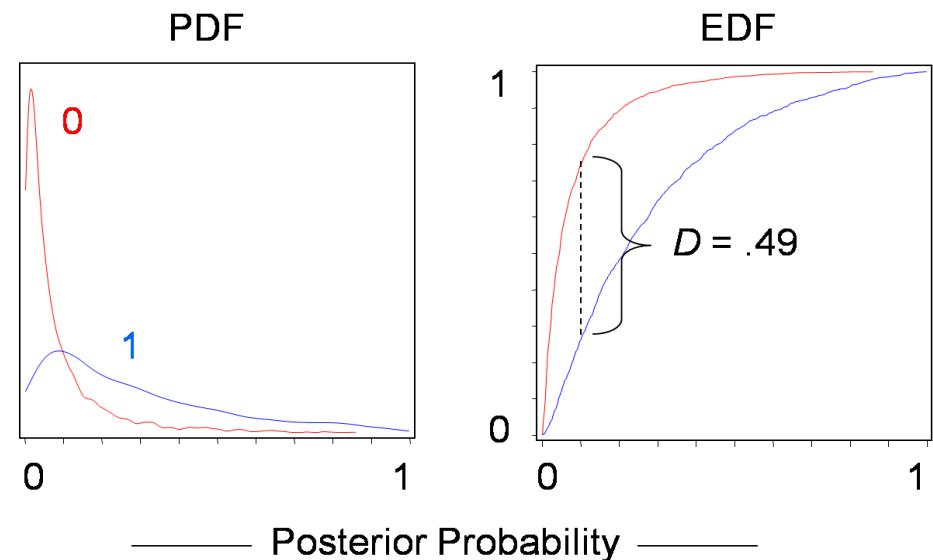
	0	1	Se	Sp
70	5	.64	.93	
9	16			
66	9	.84	.88	
4	21			
57	18	.96	.76	
1	24			

# Выбор порога

Усредненная (иногда взвешенная)  
чувствительность и  
специфичность



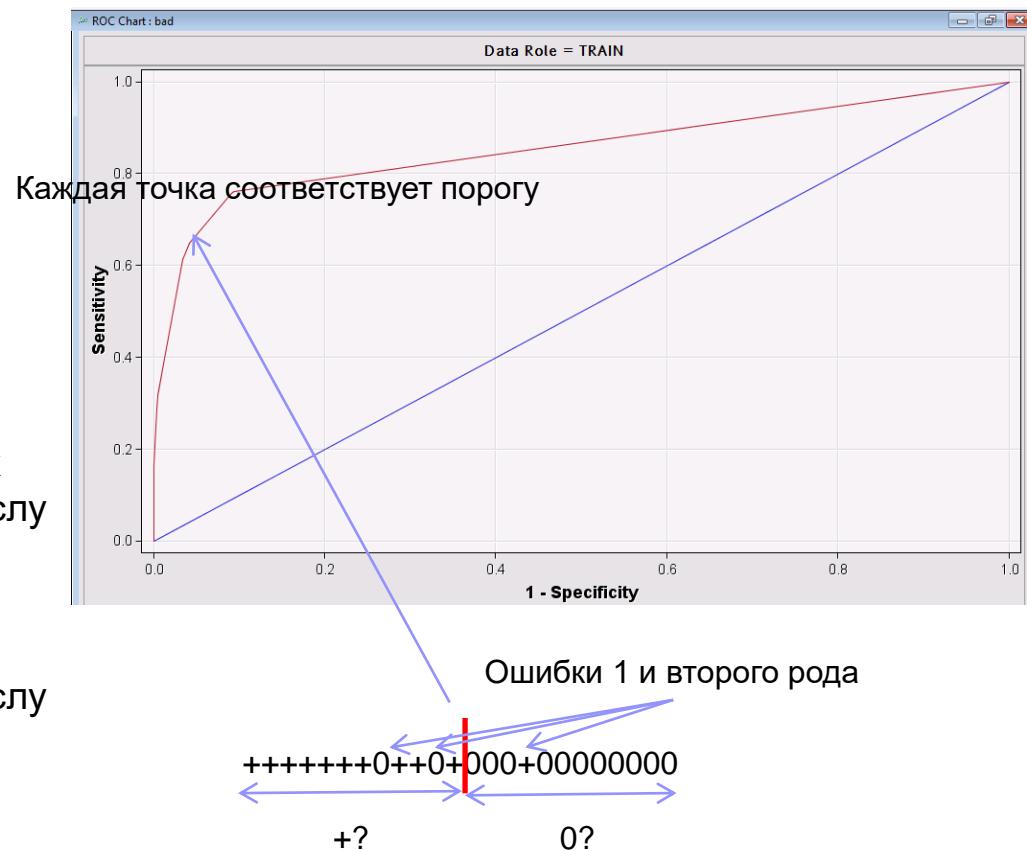
Критерий Колмогорова-Смирнова



# ROC кривая и AUC

## ■ Процедура построения:

- Сортируем набор по убыванию спрогнозированной оценки (вероятности положительного отклика)
- Идем порогом отсечения по отсортированному набору
- Для каждого положения порога считаем:
  1. отношение числа положительных примеров «слева» от порога к числу всех положительных примеров – detection rate
  2. отношение числа отрицательных примеров «слева» от порога к числу всех отрицательных примеров – false positive
- Ставим точку на графике

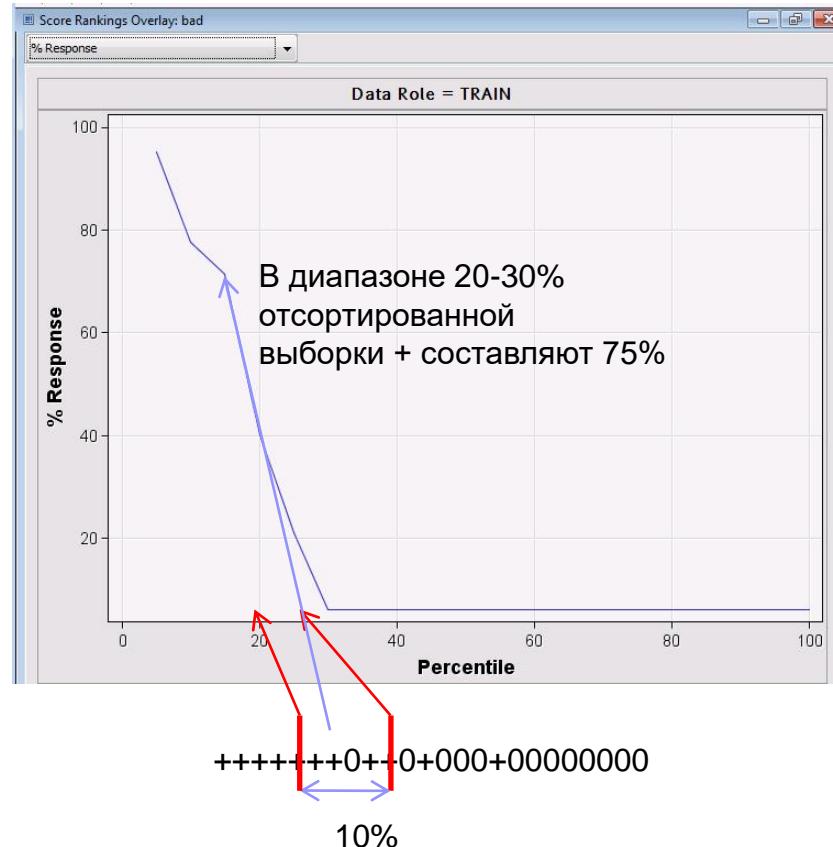


Оценка на основе согласованности всевозможных пар наблюдений (правильной упорядоченности наблюдений в паре), принадлежащих разным классам.

# Графические средства сравнения моделей: Response (отклик)

## ■ Процедура построения:

- Сортируем (например, слева направо) набор по убыванию спрогнозированной оценки (вероятности положительного отклика)
- 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

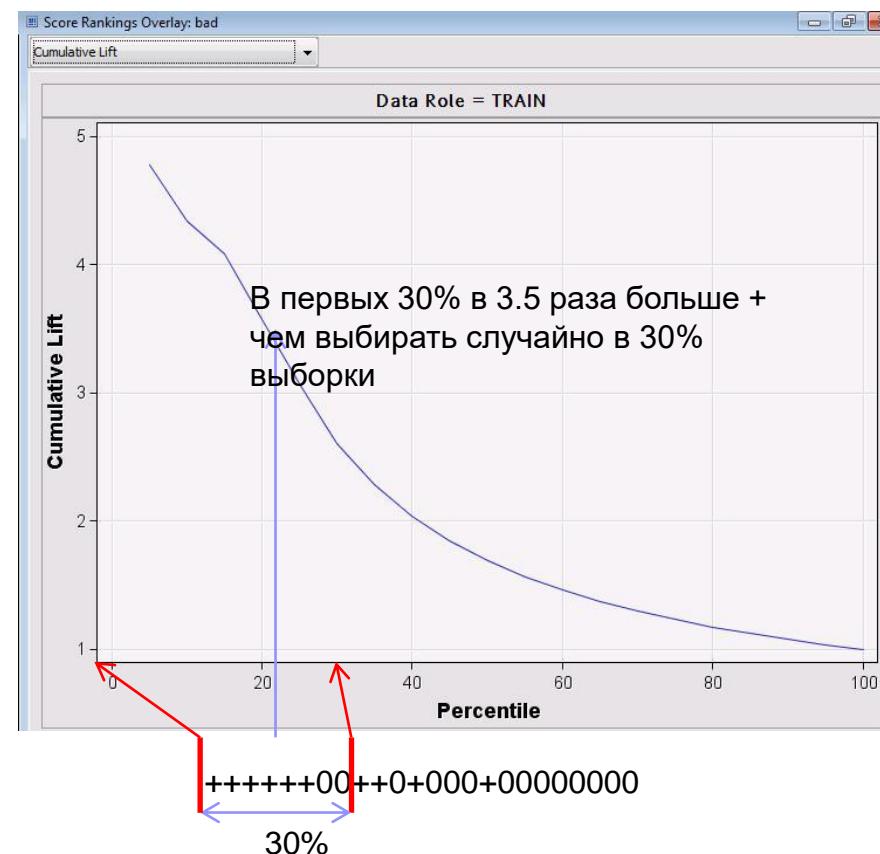
# Графические средства сравнения моделей: Lift (подъем)

## ■ Процедура построения:

- Сортируем (например, слева направо) набор по убыванию спрогнозированной оценки (вероятности положительного отклика)
- 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



# Методы Байеса

- Вероятностная постановка задачи прогнозирования:
  - В явном виде вычисляется вероятность для каждой гипотезы (варианта прогноза), очень важно для многих задач
- Возможность дообучения:
  - Каждый размеченный пример итеративно увеличивает/уменьшает вероятность корректности гипотезы
  - Априорные знания легко внедрить в модель
- Вероятностный прогноз:
  - Несколько вариантов гипотез с оценкой достоверности
- Де факто - эталон:
  - Даже когда методы Байеса вычислительно трудоемки, они могут быть полезны в качестве разовой оценки для сравнения с более быстрыми методами в плане оценки точности последних

# Теорема Байеса

- Теорема Байеса:
  - Если  $D$  - тренировочный набор, то апостериорная вероятность гипотезы  $h$ ,  $P(h|D)$  удовлетворяет соотношению:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- MAP (maximum posterior) гипотеза

$$h_{MAP} \equiv \operatorname{argmax}_{h \in H} P(h|D) = \operatorname{argmax}_{h \in H} P(D|h)P(h).$$

- «Необходимые» оценки вероятностей:
  - $P(D)$  – константа для всех классов
  - $P(h)$  - относительная частота класса в выборке
  - Надо найти  $h$  такое, что  $P(h|D)$  максимально =  $h$  такое, что  $P(D|h) \cdot P(h)$  максимально
- Проблема:
  - вычисление  $P(D|h)$  на всех комбинациях переменных вычислительно сложно или невозможно

# «Наивный» Байес

- Naïve упрощение:
  - Атрибуты независимы:  $P(x_1, \dots, x_n | C) = P(x_1 | C) \cdot \dots \cdot P(x_n | C)$
- Оценка  $P(x_i | C)$ :
  - Если атрибут категориальный, то относительная частота того, сколько примеров с классом С имеют значение атрибута  $x_i$
  - Если атрибут числовой, то по плотности распределения, например:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$$

где  $\mu_k$  – это среднее, и  $\sigma_k^2$  - дисперсия (в классе  $k$ ).

- И так, и так считается быстро
- Предположение о независимости
  - Делает метод вычислительно эффективным
  - Приводит к оптимальному классификатору (если и правда независимые атрибуты), но на практике почти всегда зависимы
- Попытки преодолеть эти ограничения:
  - Байесовские сети

# Пример: построить модель прогноза возможности поиграть в теннис

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

$$P(p) = 9/14$$

$$P(n) = 5/14$$

outlook	
$P(\text{sunny} p) = 2/9$	$P(\text{sunny} n) = 3/5$
$P(\text{overcast} p) = 4/9$	$P(\text{overcast} n) = 0$
$P(\text{rain} p) = 3/9$	$P(\text{rain} n) = 2/5$
temperature	
$P(\text{hot} p) = 2/9$	$P(\text{hot} n) = 2/5$
$P(\text{mild} p) = 4/9$	$P(\text{mild} n) = 2/5$
$P(\text{cool} p) = 3/9$	$P(\text{cool} n) = 1/5$
humidity	
$P(\text{high} p) = 3/9$	$P(\text{high} n) = 4/5$
$P(\text{normal} p) = 6/9$	$P(\text{normal} n) = 2/5$
windy	
$P(\text{true} p) = 3/9$	$P(\text{true} n) = 3/5$
$P(\text{false} p) = 6/9$	$P(\text{false} n) = 2/5$

# Пример

- Новый пример:
  - $x = \langle \text{rain}, \text{hot}, \text{high}, \text{false} \rangle$
- Расчет вероятностей:
  - $P(x|p) \cdot P(p) = P(\text{rain}|p) \cdot P(\text{hot}|p) \cdot P(\text{high}|p) \cdot P(\text{false}|p) \cdot P(p) = \frac{3}{9} \cdot \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} = 0.010582$
  - $P(x|n) \cdot P(n) = P(\text{rain}|n) \cdot P(\text{hot}|n) \cdot P(\text{high}|n) \cdot P(\text{false}|n) \cdot P(n) = \frac{2}{5} \cdot \frac{2}{5} \cdot \frac{4}{5} \cdot \frac{2}{5} \cdot \frac{5}{14} = 0.018286$
- Вывод:
  - $P(x|n) \cdot P(n) > P(x|p) \cdot P(p)$
  - Значит скорее всего не поиграть

# Пример

```
from sklearn.datasets import load_iris
from sklearn import naive_bayes
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.inspection import DecisionBoundaryDisplay

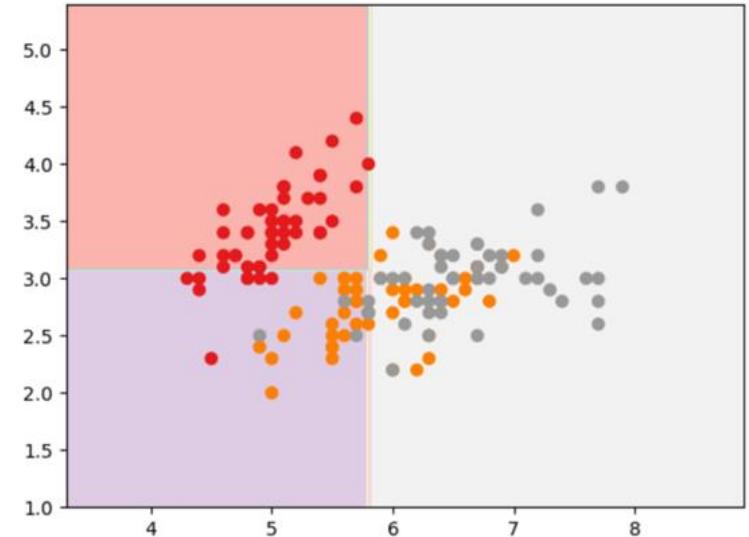
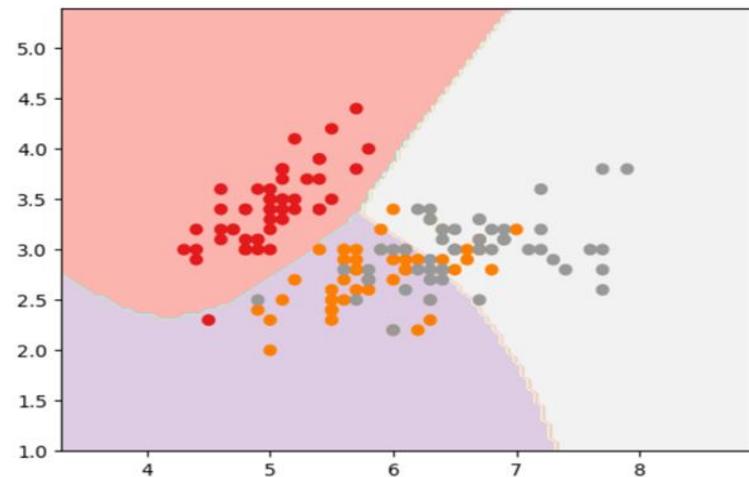
X, y = load_iris(return_X_y=True)
X = X[:, :2]

model = Pipeline([
    ("scaler", StandardScaler()),
    ("bayes", naive_bayes.GaussianNB())
])

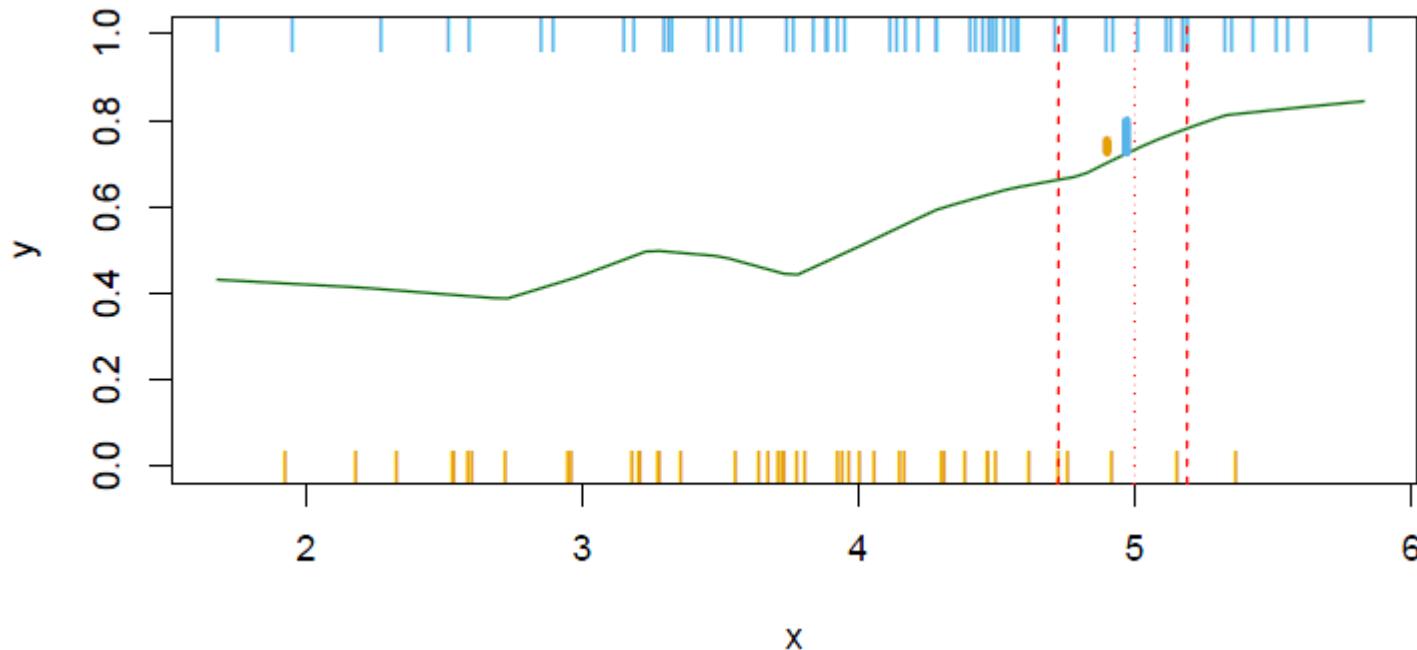
model.fit(X, y)

DecisionBoundaryDisplay.from_estimator(model, X, cmap="Pastel1")
plt.scatter(*X.T, c=y, cmap="Set1")

DecisionBoundaryDisplay.from_estimator(Pipeline([
    ("1", StandardScaler(),),
    ("2", naive_bayes.BernoulliNB(),)]).fit(X, y), X, cmap="Pastel1")
plt.scatter(*X.T, c=y, cmap="Set1")
```



