Лекция 8

Классификация

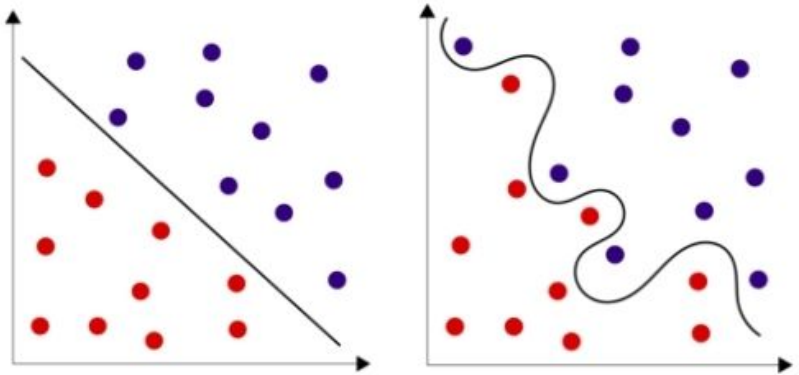
Классификация

- Задача классификации - является задачей обучения с учителем
- Цель классификации - нахождения зависимости между независимыми переменными X и метками класса Y.
- В отличии от задачи регрессии, зависимая переменная Y является дискретным конечным множеством.
- Некоторые классификаторы могут использоваться для решения задачи регрессии
- Примеры задачи классификации:
    - Определение вида хим. вещества по его свойствам
    - Предсказание рейтинга фильма по его характеристикам

Классификация

- Если в задаче классификации только 2 класса, то такая классификация называется бинарной
- Если в задаче классификации больше 2 классов, то такая классификация называется множественной

- Многие стандартные классификаторы основываются на том, что классы линейно разделимы.



Какой самый простой способ классифицировать наблюдения?

K-ближайших соседей (kNN)

- Чтобы классифицировать новый экземпляр, необходимо посмотреть на k ближайших точек и посчитать, какого класса точек больше
- kNN основывается на предположении о том, что в рамках одного класса расстояния между точками небольшие, а между точками разных классов расстояния большие
- Количество соседей k обычно устанавливается нечетным, чтобы минимизировать шанс того, что наблюдение можно будет отнести к нескольким классам с равной вероятность

К-ближайших соседей - алгоритм

1. Дана размеченная выборка размера m $X^m = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$
2. Для нового образца u считаются расстояния $d(u, x)$
3. Расстояния ранжируются по возрастанию

$$d(u, x_{1;u}) \leqslant d(u, x_{2;u}) \leqslant ... \leqslant d(u, x_{m;u})$$

4. Класс **c** образца u определяется как

$$c(u) = \underset{y \in Y}{\operatorname{argmin}} \sum_{i=1}^{m} \left[x_{i;u} = y\right] w(i, u)$$

$w(i, u)$ весовая функция между

Выбор весовой функции

- $w(i,u) = [i = 1]$ - алгоритм ближайшего соседа. k = 1
- $w(i,u) = [i \leqslant k]$ - метод k ближайших соседей
- $w(i,u) = [i \leqslant k]q^i, q \in (0,1)$ - метод k экспоненциально взвешенных ближайших соседей
- $w(i,u) = [i \leqslant k]\frac{1}{d(u, x_{i;u})}$ - взвешенный метод k ближайших соседей

- При k = 1 метод неустойчив в шумовы выбросам.
- При k = m вырождается в константу

kNN в SKlearn

- sklearn.neighbors.KNeighborsClassifier
- Параметры:
  - n_neighbors - кол-во соседей
  - weights - веса точек ‘uniform’/‘distance’
  - algorithm - алгоритм оптимизации ‘auto’/‘ball_tree’/‘kd_tree’/‘brute’
  - metric - метрика расстояния. По умолчанию расстояние Минковского со степень 2

