

• Введение в машинное обучение •  
Инкрементное и онлайнное обучение

Воронцов Константин Вячеславович

`k.v.vorontsov@phystech.edu`

`http://www.MachineLearning.ru/wiki?title=User:Vokov`

Этот курс доступен на странице вики-ресурса

`http://www.MachineLearning.ru/wiki`

«Введение в машинное обучение (курс лекций, К.В.Воронцов)»

МФТИ.ФПМИ.ИС.ИАД • 7 мая 2026

- 1 Задачи инкрементного и онлайнного обучения**
  - Постановка задачи и проблематика IL/OL
  - Ленивое обучение и отбор эталонных объектов
  - Онлайнный наивный байесовский классификатор
- 2 Градиентные и точные инкрементные методы**
  - Онлайнный градиентный спуск
  - Инкрементный метод наименьших квадратов
  - Инкрементные решающие деревья
- 3 Прогнозирование, ансамбли, нейросетевые модели**
  - Адаптивные модели краткосрочного прогнозирования
  - Адаптивная селекция и композиция моделей
  - Онлайнное глубокое обучение

## Задача онлайнного обучения с учителем

**Дано:**  $(x_i, y_i)_{i=1}^{\ell}$  — поток объектов с ответами

**Найти:**  $a(x, w)$  — адаптивную модель  $y(x)$  с параметром  $w$  и правило обновления параметров модели  $w_{i-1} \mapsto w_i$

**Критерий:** убывание кривой обучения (LOO learning curve)

$$Q(t) = \frac{1}{t} \sum_{i=1}^t \mathcal{L}(a(x_i, w_{i-1}), y_i) \rightarrow \min,$$

где  $\mathcal{L}(a, y)$  — потеря модели  $a$  в сравнении с  $y$

инициализировать параметры модели  $w_0$ ;

для всех  $i = 1, \dots, \ell$

получить объект  $x_i$  и предсказать  $a_i := a(x_i, w_{i-1})$ ;

получить ответ  $y_i$  и оценить потерю  $\mathcal{L}_i := \mathcal{L}(a_i, y_i)$ ;

обновить модель  $w_i := \text{Update}(w_{i-1}, x_i, y_i)$ ;

## Проблематика инкрементного и онлайнного обучения

- Как эффективно обновить модель по одному прецеденту?
- Как усложнять модель по мере роста объёма данных?
- Как обеспечить то же качество, что в оффлайне?
- Как избежать хранения всей выборки данных?
- Как при этом не забывать ранее выученный материал?
- Либо, наоборот, как забывать самые старые объекты?

**Что может добавляться в задачах машинного обучения:**

- объекты — **основной, но не единственный случай**
- признаки
- размерность модели
- классы/кластеры
- подвыборки/подзадачи
- области пространства данных, разладки (concept drift)

## Online Learning $\neq$ Incremental Learning. В чём отличия?

- **Online** обрабатывает объекты в потоке, по одному  
**Incremental** может накапливать пакеты обновлений
- **Online** может забывать старые данные (catastrophic forgetting)  
**Incremental** часто подразумевает эквивалентность результата оффлайновому обучению по полной выборке
- **Online** исследования озабочены теоретическими гарантиями  
**Incremental** сосредоточен на реализации быстрых алгоритмов
- **Online** обязательно является Incremental  
**Incremental** НЕ обязательно является Online

*Continual (lifelong) learning* — обучение одной модели разным задачам так, чтобы новые задачи не вытесняли старые

*Anytime algorithm* — алгоритм, который обучается по потоку, но в любой момент может быть использован для предсказаний

## Напоминание. Ленивое обучение (lazy learning)

$U \subseteq X^{\ell}$  — множество хранимых эталонов (prototypes)

$K_h(x, x_j)$  — ядро ширины  $h$ , сходство пары объектов  $x$  и  $x_j$

Метрическая классификация (kNN, окно Парзена, RBF):

$$a(x) = \arg \max_{y \in Y} \sum_{j \in U} [y_j = y] K_h(x, x_j)$$

Непараметрическая регрессия (Надара-Уотсона):

$$a(x) = \frac{\sum_{j \in U} y_j K_h(x, x_j)}{\sum_{j \in U} K_h(x, x_j)}$$

Непараметрическая оценка плотности (Парзена-Розенблатта):

$$a(x) = \frac{1}{|U|V_h} \sum_{j \in U} K_h(x, x_j)$$

## Онлайн-отбор эталонов (prototype selection)

$B$  — «бюджет», максимальное число хранимых объектов  $|U|$

$\Delta_j$  — накапливаемая оценка полезности объекта  $x_j$

$C_j$  — счётчик, сколько раз объект  $x_j$  влиял на другого

$C_{\min}$  — минимальное значение счётчика влияний

$K_{\min}$  — минимальное влияние  $K_h(x_i, x_j)$  объекта  $x_j$  на  $x_i$

$\mathcal{L}_{i \setminus j}$  — потеря на объекте  $x_i$  при исключении объекта  $x_j$  из  $U$

