

# Содержание

<b>1</b>	<b>Лекция 1.</b>	<b>2</b>
1.1	Введение. . . . .	2
1.2	По сути . . . . .	2
1.3	Метрики качества . . . . .	3
<b>2</b>	<b>Машинное обучение</b>	<b>3</b>
<b>3</b>	<b>Лекция 2. Модели, ансамбли моделей</b>	<b>4</b>
3.1	Зоопарк моделей . . . . .	4
3.1.1	Наивный байесовский классификатор . . . . .	4
3.1.2	Линейные модели . . . . .	4
3.1.3	Линейная регрессия . . . . .	4
3.1.4	Градиентный спуск . . . . .	4
3.1.5	SVM. Метод опорных векторов . . . . .	5

# 1 Лекция 1.

## 1.1 Введение.

Где применяется машинное обучение:

- Цены на нефть (есть данные нужно предсказать дальнейшее развитие);
- Кредитный скоринг (есть человек, есть кредит — надо понять давать ему кредит или нет. Предсказываем вероятность того, насколько он хороший заёмщик);
- Предсказание температуры (диапазон значений);
- Таргетированная реклама (на основе предпочтений подбираем наиболее подходящие товары);
- Информационный поиск;
- Персонализация (наиболее интересные новости VK — градиентный бустинг деревьев);
- Чат-боты (NN);
- Призма;
- Генерация изображений (GAN); <https://deepmind.com/blog>
- Боты в играх (reinforcement learning);

*Замечание:* Анонсирован градиентный бустинг на 2-ом занятии.

*Замечание:* Мода на сеточки :D надо доказать людям, что заниматься не нейронками тоже прикольно.

## 1.2 По сути

Пусть у нас есть тренировочные данные кредитного скоринга:

- Пол;
- Возраст;
- Возвращают кредит или нет;

Хотим предсказать вернёт он нам кредит или нет.

*Определение.* Дано  $X$  — обучающая выборка (множество объектов), где  $x \in X = \{x\}$  — прецедент, модель ML, после чего **модель** обучается — получается алгоритм машинного обучения.  $Y$  — множество допустимых ответов (target).

Данные необходимо проверять — модель хочет понять насколько хорошо модель «понимает» данные, которые она никогда не видела. Проверяем обобщающую способность данных.

*Определение.*  $y^* : X \rightarrow Y$  — **целевая функция**. Имеем значения только на конечном наборе данных  $x_i \in X : y^*(x_i) = y_i$ . Сопоставляем входным данным соответствующие ответы.

*Определение.*  $a$  — **решающая функция (алгоритм)** — обобщение функции  $y^*$  на всё множество объектов.

$$X = X_{\text{train}} \cup X_{\text{test}}$$

Делим выборку на обучающую + тестовую.

*Замечание* **Kaggle**. Титаник.

*Определение.* **Признак** (feature) объекта  $x$  — это результаты измерения характеристики объекта. Для  $x_j = \{x_j^1, \dots, x_j^n\}$  — признаковое описание объекта (вектор).

Можно генерировать новые признаки различными алгоритмами (feature engineering).

Очень круто выяснить какие features наиболее важны для модели.

**Формализуем:** по выборке  $X_{\text{train}}$  построить решающую функцию которая хорошо приближает  $y^*$  как на  $X_{\text{train}}$ , так и на всём множестве  $X$ .

Чтобы решать задачу необходимо ввести функционал качества. Для этого необходимо ввести метрику и функцию ошибки. Решаем задачу оптимизации: ищем набор параметров в пространстве параметров такой, чтобы Максимизировать функционал качества, минимизировать функцию ошибок (*Loss*).

### 1.3 Метрики качества

$$MAE = \frac{1}{N} \sum_1^N |y_i - y_i^*|$$

$$MSE = \frac{1}{N} \sum_1^N (y_i - y_i^*)^2$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$PRE = \frac{TP}{TP + FP}$$

$$REC = \frac{TP}{TP + FN}$$

$$F = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

- TP — 1 и алгоритм говорит, что 1;
- TN — 0 и алгоритм говорит, что 0;
- FP — 0 алгоритм говорит, что 1;
- FN — 1 алгоритм говорит, что 0;

Для оценки качества работы алгоритма на каждом из классов по отдельности введем метрики precision (точность) и recall (полнота).

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

Почему нужны эти метрики: Если данных одного класса сильно больше, чем другого, то алгоритму проще обучиться всегда выводить первый класс (вероятность ошибки ниже при accuracy). А precision и recall смещают данные.

## 2 Машинное обучение

- Классическое обучение (с учителем — без);
- Ансамблевые методы (много моделей);

*Определение.* Классификация. есть множество объектов и множество классов (конечное). Обучаем модель, разделяющую данные на группы.

*Определение.* Регрессия. отличается тем, что допустимым ответом является число или числовой вектор (пространство непрерывно, бесконечное множество ответов).