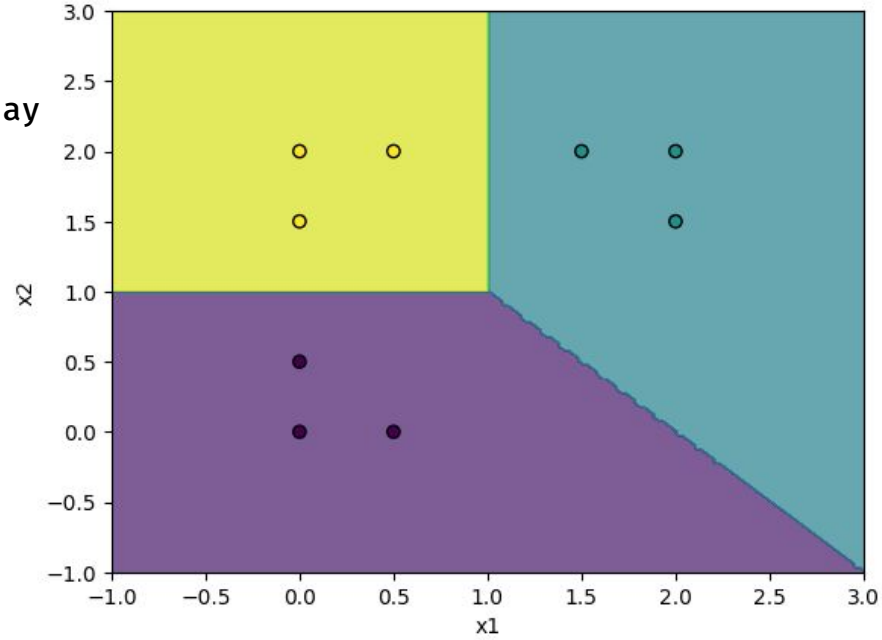
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(3,
                           weights = 'uniform')
knn.fit(x,y)
u = np.array([[0,1.1],[1,1.1],
              [2,0.1],[0.5,0.5]],
              dtype = np.float32)
y_pred = knn.predict(u).reshape(-1,1)
```

Недостатки kNN

- Сильная неустойчивость к шуму. kNN часто требует предобработки по избавлению от шума.
- Выбор метрики. В многомерных пространствах не всегда возможно оценить подходящую метрику. По умолчанию часто берут Евклидово расстояние. Также нормировка не всегда помогает, так как не обязательно, что все признаки одинаково значимы. Для выделение значимых признаков предварительно понижают размерность пространства.
- Вычислительная сложность для больших выборок. Для оптимизации точки хранят в виде дерева. **BallTree** - разбиение данных с выделением гиперсфер, **KD-Tree** - разбиение данных гиперплоскостями. **Brute-Force** - алгоритм с полным перебором.

Визуализация границ классов

