

# Введение в искусственный интеллект. Машинное обучение

Лекция 5. Линейные классификаторы и стохастический градиентный спуск

Бабин Д.Н., Иванов И.Е., Петюшко А.А.

кафедра Математической Теории Интеллектуальных Систем

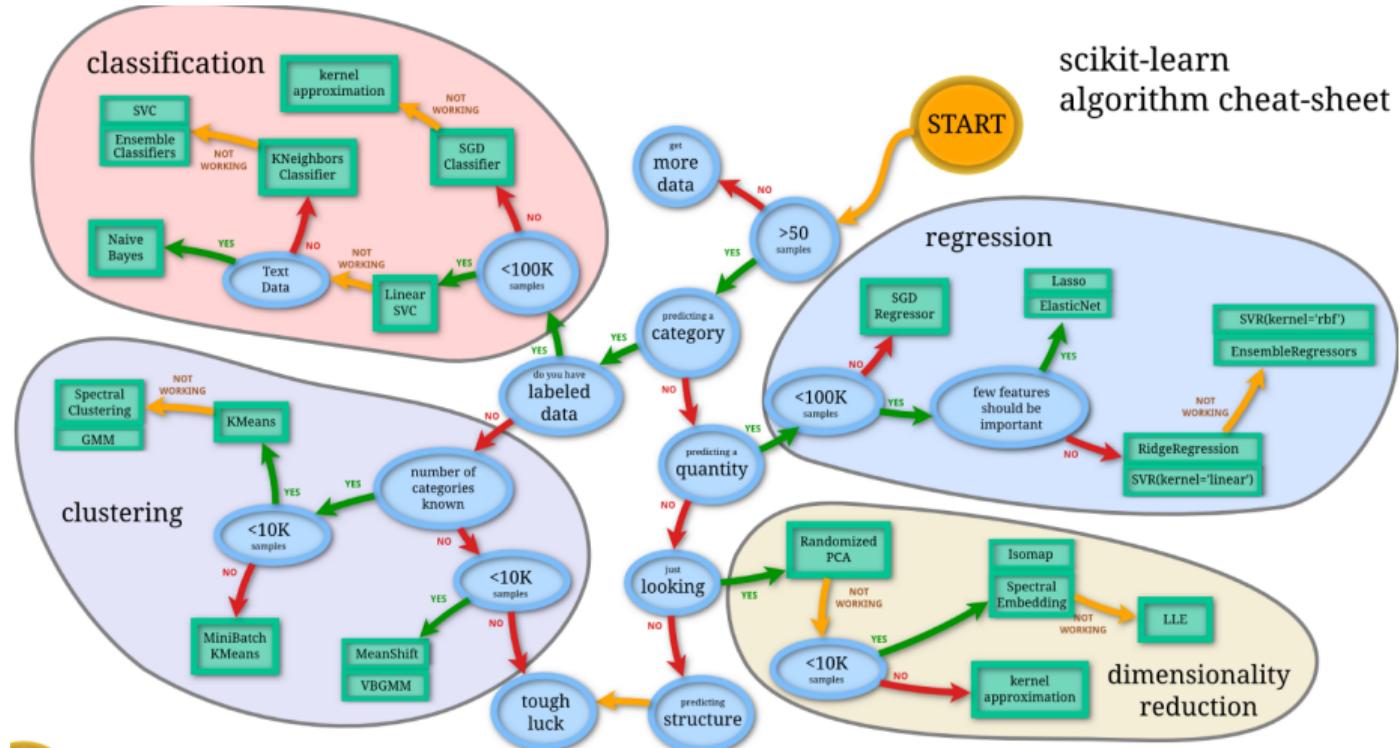
22 марта 2019г.



# План лекции

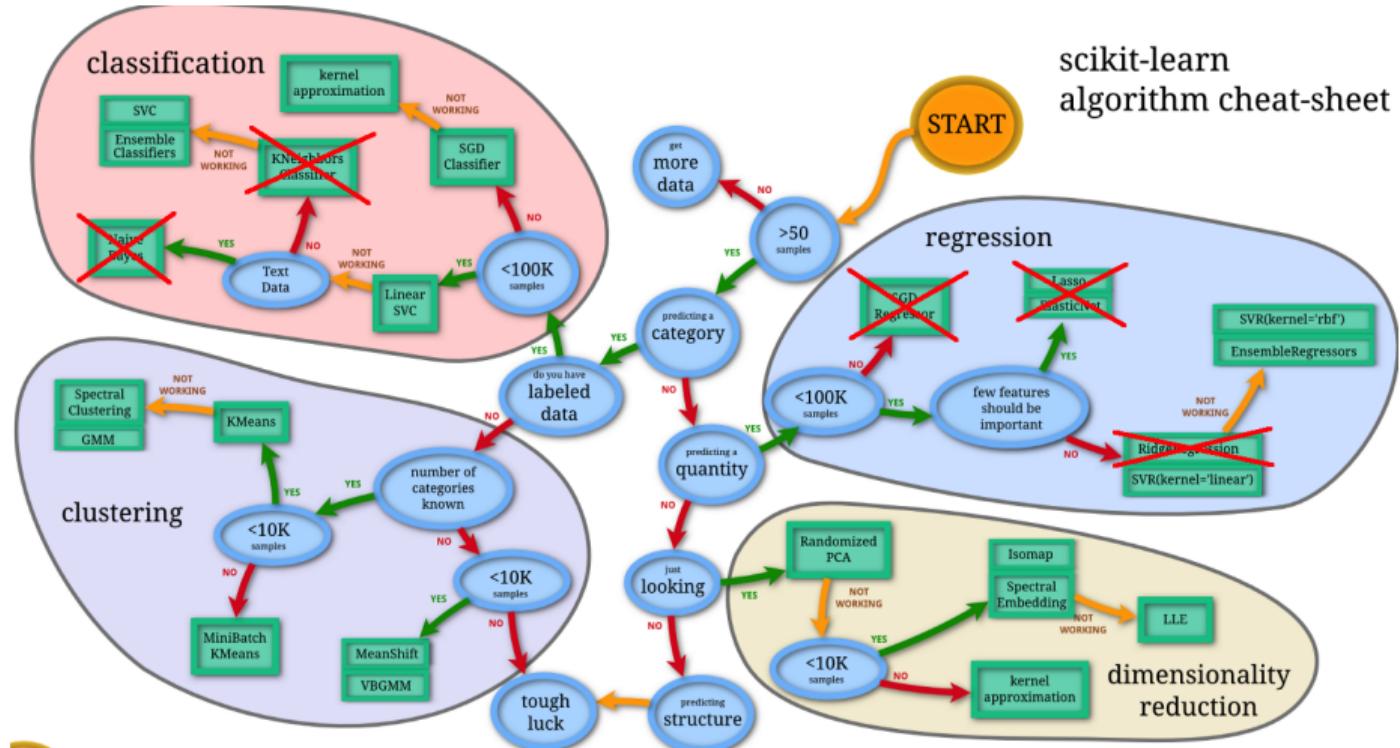
- ① Эмпирический риск и его минимизация
- ② Разделяющая поверхность
- ③ GD и SGD
- ④ Линейный классификатор и примеры (нейрон, логистическая регрессия, оптимальный байесовский классификатор)

# Дорожная карта Scikit-Learn<sup>1</sup>



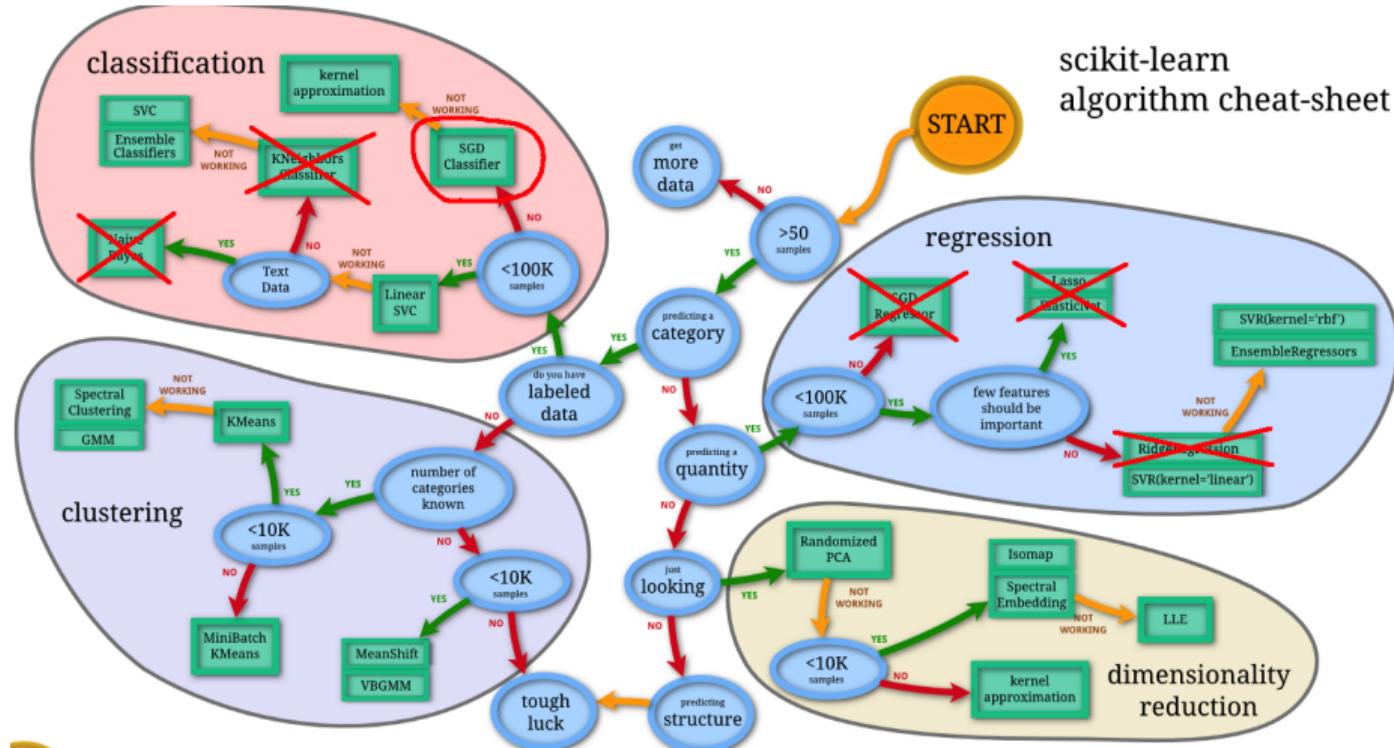
<sup>1</sup>[https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/](https://scikit-learn.org/stable/tutorial/machine_learning_map/)