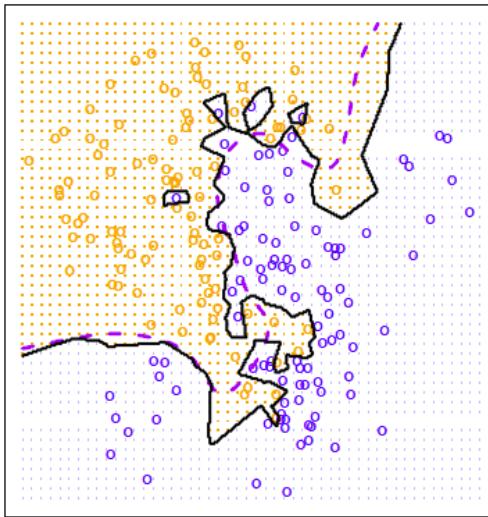
# Задача классификации KNN



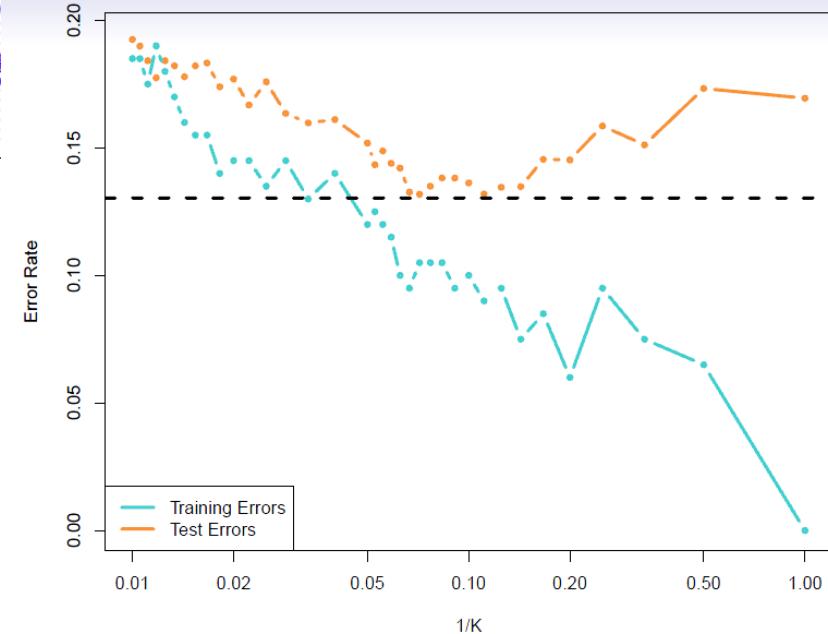
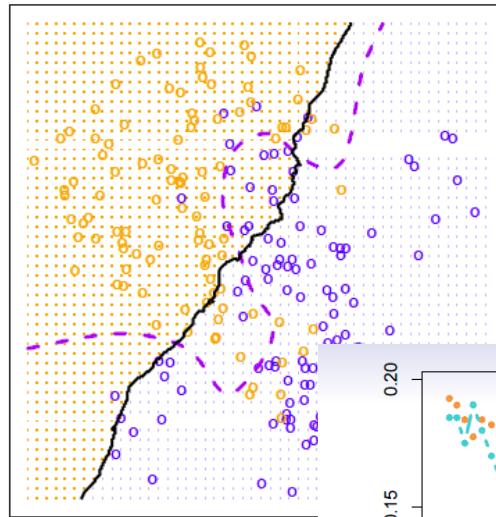
- Также может быть использовано усреднение методом ближайших соседей.
- Хотя такой подход плохо применим при возрастании размерности.

# Пример: KNN для двух измерений

KNN: K=1



KNN: K=100



# Пример: KNN (Python) (1)

```
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.datasets import load_iris  
from sklearn.inspection import DecisionBoundaryDisplay
```

```
X, y = load_iris(return_X_y=True)
```

```
X, y = X[y < 2][:, :2], y[y < 2]  
X.shape, y.shape
```

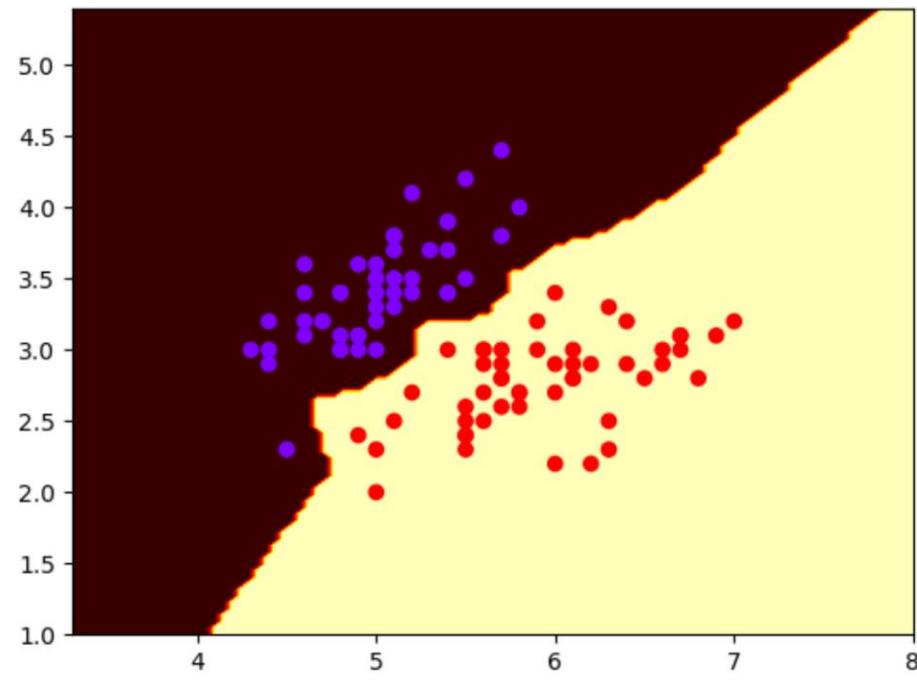
```
((100, 2), (100,))
```

```
classifier_1 = KNeighborsClassifier(n_neighbors=1).fit(X, y)  
classifier_5 = KNeighborsClassifier(n_neighbors=5).fit(X, y)
```

# Пример: KNN (Python) (2)

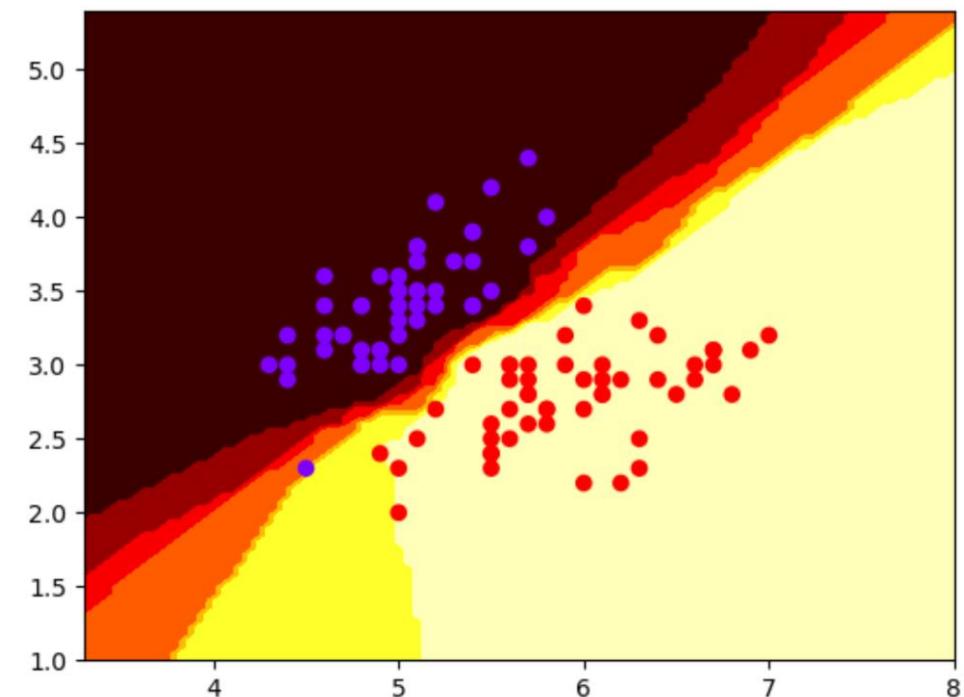
```
DecisionBoundaryDisplay.from_estimator(classifier_1, X, cmap="hot")
plt.scatter(*X.T, c=y, cmap="rainbow")
```

<matplotlib.collections.PathCollection at 0x7fb1921b9fa0>



```
DecisionBoundaryDisplay.from_estimator(classifier_5, X, cmap="hot")
plt.scatter(*X.T, c=y, cmap="rainbow")
```

<matplotlib.collections.PathCollection at 0x7fb18ff84220>



# Логистическая регрессия

Почему нельзя моделировать вероятность отклика  $p$  как непрерывный отклик с помощью линейной регрессии?

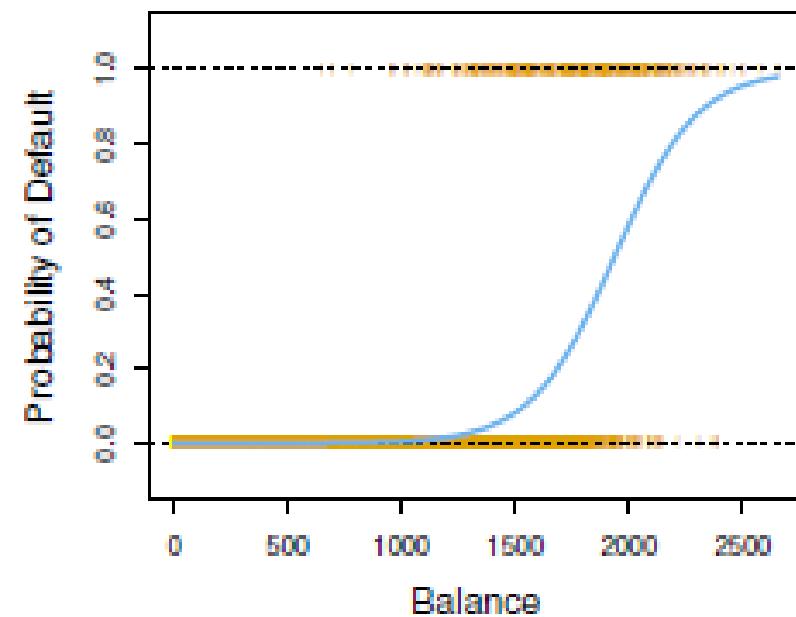
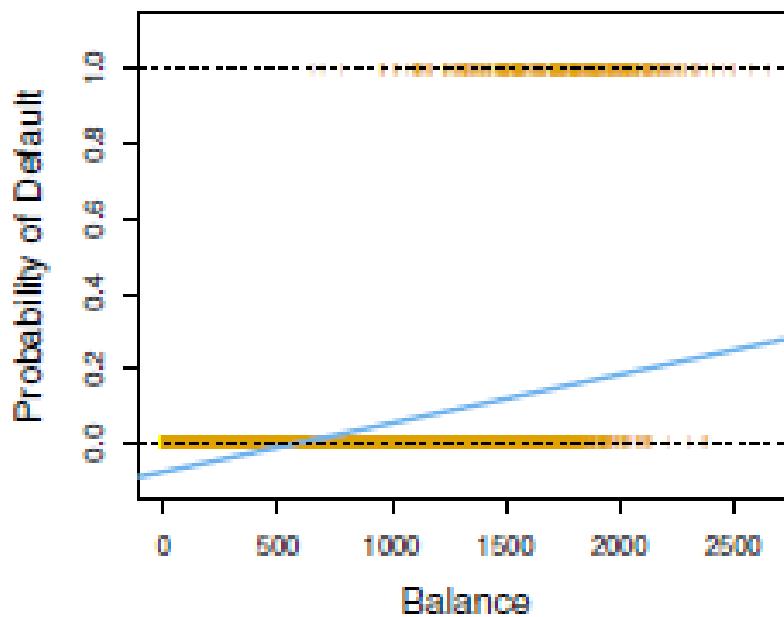
OLS Reg:  $Y_i = \beta_0 + \beta_1 X_{1i} + \varepsilon_i$

- Если целевая переменная категориальная, как представить ее в виде числовой?
- Если целевая закодирована (1=Yes and 0=No) а результат модели 0.5 или 1.1 или -0.4, что это означает?
- Если переменная имеет только два значения (или несколько), имеет ли смысл требовать постоянства дисперсии или нормальности ошибок?

Linear Prob. Model:  $p_i = \beta_0 + \beta_1 X_{1i}$

- Вероятность ограничена, а линейная функция принимает любые значения.
- Принимая во внимание ограниченность вероятности, можно ли предполагать линейную связь между  $X$  и  $p$ ?
- Можно ли предполагать ошибку с постоянной дисперсией?
- Что такое наблюдаемая вероятность для конкретного наблюдения? 0 и 1?

# Линейная vs логистическая регрессия



Логистическая регрессия гарантирует, что оценка  $p(X)$  находится в диапазоне между 0 и 1.

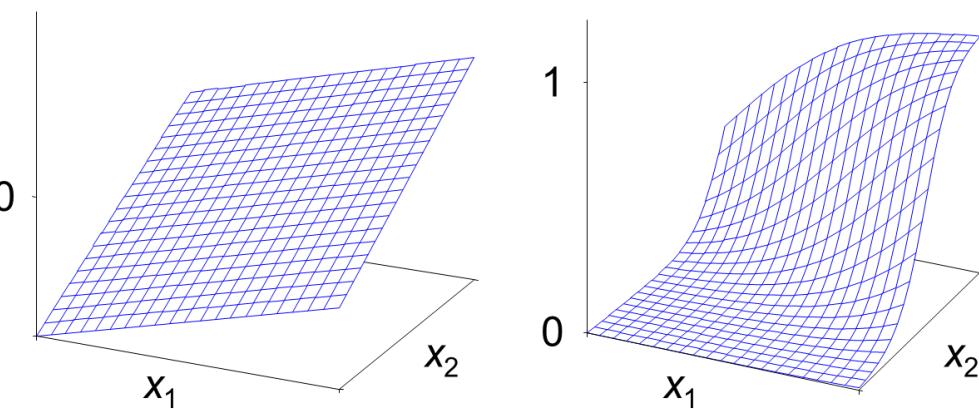
# Логистическая регрессия

**Уравнение логистической регрессии:** **Функция связи** (логит) и обратная ей (логистическая):

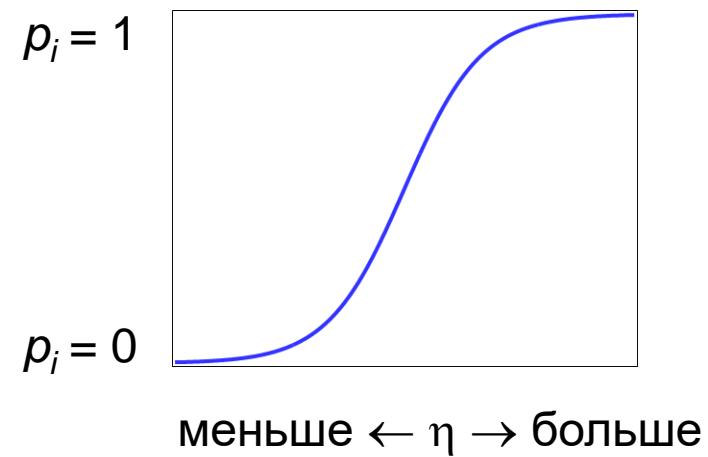
$$\text{вероятность}$$
$$\logit(p_i) = \beta_0 + \beta_1 x_{1i} + \dots + \beta_k x_{ki}$$
$$\text{параметр}$$
$$\text{предиктор}$$

Основное предположение линейной логистической регрессии (линейная зависимость логита от предикторов):

$$\logit(p)$$



$$\logit(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \eta$$
$$\Leftrightarrow p_i = \frac{1}{1+e^{-\eta}}$$



Ограничивает значение отклика

# Максимальное правдоподобие

Максимальное правдоподобие для оценки параметров.

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)).$$

Это правдоподобие дает вероятность наблюдаемых нулей и единиц в данных. Мы выбираем параметры модели, чтобы максимизировать вероятность наблюдаемых данных.

$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}$$

	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-10.8690	0.4923	-22.08	< 0.0001
balance	0.0057	0.0002	24.74	< 0.0001
income	0.0030	0.0082	0.37	0.7115
student [Yes]	-0.6468	0.2362	-2.74	0.0062

# Логистическая регрессия с более чем двумя классами

- Логистическая регрессия с двумя классами легко обобщается на более чем два класса.

$$\Pr(Y = k|X) = \frac{e^{\beta_{0k} + \beta_{1k}X_1 + \dots + \beta_{pk}X_p}}{\sum_{\ell=1}^K e^{\beta_{0\ell} + \beta_{1\ell}X_1 + \dots + \beta_{p\ell}X_p}}$$

- Здесь для каждого класса существует линейная функция.
- Мноклассовая логистическая регрессия также называется *мультиномиальной регрессией*.

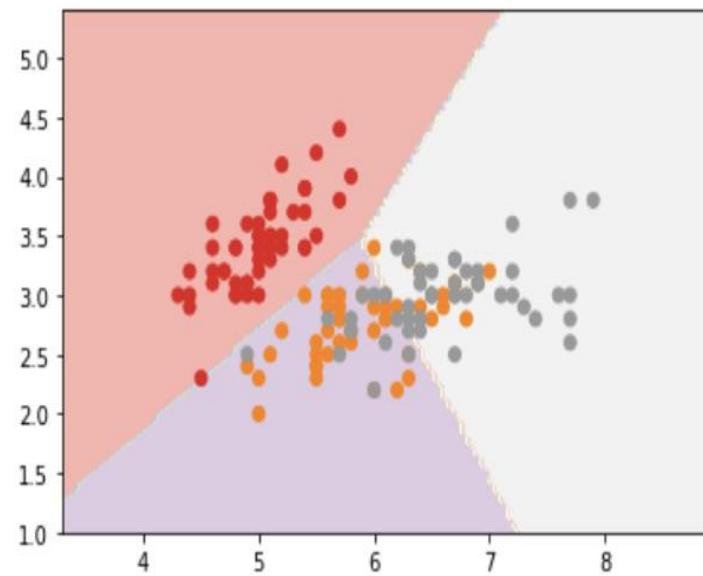
# Пример

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.inspection import DecisionBoundaryDisplay

iris = datasets.load_iris()
X = iris.data[:, :2]
Y = iris.target

logreg = LogisticRegression()
logreg.fit(X, Y)

DecisionBoundaryDisplay.from_estimator(
    logreg, X, cmap="Pastel1")
plt.scatter(X[:, 0], X[:, 1], c=Y, cmap="Set1")
plt.show()
```



# Отношение шансов

- Показывает как изменится отношение шансов при изменении i-ой переменной на 1 unit (равно exp от коэф.)

$$\text{logit}(\hat{p}) = \log(odds) = \beta_0 + \beta_i * x_i + \sum_{j \neq i} \beta_j * x_j$$

$$odds = \exp(\beta_0 + \beta_i * x_i + \sum_{j \neq i} \beta_j * x_j)$$

$$\text{logit}(\hat{p}') = \log(odds) = \beta_0 + \beta_i * (x_i + 1) + \sum_{j \neq i} \beta_j * x_j$$

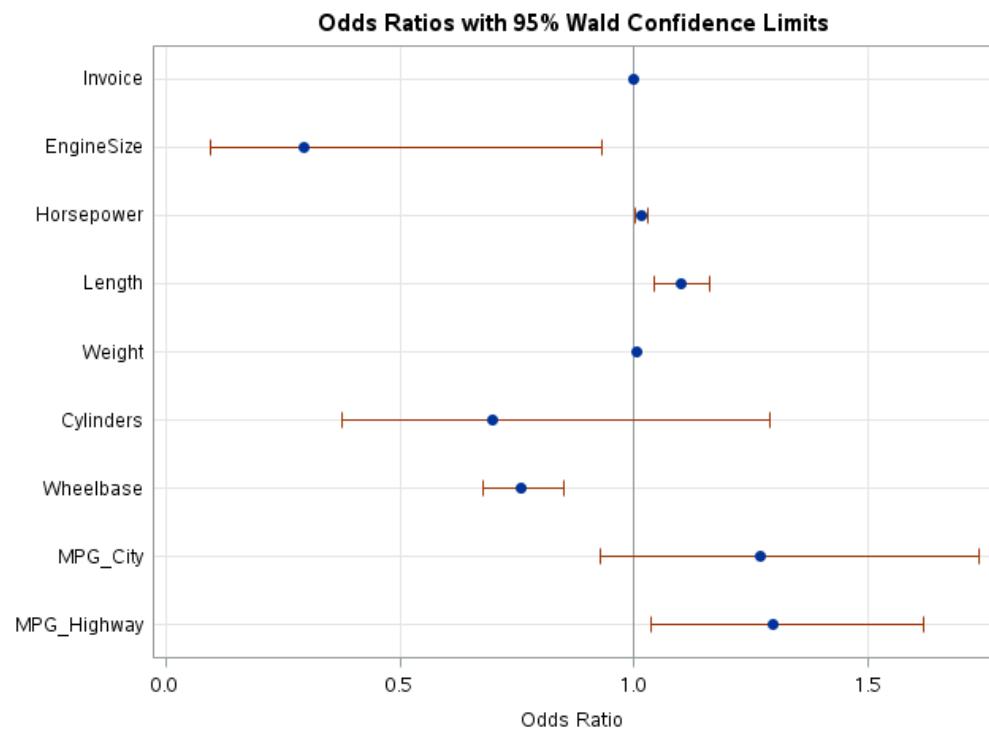
$$odds' = \exp(\beta_0 + \beta_i * (x_i + 1) + \sum_{j \neq i} \beta_j * x_j)$$

$$odds\ ratio = odds' / odds = \boxed{\exp(\beta_i)}$$

Больше 1 – отношение шансов увеличивается, если меньше, то уменьшается

# Отношение шансов (пример)

Odds Ratio Estimates			
Effect	Point Estimate	95% Wald Confidence Limits	
Invoice	1.000	1.000	1.000
EngineSize	0.295	0.094	0.931
Horsepower	1.016	1.003	1.029
Length	1.100	1.044	1.160
Weight	1.005	1.004	1.007
Cylinders	0.696	0.376	1.289
Wheelbase	0.757	0.676	0.849
MPG_City	1.270	0.929	1.736
MPG_Highway	1.295	1.036	1.618



# Категориальные предикторы

## □ Схемы кодировки:

- Effect coding (относительно «среднего»)

<u>CLASS</u>	<u>Value</u>	<u>Label</u>	<u>1</u>	<u>2</u>
<b>IncLevel</b>	1	Low Income	1	0
	2	Medium Income	0	1
	3	High Income	-1	-1

- Reference coding (относительно «базового»)

<u>CLASS</u>	<u>Value</u>	<u>Label</u>	<u>1</u>	<u>2</u>
<b>IncLevel</b>	1	Low Income	1	0
	2	Medium Income	0	1
	3	High Income	0	0

# Effect Coding: Пример

$$\text{logit}(p) = \beta_0 + \beta_1 * D_{\text{Low income}} + \beta_2 * D_{\text{Medium income}}$$

$\beta_0$ = Средний логит по всем категориям

$\beta_1$ = Разница между логитом для Low income и средним логитом

$\beta_2$ = разница между Medium income и средним логитом

Analysis of Maximum Likelihood Estimates						
Parameter	DF	Estimate	Standard Error	Chi-Square	Wald Chi-Square	Pr > ChiSq
Intercept	1	-0.5363	0.1015	27.9143		<.0001
IncLevel 1	1	-0.2259	0.1481	2.3247		0.1273
IncLevel 2	1	-0.2200	0.1447	2.3111		0.1285

# Reference Coding: Пример

$$\text{logit}(p) = \beta_0 + \beta_1 * D_{\text{Low income}} + \beta_2 * D_{\text{Medium income}}$$

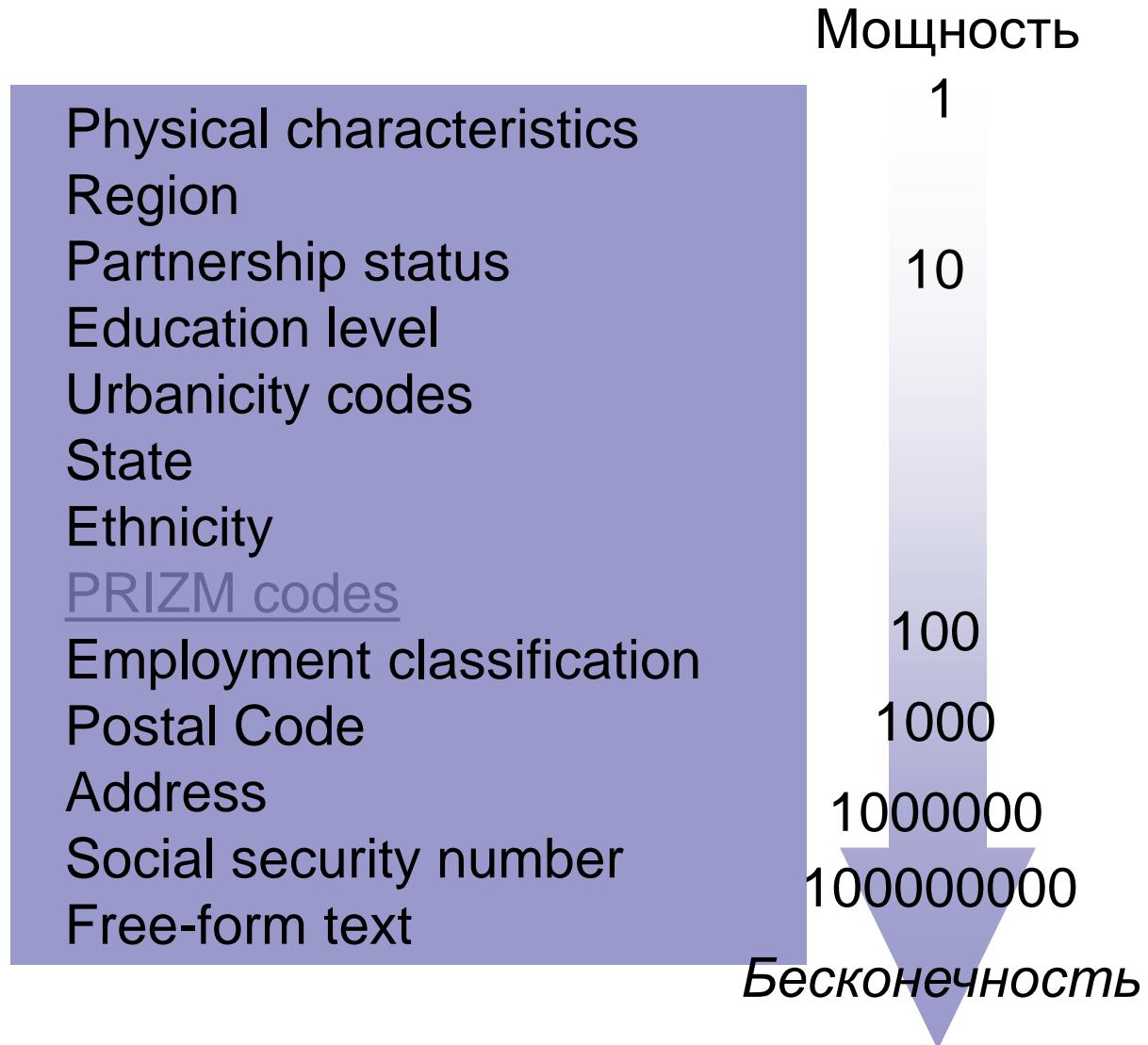
$\beta_0$ = Логит для High

$\beta_1$ = Разница логитов между Low и High

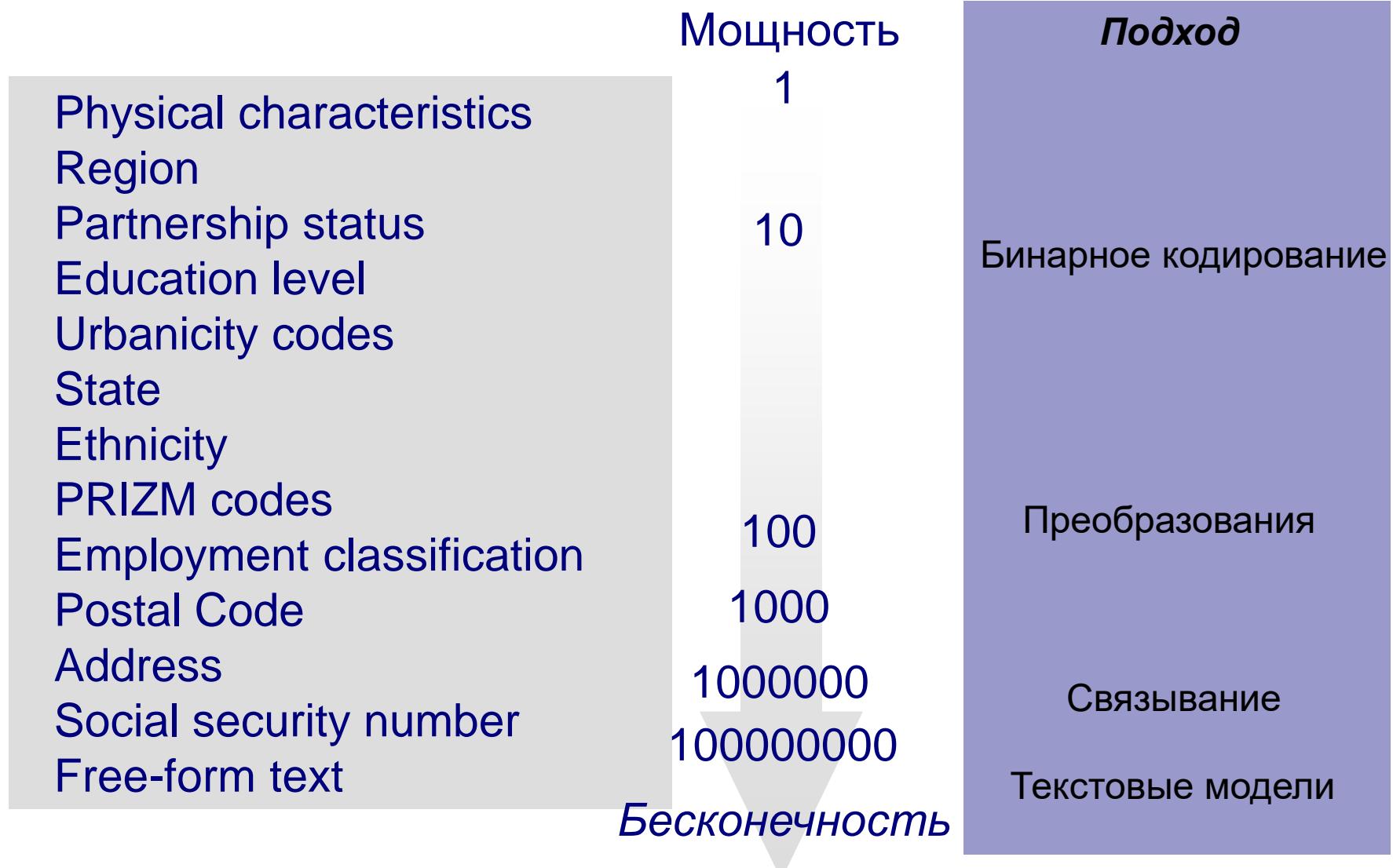
$\beta_2$ = Разница логитов между Medium и High

Analysis of Maximum Likelihood Estimates						
Parameter	DF	Estimate	Standard Error	Chi-Square	Wald	Pr > ChiSq
Intercept	1	-0.0904	0.1608	0.3159		0.5741
IncLevel 1	1	-0.6717	0.2465	7.4242		0.0064
IncLevel 2	1	-0.6659	0.2404	7.6722		0.0056

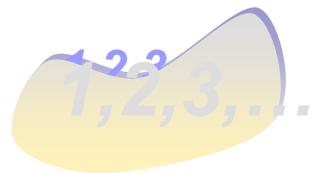
# Категориальные признаки



# Категориальные признаки



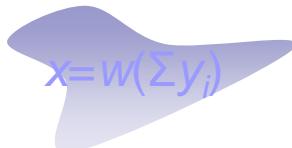
# Подходы к перекодировке



Случайное кодирование



Бинарное кодирование



Преобразование  
с учётом отклика

[https://github.com/scikit-learn-contrib/category\\_encoders](https://github.com/scikit-learn-contrib/category_encoders)

## Unsupervised:

- Backward Difference Contrast [2][3]
- BaseN [6]
- Binary [5]
- Gray [14]
- Count [10]
- Hashing [1]
- Helmert Contrast [2][3]
- Ordinal [2][3]
- One-Hot [2][3]
- Rank Hot [15]
- Polynomial Contrast [2][3]
- Sum Contrast [2][3]

## Supervised:

- CatBoost [11]
- Generalized Linear Mixed Model [12]
- James-Stein Estimator [9]
- LeaveOneOut [4]
- M-estimator [7]
- Target Encoding [7]
- Weight of Evidence [8]
- Quantile Encoder [13]
- Summary Encoder [13]

# Преобразование с учётом отклика

<i>Level</i>	$N_i$	$\Sigma Y_i$	$p_i$
A	1562	430	0.28
B	970	432	0.45
C	223	45	0.20
D	111	36	0.32
E	85	23	0.27
F	50	20	0.40
G	23	8	0.35
H	17	5	0.29
I	12	6	0.50
J	5	5	1.00

# Преобразование с учётом отклика

<i>Level</i>	$N_i$	$\Sigma Y_i$	$p_i$
J	5	5	1.00
I	12	6	0.50
B	970	432	0.45
F	50	20	0.40
G	23	8	0.35
D	111	36	0.32
H	17	5	0.29
A	1562	430	0.28
E	85	23	0.27
C	223	45	0.20

# Преобразование с учётом отклика

$X$	$N_i$	$\Sigma Y_i$	$p_i$
1	5	5	1.00
2	12	6	0.50
3	970	432	0.45
4	50	20	0.40
5	23	8	0.35
6	111	36	0.32
7	17	5	0.29
8	1562	430	0.28
9	85	23	0.27
10	223	45	0.20

# Weight of Evidence

<i>Level</i>	$N_i$	$\Sigma Y_i$	$p_i$	$\log(p_i/1-p_i)$
J	5	5	1.00	.
I	12	6	0.50	0.00
B	970	432	0.45	-0.10
F	50	20	0.40	-0.18
G	23	8	0.35	-0.27
D	111	36	0.32	-0.32
H	17	5	0.29	-0.38
A	1562	430	0.28	-0.42
E	85	23	0.27	-0.43
C	223	45	0.20	-0.60

# Кластеризация уровней

<i>Level</i>	$N_i$	$\Sigma Y_i$	$p_i$	$\log(p_i/1-p_i)$
J	5	5	1.00	.
I	12	6	0.50	0.00
B	970	432	0.45	-0.10
F	50	20	0.40	-0.18
G	23	8	0.35	-0.27
D	111	36	0.32	-0.32
H	17	5	0.29	-0.38
A	1562	430	0.28	-0.42
E	85	23	0.27	-0.43
C	223	45	0.20	-0.60

# Кластеризация уровней

<i>Level</i>	$N_i$	$\Sigma Y_i$	$p_i$	$\log(p_i/1-p_i)$
CL1	1037	463	0.45	-0.09
CL2	134	44	0.33	-0.31
CL3	1664	458	0.28	-0.42
CL4	223	45	0.20	-0.60

# Сглаженный Weight of Evidence

<i>Level</i>	$N_i$	$\Sigma Y_i$	$p_i$	$\log(p_i/1-p_i)$
J	5	5	1.00	.
I	12	6	0.50	0.00
B	970	432	0.45	-0.10
F	50	20	0.40	-0.18
G	23	8	0.35	-0.27
D	111	36	0.32	-0.32
H	17	5	0.29	-0.38
A	1562	430	0.28	-0.42
E	85	23	0.27	-0.43
C	223	45	0.20	-0.60

# Сглаженный Weight of Evidence

<i>Level</i>	<i>N<sub>i</sub></i>	$\Sigma Y_i$	<i>p<sub>i</sub></i>	$\log(p_i/1-p_i)$
J	5	+24	0.45	-0.09
I	12	+8	0.39	-0.19
B	970	+24	0.44	-0.10
F	50	+24	0.38	-0.22
G	23	+24	0.34	-0.29
D	111	+24	0.33	-0.32
H	17	+24	0.32	-0.33
A	1562	+24	0.28	-0.42
E	85	+24	0.28	-0.40
C	223	+24	0.21	-0.56

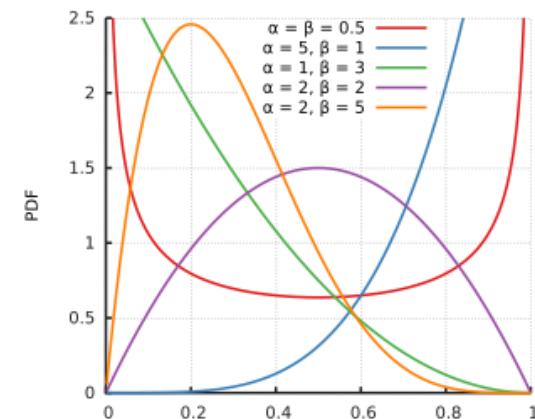
# Бета распределение для кодирования поведения категориальных предикторов

Двухпараметрическое (альфа и бета параметры), используется для описания случайных величин, заданных на интервале.

$$\text{Мат. ожидание: } \frac{a}{a+b}$$

Является сопряжённым априорным распределением для распределения Бернулли, биномиального и геометрического (принадлежат одному семейству распределений)...

Задача – найти распределение параметра  $\theta$  (рассматриваемого как случайная величина) по имеющемуся наблюдению  $x$ , где  $p(x|\theta)$  – функция правдоподобия,  $p(\theta)$  – априорное распределение параметра

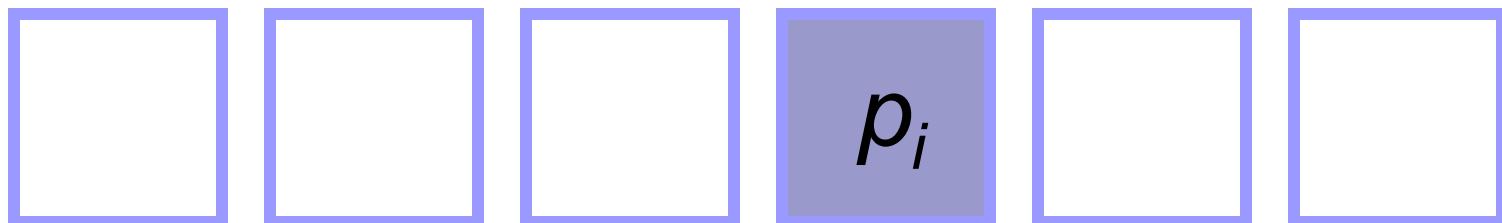


$$p(\theta|x) = \frac{p(x|\theta) p(\theta)}{\int_{\text{range } \theta} p(x|\theta) p(\theta) d\theta}$$

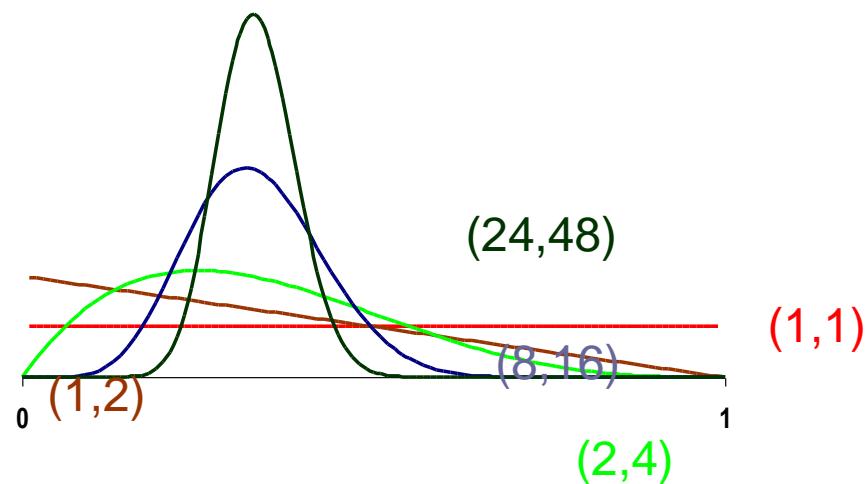
# Сглаженный WOE

Значения  $X$

$X$



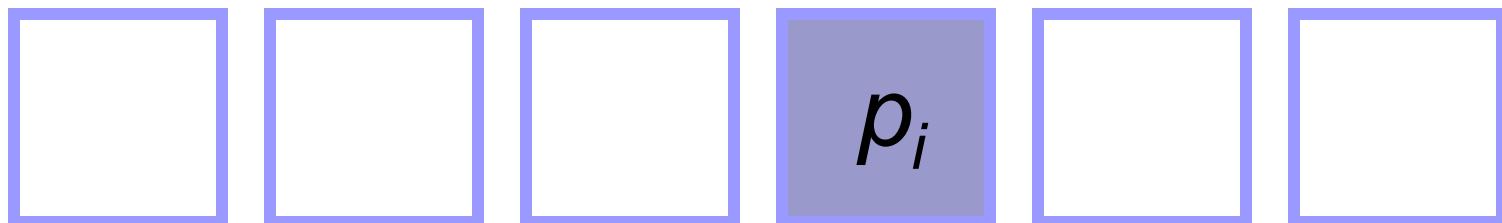
$$p_i \sim \text{Beta}(a, b)$$



# Сглаженный WOE

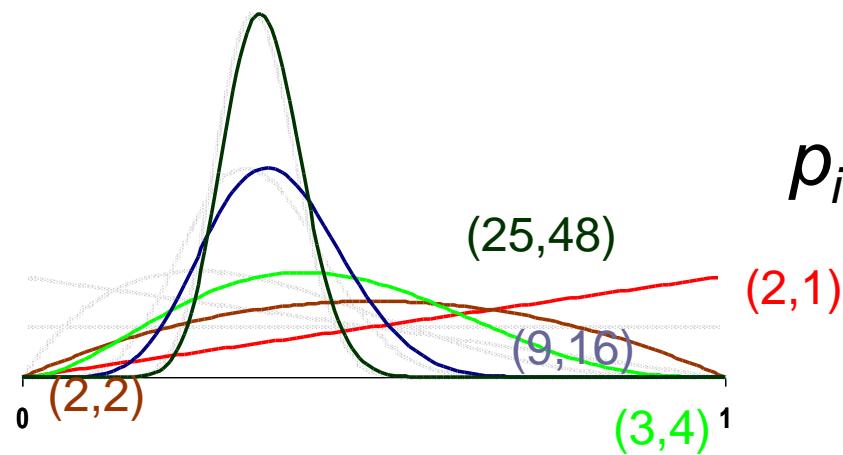
Значение  $X$

$X$



$$p_i \sim \text{Beta}(a, b)$$

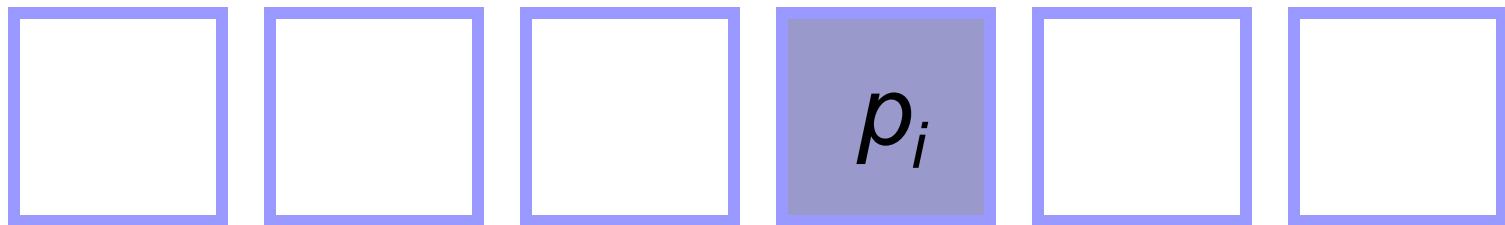
$$p_i | Y=1 \sim \text{Beta}(a+1, b)$$



# Сглаженный WOE

Значения  $X$

$X$

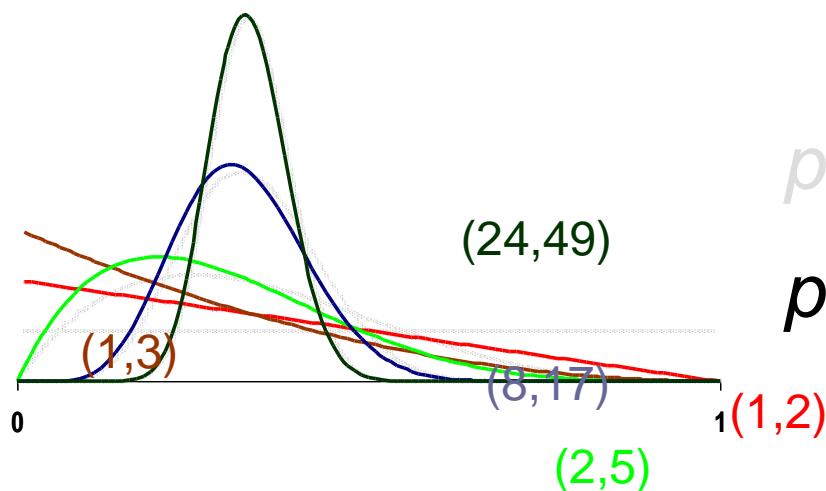


$p_i$

$p_i \sim \text{Beta}(a, b)$

$p_i | Y=1 \sim \text{Beta}(a+1, b)$

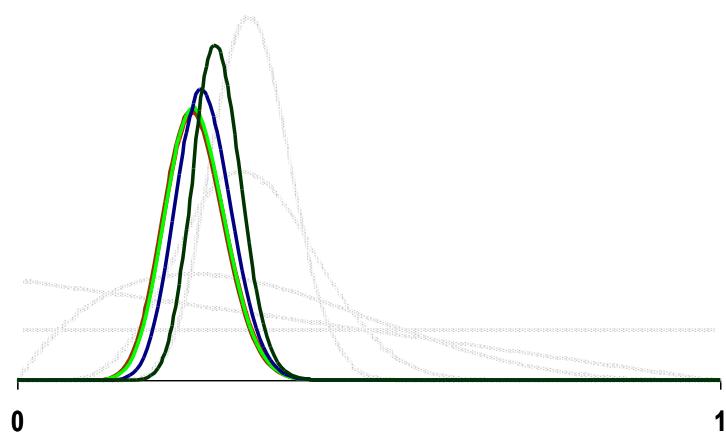
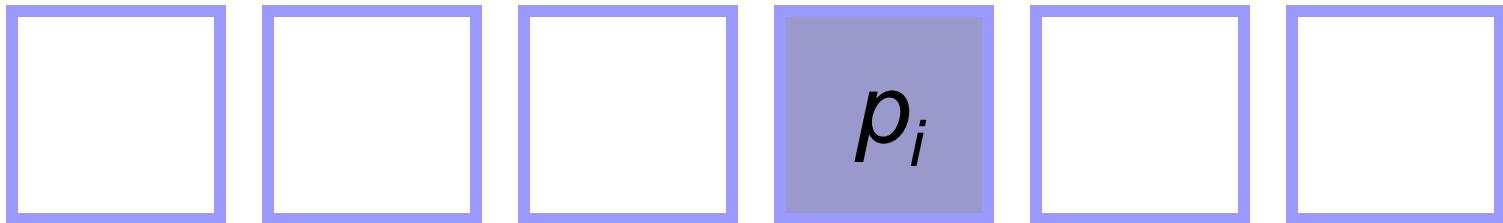
$p_i | Y=0 \sim \text{Beta}(a, b+1)$



# Сглаженный WOE

## Значения $X$

$X$



Наблюдаемое число  
событий

Априорная  
вероятность события

$$SWOE = \ln \left( \frac{\#events + c\rho_1}{\#nonevents + c(1 - \rho_1)} \right)$$

Параметр регуляризации

$$p_i \sim \text{Beta}(a, b)$$

$$p_i | \text{data} \sim \text{Beta}(a+n_1, b+n_0)$$

Оценка

$$E(p_i) \quad (n_1+a)/(n_1+n_0+a+b)$$

WOE

$$\log(E(p_i)/(1-E(p_i)))$$

$$IV = \sum_i (P(X = x_i | Y = 1) - P(X = x_i | Y = 0)) \cdot WOE_i$$

# Пример

```
import pandas as pd
import numpy as np
mydata = pd.read_csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
```

```
mydata.head(10)
```

	admit	gre	gpa	rank
0	0	380	3.61	3
1	1	660	3.67	3
2	1	800	4.00	1
3	1	640	3.19	4
4	0	520	2.93	4
5	1	760	3.00	2
6	1	560	2.98	1
7	0	400	3.08	2
8	1	540	3.39	3
9	0	700	3.92	2

```
def iv_woe(data, target, bins=10, show_woe=False):

    #Empty Dataframe
    newDF,woeDF = pd.DataFrame(), pd.DataFrame()

    #Extract Column Names
    cols = data.columns

    #Run WOE and IV on all the independent variables
    for ivars in cols[~cols.isin([target])]:
        if (data[ivars].dtype.kind in 'bifc') and (len(np.unique(data[ivars]))>10):
            binned_x = pd.qcut(data[ivars], bins, duplicates='drop')
            d0 = pd.DataFrame({'x': binned_x, 'y': data[target]})

        else:
            d0 = pd.DataFrame({'x': data[ivars], 'y': data[target]})

        d0 = d0.astype({"x": str})
        d = d0.groupby("x", as_index=False, dropna=False).agg({"y": ["count", "sum"]})
        d.columns = ['Cutoff', 'N', 'Events']
        d['% of Events'] = np.maximum(d['Events'], 0.5) / d['Events'].sum()
        d['Non-Events'] = d['N'] - d['Events']
        d['% of Non-Events'] = np.maximum(d['Non-Events'], 0.5) / d['Non-Events'].sum()
        d['WoE'] = np.log(d['% of Non-Events']/d['% of Events'])
        d['IV'] = d['WoE'] * (d['% of Non-Events']-d['% of Events'])
        d.insert(loc=0, column='Variable', value=ivars)
        print("Information value of " + ivars + " is " + str(round(d['IV'].sum(),6)))
        temp =pd.DataFrame({"Variable" : [ivars], "IV" : [d['IV'].sum()]}, columns = ["Variable", "IV"])
        newDF=pd.concat([newDF,temp], axis=0)
        woeDF=pd.concat([woeDF,d], axis=0)

    #Show WOE Table
    if show_woe == True:
        print(d)
    return newDF, woeDF
```

```
iv, woe = iv_woe(data = mydata, target = 'admit', bins=10, show_woe = True)
```

# Пример

woe

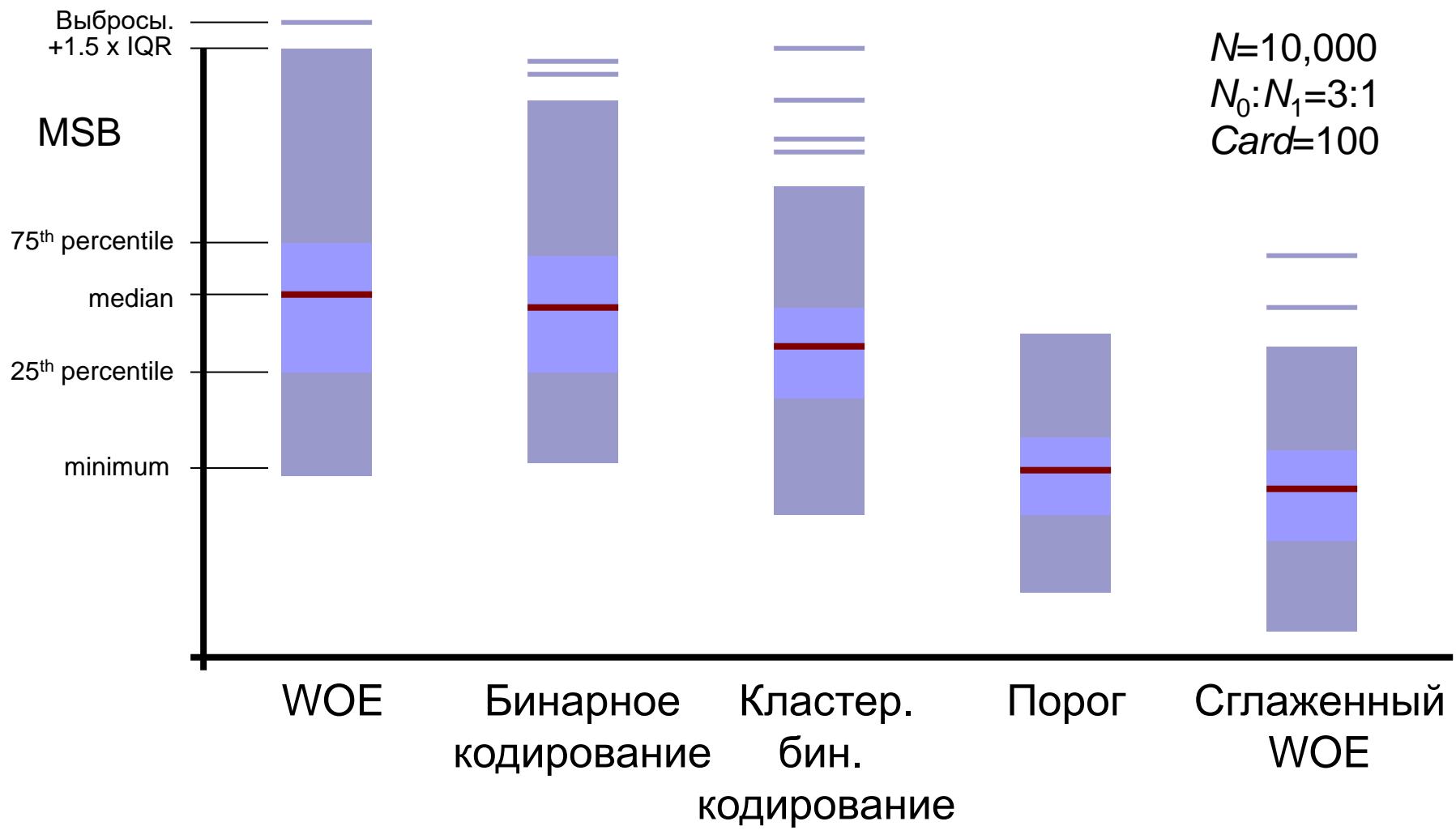
|--|--|--|--|--|--|--|--|--|--|

	Variable	Cutoff	N	Events	% of Events	Non-Events	% of Non-Events	WoE	IV
0	gre	(219.999, 440.0]	48	6	0.047244	42	0.153846	1.180625	0.125857
1	gre	(440.0, 500.0]	51	12	0.094488	39	0.142857	0.413370	0.019994
2	gre	(500.0, 520.0]	24	10	0.078740	14	0.051282	-0.428812	0.011774
3	gre	(520.0, 560.0]	51	15	0.118110	36	0.131868	0.110184	0.001516
4	gre	(560.0, 580.0]	29	6	0.047244	23	0.084249	0.578450	0.021406
5	gre	(580.0, 620.0]	53	21	0.165354	32	0.117216	-0.344071	0.016563
6	gre	(620.0, 660.0]	45	17	0.133858	28	0.102564	-0.266294	0.008333
7	gre	(660.0, 680.0]	20	9	0.070866	11	0.040293	-0.564614	0.017262
8	gre	(680.0, 740.0]	44	12	0.094488	32	0.117216	0.215545	0.004899
9	gre	(740.0, 800.0]	35	19	0.149606	16	0.058608	-0.937135	0.085278
0	gpa	(2.259, 2.9]	43	8	0.062992	35	0.128205	0.710622	0.046342
1	gpa	(2.9, 3.048]	37	11	0.086614	26	0.095238	0.094917	0.000819
2	gpa	(3.048, 3.17]	42	8	0.062992	34	0.124542	0.681634	0.041955
3	gpa	(3.17, 3.31]	42	10	0.078740	32	0.117216	0.397866	0.015308
4	gpa	(3.31, 3.395]	36	8	0.062992	28	0.102564	0.487478	0.019290
5	gpa	(3.395, 3.494]	40	14	0.110236	26	0.095238	-0.146246	0.002193
6	gpa	(3.494, 3.61]	41	16	0.125984	25	0.091575	-0.318998	0.010976
7	gpa	(3.61, 3.752]	39	20	0.157480	19	0.069597	-0.816578	0.071764
8	gpa	(3.752, 3.94]	42	13	0.102362	29	0.106227	0.037062	0.000143
9	gpa	(3.94, 4.0]	38	19	0.149606	19	0.069597	-0.765285	0.061230
0	rank	1	61	33	0.259843	28	0.102564	-0.929588	0.146204
1	rank	2	151	54	0.425197	97	0.355311	-0.179558	0.012548
2	rank	3	121	28	0.220472	93	0.340659	0.435110	0.052295
3	rank	4	67	12	0.094488	55	0.201465	0.757142	0.080997

iv

	Variable	IV
0	gre	0.312882
0	gpa	0.270020
0	rank	0.292044

# Имитационный эксперимент



# Дискриминантный анализ

- Этот подход состоит в том, чтобы моделировать распределение  $X$  в каждом из классов по отдельности, а затем использовать *теорему Байеса* для получения условных вероятностей  $\Pr(Y|X)$ .
- Когда мы используем нормальные (гауссовские) распределения для каждого класса, это приводит к линейному или квадратичному дискриминантному анализу.
- теорема Байеса :

$$\Pr(Y = k|X = x) = \frac{\Pr(X = x|Y = k) \cdot \Pr(Y = k)}{\Pr(X = x)}$$

- записывается так для дискриминантного анализа:

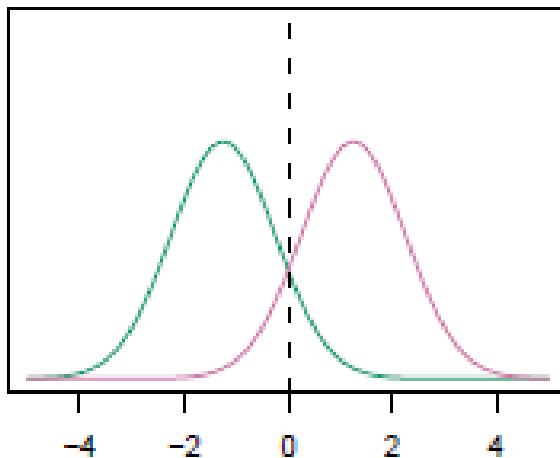
$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)},$$

где

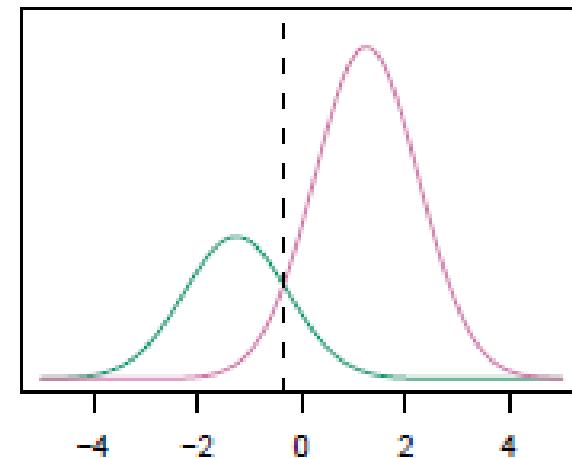
- $f_k(x) = \Pr(X = x|Y = k)$  – плотность вероятности для  $X$  в классе  $k$ .
- $\pi_k = \Pr(Y = k)$  – априорные вероятности для класса  $k$ .

# Классификация по максимальной апостериорной вероятности

$$\pi_1=.5, \quad \pi_2=.5$$



$$\pi_1=.3, \quad \pi_2=.7$$



- классифицируем новое наблюдение в соответствии с максимальной плотностью.
- Когда априорные вероятности различны, мы учитываем их и сравниваем  $\pi_k f_k(x)$ .
- На правом рисунке мы предпочитаем розовый класс - граница решения смешена влево.

# Почему используем дискриминантный анализ?

- Когда классы хорошо отделимы, оценки параметров модели логистической регрессии очень неустойчивы. Линейный дискриминантный анализ не страдает от этой проблемы.
- Если  $n$  мало и распределение предикторов  $X$  близко к нормальному в каждом из классов, линейная дискриминантная модель снова более устойчива, чем модель логистической регрессии.
- Линейный дискриминантный анализ хорошо применим, когда есть более двух классов.

# Одномерный линейный дискриминантный анализ

- Гауссова плотность имеет вид

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$$

- $\mu_k$  – это среднее, и  $\sigma_k^2$  - дисперсия (в классе  $k$ ). Предположим, что все  $\sigma_k = \sigma$  одинаковы.
- Подставив это выражение в формулу Байеса, получим выражение для  $p_k(x) = \Pr(Y = k | X = x)$ :

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_l}{\sigma}\right)^2}}$$

- эта форма может быть упрощена.

# Дискриминантные функции

- Чтобы классифицировать значение  $X = x$ , нам нужно понять, какая из величин  $p_k(x)$  наибольшая.
- Применяя логарифмирование и отбрасывая члены, не зависящие от  $k$ , мы видим, что это эквивалентно присвоению  $x$  классу с наибольшим *дискриминантной оценкой*:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

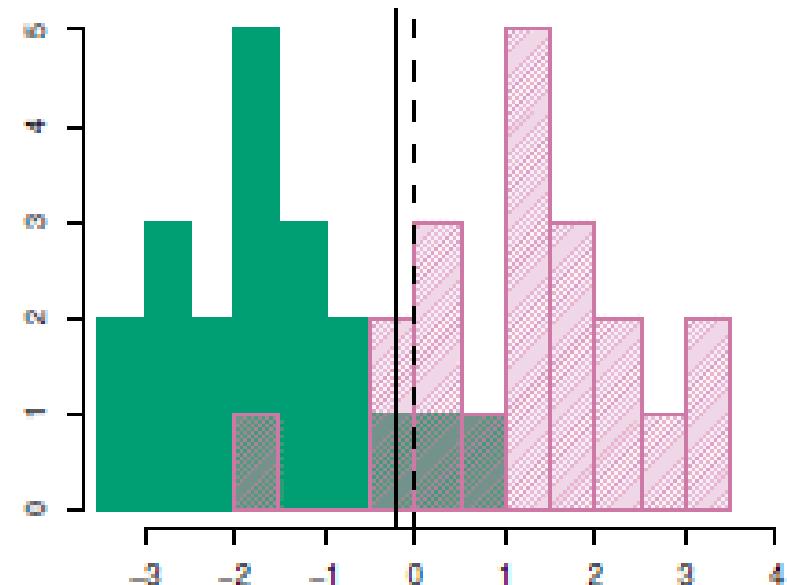
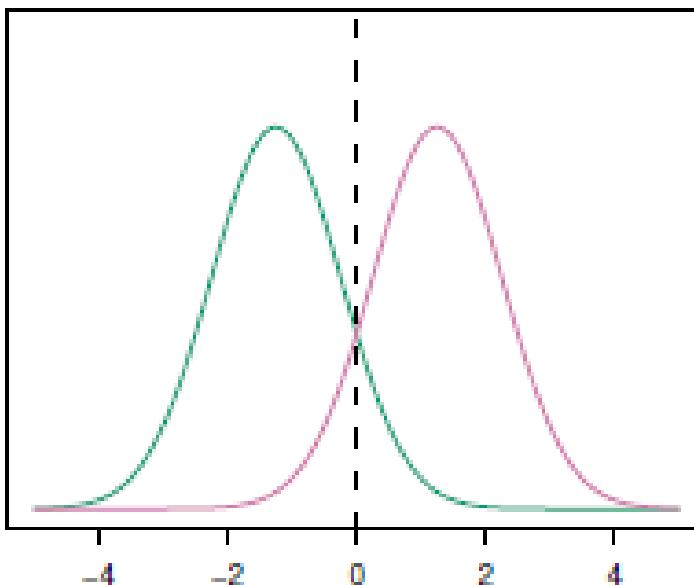
- Заметим, что  $\delta_k(x)$  является *линейной* функцией от  $x$ . Если число классов  $K = 2$  и  $\pi_1 = \pi_2 = 0,5$ , то может видеть, что *граница решения* находится в точке

$$x = \frac{\mu_1 + \mu_2}{2}.$$

# Пример

Пример:  $\mu_1 = -1.5$ ,  $\mu_2 = 1.5$ ,  $\pi_1 = \pi_2 = 0.5$  и  $\sigma^2 = 1$ .

Обычно мы не знаем этих параметров; у нас есть только обучающие данные. В этом случае мы просто оцениваем параметры.



# Оценка параметров

$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i: y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n - K} \sum_{k=1}^K \sum_{i: y_i=k} (x_i - \hat{\mu}_k)^2$$

$$= \sum_{k=1}^K \frac{n_k - 1}{n - K} \cdot \hat{\sigma}_k^2$$

формула для оцениваемой дисперсии в классе  $k$ .

$$\hat{\sigma}_k^2 = \frac{1}{n_k - 1} \sum_{i: y_i=k} (x_i - \hat{\mu}_k)^2$$

# Многомерный линейный дискриминантный анализ

Плотность:

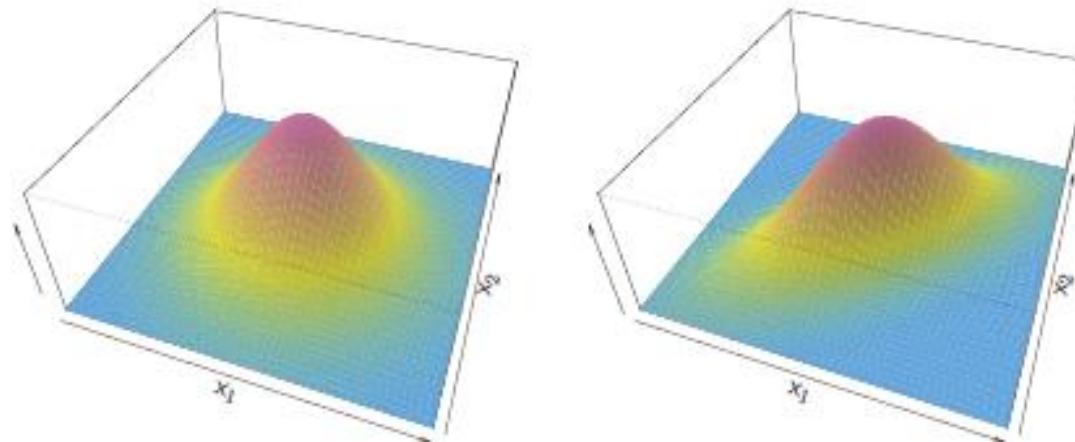
$$f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Функция дискриминанта:

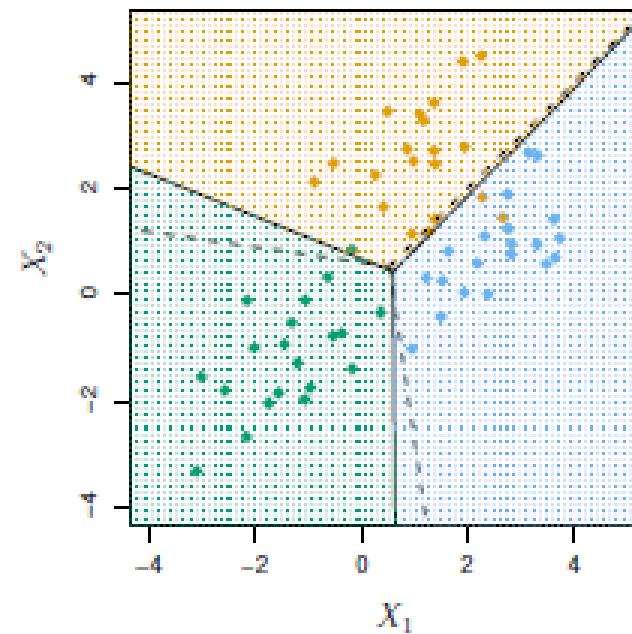
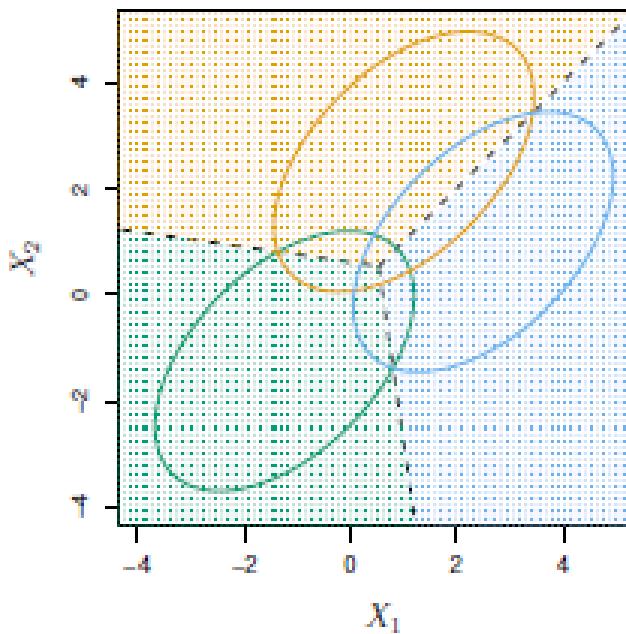
$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

- линейная функция

$$\delta_k(x) = c_{k0} + c_{k1}x_1 + c_{k2}x_2 + \dots + c_{kp}x_p$$



# Иллюстрация: $p = 2$ и $K = 3$ классов



- Здесь  $\pi_1 = \pi_2 = \pi_3 = 1/3$ .
- Пунктирные линии - границы решений Байеса (*Bayes decision boundaries*). Если бы они были известны, они бы дали наименьшее число ошибочных классификаций среди всех возможных классификаторов.

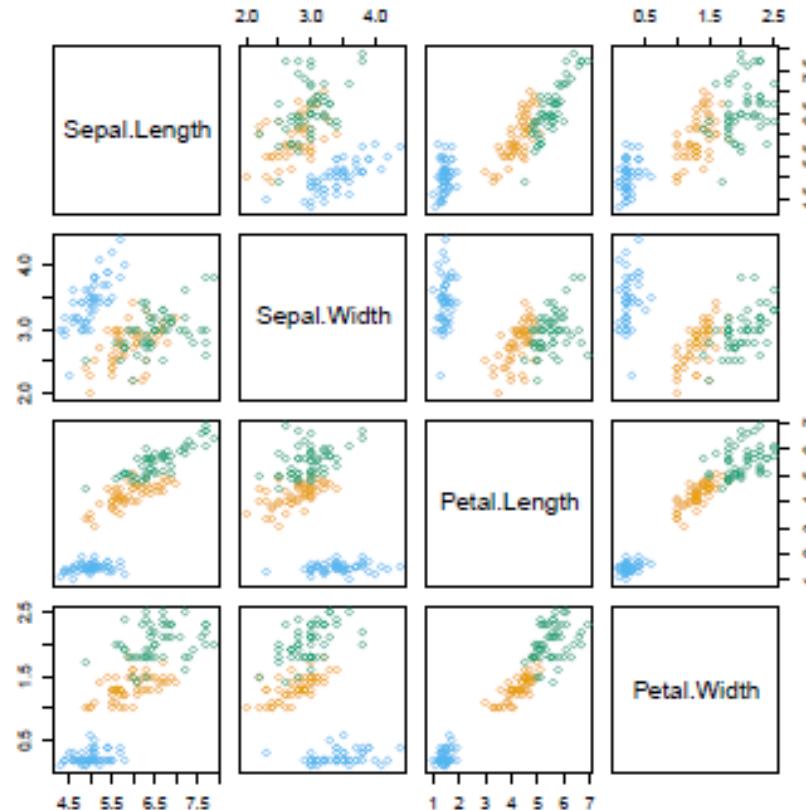
# Набор данных Fisher's Iris

4 переменных

3 класса

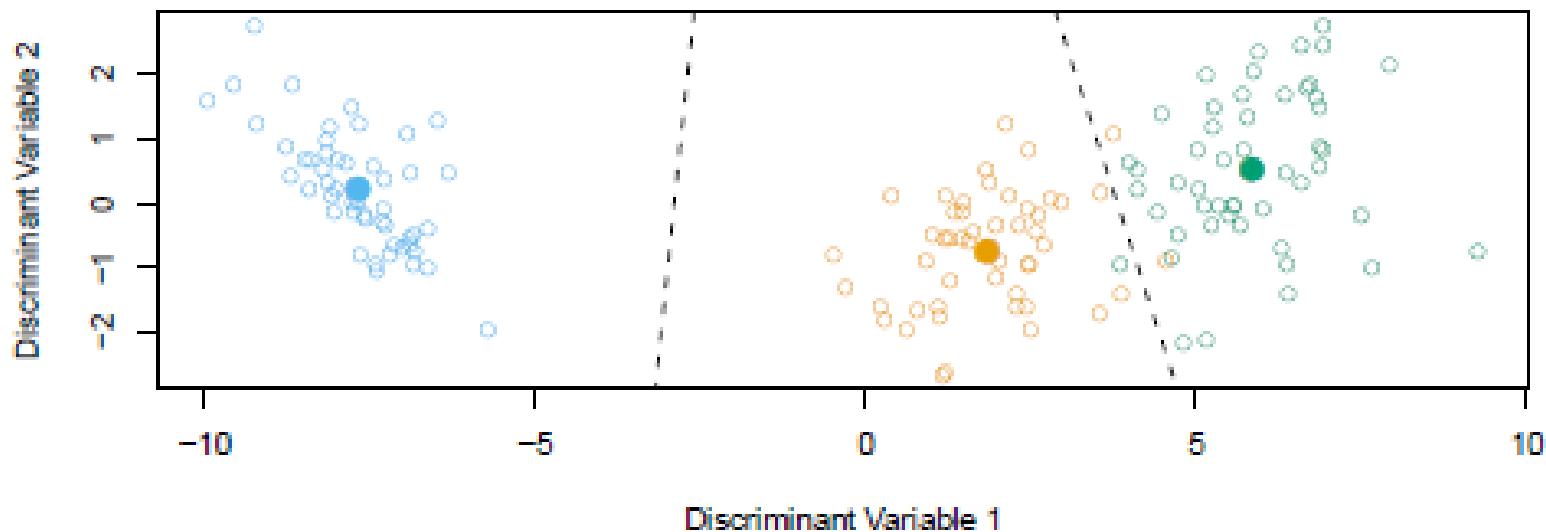
50 примеров на класс

- Setosa
- Versicolor
- Virginica



LDA правильно классифицирует все кроме трех из 150 обучающих примеров.

# Дискриминантная диаграмма



- Как в РСА строятся ортогональные главные компоненты по направлениям, наилучшим образом отделяющие классы.
- Когда есть  $K$  классов, линейный дискриминантный анализ можно рассматривать в  $K-1$ -мерной проекции.
- Даже при  $K > 3$  мы можем определить «лучшую» двумерную плоскость для визуализации дискриминантного правила.

# От $\delta_k(x)$ к вероятностям

- Как только мы получим оценки  $\hat{\delta}_k(x)$  мы можем преобразовать их в оценки вероятностей классов::

$$\widehat{\Pr}(Y = k|X = x) = \frac{e^{\hat{\delta}_k(x)}}{\sum_{l=1}^K e^{\hat{\delta}_l(x)}}.$$

- Таким образом, относим к классу, для которого наибольшая  $\hat{\delta}_k(x)$

$$\widehat{\Pr}(Y = k|X = x)$$

- Когда  $K = 2$ , мы классифицируем по правилу.

$$\widehat{\Pr}(Y = 2|X = x) \geq 0.5$$

# LDA – Пример

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis  
from sklearn.datasets import fetch_covtype
```

```
X, y = load_iris(return_X_y=True)
```

```
X = X[:, :2]  
X.shape, y.shape, np.unique(y)
```

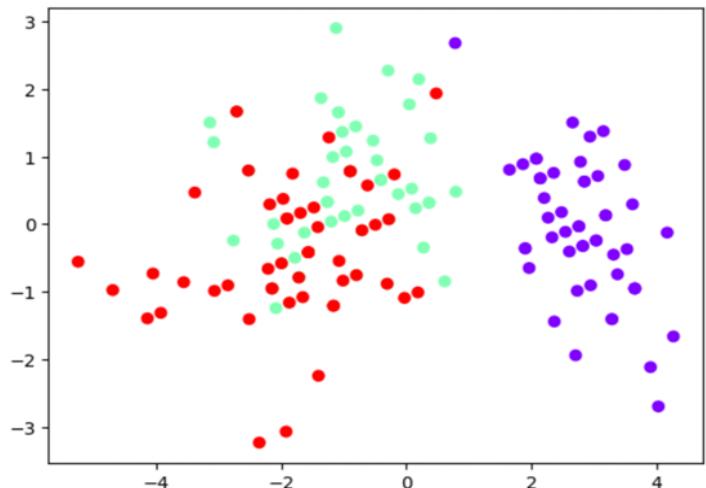
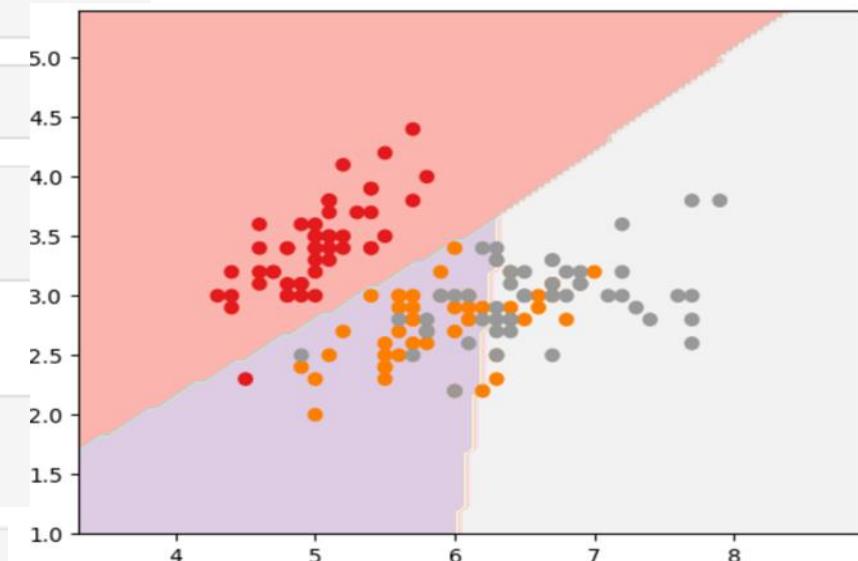
```
((150, 2), (150,), array([0, 1, 2]))
```

```
lda = LinearDiscriminantAnalysis(n_components=2)  
lda.fit(X, y)
```

```
DecisionBoundaryDisplay.from_estimator(lda, X, cmap="Pastel1")  
plt.scatter(*X.T, c=y, cmap="Set1")
```

```
transform = lda.transform(X)  
transform.shape  
(150, 2)
```

```
plt.scatter(*transform.T, c=y, cmap="rainbow")
```