Без учителя:

- Кластеризация;
- Поиск правил;
- Уменьшение размерности;

*Определение.* **Переобучение** — модель находит зависимости там, где их нет (например у всех плохих заёмщиков были красные ботинки); Причины:

- Слишком много степеней свободы (слишком большая модель);
- Мало данных (general зависимости не находим, а находим локальные зависимости);

*Определение.* Валидационная выборка — знаем на ней ответы (на тестовой не знаем);

*Определение.* Кросс-валидация — ошибка на валидационных данных не отображает реальной картины мира. Делим N раз на валидационную и тестовую, N раз тренируем алгоритм, считаем среднюю ошибку.

**Методы борьбы**

*Регуляризация* (добавляем функцию от весов в loss, сдерживаем их рост)

## 3 Лекция 2. Модели, ансамбли моделей

### 3.1 Зоопарк моделей

Занимаемся моделями обучения с учителем.

Есть 2 варианта постановки задачи:

- Классификация;
- Регрессия;

#### 3.1.1 Наивный байесовский классификатор

Самый простой способ, как построить себе модель.

**Замечание.** Только классификация.

Для класса определяем вероятность. Есть условная вероятность, как часто встречается фича с фиксированной величиной для каждого класса.

По формуле Байеса найдём вероятность ответа при значении фичей.

$$p(A|B) = \frac{p(A) \cdot p(B|A)}{p(B)}$$

**Определение.** *Априорная вероятность.* – это вероятность, присвоенная событию при отсутствии знания, поддерживающего его наступление  $p(A)$ ;

**Определение.** *Апостериорная вероятность.* – это условная вероятность события  $p(A|B)$ .

- Spam detection;
- Сегментация новостных статей по теме;
- Определение эмоционального окраса текста;

#### 3.1.2 Линейные модели

**Определение.** *Линейная модель.* выражается линейной функцией.

$$Y_i = \beta_0 + \beta_1 f(X_{i1}) + \dots + \beta_p f(X_{ip}) + \varepsilon_i$$

Где  $\varepsilon_i$  – шум (гауссово распределение),  $i = 1, \dots, N$ .

**Замечание.** Функции  $f_1, \dots, f_p$  не обязательно должны быть линейными.

#### 3.1.3 Линейная регрессия

$$y = X \cdot w^T + \varepsilon$$

В данном случае решение будет решением СЛАУ:

$$w^T = (X^T X)^{-1} X^T y$$

**Проблема 1.** решение будет «осмысленным» только тогда, когда в данных реально есть линейная зависимость. В реальных данных почти не встречается.

**Проблема 2.** Линейная зависимость признаков,  $\det X^T X$  должен быть отличен от 0.

Решение проблемы: **регуляризация:**

$$w^T = (X^T X + \lambda \cdot \text{Id})^{-1} X^T y$$

#### 3.1.4 Градиентный спуск

Веса модели ищутся с помощью градиентного спуска

1. Инициализация  $W$  случайными значениями;
2. Итеративно для всех сэмплов  $x \in X$ :
  - (a) Вычислить функцию потерь на  $x$ ;
  - (b) Вычислить производные функции потерь  $\text{grad}_w$  по каждому из  $w \in W$ ;

(с) Обновляем веса:  $w = w - \text{lr} \cdot \text{grad}_w$

3. Полученное  $W = \{w\}$  — искомое.

Для линейной регрессии функция потерь:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - (Wx_i + b))^2$$

**Проблемы:**

- Переобучение

*Как бороться:* при переобучении коэффициенты стремятся к бесконечности. Лечится регуляризацией.

$$\text{Lasso Regression (L1): } MSE = \frac{1}{N} \sum_{i=1}^N (y_i - (Wx_i + b))^2 + \lambda \sum_{i=1}^N |W_i|$$

$$\text{Ridge Regression (L2): } MSE = \frac{1}{N} \sum_{i=1}^N (y_i - (Wx_i + b))^2 + \lambda \sum_{i=1}^N W_i^2$$

Эта красота эквивалентна задаче оптимизации:

$$\begin{cases} MSE = \frac{1}{N} \sum_{i=1}^N (y_i - (Wx_i + b))^2 \rightarrow 0 \\ \lambda \sum_{i=1}^N |W_i| \leq C (= \text{const}) \end{cases}$$

### 3.1.5 SVM. Метод опорных векторов

**Цель:** построить линейную модель, которая делит наши выборки наилучшим образом. (ищем оптимальную разделяющую поверхность, наиболее удалённую от всех точек выборки)

**Пример:** пусть 2 класса -1 и 1. Хотим разделить:  $y \cdot Xw > c$ .

Решаем задачу оптимизацию:

$$\begin{cases} \text{loss} \rightarrow \min \\ ||w|| < b \end{cases}$$

**Цитата:** Люди придумали нейронки и про это все забыли.

### 3.1.6 Логистическая регрессия

$$y = \sigma(WX), \quad \sigma(x) = \frac{1}{1 + e^{-x}}$$

Задача ставится следующим образом:

$$y = \text{sgn}(WX - b)$$

Решающее правило: