OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

Логистическая регрессия

Ксения Стройкова

Сегодня на лекции

- Задачи машинного обучения, задача классификации
- Отступ, zero-one loss
- Предсказание вероятности
- Мультиклассовая регрессия, softmax
- Cross entropy
- Градиентный спуск
- Стохастический градиентный спуск
- Случай линейно неразделимых данных

Задачи машинного обучения

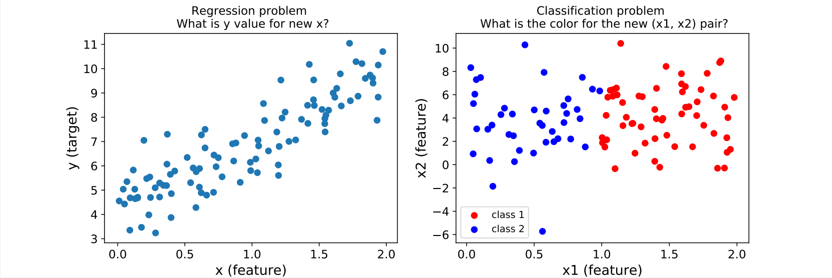
- Обучение с учителем
  - Регрессия
  - Классификация
- Обучение без учителя
  - Кластеризация
  - Снижение размерности
- Обучение с подкреплением

Обучение с учителем

есть некоторое количество примеров, для которых известны ответы

- ответы числа - регрессия
- ответы классы - классификация

Обучение с учителем



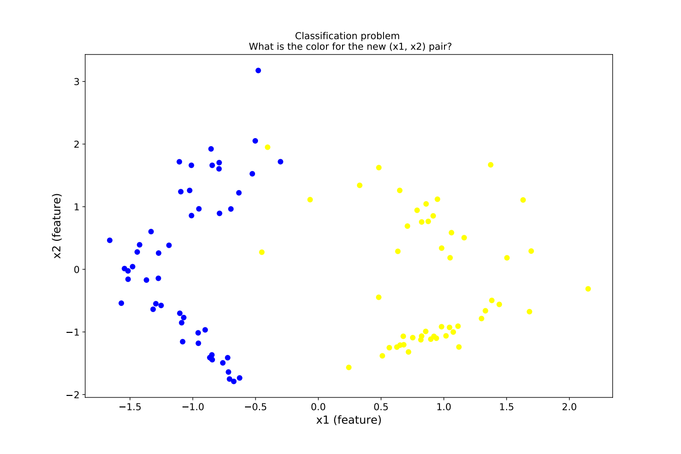
С учителем - классификация

```
X, y = datasets.make_classification(
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_repeated=0,
    random_state=1
)
```

С учителем - классификация

x	y	
1.30022717	-0.7856539	1
1.44184425	-0.56008554	1
-0.84792445	-1.36621324	0
-0.72215015	-1.41129414	0
-1.27221465	0.25945106	0

С учителем - классификация



Логистическая регрессия

Построим случайную прямую.

Принятие решения

Простой вариант - узнать, с какой стороны от гиперплоскости находится точка

$$\hat{y} = sign(x\theta)$$

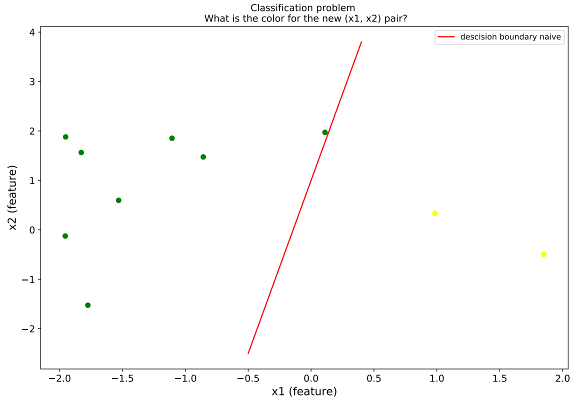
Уравнение прямой

$$Ax + By + C = 0$$

Расстояние от точки $(x0, y0)$ до прямой $Ax + By + C = 0$ это

$$\frac{|Ax0 + By0 + C|}{\sqrt{(A^2 + B^2)}}$$

Принятие решения



Упражнение 1

реализовать predict

Упражнение 1

```
def predict(x, w):
    return np.sign(x.dot(w))
```

Отступ - простая оценка результата

Отступ (margin) - величина $M_i = y_i \cdot x_i\theta$ (для $y = 1$ или $y = -1$), где x_i - элемент обучающей выборки, y_i - его класс

$$M_i \leq 0 \Rightarrow y_i \neq \hat{y}_i$$

$$M_i > 0 \Rightarrow y_i = \hat{y}_i$$

Zero-one loss

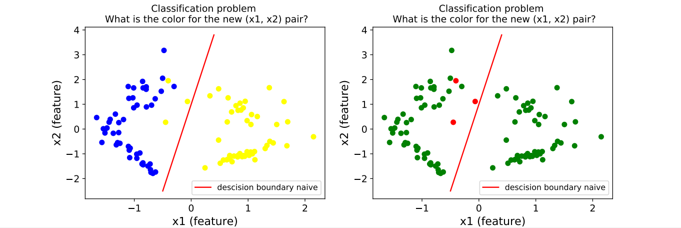
Функция потерь zero-one loss:

$$f(x) = \begin{cases} 1, & \text{если} \hat{y} \neq y, \\ 0, & \text{если} \hat{y} = y \end{cases}$$

Эмпирический риск - zero one loss

$$Q(\theta, x) = \frac{1}{n}\sum_{i=1}^{n} f(x) = \frac{1}{n}\sum_{i=1}^{n}[M_i < 0]$$

Эмпирический риск - zero one loss



Эмпирический риск - zero one loss

```
from sklearn.metrics import zero_one_loss
zero_one_loss(y, y_pred)

# 0.030000000000000027
```

Logit regression

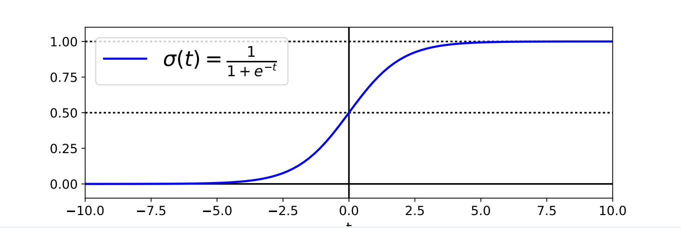
Переформулируем задачу Вместо класса будем предсказывать вероятность принадлежности классу

$$\hat{p} = \sigma(x\theta)$$

где

$$\sigma(t) = \frac{1}{1 + exp(-t)}$$

Сигмоид



Предсказание

$$y = \begin{cases} 0, & \text{если} \hat{p} < 0.5, \\ 1, & \text{если} \hat{p} \geq 0.5 \end{cases}$$

ONLINE-ОБРАЗОВАНИЕ

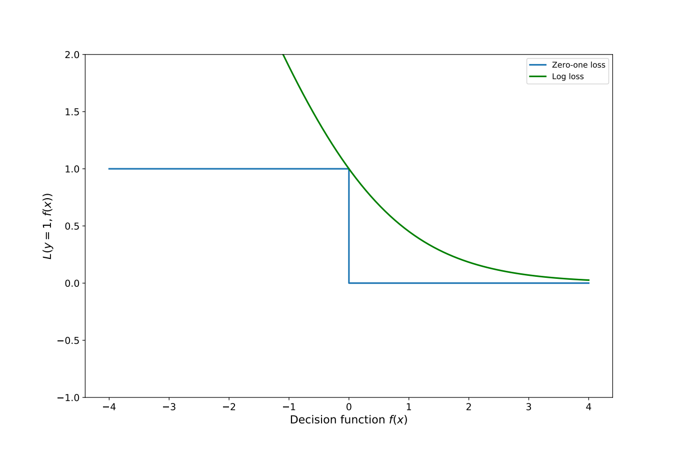
Оценка одного элемента выборки

$$Q(\theta, x_i) = \begin{cases} -log(\hat{p}), & \text{если} y = 1, \\ -log(1 - \hat{p}), & \text{если} y = 0 \end{cases}$$