```
from sklearn.inspection import DecisionBoundaryDisplay
display = DecisionBoundaryDisplay.from_estimator(
    knn, x, response_method='predict',
    xlabel='x1', ylabel='x2',
    grid_resolution=200, alpha = 0.7)
display.ax_.scatter(x[:,0],x[:,1],c=y,
                    edgecolors = 'k')
plt.show()
```



Бинарная классификация

- Если имеется всего 2 класса, то классификатор может обучаться для того, чтобы уметь определять принадлежность наблюдения к первому классу.
- Метки классов в бин. классификации кодируются как 0 и 1 (иногда -1 и 1). Для кодирования можно использовать LabelEncoder
- Если метки классов закодированы как 0 и 1, то можно значения меток рассматривать как вероятность принадлежности к одному из классов.
- Классификатор может иметь вероятностный характер, предсказывая вероятность принадлежности ***p*** к первому классу, и соответственно вероятность принадлежности ко второму классу будет равна ***(1-p)***

Логистическая регрессия

- Логистическая регрессия - бинарный классификатор, который основывается на линейной комбинации признаков (лин. регрессии) и оценке вероятности принадлежности к классу на основе значения лин. комбинации.

$$\widehat{p} = h_\theta(x) = \sigma(x^T\theta)$$

- $\widehat{p}$ - предсказанная вероятность принадлежности к классу
- $\sigma(\cdot)$ - сигмоидальная функция (сигмоида)
- θ - параметры модели
- Предсказание класса обычно основывается на пороге с вероятностью 0.5

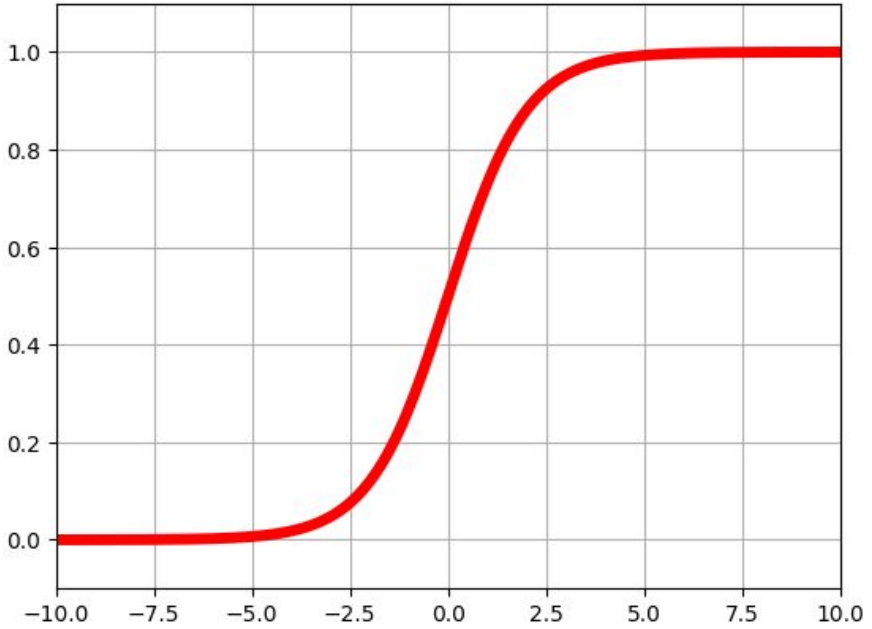
$$\widehat{y} = \begin{cases} 0 & , p < 0.5 \\ 1 & , p \geqslant 0.5 \end{cases}$$

Сигмоидальная функция

- Сигмоида приводит любое значения к диапазону (0, 1). Причем никогда не принимает значения равное 0 или 1