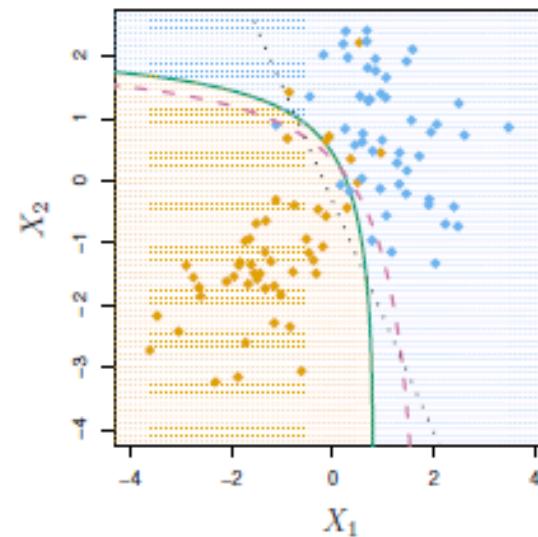
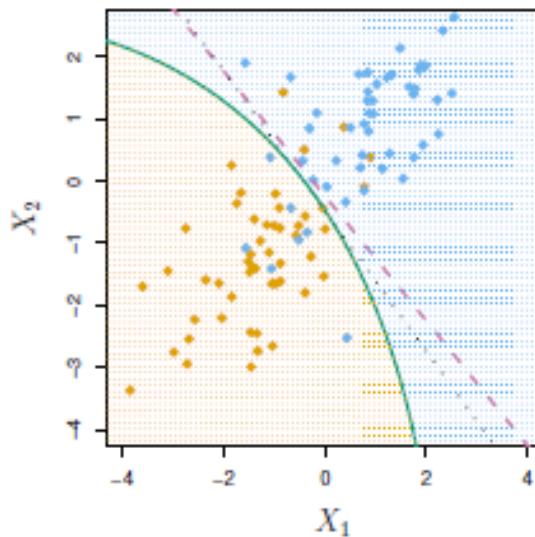


# Другие виды дискриминантного анализа

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

- Если  $f_k(x)$  - гауссовские плотности с одинаковой ковариационной матрицей в каждом классе, то мы имеем дело с линейным дискриминантным анализом.
- Изменяя формы для  $f_k(x)$ , мы получаем разные классификаторы.
  - С гауссским распределением, но разными  $\Sigma_k$  в каждом классе, мы получаем *квадратичный дискриминантный анализ*.
  - С  $f_k(x) = \prod_{j=1}^p f_{jk}(x_j)$  (условно независимая модель) в каждом классе мы получаем *наивный Байес*.
  - Для гауссиана это означает, что  $\Sigma_k$  диагональная.
  - Многие другие формы, предлагают конкретные модели плотности для  $f_k(x)$ , включая непараметрические подходы.

# Квадратичный дискриминантный анализ

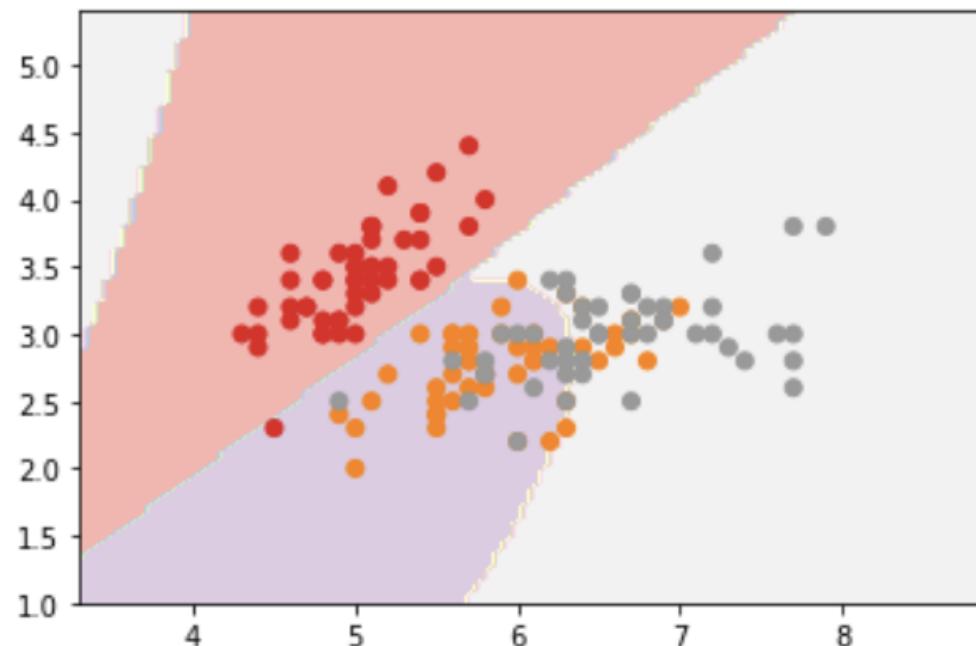


$$\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log \pi_k$$

Поскольку  $\Sigma_k$  различны, квадратичные члены значимы.

# Пример

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis  
qda = QuadraticDiscriminantAnalysis()  
qda.fit(X, y)  
DecisionBoundaryDisplay.from_estimator(qda, X, cmap="Pastel1")  
plt.scatter(*X.T, c=y, cmap="Set1")  
  