Для многих элементов выборки (log loss)

$$Q(\theta, x) = -\frac{1}{m}\sum_{i=1}^{n}[y_i \log(\hat{p}_i) + (1 - y_i) log(1 - \hat{p}_i)]$$

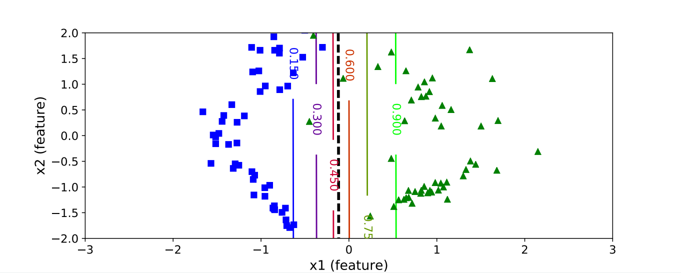
Log loss


Zero-one loss
Log loss
$L(y=1, f(x))$
Decision function $f(x)$
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−4
−3
−2
−1
0
1
2
3
4

Логистическая регрессия в sklearn

```
from sklearn.linear_model import LogisticRegressio
log_reg = LogisticRegression()
log_reg.fit(X, y)
y_proba = log_reg.predict_proba(X_new)
```

Логистическая регрессия в sklearn



Мультиклассовая регрессия

Для обучения модели предсказывать K классов можно натренировать

- K классификаторов 1 против всех (one vs rest)
- $K(K-1)/2$ классификаторов one vs one.

При предсказании брать максимальное значение. Вероятности нормализуются.

Мультиклассовая регрессия, softmax

Нам необходимо получить значения для k классов - составим матрицу параметров Θ

$$x = \begin{bmatrix} 1 & x_{11} & \dots & x_{p1} \\ 1 & x_{12} & \dots & x_{p2} \\ \dots & \dots & \dots & \dots \\ 1 & x_{n1} & \dots & x_{n1} \end{bmatrix} \quad \Theta = \begin{bmatrix} \theta_{01} & \dots & \theta_{0k} \\ \theta_{11} & \dots & \theta_{1k} \\ \dots & \dots & \dots \\ \theta_{p1} & \dots & \theta_{pk} \end{bmatrix} \quad f = \begin{bmatrix} f_{01} & \dots & f_{0k} \\ f_{11} & \dots & f_{1k} \\ \dots & \dots & \dots \\ f_{n1} & \dots & f_{nk} \end{bmatrix}$$

Мультиклассовая регрессия, softmax

$$\hat{p}_k = \frac{e^{x\theta_k}}{\sum_{j=1}^{K} e^{x\theta_j}}$$

$$\hat{y}_k = argmax_k \hat{p}_k$$

Оценка для случая многих классов - cross entropy

$$Q(\Theta, x) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \log \hat{p}_{ik}$$

Логистическая регрессия в sklearn

```
from sklearn import linear_model, datasets
iris = datasets.load_iris()

X = iris.data[:, :2]  # we only take the first two
Y = iris.target
logreg = linear_model.LogisticRegression(C=1e5)
logreg.fit(X, Y)
```