# Дорожная карта Scikit-Learn<sup>1</sup>



<sup>1</sup>[https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/](https://scikit-learn.org/stable/tutorial/machine_learning_map/)

# Дорожная карта Scikit-Learn<sup>1</sup>



<sup>1</sup>[https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/](https://scikit-learn.org/stable/tutorial/machine_learning_map/)

# Обучение по прецедентам

- $X$  – множество описаний объектов,  $Y$  – множество допустимых ответов
- Неизвестная целевая зависимость: отображение  $y^* : X \rightarrow Y$
- Конечная обучающая выборка:  $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$ , т.ч.  $y_i = y^*(x_i)$

# Обучение по прецедентам

- $X$  – множество описаний объектов,  $Y$  – множество допустимых ответов
- Неизвестная целевая зависимость: отображение  $y^* : X \rightarrow Y$
- Конечная обучающая выборка:  $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$ , т.ч.  $y_i = y^*(x_i)$
- **Задача обучения по прецедентам** состоит в том, чтобы построить алгоритм  $a : X \rightarrow Y$ , который приближал бы целевую зависимость  $y^*$  как на обучающей выборке  $X^m$ , так и на всём множестве  $X$

# Обучение по прецедентам

- $X$  – множество описаний объектов,  $Y$  – множество допустимых ответов
- Неизвестная целевая зависимость: отображение  $y^* : X \rightarrow Y$
- Конечная обучающая выборка:  $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$ , т.ч.  $y_i = y^*(x_i)$
- **Задача обучения по прецедентам** состоит в том, чтобы построить алгоритм  $a : X \rightarrow Y$ , который приближал бы целевую зависимость  $y^*$  как на обучающей выборке  $X^m$ , так и на всём множестве  $X$
- **Эмпирический риск** – это средняя величина ошибки  $a$  на  $X^m$

# Обучение по прецедентам

- $X$  – множество описаний объектов,  $Y$  – множество допустимых ответов
- Неизвестная целевая зависимость: отображение  $y^* : X \rightarrow Y$
- Конечная обучающая выборка:  $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$ , т.ч.  $y_i = y^*(x_i)$
- **Задача обучения по прецедентам** состоит в том, чтобы построить алгоритм  $a : X \rightarrow Y$ , который приближал бы целевую зависимость  $y^*$  как на обучающей выборке  $X^m$ , так и на всём множестве  $X$
- **Эмпирический риск** – это средняя величина ошибки  $a$  на  $X^m$
- **Метод минимизации эмпирического риска** – это общий подход к решению широкого класса задач обучения по прецедентам (задачи классификации и регрессии)

## Функция потерь $L(y, y')$

Характеризует величину отклонения ответа  $y = a(x)$  от правильного ответа  $y' = y^*(x)$  на объекте  $x \in X$



# Эмпирический риск – определения

## Функция потерь $L(y, y')$

Характеризует величину отклонения ответа  $y = a(x)$  от правильного ответа  $y' = y^*(x)$  на объекте  $x \in X$

## Множество алгоритмов $A = \{a : X \rightarrow Y\}$

В этом множестве будет вестись поиск отображения, приближающего неизвестную целевую зависимость



# Эмпирический риск – определения

## Функция потерь $L(y, y')$

Характеризует величину отклонения ответа  $y = a(x)$  от правильного ответа  $y' = y^*(x)$  на объекте  $x \in X$

## Множество алгоритмов $A = \{a : X \rightarrow Y\}$

В этом множестве будет вестись поиск отображения, приближающего неизвестную целевую зависимость

## Эмпирический риск

Функционал качества, характеризующий среднюю ошибку алгоритма  $a$  на выборке  $X^m$ :  
 $R(a, X^m) = \frac{1}{m} \sum_{i=1}^m L(a(x_i), y^*(x_i))$

## Минимизация эмпирического риска (М.Э.Р.)

В заданном множестве алгоритмов  $A$  найти алгоритм, минимизирующий эмпирический риск:

$$a = \arg \min_{a \in A} R(a, X^m)$$



# Минимизация эмпирического риска

## Минимизация эмпирического риска (М.Э.Р.)

В заданном множестве алгоритмов  $A$  найти алгоритм, минимизирующий эмпирический риск:

$$a = \arg \min_{a \in A} R(a, X^m)$$

### Достоинство М.Э.Р.

Конструктивный и универсальный подход, позволяющий сводить задачу обучения к задачам численной оптимизации



# Минимизация эмпирического риска

## Минимизация эмпирического риска (М.Э.Р.)

В заданном множестве алгоритмов  $A$  найти алгоритм, минимизирующий эмпирический риск:

$$a = \arg \min_{a \in A} R(a, X^m)$$

### Достоинство М.Э.Р.

Конструктивный и универсальный подход, позволяющий сводить задачу обучения к задачам численной оптимизации

### Недостаток М.Э.Р.

Явление переобучения, которое возникает практически всегда при использовании метода м.э.р.

## Задача классификации

- Пороговая функция  $L(y, y') = [y \neq y']$
- Функция разрывна  $\Rightarrow$  минимизация эмпирического риска – это задача комбинаторной оптимизации  $\Rightarrow$  во многих практически важных случаях сводится к поиску максимальной совместной подсистемы в системе неравенств (число неравенств совпадает с число объектов обучения  $m$ ) и является NP-полной

# Примеры функций потерь

## Задача классификации

- Пороговая функция  $L(y, y') = [y \neq y']$
- Функция разрывна  $\Rightarrow$  минимизация эмпирического риска – это задача комбинаторной оптимизации  $\Rightarrow$  во многих практически важных случаях сводится к поиску максимальной совместной подсистемы в системе неравенств (число неравенств совпадает с число объектов обучения  $m$ ) и является NP-полной

## Задача регрессии

Квадратичная функция потерь  $L(y, y') = (y - y')^2$



# Разделяющая поверхность

- Рассмотрим задачу бинарной классификации:  $X \rightarrow Y$ ,  $Y = \{+1, -1\}$  на обучающей выборке  $X^m = (x_i, y_i)_{i=1}^m$
- Будем алгоритм искать в виде  $a(x, w) = \text{sign } g(x, w)$ , где  $g(x, w)$  - дискриминантная функция, а  $w$  - вектор параметров
- $g(x, w) = 0$  - разделяющая поверхность;  
 $M_i(w) = y_i g(x_i, w)$  - отступ объекта  $x_i$ ;  
 $M_i(w) < 0 \Leftrightarrow a(x_i, w) \neq y_i$ .

# Разделяющая поверхность

- Рассмотрим задачу бинарной классификации:  $X \rightarrow Y$ ,  $Y = \{+1, -1\}$  на обучающей выборке  $X^m = (x_i, y_i)_{i=1}^m$
- Будем алгоритм искать в виде  $a(x, w) = \text{sign } g(x, w)$ , где  $g(x, w)$  - дискриминантная функция, а  $w$  - вектор параметров
- $g(x, w) = 0$  - разделяющая поверхность;  
 $M_i(w) = y_i g(x_i, w)$  - отступ объекта  $x_i$ ;  
 $M_i(w) < 0 \Leftrightarrow a(x_i, w) \neq y_i$ .
- М.Э.Р.:  $R(a, X^m) = \frac{1}{m} \sum_{i=1}^m [M_i(w) < 0] \leq \tilde{R}(a, X^m) = \frac{1}{m} \sum_{i=1}^m L(M_i(w))$ ,  
где новая функция потерь  $L(M)$  - невозрастающая и неотрицательная  
аппроксимация функции  $[M < 0]$ , т.ч.:  $L(M) \geq [M < 0]$

# Разделяющая поверхность

- Рассмотрим задачу бинарной классификации:  $X \rightarrow Y$ ,  $Y = \{+1, -1\}$  на обучающей выборке  $X^m = (x_i, y_i)_{i=1}^m$
- Будем алгоритм искать в виде  $a(x, w) = \text{sign } g(x, w)$ , где  $g(x, w)$  - дискриминантная функция, а  $w$  - вектор параметров
- $g(x, w) = 0$  - разделяющая поверхность;  
 $M_i(w) = y_i g(x_i, w)$  - отступ объекта  $x_i$ ;  
 $M_i(w) < 0 \Leftrightarrow a(x_i, w) \neq y_i$ .
- М.Э.Р.:  $R(a, X^m) = \frac{1}{m} \sum_{i=1}^m [M_i(w) < 0] \leq \tilde{R}(a, X^m) = \frac{1}{m} \sum_{i=1}^m L(M_i(w))$ ,  
где новая функция потерь  $L(M)$  - невозрастающая и неотрицательная  
аппроксимация функции  $[M < 0]$ , т.ч.:  $L(M) \geq [M < 0]$

**Замечание.** В дальнейшем будем предполагать, что мы работаем сразу с аппроксимаций Э.Р.  $\tilde{R}$ , поэтому знак  $\sim$  будем опускать.

# Вероятностный смысл минимизации аппроксимированного Э.Р.

Рассмотрим принцип максимизации правдоподобия, или MLE (Maximum Likelihood Estimation).

- Параметрическая модель плотности распределения  $p(x, y|w)$
- Максимизация логарифма правдоподобия

$$L(w, X^m) = \ln \prod_{i=1}^m p(x_i, y_i|w) = \sum_{i=1}^m \ln p(x_i, y_i|w) \rightarrow \max_w$$

# Вероятностный смысл минимизации аппроксимированного Э.Р.

Рассмотрим принцип максимизации правдоподобия, или MLE (Maximum Likelihood Estimation).

- Параметрическая модель плотности распределения  $p(x, y|w)$
- Максимизация логарифма правдоподобия

$$L(w, X^m) = \ln \prod_{i=1}^m p(x_i, y_i|w) = \sum_{i=1}^m \ln p(x_i, y_i|w) \rightarrow \max_w$$

- Минимизация аппроксимированного Э.Р.

$$R(w, X^m) = \frac{1}{m} \sum_{i=1}^m L(M_i(w)) \rightarrow \min_w$$

# Вероятностный смысл минимизации аппроксимированного Э.Р.

Рассмотрим принцип максимизации правдоподобия, или MLE (Maximum Likelihood Estimation).

- Параметрическая модель плотности распределения  $p(x, y|w)$
- Максимизация логарифма правдоподобия

$$L(w, X^m) = \ln \prod_{i=1}^m p(x_i, y_i|w) = \sum_{i=1}^m \ln p(x_i, y_i|w) \rightarrow \max_w$$

- Минимизация аппроксимированного Э.Р.

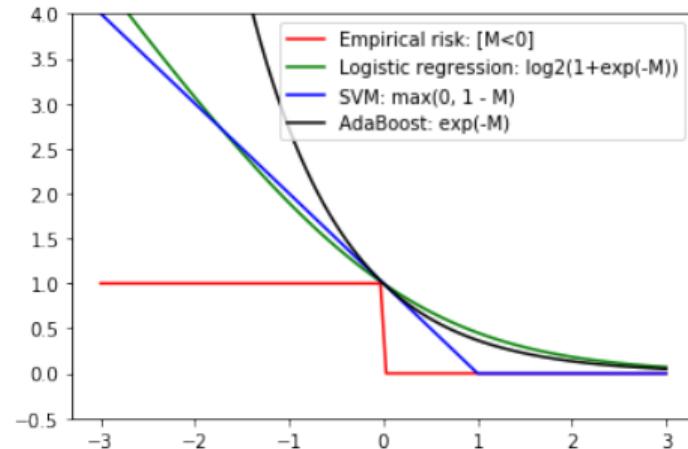
$$R(w, X^m) = \frac{1}{m} \sum_{i=1}^m L(M_i(w)) \rightarrow \min_w$$

Вывод. Эти два принципа эквивалентны при  $L(M_i(w)) = -\ln p(x_i, y_i|w)$  (коэффициент  $\frac{1}{m}$  не влияет на вывод).

# Об аппроксимации

- Рассмотрим аппроксимацию функции ошибки на обучающем примере:  $L(M) \geq [M < 0]$
- В дальнейшем будем рассматривать в основном достаточно гладкие (непрерывные и дифференцируемые) функции  $L(M)$
- Некоторые аппроксимации способны улучшать обобщающую способность классификатора
- Непрерывные аппроксимации позволяют применять известные численные методы оптимизации для настройки весов  $w$  (например, градиентные методы и методы выпуклого программирования)

Примеры аппроксимации функции  $[M < 0]$ :



# Классический градиентный спуск

Задача: минимизировать аппроксимированный Э.Р. (выбор алгоритма осуществляется по  $w$ ):

$$R(w) = \frac{1}{m} \sum_{i=1}^m L(M_i(w)) = \frac{1}{m} \sum_{i=1}^m L_i(w) \rightarrow \min_w$$

# Классический градиентный спуск

Задача: минимизировать аппроксимированный Э.Р. (выбор алгоритма осуществляется по  $w$ ):

$$R(w) = \frac{1}{m} \sum_{i=1}^m L(M_i(w)) = \frac{1}{m} \sum_{i=1}^m L_i(w) \rightarrow \min_w$$

## Численная оптимизация методом градиентного спуска

- $w^{(0)}$  := начальное приближение
- $w^{(t+1)} := w^{(t)} - \eta \cdot \nabla R(w^{(t)})$  - итерация алгоритма
- $\eta$  - градиентный шаг



# Классический градиентный спуск

Задача: минимизировать аппроксимированный Э.Р. (выбор алгоритма осуществляется по  $w$ ):

$$R(w) = \frac{1}{m} \sum_{i=1}^m L(M_i(w)) = \frac{1}{m} \sum_{i=1}^m L_i(w) \rightarrow \min_w$$

## Численная оптимизация методом градиентного спуска

- $w^{(0)}$  := начальное приближение
- $w^{(t+1)} := w^{(t)} - \eta \cdot \nabla R(w^{(t)})$  - итерация алгоритма
- $\eta$  - градиентный шаг

**Проблема:** сложно считать в условиях большого количества объектов в обучающей выборке.



# Стохастический градиентный спуск

## Алгоритм стохастического градиентного спуска

- Инициализация весов  $w$
- Инициализация Э.Р.  $R := \frac{1}{m} \sum_{i=1}^m L_i(w)$

## Итерации

- Выбор объекта  $x_i \in X^m$  (например, случайным образом)
- Вычисление ошибки на данном объекте:  $\varepsilon_i = L_i(w)$
- Шаг градиентного спуска:  $w := w - \eta \cdot \nabla L_i(w)$
- Вычисление сглаженного Э.Р.:  $R := (1 - \lambda)R + \lambda \varepsilon_i$

**Замечание:** параметр сглаживания  $\lambda \in [0, 1]$  (можно использовать, например, 0.1).

## Инициализация

- $w_j = 0 \quad \forall j = 1, \dots, n$  (где  $n$  - число весов)
- $w_j = \text{rand}\left(-\frac{1}{2n}, \frac{1}{2n}\right)$
- Предобучение на другой обучающей выборке

# Вариативность SGD

## Инициализация

- $w_j = 0 \quad \forall j = 1, \dots, n$  (где  $n$  - число весов)
- $w_j = rand(-\frac{1}{2n}, \frac{1}{2n})$
- Предобучение на другой обучающей выборке

## Порядок выбора объектов $x_i$

- Случайная перетасовка: попаременно брать объекты разных классов
- Чаще брать объекты с большой ошибкой (маленькое значение  $M_i$ )
- Чаще брать объекты с большой неуверенностью (маленькое значение  $|M_i|$ )



# Вариативность SGD

## Инициализация

- $w_j = 0 \quad \forall j = 1, \dots, n$  (где  $n$  - число весов)
- $w_j = rand(-\frac{1}{2n}, \frac{1}{2n})$
- Предобучение на другой обучающей выборке

## Порядок выбора объектов $x_i$

- Случайная перетасовка: попаременно брать объекты разных классов
- Чаще брать объекты с большой ошибкой (маленькое значение  $M_i$ )
- Чаще брать объекты с большой неуверенностью (маленькое значение  $|M_i|$ )

## Критерий остановки

- Исчерпали лимит по числу шагов
- Значение Э.Р. либо весов перестало меняться

# Пакетный (mini-batch) SGD

## Пакетный SGD

**Идея:** на каждом шаге использовать более надежную оценку градиента не на одном примере, а на нескольких

### Итерации

- Выбор подмножества объектов мощности  $1 < k < m$ :  $J = \{i_1, \dots, i_k\}$
- Вычисление ошибки на этих объектах:  $L_{i_1}(w^{(t)}), \dots, L_{i_k}(w^{(t)})$
- Шаг градиентного спуска:  $w^{(t+1)} := w^{(t)} - \eta \cdot \frac{1}{k} \sum_{j=1}^k \nabla_w L_{i_j}(w^{(t)})$



# Выбор шага SGD

- Сходимость гарантируется<sup>2</sup> для выпуклых функций Э.Р. и ограниченного градиента при  
 $\eta_t \rightarrow 0, \sum_{t=0}^{\infty} \eta_t = \infty, \sum_{t=0}^{\infty} \eta_t^2 < \infty$  (например, подходит  $\eta_t = \frac{1}{t}$ )
  - При этом скорость сходимости будет также  $O(\frac{1}{t})$
- Метод скорейшего градиентного спуска  
 $R(w - \eta \nabla R(w)) \rightarrow \min_{\eta}$  позволяет найти оптимальный  $\eta^*$
- Время от времени бывает полезно делать большие шаги для “выпрыгивания” из локальных минимумов

<sup>2</sup>Robbins, H. and Monro, S. (1951). “A stochastic approximation method”

# Плюсы и минусы SGD

## Плюсы

- Легко реализуется на практике;
- Легко обобщается на любые алгоритмы и функции потерь;
- Возможно онлайн до-обучение (для нового  $x_i$ );
- Необязательно использовать все объекты  $x_i$ .