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

- В точке 0 имеет значение равное 0.5
- Соответственно, в логистической регрессии вероятность больше 0.5, если линейная комбинация признаков дала положительное число.



Функция потерь бинарной классификации

- Так как в бинарной классификации необходимо максимизировать вероятности правильной классификации, то потери для одного наблюдения

$$c(\theta) = \begin{cases} -\log(\widehat{p}) & , y = 1 \\ -\log(1 - \widehat{p}) & , y = 0 \end{cases}$$

- Полная функция потерь для всего набора данных называется *бинарной кросс-энтропией* или *log-loss*

$$J(\theta) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\widehat{p}_i) + (1 - y_i) \log(1 - \widehat{p}_i)]$$

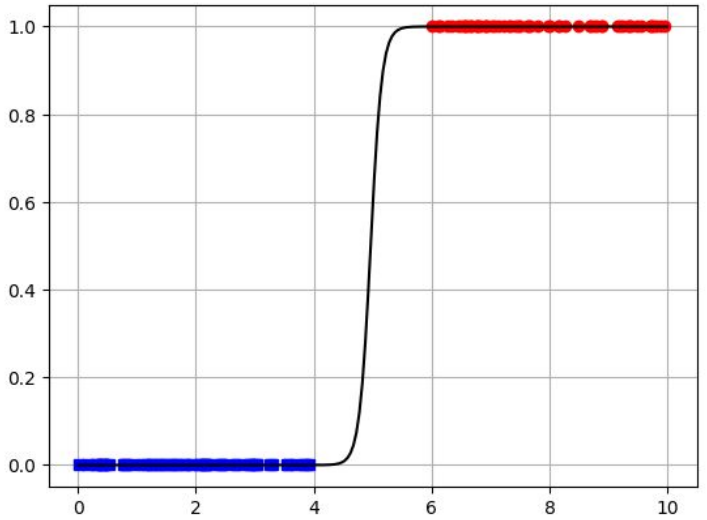
- Основная проблема, что для такой ф-ции потерь нет аналитического решения. Поэтому логистическая регрессия оптимизируется градиентным спуском, а частная производная по j-му параметру имеет вид

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left[\sigma(\theta^T x_i) - y_i \right] x_{i,j}$$

Логистическая регрессия в SKlearn

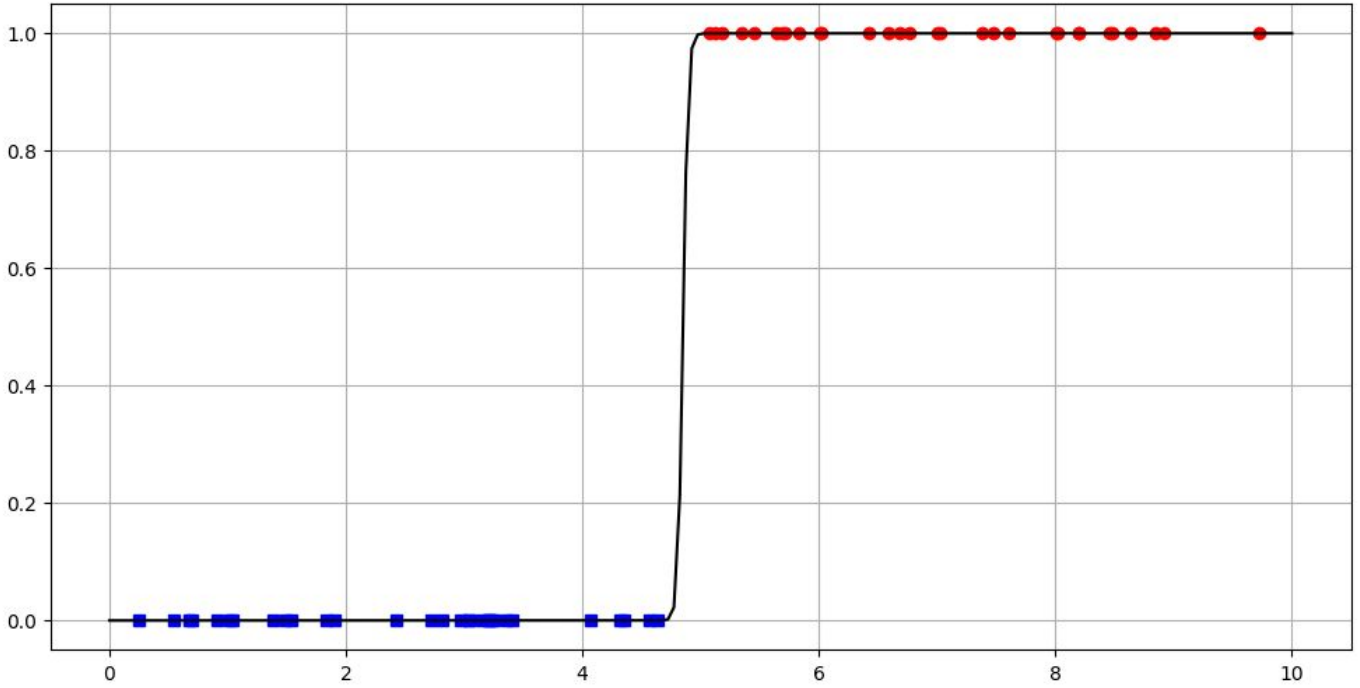
- sklearn.linear_model.LogisticRegression
- Параметры:
  - penalty - регуляризация
- Атрибуты:
  - coef_/intercept_ - как в лин. регрессии
  - classes_ - метки классов для обуч. в.
- Методы:
  - predict - предсказание метки класса
  - predict_proba - вероятности предсказания

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(penalty = None)
logreg.fit(lrx, lry)
u = np.array([[2],[5],[7]])
logreg.predict(u)
logreg.predict_proba(u)
```



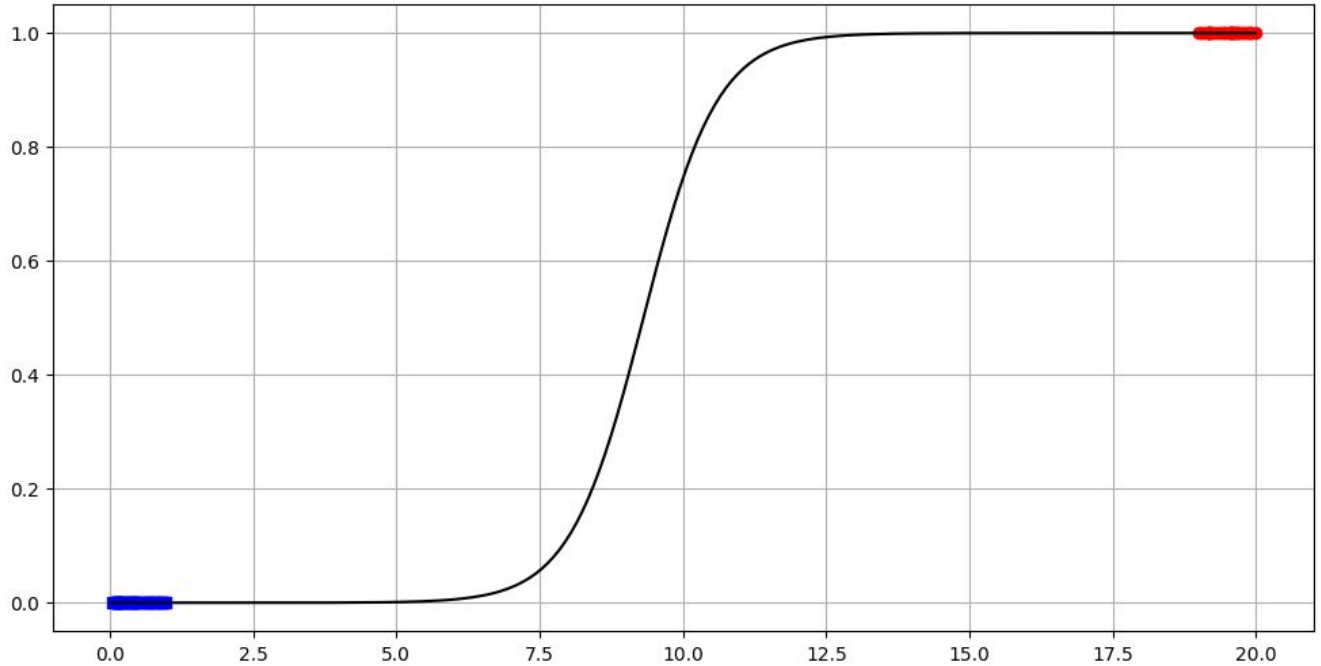
Признак класса соприкасается

В данном случае логистическая функция приближается к функции Хевисайда



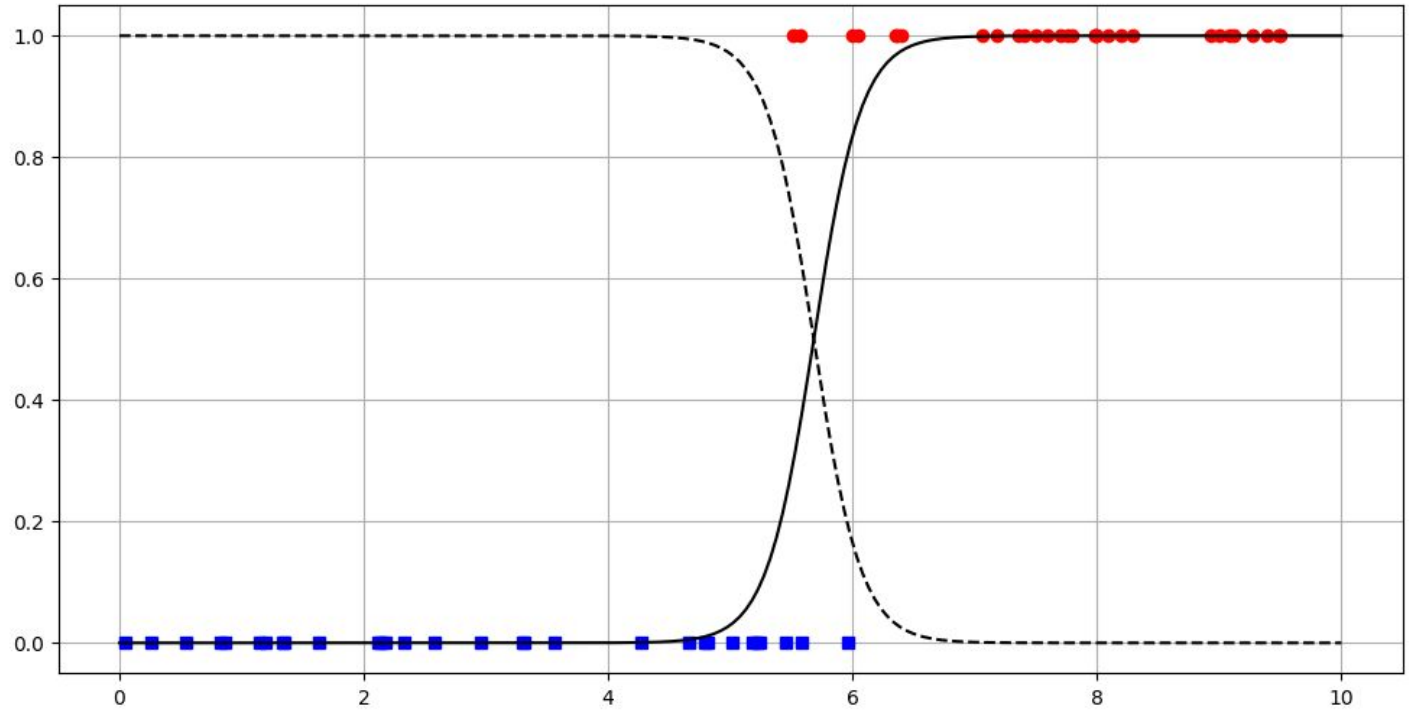
Признак класса разреженный

В данном случае появляется большая область неопределенности



Граница принятия решения

Если значение признака пересекается, то выбирается граница разделения и будет не точная классификация