Логистическая регрессия в sklearn

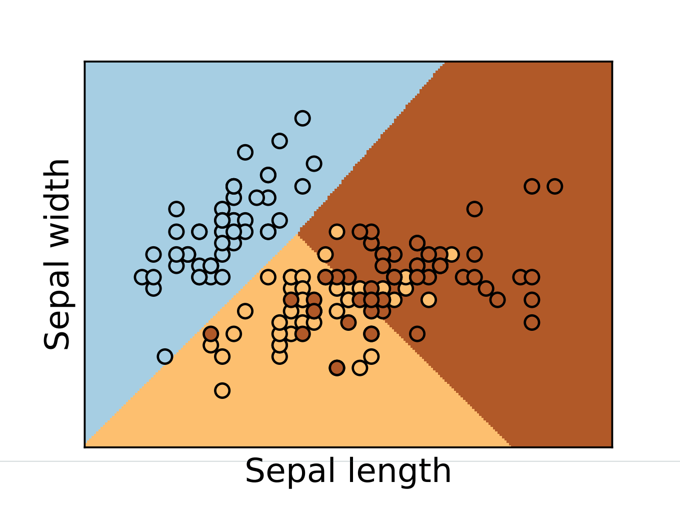
```
[[ 5.1  3.5]
 [ 4.9  3. ]
 [ 4.7  3.2]
 [ 4.6  3.1]
 [ 5.   3.6]
 [ 5.4  3.9]
 [ 4.6  3.4]
 [ 5.   3.4]
 [ 4.4  2.9]
 [ 4.9  3.1]]
[0 1 2]
```

Логистическая регрессия в sklearn

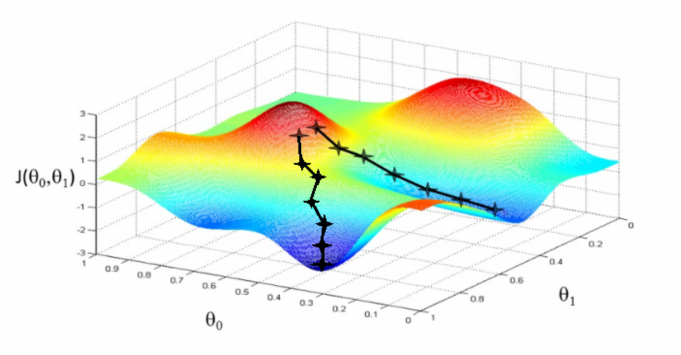
```
LogisticRegression(C=100000.0, class_weight=None,
  dual=False, fit_intercept=True,
  intercept_scaling=1, max_iter=100,
  multi_class='ovr', n_jobs=1, penalty='l2',
  random_state=None, solver='liblinear',
  tol=0.0001, verbose=0, warm_start=False)

print logreg.coef_
[[-30.619  27.55 ]
 [  0.14   -3.214]
 [  2.604  -0.743]]
```

Логистическая регрессия в sklearn



Градиентный спуск



Градиентный спуск

$$\theta := \theta - \alpha \frac{\partial L}{\partial \theta}$$

α - скорость спуска

Градиент функции потерь $RSS(\theta)$

$$RSS = \mathcal{L}(\theta) = (\hat{y} - y)^2$$

$$\frac{\partial L}{\partial \theta_i} = 2(\hat{y} - y)\frac{\partial L}{\partial \theta_i}(\hat{y} - y) = 2(\hat{y} - y)\frac{\partial L}{\partial \theta_i}(\theta_0 x_0 + ... + \theta_1 x_1 - y) = 2(\hat{y} - y) \cdot x_i$$

$$\theta_i := \theta_i - \alpha(\hat{y} - y) \cdot x_i$$

α - скорость спуска

Градиент функции потерь $RSS(\theta)$

$$\frac{\partial RSS(\theta)}{\partial \theta_i} = 2\sum_{i=1}^{n}(\theta^T \cdot x_i - y_i)x_i$$

$$\nabla_\theta RSS(\theta) = \begin{pmatrix} \frac{\partial L}{\partial \theta_0} \\ \frac{\partial L}{\partial \theta_1} \\ \dots \\ \frac{\partial L}{\partial \theta_p} \end{pmatrix} = x^\top(x\theta - y)$$

α - скорость спуска

Градиент функции потерь $MSE(\theta)$

$$\frac{\partial L}{\partial \theta} = \frac{1}{n} X^\top (X\theta - y)$$

Псевдокод алгоритма

```
function gd(X, alpha, epsilon):
    initialise theta
    do:
        theta = new_theta
        new_theta = theta - alpha * grad(X, thet
    until dist(new_theta, theta) < epsilon
    return theta
```