# Плюсы и минусы SGD

## Плюсы

- Легко реализуется на практике;
- Легко обобщается на любые алгоритмы и функции потерь;
- Возможно онлайн до-обучение (для нового  $x_i$ );
- Необязательно использовать все объекты  $x_i$ .

## Минусы

- На практике возможно расходимость / медленная сходимость;
- Локальные минимумы!!!
- Подбор шага градиента, условия остановки неочевидны;
- Переобучение.

# Понятие линейной классификации

## Линейный классификатор

Это алгоритм классификации, основанный на построении **линейной** разделяющей поверхности



## Линейный классификатор

Это алгоритм классификации, основанный на построении **линейной** разделяющей поверхности

- В случае **двух** классов разделяющей поверхностью является **гиперплоскость**, которая делит пространство признаков на два полупространства

## Линейный классификатор

Это алгоритм классификации, основанный на построении **линейной** разделяющей поверхности

- В случае **двух** классов разделяющей поверхностью является **гиперплоскость**, которая делит пространство признаков на два полупространства
- В случае числа классов **больше двух** разделяющая поверхность **кусочно-линейна**



## Два класса

Дискриминантная функция:

$g(x, w) = \sum_{j=1}^n w_j f_j - w_0$ , где  
 $f_j : X \rightarrow \mathbb{R}$  – числовые признаки.

Алгоритм классификации

$a(x, w) = \text{sign}(\sum_{j=1}^n w_j f_j - w_0)$ .

Если ввести константный признак  $f_0 \equiv -1$ , то

$x = (f_0(x), \dots, f_n(x))$ ,

и алгоритм в векторной записи:

$a(x, w) = \text{sign}(\langle w, x \rangle)$ .

Отступ объекта  $x_i$ :  $M_i = \langle w, x_i \rangle y_i$

# Линейная классификация: определения

## Два класса

Дискриминантная функция:

$g(x, w) = \sum_{j=1}^n w_j f_j - w_0$ , где  
 $f_j : X \rightarrow \mathbb{R}$  – числовые признаки.

Алгоритм классификации

$a(x, w) = \text{sign}(\sum_{j=1}^n w_j f_j - w_0)$ .

Если ввести константный признак  $f_0 \equiv -1$ , то

$x = (f_0(x), \dots, f_n(x))$ ,

и алгоритм в векторной записи:

$a(x, w) = \text{sign}(\langle w, x \rangle)$ .

Отступ объекта  $x_i$ :  $M_i = \langle w, x_i \rangle y_i$

## Произвольное число классов

У каждого класса  $c \in Y$  свой вектор весов:  $w^c = (w_0^c, \dots, w_n^c)$ .

Линейный классификатор:

$a(x, w) = \arg \max_{c \in Y} \sum_{j=0}^n w_j^c f_j(x) = \arg \max_{c \in Y} \langle w^c, x \rangle$ .

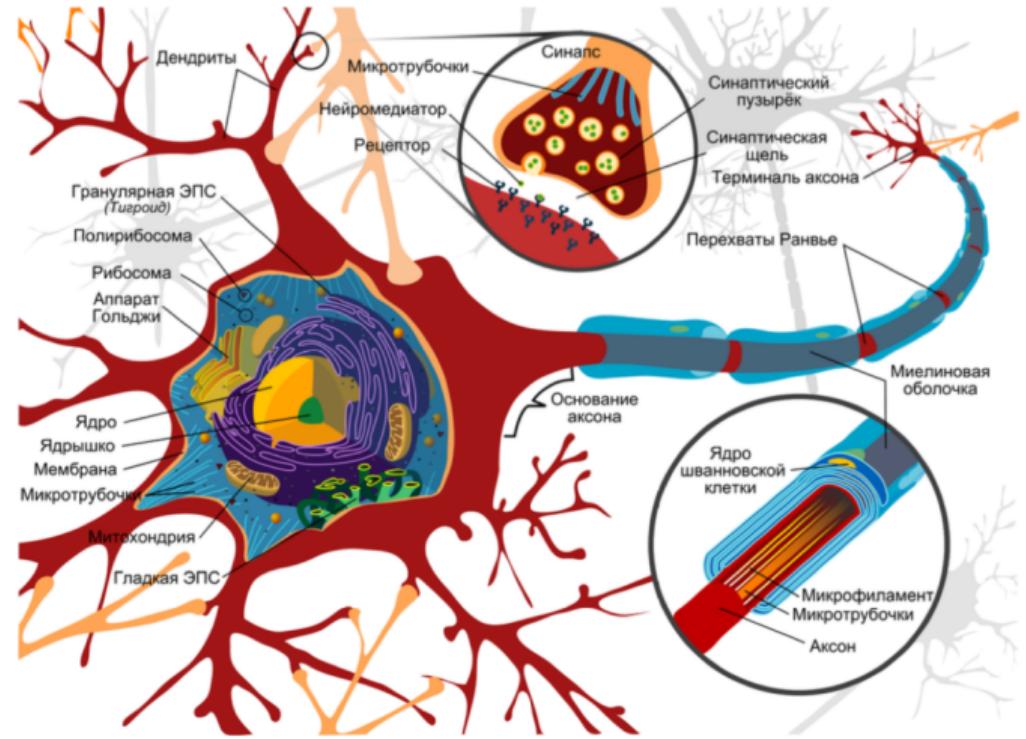
Отступ:

$M_i(w) = \langle x_i, w^{y_i} \rangle - \max_{c \in Y, c \neq y_i} \langle x_i, w^c \rangle$

**Замечание.** Обратите внимание на разницу со случаем двух классов!

## Биологический нейрон

- Кора головного мозга содержит  $10^{11}$  нейронов
  - Каждый нейрон связан синапсами с  $10^3 - 10^4$  другими нейронами
  - Скорость распространения импульсов 100 м/с
  - Входы (много) - дендриты
  - Выход (один) - аксон

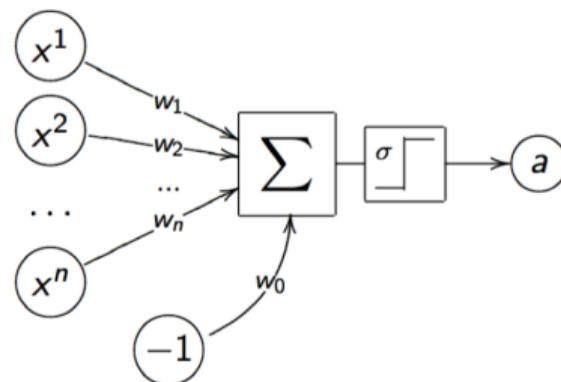


# Математическая модель нейрона

Предложена МакКалоком и Питтсом в 1943 году<sup>3</sup>.

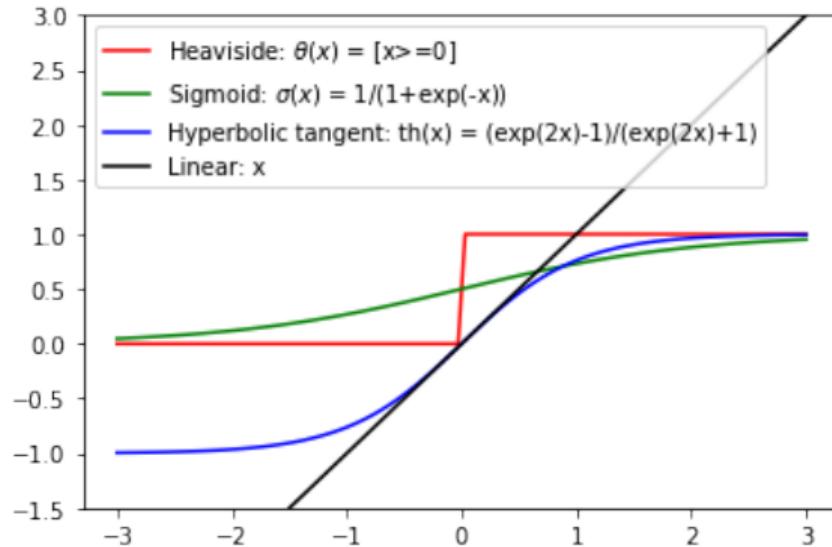
$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^n w_j f_j - w_0\right)$$

где  $\sigma(x)$  - некоторая функция активации (например, sign).