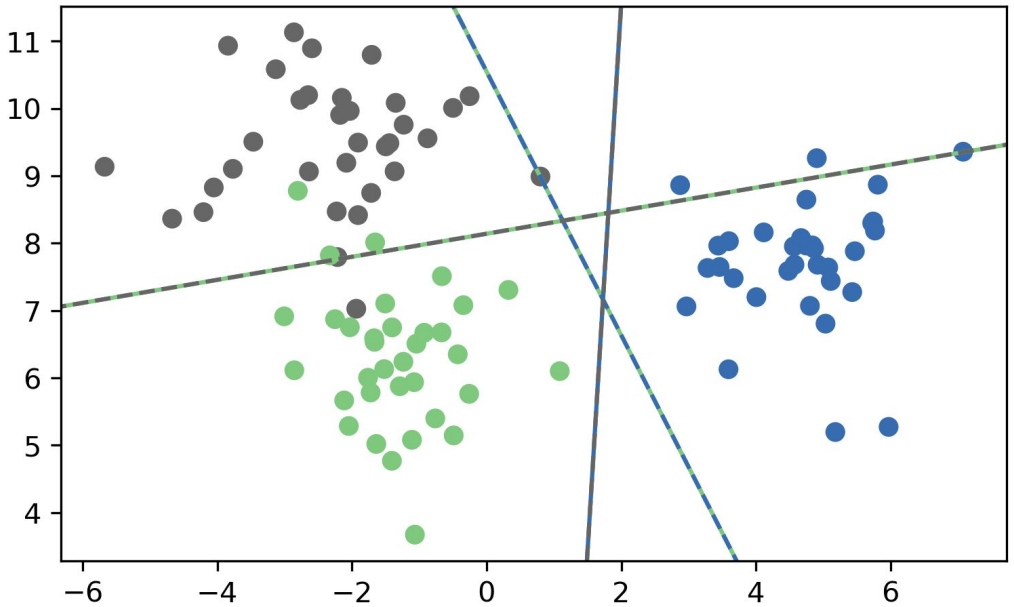


Множественная классификация

- Не все классификаторы номинально работают с более чем 2 классами
- В таких случаях можно для множественной классификации можно использовать множество бинарных классификаторов
- Существует 2 стратегии перехода от бинарной к множественной линейной классификации:
    - Один против одного - One vs One (OvO)
    - Одни против всех - One vs All (OvA)

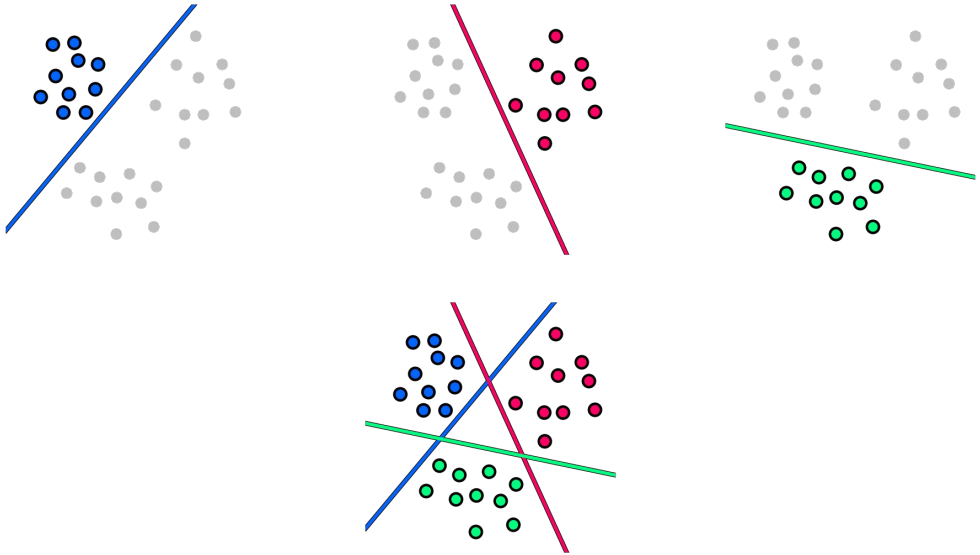
Один против одного

- Для каждой пары классов строится свой классификатор, которые позволяет их различать. Количество классификаторов в данном случае K * (K-1) / 2



Один против всех

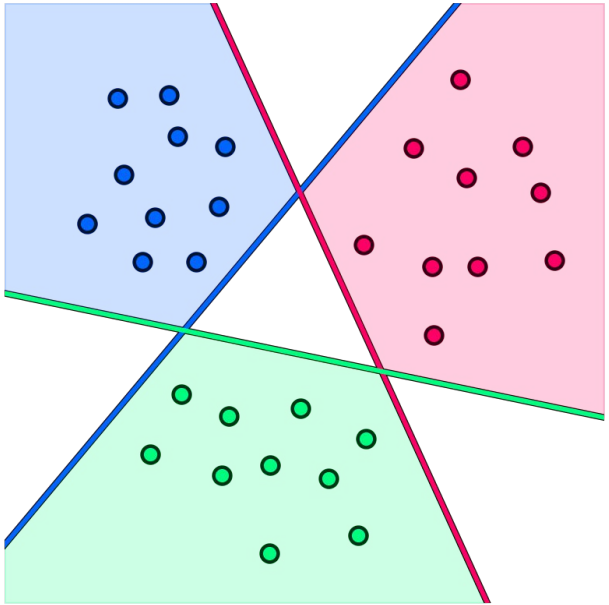
- Для каждого класса строится классификатор отделяющий его от всех остальных классов. Основная проблема, области неопределенности



https://jermwatt.github.io/machine_learning_refined/notes/7_Linear_multiclass_classification/7_2_OvA.html