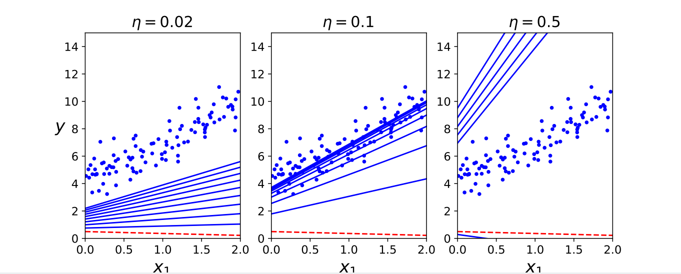
Упражнение 2

Реализовать алгоритм GD

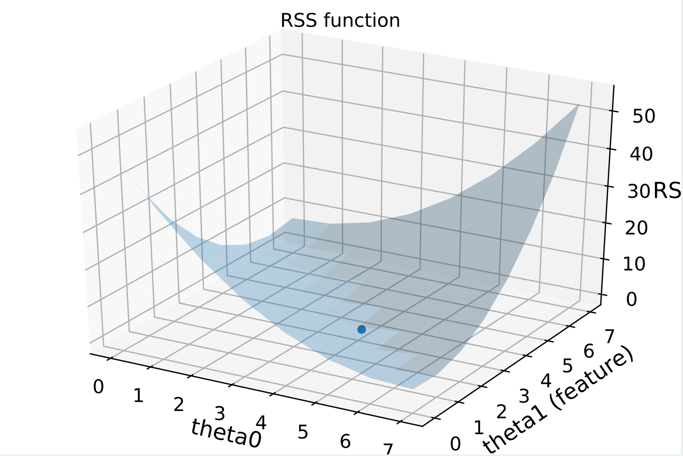
Упражнение 2

```
for iteration in range(n_iterations):
    gradients = 2. / m * X_b.T.dot(X_b.dot(theta)
    theta_old = theta
    theta = theta - alpha * gradients
    dist = np.linalg.norm(theta - theta_old)
    if dist < eps:
        break
print iteration, dist
```

Шаг алгоритма



Минимизация ошибки



Стохастический градиентный спуск

Проблема - используется вся обучающая выборка на каждом шаге алгоритма Решение - использовать один случайный элемент выборки

Градиентный спуск

```
function gd(X, alpha, epsilon):
    initialise theta
    do:
        theta = new_theta
        new_theta = theta - alpha * grad(X, thet
    until dist(new_theta, theta) < epsilon
    return theta
```

Стохастический градиентный спуск

```
function sgd(X, alpha, epsilon):
  initialise theta
  do:
        X = shuffle(X)
        for x in X:
            theta = new_theta
            new_theta = theta - alpha * grad(x,
  until dist(new_theta, theta) < epsilon
  return theta
```

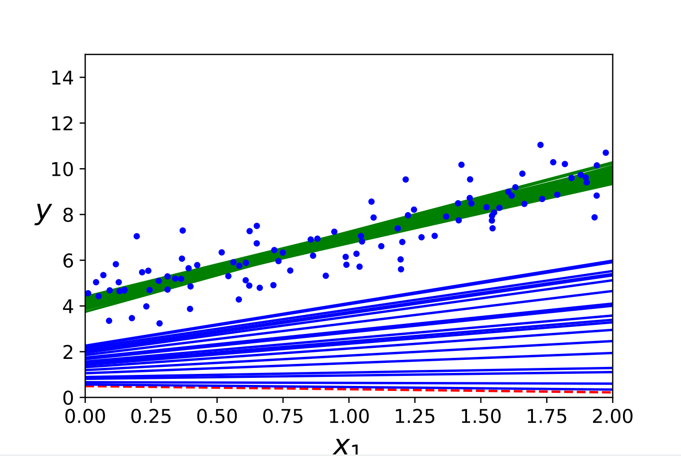
Упражнение 3

Реализовать SGD

Упражнение 3

```
for epoch in range(n_epochs):
    p = np.random.permutation(m)
    for idx in p:
        random_index = np.random.randint(m)
        xi = X_b[[idx], :]
        yi = y[[idx], :]
        gradients = 2 * xi.T.dot(xi.dot(theta) - y
        theta = theta - alpha * gradients
print theta
```

Стохастический градиентный спуск



Упражнение

Найти формулы для градиентного спуска для линейной регрессии с регуляризацией

Градиентный спуск для логистической регрессии

Бинарная классификация, log loss

$$Q(\theta, x) = -\frac{1}{m}\sum_{i=1}^{n}[y_i \log(\hat{p}_i) + (1 - y_i)log(1 - \hat{p}_i)]$$

$$\frac{\partial Q(\theta_j, x)}{\partial \theta_i} = \frac{1}{n}\sum_{i=1}^{n}(\sigma(\theta^T \cdot x_i) - y_i)x_ij$$

Градиентный спуск для логистической регрессии

cross entropy

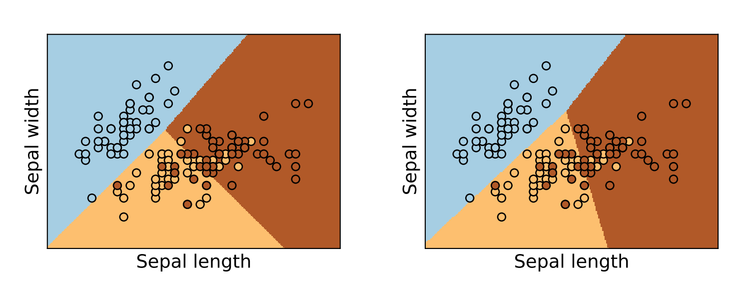
$$Q(\Theta, x) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \log \hat{p}_{ik}$$

$$\nabla_{\theta_k} Q(\Theta, x) = \frac{1}{n}\sum_{i=1}^{n}(\hat{p}_{ki} - y_{ki})x_i$$

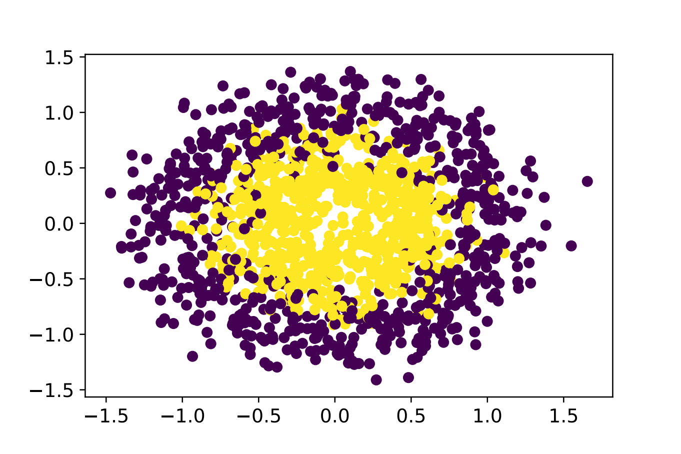
Градиентный спуск для логистической регрессии

```
X = iris.data[:, :2]  # we only take the first two
Y = iris.target
softmax_reg = LogisticRegression(multi_class="mult
softmax_reg.fit(X, Y.ravel())
```

Градиентный спуск для логистической регрессии



Линейно неразделимый случай

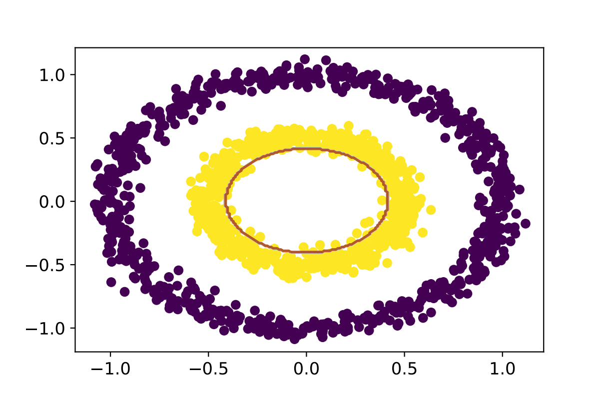


Линейно неразделимый случай

```
n_samples = 1500
x, y = datasets.make_circles(n_samples=n_samples,
    factor=.5, noise=noise)
poly_features = PolynomialFeatures(degree=2,
    include_bias=False)
X_poly = poly_features.fit_transform(x)
logit = LogisticRegression(C=c)
logit.fit(X_poly, y)
plot_boundary(...)
```

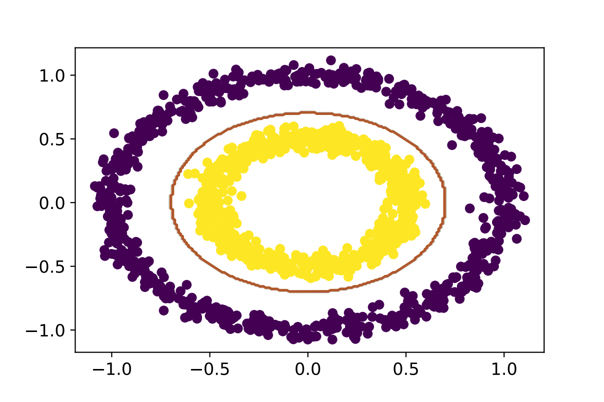
Линейно неразделимый случай

$noise = .05 \; c = 0.001$



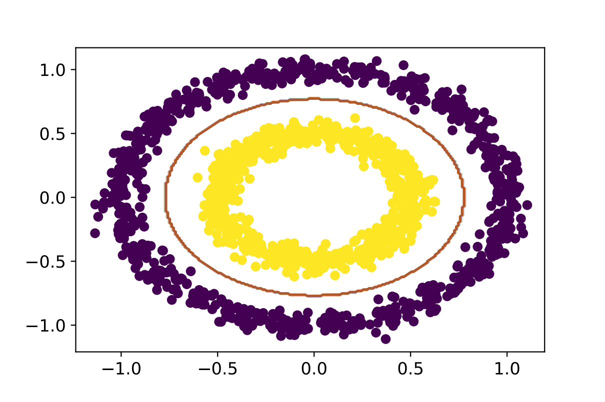
Линейно неразделимый случай

$noise = .05 \ c = 0.01$



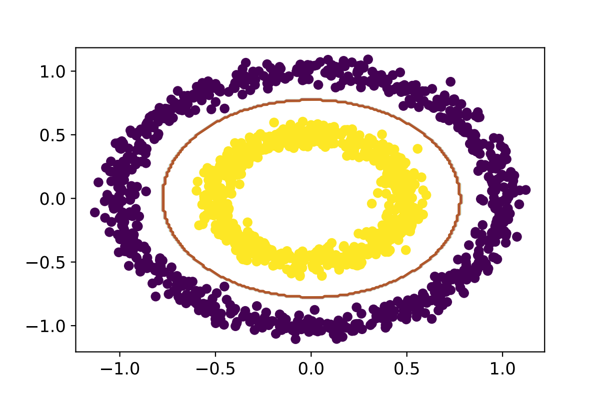
Линейно неразделимый случай

$noise = .05 \ c = 0.1$



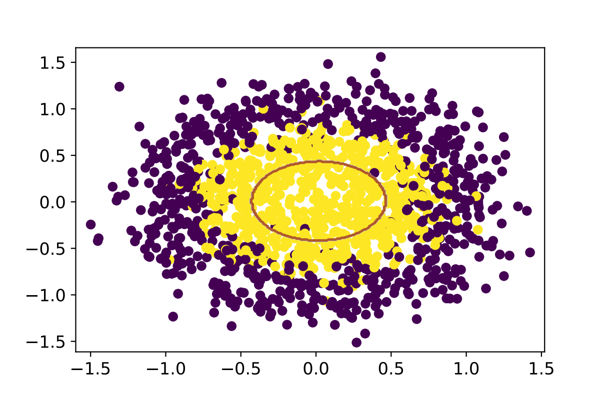
Линейно неразделимый случай

$noise = .05 \; c = 1$



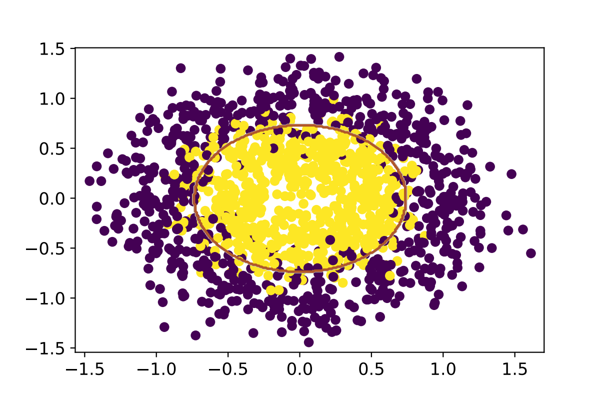
Линейно неразделимый случай

$noise = .20 \ c = 0.001$



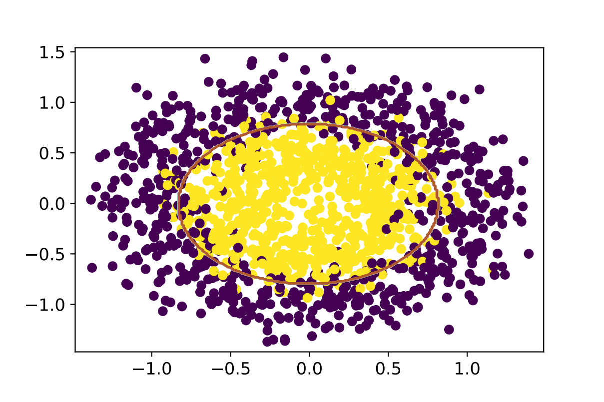
Линейно неразделимый случай

$noise = .20 \ c = 0.01$



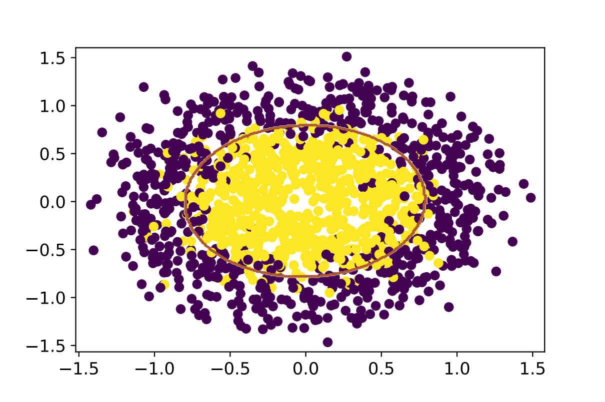
Линейно неразделимый случай

$noise = .20 \; c = 0.1$



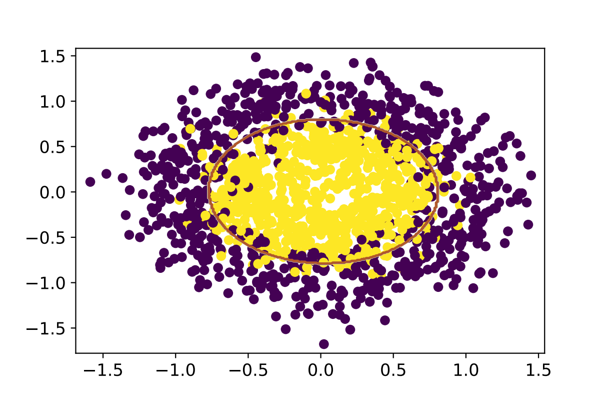
Линейно неразделимый случай

$noise = .20 \; c = 1$



Линейно неразделимый случай

$noise = .20 \; c = 10000$



Материалы

https://habrahabr.ru/company/ods/blog/322076/
Aurélien Géron - Hands-on Machine Learning with Scikit-Learn and TensorFlow

Вопросы