<sup>3</sup>McCulloch, W. S. and Pitts, W. (1943). "A logical calculus of the ideas immanent in nervous activity"

# Примеры функций активаций



# SGD для линейной регрессии: ADALINE

В задаче линейной регрессии функция потерь:

$$L(a(x_i, w), y_i) = (a(x_i, w) - y_i)^2$$

Адаптивный линейный нейрон (ADaptive Linear NEuron) ADALINE предложен Видроу и Хоффом в 1960<sup>4</sup>:

$$a(x, w) = \langle w, x \rangle$$

Градиентный шаг стохастического градиентного спуска - т.н. **дельта-правило**:

$$w^{(t+1)} = w^{(t)} - \eta(\langle w^{(t)}, x_i \rangle - y_i)x_i$$

---

<sup>4</sup>Widrow, B. and Hoff, M. E. (1960). "Adaptive switching circuits"



# SGD для линейного классификатора

Алгоритм стохастического градиентного спуска в общем виде для линейного классификатора:

$$L = L(\langle w, x_i \rangle y_i) \Rightarrow w^{(t+1)} = w^{(t)} - \eta L'(z) \Big|_{z=\langle w^{(t)}, x_i \rangle y_i} x_i y_i$$

Рассмотрим подробнее некоторые частные случаи.

## Правило Хэбба, 1949<sup>5</sup>

В задаче бинарной ( $Y = \{-1, +1\}$ ) классификации линейный классификатор:

$$a(x, w) = \text{sign}(\langle w, x \rangle)$$

Функция потерь:  $L(a(x_i, w), y_i) = [a(x_i, w) \neq y_i]$ .

Градиентный шаг: если  $a(x_i, w^{(t)}) \neq y_i \Leftrightarrow a(x_i, w^{(t)})y_i < 0$ , то  $w^{(t+1)} = w^{(t)} + \eta x_i y_i$

<sup>5</sup>Hebb, D. O. (1949). "The organization of behavior: a neuropsychological theory."

<sup>6</sup>Rosenblatt, F. (1957). "The perceptron, a perceiving and recognizing automaton"

# История: правила Хэбба и Розенблатта

## Правило Хэбба, 1949<sup>5</sup>

В задаче бинарной ( $Y = \{-1, +1\}$ ) классификации линейный классификатор:

$$a(x, w) = \text{sign}(\langle w, x \rangle)$$

Функция потерь:  $L(a(x_i, w), y_i) = [a(x_i, w) \neq y_i]$ .

Градиентный шаг: если  $a(x_i, w^{(t)}) \neq y_i \Leftrightarrow a(x_i, w^{(t)})y_i < 0$ , то  $w^{(t+1)} = w^{(t)} + \eta x_i y_i$

## Правило перцептрана Розенблатта, 1957<sup>6</sup>

Пусть  $X = \{0, 1\}^n$ ,  $Y = \{0, +1\}$ . Тогда:

если  $a(x_i, w^{(t)}) \neq y_i$ :  $w^{(t+1)} = w^{(t)} + \eta x_i$ , если  $y_i = 1$ , и  $w^{(t+1)} = w^{(t)} - \eta x_i$ , если  $y_i = 0$

<sup>5</sup>Hebb, D. O. (1949). "The organization of behavior: a neuropsychological theory."

<sup>6</sup>Rosenblatt, F. (1957). "The perceptron, a perceiving and recognizing automaton"

# История: правила Хэбба и Розенблатта

## Правило Хэбба, 1949<sup>5</sup>

В задаче бинарной ( $Y = \{-1, +1\}$ ) классификации линейный классификатор:

$$a(x, w) = \text{sign}(\langle w, x \rangle)$$

Функция потерь:  $L(a(x_i, w), y_i) = [a(x_i, w) \neq y_i]$ .

Градиентный шаг: если  $a(x_i, w^{(t)}) \neq y_i \Leftrightarrow a(x_i, w^{(t)})y_i < 0$ , то  $w^{(t+1)} = w^{(t)} + \eta x_i y_i$

## Правило перцептрана Розенблатта, 1957<sup>6</sup>

Пусть  $X = \{0, 1\}^n$ ,  $Y = \{0, +1\}$ . Тогда:

если  $a(x_i, w^{(t)}) \neq y_i$ :  $w^{(t+1)} = w^{(t)} + \eta x_i$ , если  $y_i = 1$ , и  $w^{(t+1)} = w^{(t)} - \eta x_i$ , если  $y_i = 0$

Правило Хэбба и правило Розенблатта - суть одно и то же, и совпадают с правилом ADALINE с заменой  $a(x, w)$  на соответствующее значение из  $Y$ :

$$w^{(t+1)} = w^{(t)} - \eta(a(x_i, w^{(t)}) - y_i)x_i$$

<sup>5</sup>Hebb, D. O. (1949). "The organization of behavior: a neuropsychological theory."

<sup>6</sup>Rosenblatt, F. (1957). "The perceptron, a perceiving and recognizing automaton"

# Теорема Новикова<sup>7</sup>

Задача бинарной классификации  $X = \mathbb{R}^{n+1}$ ,  $Y = \{-1, +1\}$ .

## Теорема Новикова, 1962

Пусть выборка  $X^m$  линейно разделима, т.е.  $\exists \tilde{w}, \|\tilde{w}\| = 1, \exists \delta > 0 : \langle \tilde{w}, x_i \rangle y_i > \delta$  для всех  $i = 1, \dots, m$ . Пусть начальный вектор весов  $w^0 = 0$ . Также в процедуре обучения каждый объект обучающей выборки появляется повторно через некоторый конечный интервал времени.

Тогда алгоритм SGD с правилом Хэбба находит вектор весов  $w$ :

- разделяющий выборку без ошибок,
- при любом шаге градиентного спуска  $\eta$ ,
- независимо от порядка предъявления  $x_i$ ,
- за конечное число исправлений вектора  $w$ :  $t_{max} \leq \frac{1}{\delta^2} \max_i \|x_i\|^2$

<sup>7</sup>Novikoff, A. (1962). "On Convergence Proofs on Perceptrons"

## Теорема Новикова: доказательство

С одной стороны,

$$\langle \tilde{w}, w^t \rangle = \langle \tilde{w}, w^{t-1} \rangle + \eta \langle \tilde{w}, x_i \rangle y_i > \langle \tilde{w}, w^{t-1} \rangle + \eta \delta > \dots > \langle \tilde{w}, w^0 \rangle + t \eta \delta = t \eta \delta.$$

С другой стороны, поскольку выборка конечна,  $\exists D > 0 : \|x_i\| < D$  для всех  $i$ . В силу этого  $\|w^t\|^2 = \|w^{t-1}\|^2 + \eta^2 \|x_i\|^2 + 2\eta \langle w^{t-1}, x_i \rangle y_i$ . Так как для применения правила Хэбба должно быть  $\langle w^{t-1}, x_i \rangle y_i < 0$ , то

$$\|w^t\|^2 < \|w^{t-1}\|^2 + \eta^2 D^2 < \dots < \|w^0\|^2 + t\eta^2 D^2 = t\eta^2 D^2.$$

По неравенству Коши-Буняковского  $\langle \tilde{w}, w^t \rangle \leq \|\tilde{w}\| \cdot \|w^t\|$ .

Объединяя эти неравенства, получаем  $\eta \delta t \leq \langle \tilde{w}, w^t \rangle \leq \eta D \sqrt{t} \cdot \|\tilde{w}\|$ , или  $\sqrt{t} \leq \frac{D}{\delta}$ .

Т.о. при  $t > \frac{D^2}{\delta^2}$  не найдётся ни одного  $x_i$ , т.ч.  $\langle w^t, x_i \rangle y_i < 0$ , т.е. вся выборка будет правильно классифицирована. Ч.т.д.

# Линейность байесовского классификатора

Из предыдущего материала известно, что оптимальный байесовский бинарный классификатор определяется как:

$$a(x) = \text{sign}(\lambda_+ p(y = +1|x) - \lambda_- p(y = -1|x)) = \text{sign}\left(\frac{p(y=+1|x)}{p(y=-1|x)} - \frac{\lambda_-}{\lambda_+}\right)$$

## Теорема о линейности байесовского классификатора

Если распределения  $p(y|x)$  экспонентны, параметры  $d()$ ,  $\delta$  не зависят от  $y$ , и среди признаков  $x_1, \dots, x_n$  есть константа, то байесовский классификатор линеен:

$$a(x) = \text{sign}(\langle w, x \rangle - w_0), w_0 = \ln \frac{\lambda_-}{\lambda_+};$$

при этом апостериорные вероятности классов  $p(y|x) = \sigma(\langle w, x \rangle y)$ , где  $\sigma(z) = \frac{1}{1+e^{-z}}$  – логистическая функция (сигмоид).

# Линейность байесовского классификатора

Из предыдущего материала известно, что оптимальный байесовский бинарный классификатор определяется как:

$$a(x) = \text{sign}(\lambda_+ p(y = +1|x) - \lambda_- p(y = -1|x)) = \text{sign}\left(\frac{p(y=+1|x)}{p(y=-1|x)} - \frac{\lambda_-}{\lambda_+}\right)$$

## Теорема о линейности байесовского классификатора

Если распределения  $p(y|x)$  экспонентны, параметры  $d()$ ,  $\delta$  не зависят от  $y$ , и среди признаков  $x_1, \dots, x_n$  есть константа, то байесовский классификатор линеен:

$$a(x) = \text{sign}(\langle w, x \rangle - w_0), w_0 = \ln \frac{\lambda_-}{\lambda_+};$$

при этом апостериорные вероятности классов  $p(y|x) = \sigma(\langle w, x \rangle y)$ , где  $\sigma(z) = \frac{1}{1+e^{-z}}$  – логистическая функция (сигмоид).