<matplotlib.collections.PathCollection at 0x167af6902e0>
```



# Логистическая регрессия по сравнению с LDA

- Для задачи двух классов можно показать, что для LDA

$$\log \left( \frac{p_1(x)}{1 - p_1(x)} \right) = \log \left( \frac{p_1(x)}{p_2(x)} \right) = c_0 + c_1 x_1 + \dots + c_p x_p$$

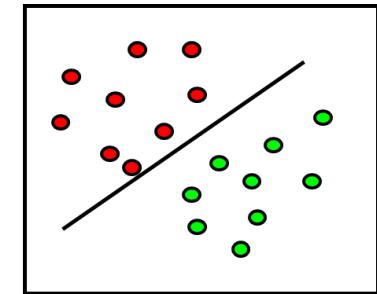
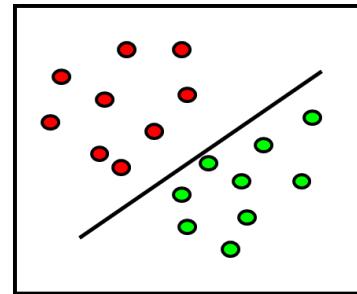
- Таким образом, он имеет ту же форму, что и логистическая регрессия.
- Разница заключается в том, как оцениваются параметры.
  - Логистическая регрессия использует условное правдоподобие, основанное на  $\text{Pr}(Y|X)$  (известное как *дискриминационное обучение*).
  - LDA использует полную правдоподобие, основанное на  $\text{Pr}(X, Y)$  (известное как *генеративное обучение*).
  - Несмотря на эти различия, на практике результаты часто очень похожи.
- Замечание: логистическая регрессия может также быть построена с квадратичными границами, такими как у QDA, путем явного включения квадратичных членов в модель.

# Метод опорных векторов (для задачи классификации)

Тренировочный набор

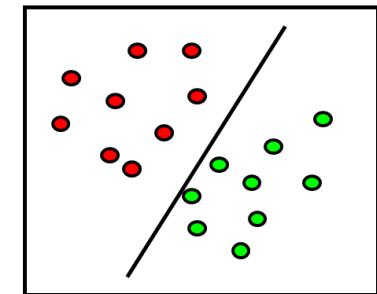
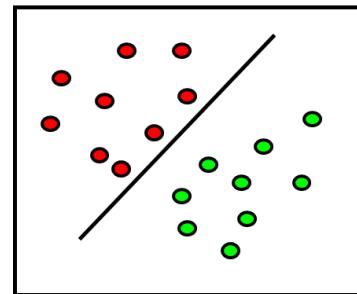
- Входные переменные

$$x_i \in \mathbb{R}^d$$



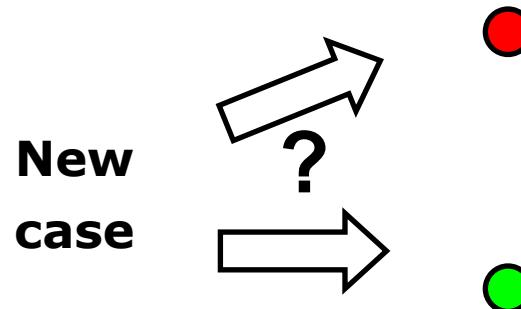
- Отклик:

$$y_i \in \{+1, -1\}$$



- Классификатор:

$$f: \mathbb{R}^d \mapsto \{+1, -1\}$$



# Линейное разделение

## □ Разделяющая гиперплоскость $H$

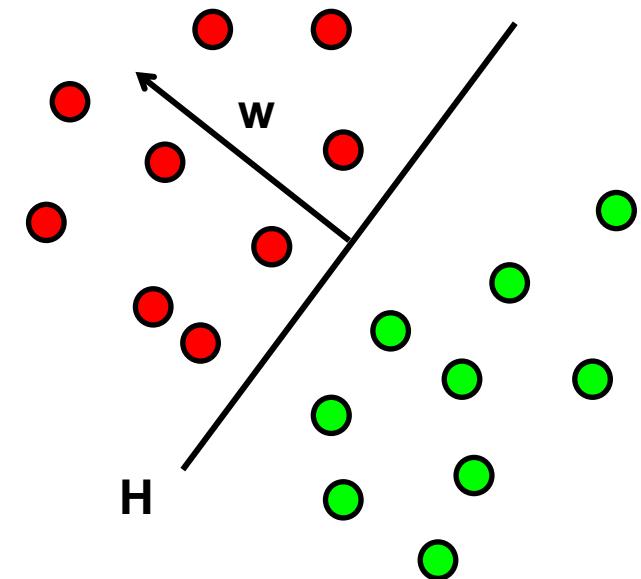
определяется:

- нормалью  $w$ ,
- и смещением  $b$ .

$$H = \{x | \langle w, x \rangle + b = 0\}$$



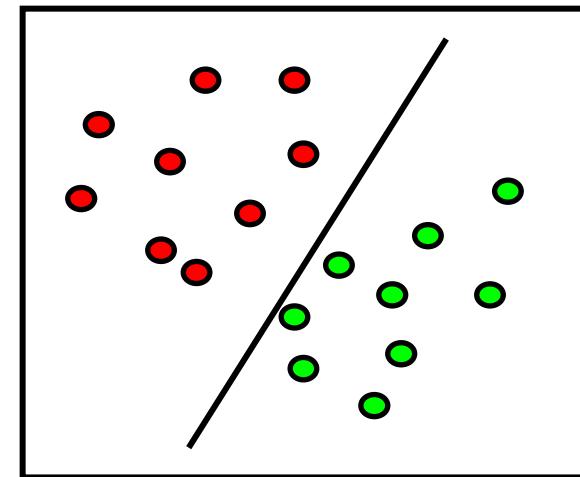
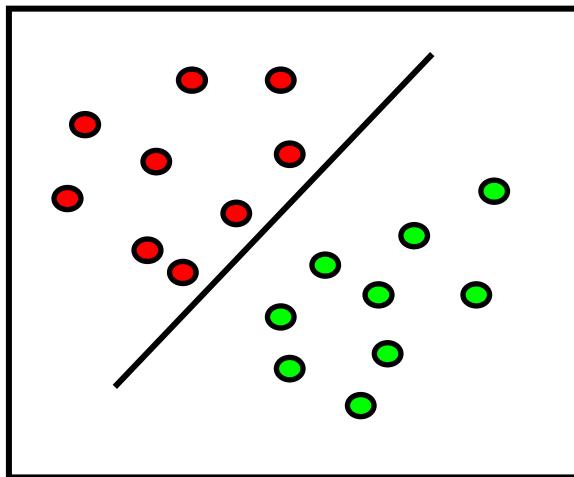
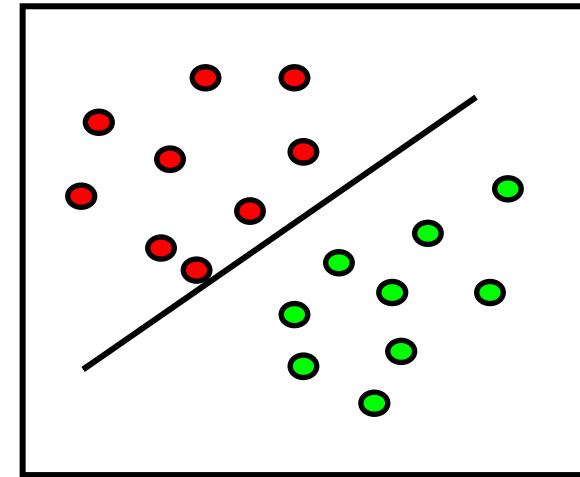
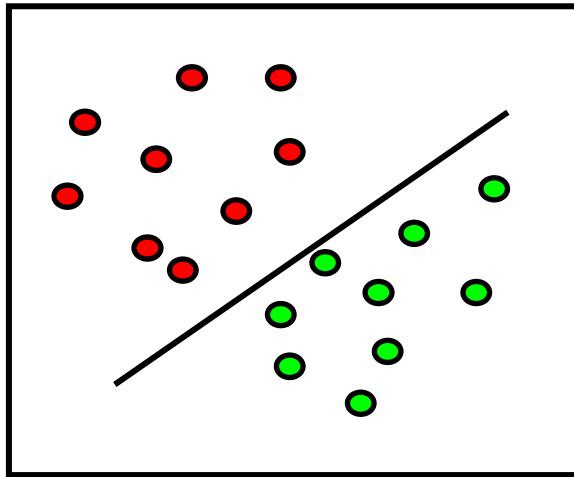
Скалярное произведение



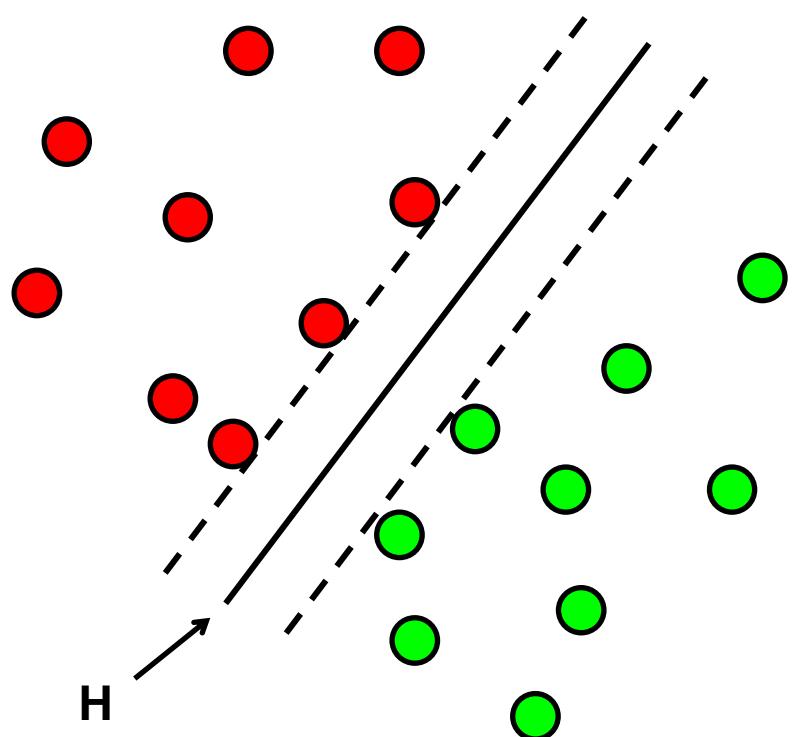
- Обучение – построение гиперплоскости, Разделяющей классы
- Прогнозирование – с какой стороны гиперплоскости пример?

# Возможные разделяющие гиперплоскости

Если образы классов линейно разделимы, то  
плоскостей может быть бесконечно много



# Разделяющая гиперплоскость с наиболее широкой разделяющей полосой (границей)



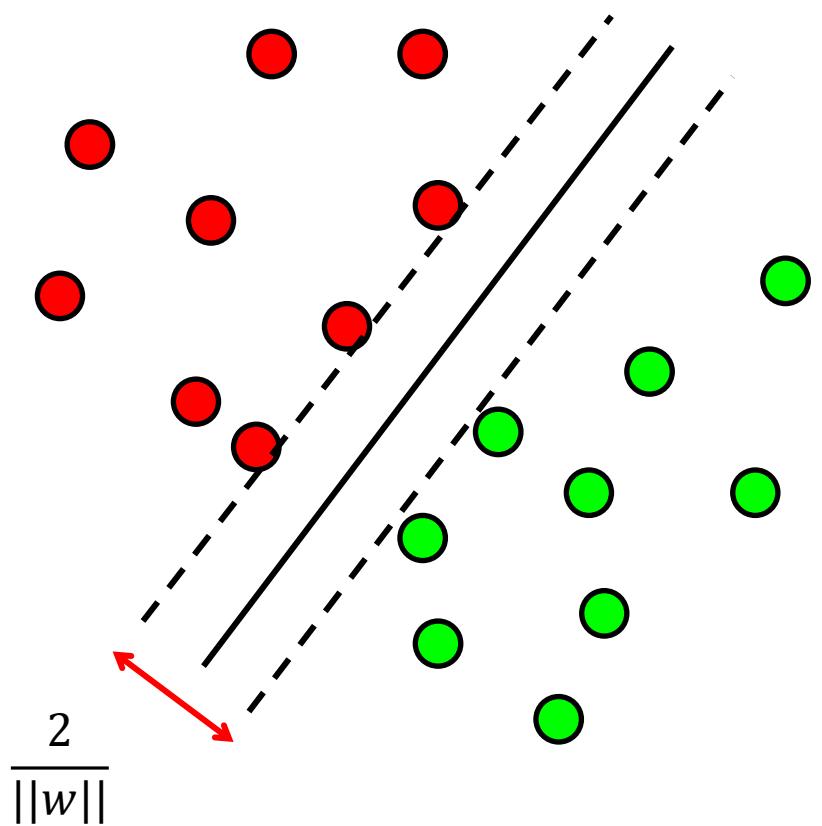
Задача оптимизации (через канонические гиперплоскости):

$$\langle w, x_i \rangle + b \geq 1 \quad y_i = 1$$

$$\langle w, x_i \rangle + b \leq -1 \quad y_i = -1$$

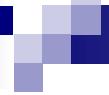
$$y_i \cdot (\langle w, x_i \rangle + b) \geq 1$$

# Максимизация ширины разделяющей полосы



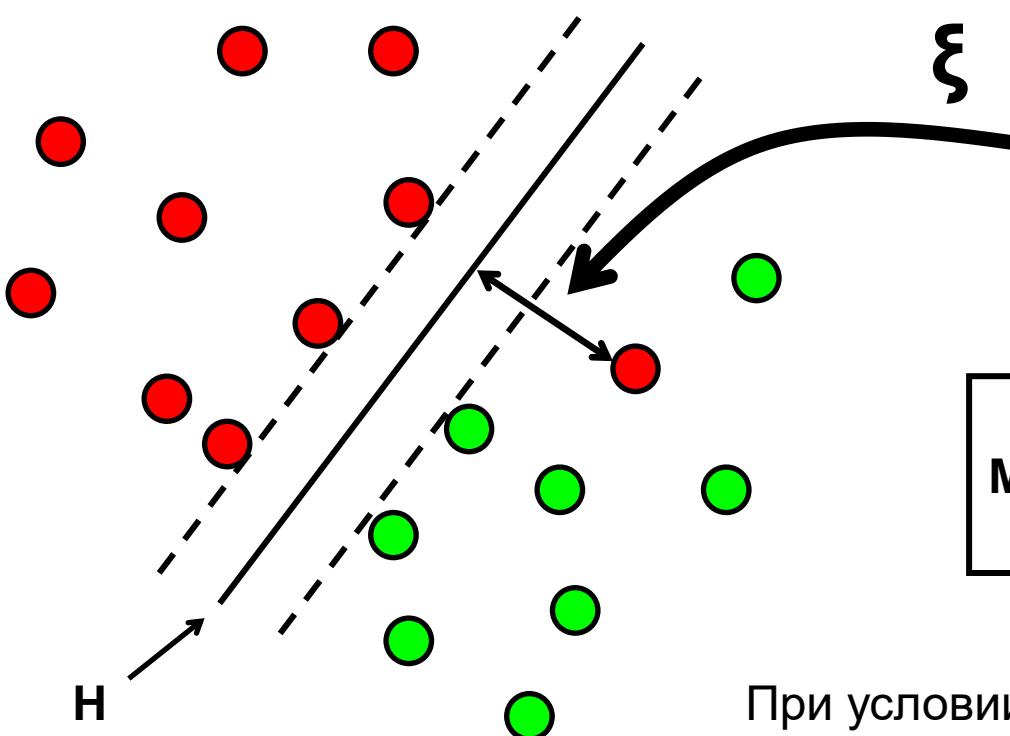
Задача оптимизации:  
Максимизация ширины  
разделяющей полосы есть  
минимизация

$$\text{Minimize } \|w\|^2$$



# Линейно неразделимые классы

Штраф С (параметр регуляризации):



«Сложность» или  
обобщающая способность

ошибка

$$\text{Min } ||w||^2 + C \cdot \sum_i \xi_i$$

При условии:

$$y_i \cdot (\langle w, x_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

# Метод множителей лагранжа

- Лагранжиан:

$$L(w, b, \alpha, \xi) = \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i (\xi_i + y_i(\langle w, x_i \rangle + b) - 1)$$

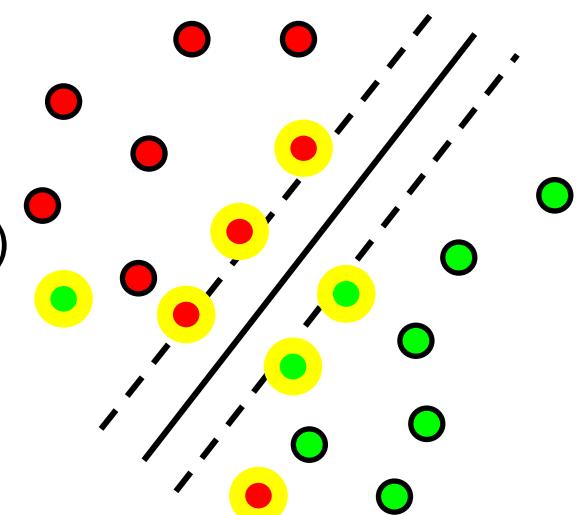
- Прямая задача:  $\min L(w, b, \alpha, \xi)$  по  $w, b, \xi$ .
- Двойственная задача:  $\max L(w, b, \alpha, \xi)$  по  $\alpha_i$

$$W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$0 \leq \alpha_i \leq C$$

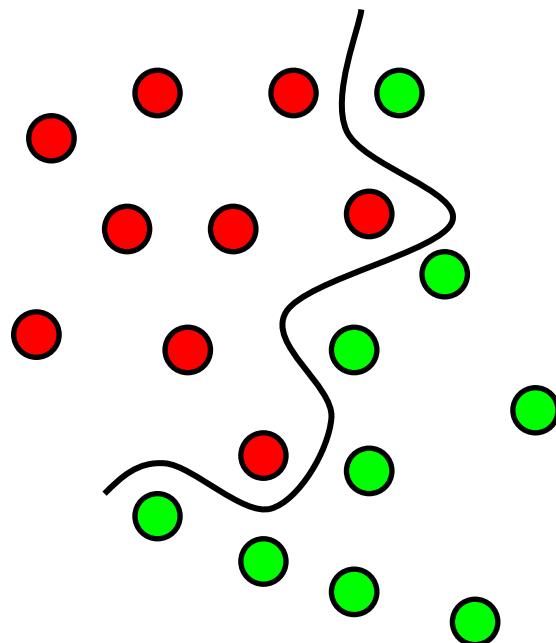
- Опорные вектора (где альфа не равны 0)
- Решающая функция:

$$w = \sum_{i=1}^{\#sv} \alpha_i y_i x_i^{sv}$$



# Неявное преобразование пространства признаков

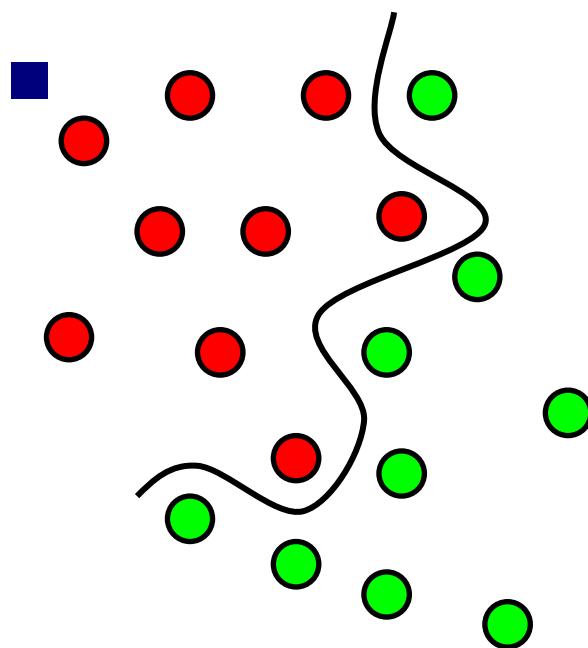
## Входное пространство признаков 2-D



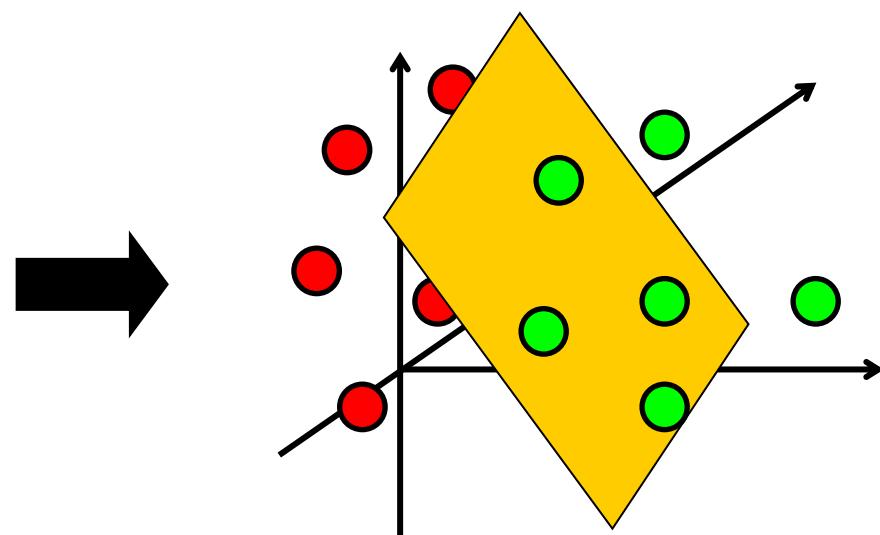
- Неявное преобразование исходного пространства признаков в новое пространство признаков большей или бесконечной размерности.
- Разделяющая плоскость строится в преобразованном пространстве
- В новом пространстве зависимость линейна, в исходном нелинейна

# Подмена ядра (kernel trick)

Input space 2-D



Feature space 3-D



# Все зависит от скалярного произведения

- Двойственная задача оптимизации:

$$W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

- Решающая функция:

Скалярное произведение

$$f(x_{new}) = \text{sign} \left( \sum_{i=1}^n \alpha_i y_i \langle x_i, x_{new} \rangle + b \right)$$

# Преобразованное пространство признаков $\mathcal{F}$

- Преобразование пространства признаков с помощью отображения  $\Phi$ :

$$\Phi: \mathbb{R}^d \mapsto \mathcal{F} \quad x \mapsto \Phi(x)$$

- Разделяем гиперплоскостью «образы» исходных наблюдений  $\Phi(x)$  в пространстве признаков  $\mathcal{F}$ , где скалярное произведение:

$$\langle \Phi(x_i), \Phi(x_j) \rangle$$

- НО: в явном виде не нужно вычислять новые признаки и образы, достаточно заменить скалярное произведение на ядерную функцию:

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

# Ядерные функции

## ■ Простые примеры:

- Линейная:  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Полиномиальная степени  $p$ :  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
- Гауссовская (radial-basis function network):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)$$

- K-spectrum (для текстов и ДНК):

$$k_n(s, t) = \sum_{u \in \Sigma^n} (\varphi_u(s) \cdot \varphi_u(t)) = \sum_{u \in \Sigma^n} \sum_{i: u=s[i]} \sum_{j: u=t[j]} \lambda^{l(i)+l(j)}$$

## ■ Методы построения ядерных функций для сложно структурированных объектов:

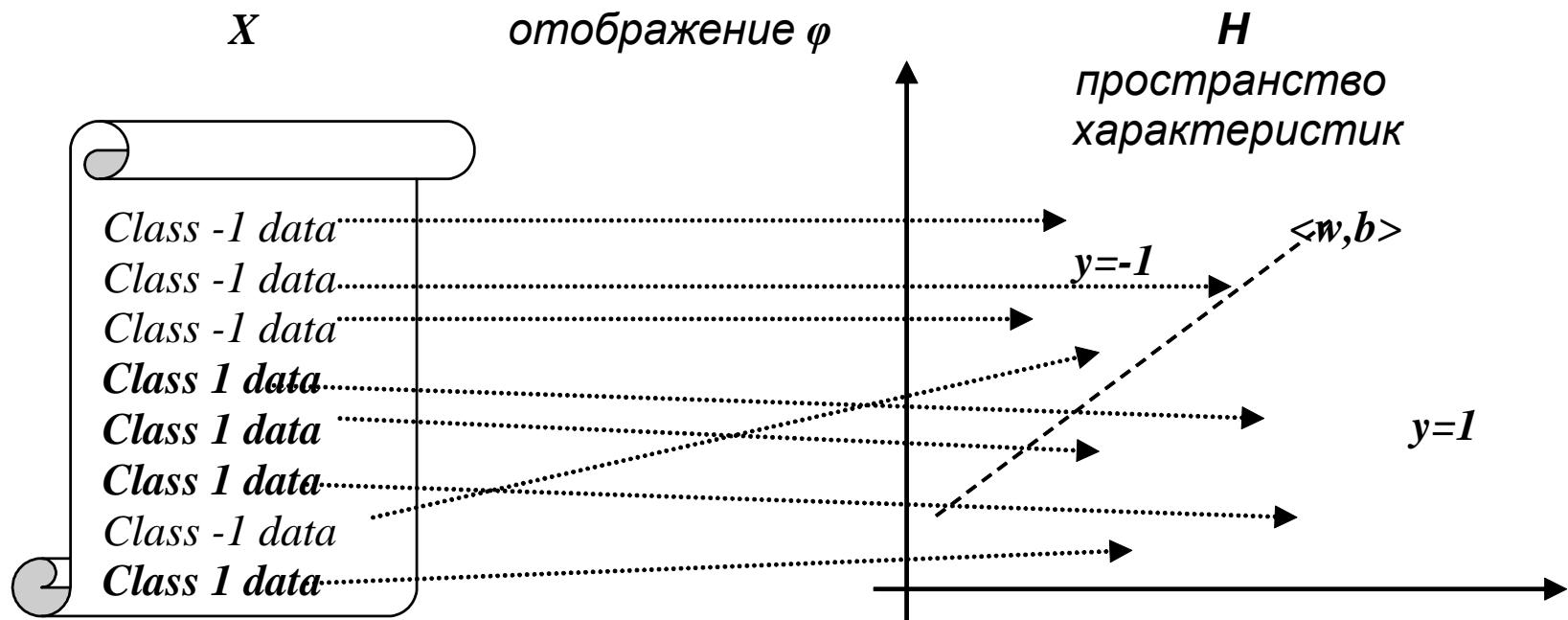
- на основе заданной вероятностной модели
- на основе экспертных попарных или групповых сравнений
- на основе R-свертки по отношениям:

$$K(x, y) = \sum_{\vec{x} \in R^{-1}(x), \vec{y} \in R^{-1}(y)} \prod_{d=1}^D K_d(x_d, y_d)$$

- Здесь  $R$  – набор всех возможных вариантов декомпозиции сложных объектов  $X$  и  $Y$  на составные части  $\vec{x} = \langle x_1, \dots, x_D \rangle$

# Классификация с помощью нелинейного SVM

- Основная идея:



В пространстве характеристик  $H$  строится гиперплоскость  $\langle w, b \rangle$ ,  
оптимально разделяющая образы классов

# Классификация с помощью нелинейного SVM

- С-Формулировка:

$$\min_{\xi \in \Re^m, w \in H} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i$$
$$(y_i(\langle \phi(x_i), w \rangle_H + b) \leq 1 - \xi_i,$$
$$\xi_i \geq 0, \forall i \in [1, N]$$

- nu-Формулировка:

$$\min_{\xi \in \Re^m, \rho \in \Re, w \in H} \frac{1}{2} \|w\|^2 + \sum_{i=1}^N \xi_i - \rho v$$
$$(y_i(\langle \phi(x_i), w \rangle_H + b) \leq \rho - \xi_i,$$
$$\rho \geq 0, \xi_i \geq 0, \forall i \in [1, N]$$

- Решающая функция:

$$f(x) = \text{sgn} \left( \sum_i \beta_i K(x_i, x) + b \right)$$

- nu – контролирует число опорных векторов

- Основные недостатки:

- бинарная решающая функция существенно зависит от параметра C (или nu), устанавливаемого априори, а переборные методы определения неприемлемы для больших объемов данных

# SVM – Пример (Python)

```
from sklearn.svm import SVC
```

```
X, y = load_iris(return_X_y=True)
```

```
X = X[:, :2]  
X.shape, y.shape, np.unique(y)
```

```
((150, 2), (150,), array([0, 1, 2]))
```

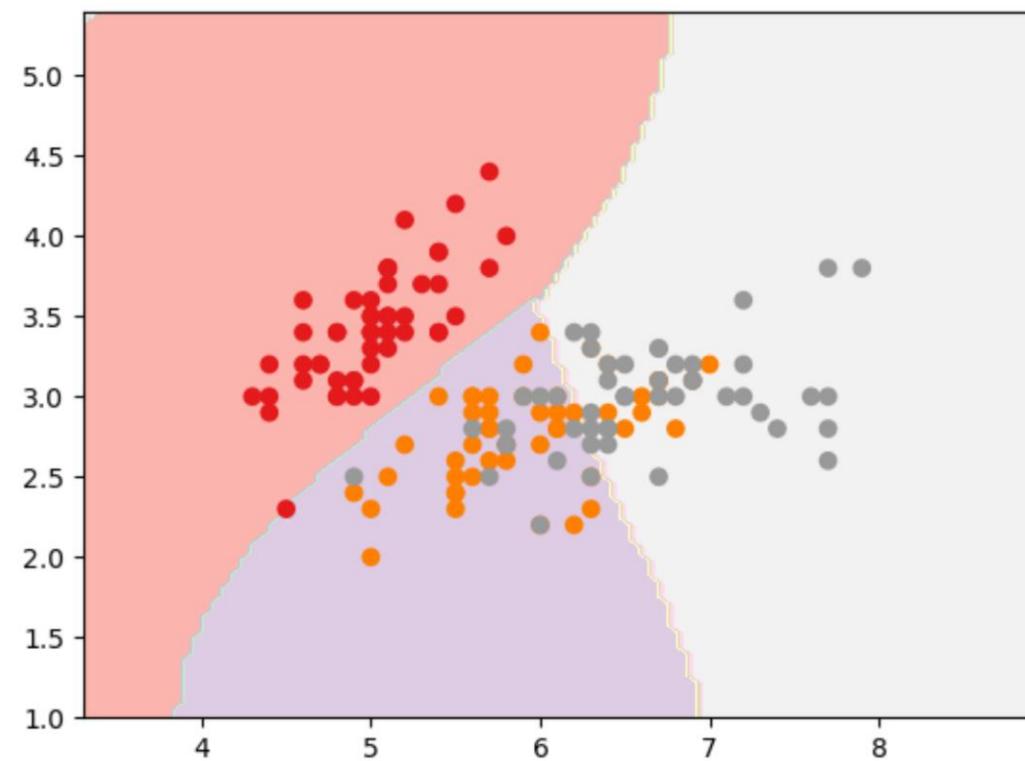
```
# auto <=> gamma = 1 / (n_features * X.var())  
svc = SVC(gamma="auto", kernel="rbf")
```

```
svc.fit(X, y)
```

```
▼ SVC
```

```
SVC(gamma='auto')
```

```
DecisionBoundaryDisplay.from_estimator(svc, X, cmap="Pastel1")  
plt.scatter(*X.T, c=y, cmap="Set1")  
<matplotlib.collections.PathCollection at 0x7fb0f2f1e4c0>
```

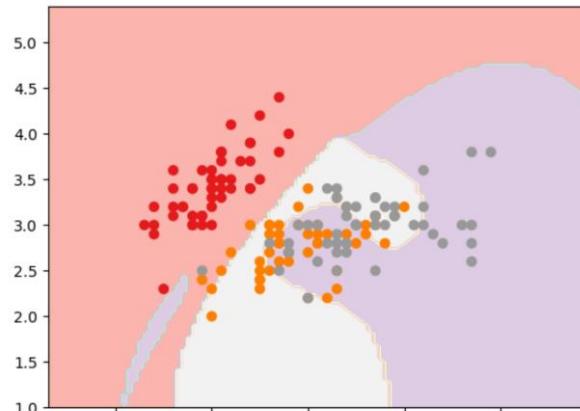


# Nu-SVM – Пример (Python)

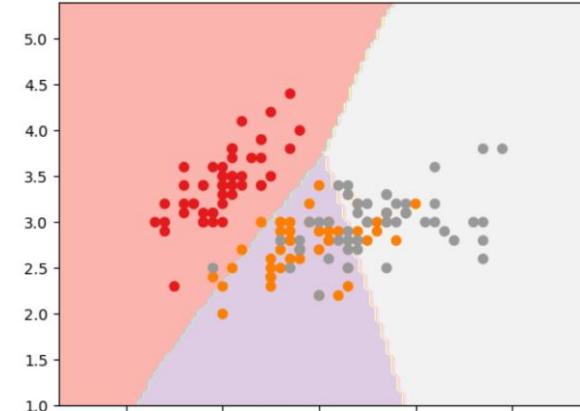
```
from sklearn.svm import NuSVC