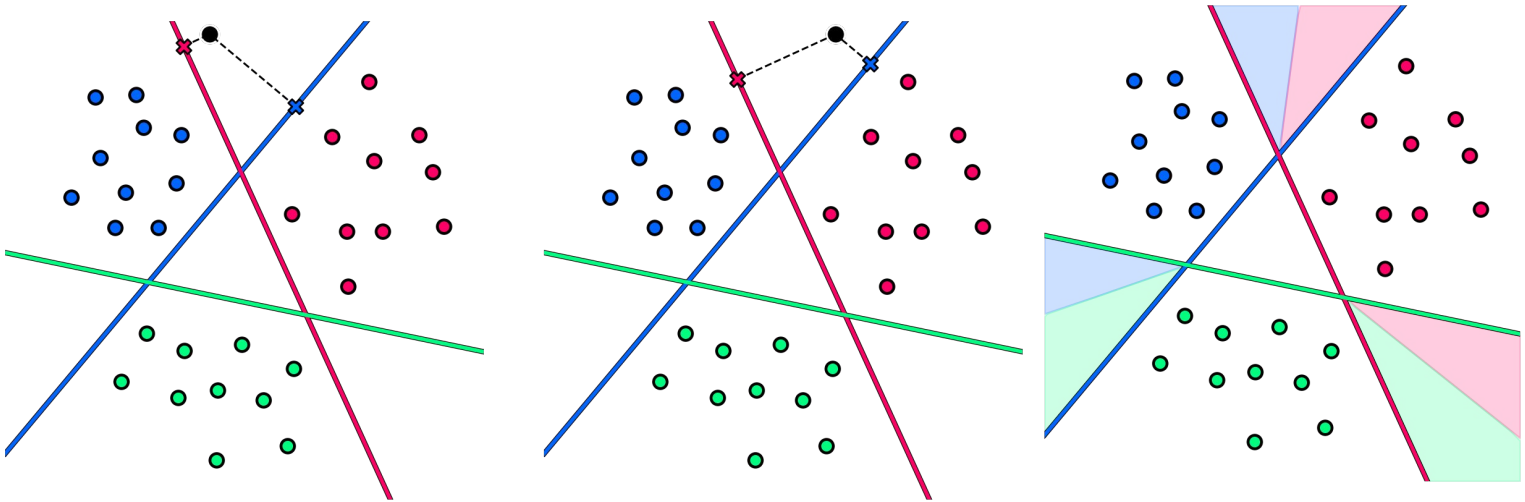
Один против всех. Одна положительная сторона

- В данном случае происходит однозначное определение целевого класса



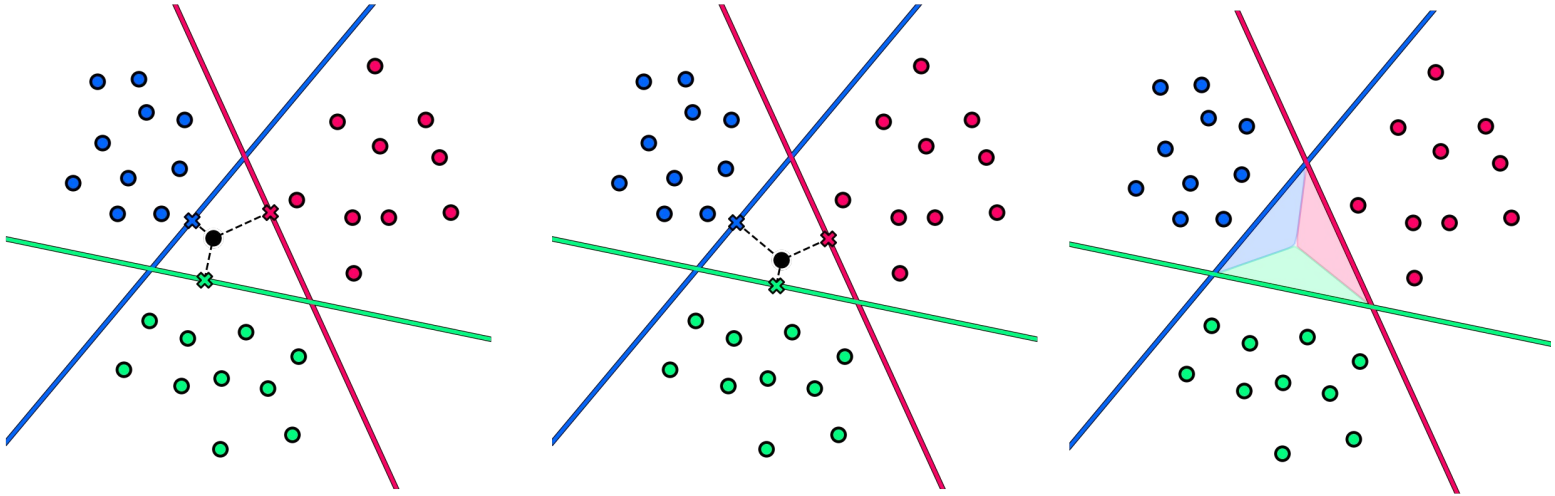
Один против всех. Несколько положительных сторон

- Для точки, попавшую в положительные области нескольких классов необходимо построить расстояние до каждой разделяющей линии. И отнести к тому классу, для которой расстояние больше