## Определение логистической регрессии

Классификационная бинарная модель, в которой вероятность принадлежности к положительному классу задаётся сигмоидом от линейной функции по входу.

# Логарифмическая функция потерь

Напоминание: максимизация логарифма правдоподобия:

- $L(w, X^m) = \log \prod_{i=1}^m p(x_i, y_i) \rightarrow \max_w$

# Логарифмическая функция потерь

Напоминание: максимизация логарифма правдоподобия:

- $L(w, X^m) = \log \prod_{i=1}^m p(x_i, y_i) \rightarrow \max_w$

Подставим в формулу выражение для логистической регрессии  
 $p(x, y) = p(y|x) \cdot p(x) = \sigma(\langle w, x \rangle) \cdot \text{const}(w)$ :

- $L(w, X^m) = \sum_{i=1}^m \log \sigma(\langle w, x_i \rangle y_i) + \text{const}(w) \rightarrow \max_w$

# Логарифмическая функция потерь

Напоминание: максимизация логарифма правдоподобия:

- $L(w, X^m) = \log \prod_{i=1}^m p(x_i, y_i) \rightarrow \max_w$

Подставим в формулу выражение для логистической регрессии  
 $p(x, y) = p(y|x) \cdot p(x) = \sigma(\langle w, x \rangle) \cdot \text{const}(w)$ :

- $L(w, X^m) = \sum_{i=1}^m \log \sigma(\langle w, x_i \rangle y_i) + \text{const}(w) \rightarrow \max_w$

Максимизация  $L$  эквивалентна минимизации аппроксимированного Э.Р.  $R$ :

$$R(w, X^m) = \sum_{i=1}^m \log(1 + \exp(-\langle w, x_i \rangle y_i)) \rightarrow \min_w$$

# Многоклассовая логистическая регрессия

Рассмотрим случай произвольного количества классов  $|Y| > 2$ . Тогда линейный классификатор (напоминание):

$$a(x) = \arg \max_{c \in Y} \langle w^c, x \rangle \quad x, w^c \in \mathbb{R}^n$$

# Многоклассовая логистическая регрессия

Рассмотрим случай произвольного количества классов  $|Y| > 2$ . Тогда линейный классификатор (напоминание):

$$a(x) = \arg \max_{c \in Y} \langle w^c, x \rangle \quad x, w^c \in \mathbb{R}^n$$

Вероятность принадлежности объекта  $x$  к классу  $c$  определяется т.н. функцией SoftMax:

$$\text{SoftMax}(\langle w^c, x \rangle) = P(y = c|x, w) = \frac{\exp(\langle w^c, x \rangle)}{\sum_{z \in Y} \exp(\langle w^z, x \rangle)}$$

Т.о. функция  $\text{SoftMax} : \mathbb{R}^{|Y|} \rightarrow \mathbb{R}^{|Y|}$  преобразует любой вещественнозначный вектор в вектор дискретного распределения.

## Причины переобучения

- Маленькая обучающая выборка; большое число признаков;
- Признаки линейно зависимы;
- Неинформационные (шумовые) признаки.

# О переобучении

## Причины переобучения

- Маленькая обучающая выборка; большое число признаков;
- Признаки линейно зависимы;
- Неинформационные (шумовые) признаки.

## Проявление переобучения

- Резкое увеличение нормы  $w$ ;
- Большая разница в ошибке классификации на тестовой и обучающей выборках;



# О переобучении

## Причины переобучения

- Маленькая обучающая выборка; большое число признаков;
- Признаки линейно зависимы;
- Неинформационные (шумовые) признаки.

## Проявление переобучения

- Резкое увеличение нормы  $w$ ;
- Большая разница в ошибке классификации на тестовой и обучающей выборках;

## Борьба с переобучением

- Ранняя остановка обучения;
- Уменьшение норм весов (**регуляризация**);

# Вероятностный смысл простой регуляризации

Рассмотрим принцип максимума совместного правдоподобия данных и модели, или MAP (Maximum A Posteriori Probability).

Дано:

- Параметрическая модель плотности распределения  $p(x, y|w)$
- Априорная информация о плотности распределения параметров модели  $p(w)$   
Например, параметрическое семейство априорных распределений  $p(w; h)$ , где  $h$  — неизвестная и неслучайная величина (гиперпараметр).

# Вероятностный смысл простой регуляризации

Рассмотрим принцип максимума совместного правдоподобия данных и модели, или МАР (Maximum A Posteriori Probability).

Дано:

- Параметрическая модель плотности распределения  $p(x, y|w)$
- Априорная информация о плотности распределения параметров модели  $p(w)$   
Например, параметрическое семейство априорных распределений  $p(w; h)$ , где  $h$  — неизвестная и неслучайная величина (гиперпараметр).

Тогда:

- Плотность  $p(X^m, w; h) = p(X^m|w)p(w; h)$
- Максимизируем логарифм совместного распределения

$$L(w, X^m) = \ln p(X^m, w; h) = \sum_{i=1}^m \ln p(x_i, y_i|w) + \ln p(w; h) \rightarrow \max_{w, h}$$

## $L_2$ -регуляризация

Рассмотрим введение квадратичного штрафа за увеличение нормы весов в функционал Э.Р.:

$$R_\tau(w, X^m) = R(w, X^m) + \frac{\tau}{2} \|w\|^2 \rightarrow \min_w$$

Тогда градиент Э.Р.:  $\nabla R_\tau(w, X^m) = \nabla R(w, X^m) + \tau w,$

## $L_2$ -регуляризация

Рассмотрим введение квадратичного штрафа за увеличение нормы весов в функционал Э.Р.:

$$R_\tau(w, X^m) = R(w, X^m) + \frac{\tau}{2} \|w\|^2 \rightarrow \min_w$$

Тогда градиент Э.Р.:  $\nabla R_\tau(w, X^m) = \nabla R(w, X^m) + \tau w$ ,  
А градиентный шаг:  $w^{(t+1)} = (1 - \tau\eta)w^{(t)} - \eta \nabla R(w^{(t)}, X^m)$ .

## $L_2$ -регуляризация

Рассмотрим введение квадратичного штрафа за увеличение нормы весов в функционал Э.Р.:

$$R_\tau(w, X^m) = R(w, X^m) + \frac{\tau}{2} \|w\|^2 \rightarrow \min_w$$

Тогда градиент Э.Р.:  $\nabla R_\tau(w, X^m) = \nabla R(w, X^m) + \tau w,$

А градиентный шаг:  $w^{(t+1)} = (1 - \tau\eta)w^{(t)} - \eta \nabla R(w^{(t)}, X^m).$

### Подбор параметра регуляризации $\tau$

- Больше значение  $\tau$  - больше штрафа за переобучение (но сходимость медленнее!)
- Методом скользящего контроля (cross-validation);



- ① Эмпирическим риском измеряем качество классификатора

- ① Эмпирическим риском измеряем качество классификатора
- ② На практике используется аппроксимационный эмпирический риск

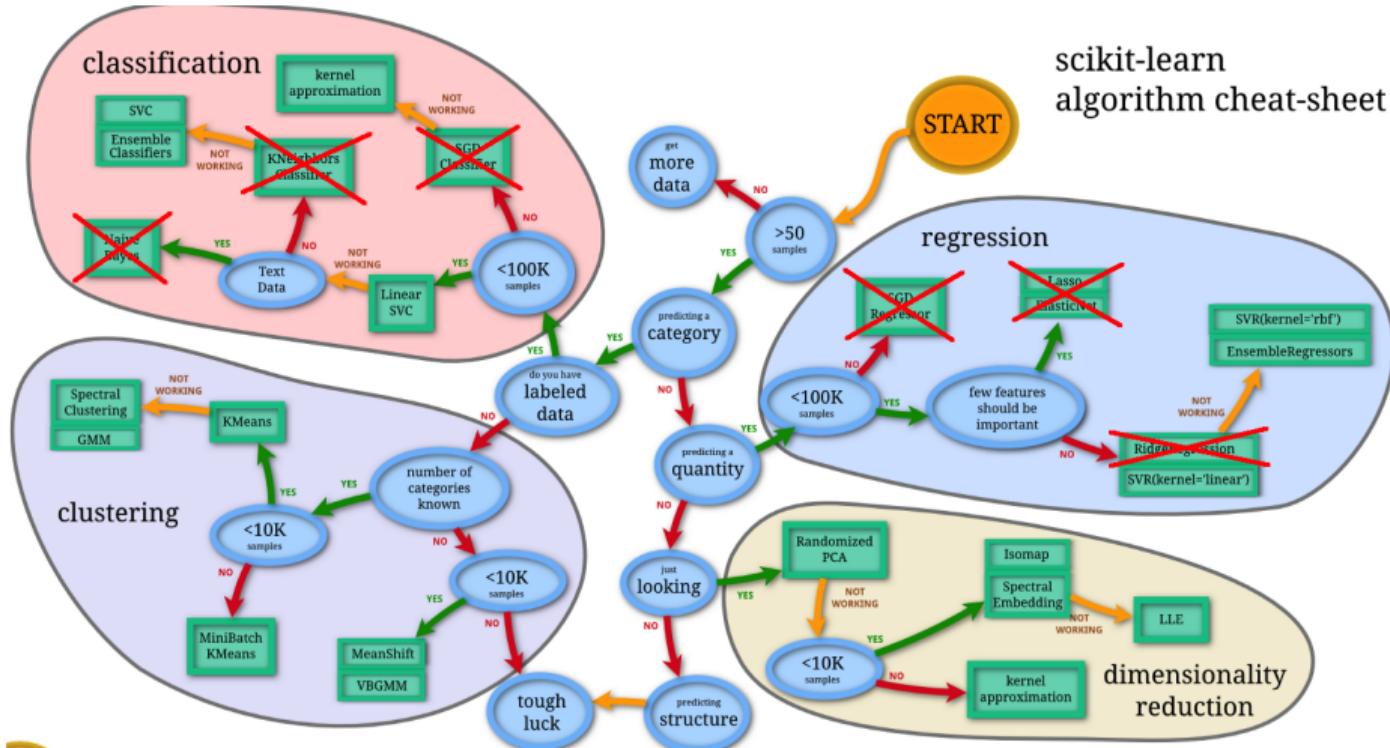
- ① Эмпирическим риском измеряем качество классификатора
- ② На практике используется аппроксимационный эмпирический риск
- ③ Линейный классификатор – предельный простой случай (тем не менее, работающий на практике!)