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

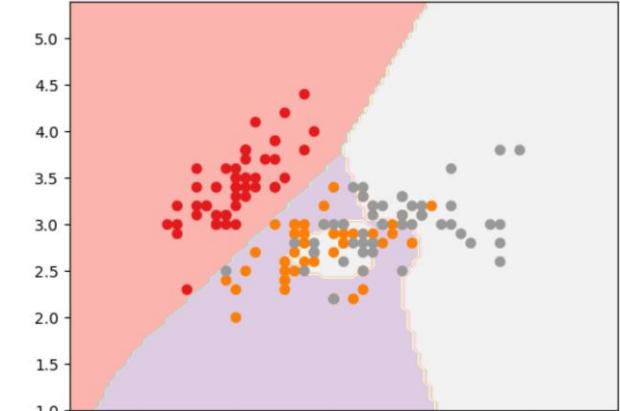
for i, nu in enumerate([0.1, 0.5, 0.9]):
    DecisionBoundaryDisplay.from_estimator(
        NuSVC(nu=nu, kernel="rbf").fit(X, y), X, cmap="Pastel1")
    plt.scatter(*X.T, c=y, cmap="Set1")
```



Nu=0.1



Nu=0.9

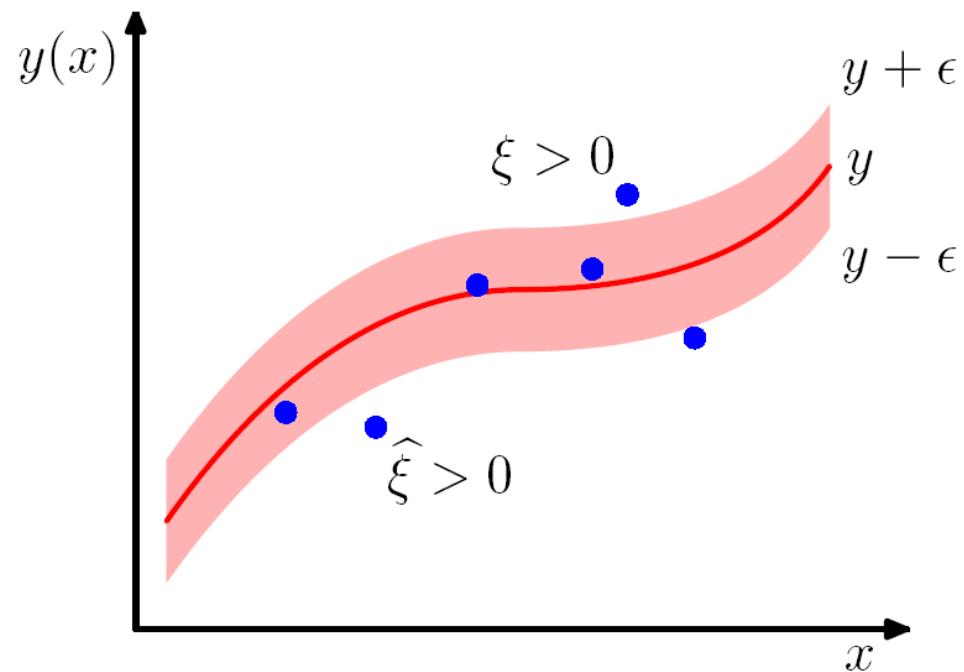


Nu=0.5

# Нелинейная регрессия с помощью SVM

$\epsilon$ -чувствительная функция потерь  
определяется как расстояние до  
регрессии за вычетом порога  $\epsilon$ :

$$L_\epsilon(y) = \begin{cases} 0, & |f(x) - y| < \epsilon \\ |f(x) - y| - \epsilon, & |f(x) - y| \geq \epsilon \end{cases}$$



Точки «внутри» полосы отступа от регрессии – не штрафуются.

Формулируется стандартный регуляризованный функционал  
эмпирического риска:

$$C \sum_{n=1}^N E_\epsilon(y(x_n) - t_n) + \frac{1}{2} \|w\|^2$$

# Нелинейная регрессия с помощью SVM

## ■ С-Формулировка:

$$\min_{\xi^{+/-} \in \Re^m, w \in H} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^-)$$
$$(\langle \phi(x_i), w \rangle_H + b) - y_i \leq \varepsilon + \xi_i, \xi_i \geq 0,$$
$$y_i - (\langle \phi(x_i), w \rangle_H + b) \leq \varepsilon + \xi_i^-, \xi_i^- \geq 0$$

## ни-Формулировка:

$$\min_{\varepsilon \in \Re, \xi^{+/-} \in \Re^m, w \in H} \frac{1}{2} \|w\|^2 + C \left( \frac{1}{N} \sum_{i=1}^N (\xi_i + \xi_i^-) + \varepsilon v \right)$$
$$(\langle \phi(x_i), w \rangle_H + b) - y_i \leq \varepsilon + \xi_i, \xi_i \geq 0,$$
$$y_i - (\langle \phi(x_i), w \rangle_H + b) \leq \varepsilon + \xi_i^-, \xi_i^- \geq 0$$
$$\varepsilon \geq 0$$

## ■ Функция регрессии:

$$y(x) = \sum_{n=1}^N (a_n - a_n^-) k(x, x_n) + b$$

## ■ ни – контролирует число опорных векторов

## ■ Основной недостаток:

- Тяжело правильно выбрать параметры регуляризации и функции потерь
- Тяжело правильно выбрать ядро

# SVR – Пример (Python)

```
from sklearn.svm import SVR
from sklearn.datasets import load_diabetes
```

```
X, y = load_diabetes(return_X_y=True)
X.shape, y.shape
```

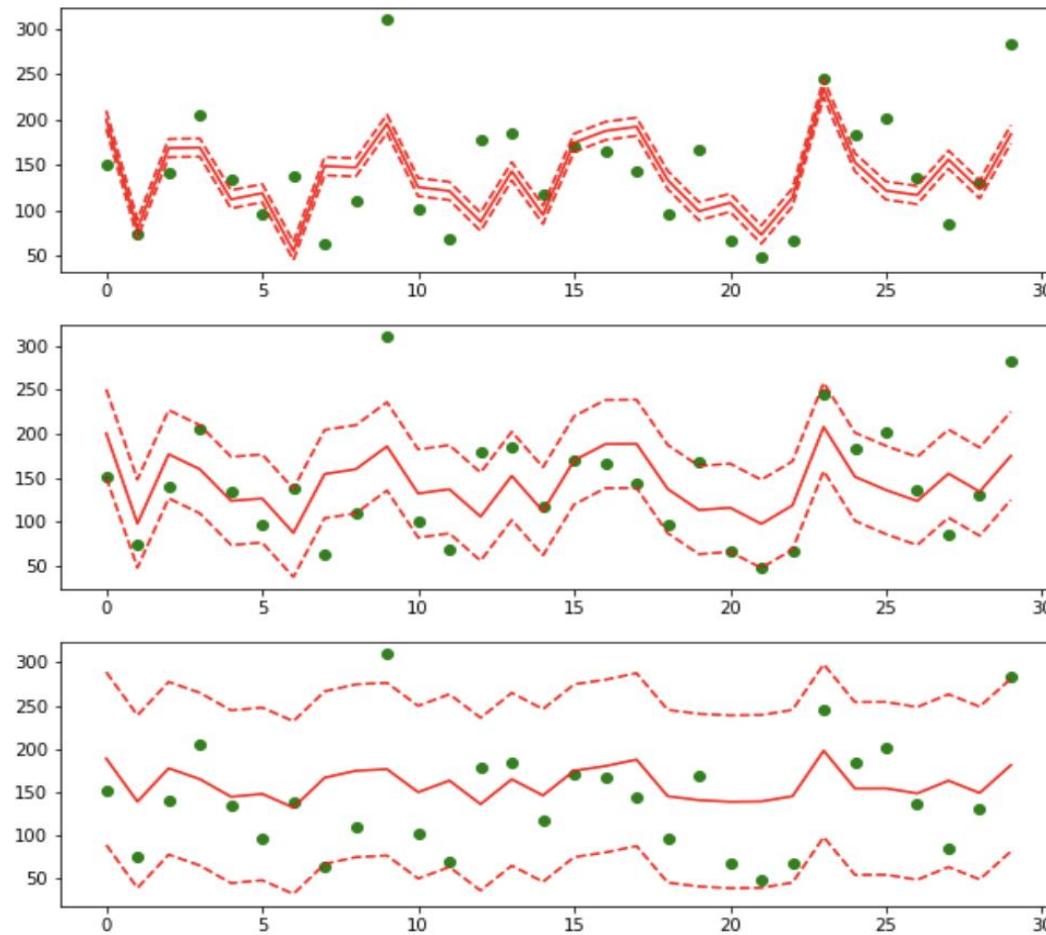
```
((442, 10), (442,))
```

```
SPAN = slice(0, 30)
for i, eps in enumerate([10, 50, 100]):
    svr = SVR(C=10, kernel="rbf", epsilon=eps).fit(X, y)

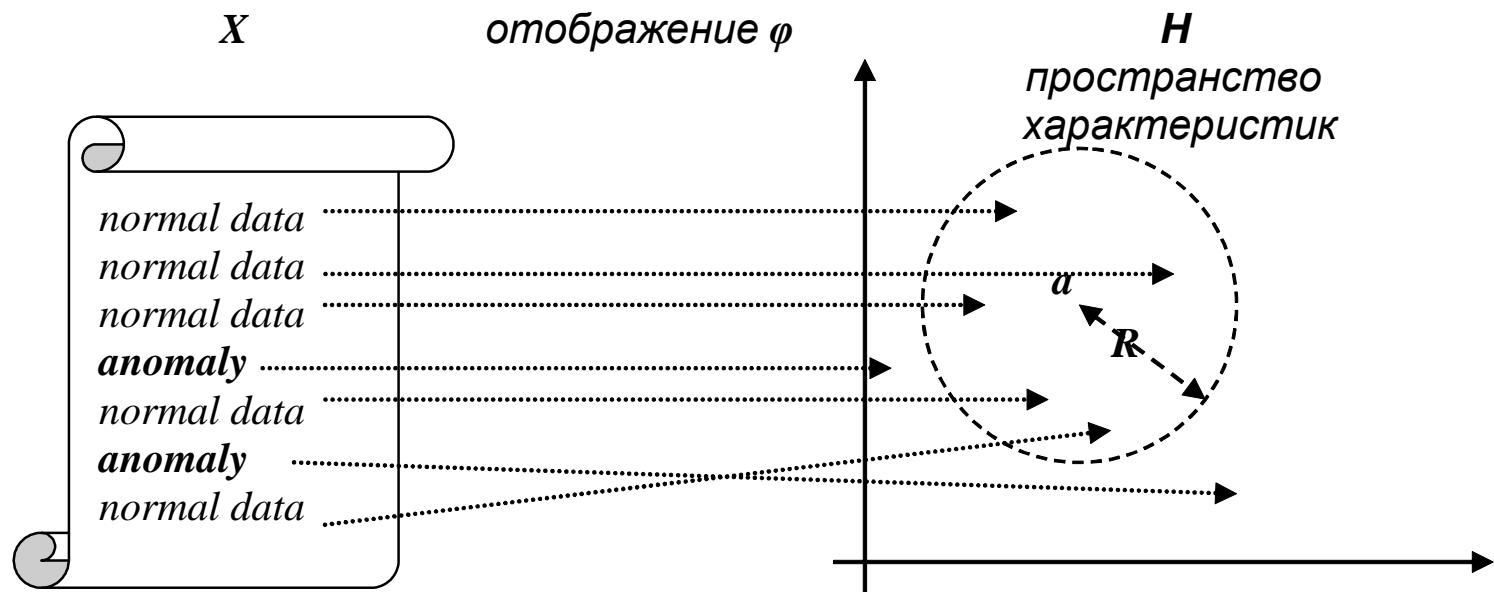
    plt.subplot(int(f"31{i+1}"))
    plt.plot(svr.predict(X)[SPAN], c="red")
    plt.plot(svr.predict(X)[SPAN] + eps, c="red", linestyle="dashed")
    plt.plot(svr.predict(X)[SPAN] - eps, c="red", linestyle="dashed")
    plt.scatter(x=range(0,30),y=y[SPAN], c="green")

plt.gcf().set_size_inches(10, 10)
```

# SVR – Пример (Python)



# Обнаружение аномалий с помощью SVM



В бесконечномерном пространстве характеристик  $H$  ищется гиперсфера с центром в точке  $a$  минимального радиуса  $R$ , включающая «основную часть»  $v$  образов событий из исходного пространства

# Обнаружение аномалий с помощью SVM

- Формулировка:

$$\min_{\xi \in \Re^m, R \in \Re, a \in H} \left[ R^2 + \frac{1}{\nu N} \sum_{i=1}^N \xi_i \right]$$
$$\|\phi(x_i) - a\|^2 \leq R^2 + \xi_i, \forall i \in [1, N]$$

- Решающая функция:

$$f(x) = \text{sgn} \left( R^2 - \sum_{i,j} \beta_i \beta_j K(x_i, x_j) + 2 \sum_i \beta_i K(x_i, x) - K(x, x) \right)$$

- Основной недостаток:

- бинарная решающая функция, которая существенно зависит от параметра  $\nu$ , устанавливаемого априори, а переборные методы определения  $\nu$  неприемлемы для больших объемов данных.

# Выводы по SVM

- SVM строит разделяющую плоскость с максимально широкой границей в преобразованном пространстве признаков, индуцированном kernel функцией, используемой вместо скалярного произведения, при этом с помощью параметра регуляризации задается компромисс между точностью и обобщающей способностью модели.
- Основные параметры, которые нужно задавать: константа регуляризации С и kernel функция.
- Основной недостаток: НЕ существует автоматических эффективных методов выбора выбора этих параметров для конкретной задачи
- Достоинств много: Единственное решение, Kernel trick, Геометрическая интерпретация, Устойчивость к проклятию размерности, Явный контроль сложности модели с помощью параметров

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

- Предложено в 1995 Dietterich и Bakiri
- 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  $l$  бинарных подзадач с помощью кодовой матрицы

	$1$	$2$	$\dots$	$s$	$\dots$	$l$
$1$	$+1$	$+1$	$\dots$	$-1$	$\dots$	$0$
$2$	$0$	$-1$	$\dots$	$0$		$+1$
$\dots$						
$j$	$0$	$-1$	$\dots$	$-1$	$\dots$	$-1$
$\dots$						
$k$	$-1$	$0$	$\dots$	$+1$	$\dots$	$-1$

$k$  классов →  $l$  бинарных задач

$j$ -й класс →  $s$ -ая бинарная задача

$M \in \{-1, 0, 1\}^{k \times l}$

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

# Кодирование в ЕСОС

## ■ “Разреженный” ЕСОС – общий случай:

- “Плотный” ЕСОС – матрица без 0
- “Каждый против всех”:

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

“Каждый против каждого”:

	$I$	2	...	$k-I$	$k$	$k+I$	...	$\binom{k}{2}$
$I$	+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
- Случайные коды: случайные длинные «хорошо разделимые» коды

# Обучение в ECOC

- $l$  бинарных задач решаются независимо:

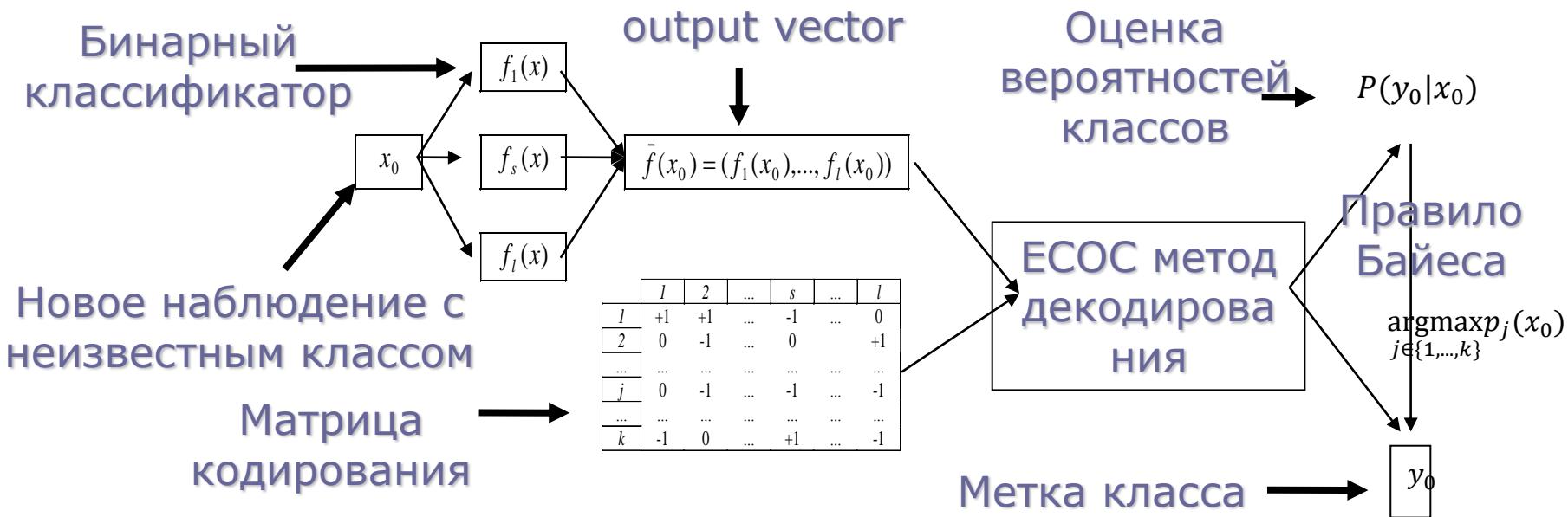
- $s$ -й бинарный классификатор отделяет  $s$ -е “положительные” примеры от  $s$ -х “отрицательных”, так что  $s$ -й тренировочный набор:  
 $Z_s = \{(x_i, M(y_i, s)) \mid (x_i, y_i) \in Z \wedge (y_i \in I_s^- \vee y_i \in I_s^+)\} \subseteq X \times \{-1, +1\}$
  - Бинарный алгоритм используется для решения  $f_1(x), \dots, f_l(x)$
  - Получаем  $l$  бинарных классификаторов ( $l$  гипотез)  
таких, что  $f_s : X \rightarrow Y_{bin}$

- Типы бинарного отклика:

- *Булевый* (hard-level):  $Y_{bin} = \{-1, 1\}$
  - *Вещественный* (soft-level):  $Y_{bin} \subseteq \mathbb{R}$
  - *Вероятностный*:  $r_s(x) = P(f_s(x) \in I_s^+ \mid f_s(x) \in I_s^+ \cup I_s^-)$

# Декодирование в ECOC

## ■ Процесс прогнозирования:



- Применить все бинарные классификаторы, получить вектор откликов длины  $l$
- 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

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

Ближайший  
код

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

Предсказанный  
класс

- Используются разные метрики:

Хэмминга (hard-level):

$$d_H(\bar{f}(x), M(y)) = \sum_{s=1}^l [1 - \text{sgn}(M(y, s)f_s(x))]$$

Минковского (probabilistic):

$$d_{L1}(\bar{r}(x), M(y)) = \sum_{s=1}^l |M(y, s) - r_s(x)|$$

- На основе функции потерь
  - С оценкой вероятности и т.п.

$$\text{Loss}(\bar{f}(x), M(r)) = \sum_{s=1}^l \text{loss}(M(y, s)f_s(x))$$

# Пример

```
from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier, OutputCodeClassifier
from sklearn.svm import LinearSVC
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(LinearSVC(C=1.0, max_iter=10000)).fit(X, y)
ovo = OneVsOneClassifier(LinearSVC(C=1.0, max_iter=10000)).fit(X, y)
ocs = OutputCodeClassifier(LinearSVC(C=1.0, max_iter=10000), 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.]]
```

```
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__)
```