Один против всех. Ни одной положительной стороны

- Для точки, попавшую в положительные области нескольких классов необходимо построить расстояние до каждой разделяющей линии. И отнести к тому классу, для которой расстояние меньше



Множественная логистическая регрессия

- Множественная логистическая регрессия работает по принципу OvA.
- Для наблюдения для каждого класса k считается “сумма очков”

$$s_k(x) = x^T \theta^{(k)}$$

- Далее оценивается вероятность принадлежности к классу k через многопеременную логистическую функцию (Softmax)

$$\widehat{p}_k = \sigma(s(x))_k = \frac{e^{s_k(x)}}{\sum_{j=1}^{K} e^{s_j(x)}}$$

- Выбор класса происходит исходя из максимальной вероятности

$$\widehat{y} = \underset{k}{\mathbf{argmax}} \sigma(s(x))_k = \underset{k}{\mathbf{argmax}} s_k(x) = \underset{k}{\mathbf{argmax}} ((\theta^{(k)})^T x)$$

Перекрестная энтропия

- Перекрестная энтропия по сути является расстоянием между битами в теории информации и выводится из расстояния Кульбака-Лейблера
- Позволяет оценить качество множественной классификации

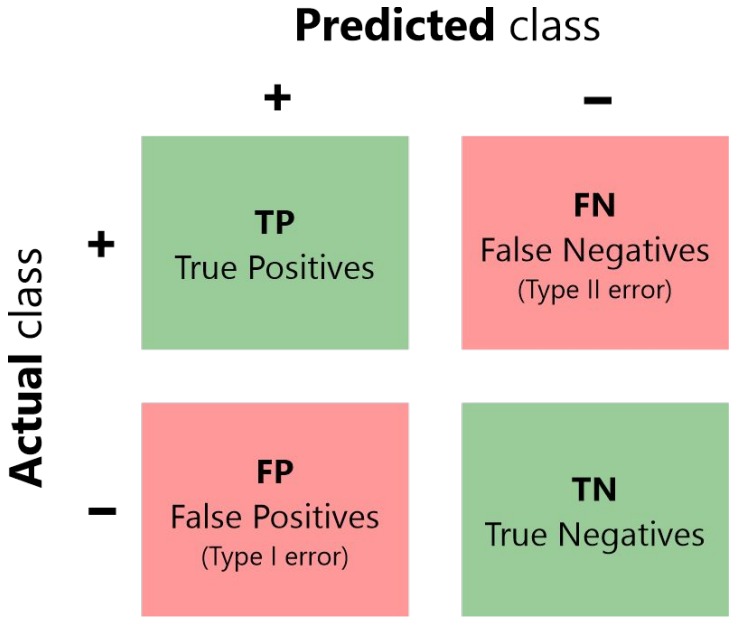
$$J(\Theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i=k}\log(\widehat{p}_{i=k})$$

- Градиент по параметрам класса k имеет вид

$$\nabla_{\theta^{(k)}} J(\Theta) = \frac{1}{N}\sum_{i=1}^{N}(\widehat{p}_{i=k} - y_{i=k})x_i$$

Метрики оценки качества

- Результаты классификации можно представить в виде матрицы ошибок/несоответствий
- Данная матрица показывает количество классифицированных тем или иным образом точек:
  - TP - правильно отнесены к классу
  - TN - правильно не отнесены к классу
  - FN - неправильно не отнесены к классу
  - FP - неправильно отнесены к классу
- sklearn.metrics.confusion_matrix и sklearn.metrics.ConfusionMatrixDisplay



Метрики оценки качества (2)

- Из матрицы ошибок можно рассчитать следующие метрики:
  - “Точность”/Accuracy (sklearn.metrics.accuracy_score) - **(TP+TN)/N** .
  - Точность/Precision (sklearn.metrics.precision_score) - **TP/(TP+FP)** - показывает сколько из истинных предсказаний являются реально истинными
  - Полнота/Recall (sklearn.metrics.recall_score) - **TP/(TP+FN)** - показывает сколько реально истинных значений были правильно определены как истинные
  - F1-мера (sklearn.metrics.f1_score) - **2*TP/(2*TP+FN+FP)** - среднее гармоническое из Precision и Recall

Precision и Recall

- Precision - показывает как мало ложных срабатываний. Например, при назначении опасного лечения, необходимо минимизировать ложные срабатывания, чтобы не назначить опасное лечение для здорового человека.

И/П	Больной	Здоров
Больной	TP= 49	FN= 1
Здоров	FP= 25	TN= 25

- Recall - показывают как мало истинных срабатываний отбросили. Например, при выпуске товаров важнее не выпустить бракованный, чем товар без дефектов отметить бракованным.

И/П	Брак	Не брак
Брак	TP= 25	FN= 25
Не брак	FP= 1	TN= 49