- ① Эмпирическим риском измеряем качество классификатора
- ② На практике используется аппроксимационный эмпирический риск
- ③ Линейный классификатор – предельный простой случай (тем не менее, работающий на практике!)
- ④ Градиентный спуск – алгоритм оптимизации первого порядка

- ① Эмпирическим риском измеряем качество классификатора
- ② На практике используется аппроксимационный эмпирический риск
- ③ Линейный классификатор – предельный простой случай (тем не менее, работающий на практике!)
- ④ Градиентный спуск – алгоритм оптимизации первого порядка
- ⑤ SGD – практическая версия GD

- ① Эмпирическим риском измеряем качество классификатора
- ② На практике используется аппроксимационный эмпирический риск
- ③ Линейный классификатор – предельный простой случай (тем не менее, работающий на практике!)
- ④ Градиентный спуск – алгоритм оптимизации первого порядка
- ⑤ SGD – практическая версия GD
- ⑥ Регуляризация изменяет коэффициенты для SGD

# Дорожная карта Scikit-Learn<sup>8</sup>



<sup>8</sup>[https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/](https://scikit-learn.org/stable/tutorial/machine_learning_map/)

# Семинар

## Метрики качества классификаторов

# Классификация ответов бинарного классификатора

- Обучающая выборка  $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$
- Задача классификации на 2 класса:  $X \rightarrow Y, Y = \{+1, -1\}$
- Алгоритм классификации  $a(x_i) = y_i$
- Класс с меткой “+1” называется “**positive**”
- Класс с меткой “-1” называется “**negative**”

# Классификация ответов бинарного классификатора

- Обучающая выборка  $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$
- Задача классификации на 2 класса:  $X \rightarrow Y, Y = \{+1, -1\}$
- Алгоритм классификации  $a(x_i) = y_i$
- Класс с меткой “+1” называется “positive”
- Класс с меткой “-1” называется “negative”

Таблица: Классификация ответов

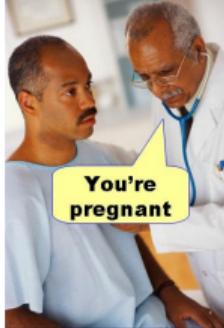
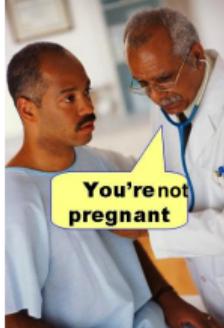
	Выход алгоритма	Правильный ответ
TP (True Positive)	$a(x_i) = +1$	$y_i = +1$
TN (True Negative)	$a(x_i) = -1$	$y_i = -1$
FP (False Positive)	$a(x_i) = +1$	$y_i = -1$
FN (False Negative)	$a(x_i) = -1$	$y_i = +1$

# Матрица ошибок

Более наглядно эти соотношения можно изобразить с помощью **матрицы ошибок** (confusion matrix)

		Правильный ответ	
		$y = +1$	$y = -1$
Выход алгоритма	$a(x) = +1$	True Positive  False Positive (Ошибка 1 рода)	False Positive (Ошибка 1 рода)
	$a(x) = -1$	False Negative (Ошибка 2 рода)	True Negative

# Матрица ошибок

	$y = +1$	$y = -1$
$a(x) = +1$		
$a(x) = -1$		

# Простейшая метрика качества

- Простейшая метрика качества - это доля правильных ответов на тесте (контрольной выборке)
- По-английски - **Accuracy**

## Формула Accuracy

$$Accuracy = \frac{1}{m} \sum_{i=1}^m [a(x_i) = y_i] = \frac{TP+TN}{TP+FP+TN+FN}$$



# Простейшая метрика качества

- Простейшая метрика качества - это доля правильных ответов на тесте (контрольной выборке)
- По-английски - **Accuracy**

## Формула Accuracy

$$Accuracy = \frac{1}{m} \sum_{i=1}^m [a(x_i) = y_i] = \frac{TP+TN}{TP+FP+TN+FN}$$

## Недостаток

- Не учитывается дисбаланс классов
- Не учитывается цена ошибки на объектах разных классов



# Метрики по положительному отклику алгоритма

Рассмотрим метрики, которые основаны на подсчёте доли положительных ответов алгоритма.

## Доля ложных положительных классификаций

Также известно как False Positive Rate, или FPR.

$$FPR(a, X^m) = \frac{\sum_{i=1}^m [y_i = -1] [a(x_i) = +1]}{\sum_{i=1}^m [y_i = -1]}$$



# Метрики по положительному отклику алгоритма

Рассмотрим метрики, которые основаны на подсчёте доли положительных ответов алгоритма.

## Доля ложных положительных классификаций

Также известно как False Positive Rate, или FPR.

$$FPR(a, X^m) = \frac{\sum_{i=1}^m [y_i = -1] [a(x_i) = +1]}{\sum_{i=1}^m [y_i = -1]}$$

## Доля верных положительных классификаций

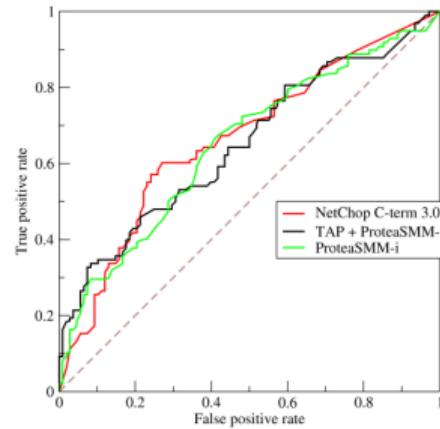
Также известно как True Positive Rate, или TPR.

$$TPR(a, X^m) = \frac{\sum_{i=1}^m [y_i = +1] [a(x_i) = +1]}{\sum_{i=1}^m [y_i = +1]}$$

**Замечание.** Обратите внимание на разные знаменатели!

# Кривая ошибок

Наиболее известна как рабочая характеристика приёмника, или Receiver Operating Characteristic (**ROC-кривая**), в который мы смотрим на компромисс между уровнем ложной тревоги и долей верного отклика.



По оси X откладывается FPR, по оси Y - TPR<sup>9</sup>.

**Замечание.** На данной кривой никак не учитываются пропуски.

<sup>9</sup><https://wikipedia.org>

## AUROC

Чем больше для каждого значения ошибки FPR значение правильного предсказания TPR, тем лучше работает классификатор.

Т.о., площадь под кривой (Area Under Curve, AUC / AUROC) необходимо максимизировать.

<sup>10</sup><https://dyakonov.org>

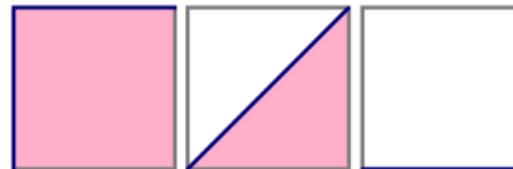
# Площадь под ROC-кривой и виды ROC-кривых

## AUROC

Чем больше для каждого значения ошибки FPR значение правильного предсказания TPR, тем лучше работает классификатор.

Т.о., площадь под кривой (Area Under Curve, AUC / AUROC) необходимо максимизировать.

Наглядны ROC-кривые для наилучшего ( $AUC=1$ ), случайного ( $AUC=0.5$ ) и наихудшего ( $AUC=0$ ) алгоритма<sup>10</sup>.



<sup>10</sup><https://dyakonov.org>

# Задача

Предположим, что алгоритм бинарной классификации  $a(x_i)$  принимает решение о присвоении класса на основе некоторого скалярного значения  $g_\theta(x_i) \in \mathbb{R}$ , где  $\theta$  - набор параметров модели, а  $g_\theta(x_i)$  - дискриминантная функция.

## Задача

- Хотим построить ROC-кривую, т.е. найти точки  $\{(FPR_i, TPR_i)\}_{i=1}^m$
- Подсчитать площадь под кривой - AUROC



# Задача

Предположим, что алгоритм бинарной классификации  $a(x_i)$  принимает решение о присвоении класса на основе некоторого скалярного значения  $g_\theta(x_i) \in \mathbb{R}$ , где  $\theta$  - набор параметров модели, а  $g_\theta(x_i)$  - дискриминантная функция.

## Задача

- Хотим построить ROC-кривую, т.е. найти точки  $\{(FPR_i, TPR_i)\}_{i=1}^m$
- Подсчитать площадь под кривой - AUROC

Подсчитаем количество правильных ответов разного типа:

- $m_+ = \sum_{i=1}^m [y(x_i) = +1]$
- $m_- = \sum_{i=1}^m [y(x_i) = -1]$  (понятно, что  $m = m_+ + m_-$ )

Упорядочим обучающую выборку  $X^m$  по убыванию значений  $g_\theta(x_i)$ .

Тогда формула для  $AUROC = \frac{1}{m_-} \sum_{i=1}^m [y_i = -1] TPR_i$ .

## Алгоритм

Первую точку ставим в начало координат:  $(FPR_0, TPR_0) = (0, 0)$ ,  $AUROC = 0$ .

# Решение задачи

## Алгоритм

Первую точку ставим в начало координат:  $(FPR_0, TPR_0) = (0, 0)$ ,  $AUROC = 0$ .

Цикл по упорядоченной выборке  $i = 1 \dots m$

Если  $y_i = -1$ :

- $(FPR_i, TPR_i) = (FPR_{i-1} + \frac{1}{m_-}, TPR_{i-1})$  (двигаемся по оси X)
- $AUROC = AUROC + \frac{1}{m_-} TPR_i$

# Решение задачи

## Алгоритм

Первую точку ставим в начало координат:  $(FPR_0, TPR_0) = (0, 0)$ ,  $AUROC = 0$ .

Цикл по упорядоченной выборке  $i = 1 \dots m$

Если  $y_i = -1$ :

- $(FPR_i, TPR_i) = (FPR_{i-1} + \frac{1}{m_-}, TPR_{i-1})$  (двигаемся по оси X)
- $AUROC = AUROC + \frac{1}{m_-} TPR_i$

Если  $y_i = +1$ :

- $(FPR_i, TPR_i) = (FPR_{i-1}, TPR_{i-1} + \frac{1}{m_+})$  (двигаемся по оси Y)

## В задачах информационного поиска

- Точность, или  $Precision = \frac{TP}{TP+FP}$  (доля релевантных объектов среди найденных)
- Полнота, или  $Recall = \frac{TP}{TP+FN}$  (доля найденных объектов среди релевантных)

# Другие важные метрики 1

## В задачах информационного поиска

- Точность, или  $Precision = \frac{TP}{TP+FP}$  (доля релевантных объектов среди найденных)
- Полнота, или  $Recall = \frac{TP}{TP+FN}$  (доля найденных объектов среди релевантных)

## Как применяются

- **Точность:** позволяет следить, чтобы было мало ложных тревог; но при этом ничего не говорит о пропусках (высока цена ложной тревоги, а цена пропуска - низкая).
- **Полнота:** позволяет следить, чтобы было мало пропусков; но при этом ничего не говорит о ложных тревогах (высока цена пропуска, а цена ложной тревоги - низкая).

**Замечание.** Зачастую задача состоит в оптимизации одной метрики при фиксации другой.

## В задачах медицинской диагностики

- Чувствительность, или  $Sensitivity = \frac{TP}{TP+FN}$  (доля верных положительных диагнозов)
- Специфичность, или  $Specificity = \frac{TN}{TN+FP}$  (доля верных отрицательных диагнозов)

# Другие важные метрики 2

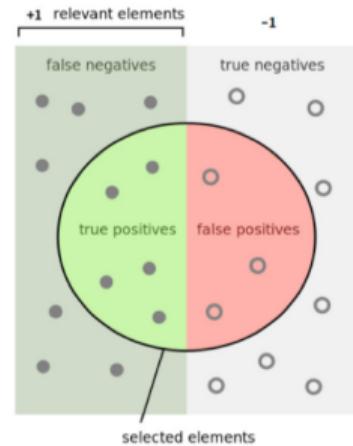
## В задачах медицинской диагностики

- Чувствительность, или  $Sensitivity = \frac{TP}{TP+FN}$  (доля верных положительных диагнозов)
- Специфичность, или  $Specificity = \frac{TN}{TN+FP}$  (доля верных отрицательных диагнозов)

## Как применяются

- **Чувствительность:** максимизируем количество верных положительных диагнозов, но не учитываем ложные диагнозы (стоимость лечения низкая, а цена пропуска - высокая).
- **Специфичность:** максимизируем количество верных отрицательных диагнозов, но не учитываем пропуски диагноза (стоимость лечения высокая, а цена пропуска - низкая).

# Иллюстрация метрик

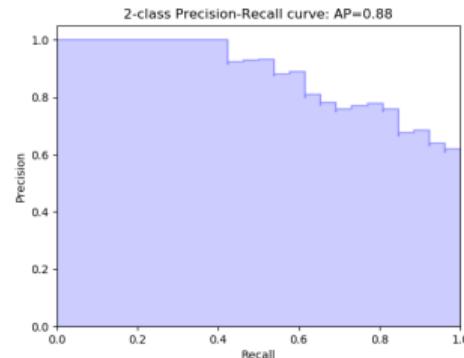


$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$
$$\text{Recall} = \text{Sensitivity} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$\text{Accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{true negatives} + \text{false positives} + \text{false negatives}}$$
$$\text{Specificity} = \frac{\text{true negatives}}{\text{true negatives} + \text{false positives}}$$

# Агрегированные метрики над Precision-Recall

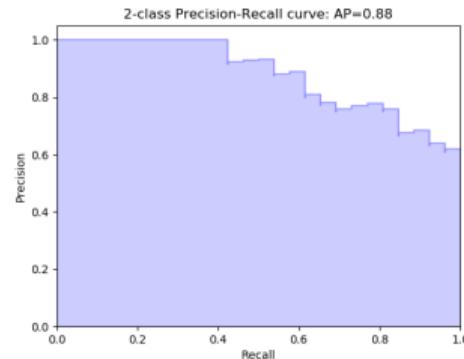
Можно построить кривую Точность-Полнота (PR-кривая) по аналогии с ROC-кривой:



**Замечание.** Обратите внимание, что в данном случае кривая не обязательно монотонна!

# Агрегированные метрики над Precision-Recall

Можно построить кривую Точность-Полнота (PR-кривая) по аналогии с ROC-кривой:



**Замечание.** Обратите внимание, что в данном случае кривая не обязательно монотонна!

## AUPRC

- Аналогично AUROC, можно вычислить площадь под PR-кривой - AUPRC
- Другое название - Average Precision (с некоторым допущениями на способ интегрирования): чем больше, тем лучше

# Многоклассовая классификация

Для каждого класса  $c \in Y$  обозначим через  $TP_c$ ,  $FP_c$  и  $FN_c$  верные положительные, ложные положительные и ложные отрицательные ответы. Тогда:

## Точность и полнота с макроусреднением

- $Precision = \frac{\sum_c TP_c}{\sum_c (TP_c + FP_c)}$
- $Recall = \frac{\sum_c TP_c}{\sum_c (TP_c + FN_c)}$
- Не чувствительно к ошибкам на маленьких классах

# Многоклассовая классификация

Для каждого класса  $c \in Y$  обозначим через  $TP_c$ ,  $FP_c$  и  $FN_c$  верные положительные, ложные положительные и ложные отрицательные ответы. Тогда:

## Точность и полнота с макроусреднением

- $Precision = \frac{\sum_c TP_c}{\sum_c (TP_c + FP_c)}$
- $Recall = \frac{\sum_c TP_c}{\sum_c (TP_c + FN_c)}$
- Не чувствительно к ошибкам на маленьких классах

## Точность и полнота с микроусреднением

- $Precision = \frac{1}{|Y|} \sum_c \frac{TP_c}{TP_c + FP_c}$
- $Recall = \frac{1}{|Y|} \sum_c \frac{TP_c}{TP_c + FN_c}$
- Чувствительно к ошибкам на маленьких классах



# Резюме по оценкам качества классификации

- Точность и полнота подходят для задач информационного поиска, когда доля объектов релевантного класса мала

# Резюме по оценкам качества классификации

- Точность и полнота подходят для задач информационного поиска, когда доля объектов релевантного класса мала
- Чувствительность и специфичность подходят для задач с несбалансированными классами (как, например, в медицине)

# Резюме по оценкам качества классификации

- Точность и полнота подходят для задач информационного поиска, когда доля объектов релевантного класса мала
- Чувствительность и специфичность подходят для задач с несбалансированными классами (как, например, в медицине)
- AUROC подходит для оценки качества при нефиксированном соотношении цены ошибок

# Резюме по оценкам качества классификации

- Точность и полнота подходят для задач информационного поиска, когда доля объектов релевантного класса мала
- Чувствительность и специфичность подходят для задач с несбалансированными классами (как, например, в медицине)
- AUROC подходит для оценки качества при нефиксированном соотношении цены ошибок
- Ещё одна агрегированная оценка качества - F-мера:  
$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$
  - Это гармоническое среднее, которое стремится к нулю когда хотя бы одно из значений стремится к нулю

# Источники

На основе материалов сайта <http://www.machinelearning.ru>.