для всех  $i = 1, \dots, \ell$

получить  $x_i$ ; вычислить  $a(x_i)$ ;  $\Delta_i := 0$ ;  $C_i := 0$ ;

$U := U \cup \{x_i\}$ ;

для всех  $x_j \in U$ , близких к  $x_i$ :  $K_h(x_i, x_j) > K_{\min}$

$\Delta_i := \Delta_i + (\mathcal{L}_{j \setminus i} - \mathcal{L}_j)$ ;  $C_i := C_i + 1$ ;

$\Delta_j := \Delta_j + (\mathcal{L}_{i \setminus j} - \mathcal{L}_i)$ ;  $C_j := C_j + 1$ ;

если  $|U| > B$  то  $U := U \setminus \{x_j: \frac{\Delta_j}{C_j} \rightarrow \min, C_j > C_{\min}\}$ ;

## Преимущества и недостатки ленивого онлайн

### Преимущества:

- простота реализации
- решения онлайн и оффлайна гарантированно совпадают (только при хранении всех данных,  $U = X^\ell$ )
- идею отбора эталонов можно переносить на другие онлайнные методы, для которых имеется быстрый способ
  - 1) оценивать влияние одних объектов на другие и
  - 2) оценивать декрементную потерю  $\mathcal{L}_{i \setminus j}$

### Недостатки:

- хранение выборки — это не настоящий онлайн
- обучение ширины окна  $h$  и других параметров функций сходства  $K_h$  могут существенно усложнять алгоритм

## Напоминание. Наивный байесовский классификатор

«Оптимальный» байесовский классификатор:

$$a(x) = \arg \max_{y \in Y} \lambda_y P(y) p(x|y)$$

«Наивное» предположение о независимости признаков:

$$a(x) = \arg \max_{y \in Y} \left( \ln(\lambda_y P(y)) + \sum_{j=1}^n \ln p(x^j|y) \right)$$

Предположение, что одномерные плотности экспоненциальны:

$$p(x^j|y; \theta_{yj}, \varphi_{yj}) = \exp \left( \frac{x^j \theta_{yj} - c(\theta_{yj})}{\varphi_{yj}} + h(x^j, \varphi_{yj}) \right)$$

Задача максимизации log-правдоподобия распадается на независимые подзадачи по классам  $y$  и признакам  $j$ :

$$L(\theta, \varphi) = \ln \prod_{i=1}^{\ell} p(x_i|y_i) = \sum_{j=1}^n \sum_{y \in Y} \left( \sum_{x_i \in X_y} \ln p(x_i^j|y; \theta_{yj}, \varphi_{yj}) \right) \rightarrow \max_{\theta, \varphi}$$

## Линейный наивный байесовский классификатор

Решение  $\theta_{yj}$  через среднее значение признака  $j$  в классе  $y$ :

$$\frac{\partial L}{\partial \theta_{yj}} = 0 \Rightarrow c'(\theta_{yj}) = \sum_{x_i \in X_y} \frac{x_i^j}{|X_y|} \equiv \bar{x}_{yj} \Rightarrow \theta_{yj} = [c']^{-1}(\bar{x}_{yj})$$

Решение  $\varphi_{yj}$  выражается из уравнения  $\frac{\partial L}{\partial \varphi_{yj}} = 0$ , например,

в случае гауссовского распределения  $\varphi_{yj} = \frac{1}{\ell} \sum_{i=1}^{\ell} (x_i^j - \bar{x}_{yj})^2$

**Naïve Bayes — линейный классификатор:**

$$a(x) = \arg \max_{y \in Y} \left( \underbrace{\sum_{j=1}^n x^j \frac{\theta_{yj}}{\varphi_{yj}}}_{w_{yj}} + \underbrace{\ln(\lambda_y P(y)) - \sum_{j=1}^n \frac{c(\theta_{yj})}{\varphi_{yj}}}_{b_y} + \underbrace{h(x^j, \varphi_{yj})}_{\substack{\text{если от } y \\ \text{не зависит}}} \right)$$

**Онлайнное обучение** — рекуррентные формулы для  $\bar{x}_{yj}$ ,  $\varphi_{yj}$

- скорость  $O(n\ell)$  как в оффлайне, так и в онлайне
- решения онлайн и оффлайна совпадают

## Онлайновый наивный байесовский классификатор (ONB)

инициализировать  $b_y := \ln(\lambda_y P(y))$ ;  $\bar{x}_{yj} := 0$ ;  $\ell_y := 0$ ;

для всех  $i = 1, \dots, \ell$

получить очередной объект  $x_i = (x_i^1, \dots, x_i^n)$ ;

сделать предсказание  $a_i := \arg \max_{y \in Y} \left( b_y + \sum_{j=1}^n x_i^j w_{yj} \right)$ ;

получить ответ  $y_i$  и оценить потерю  $\mathcal{L}_i := \mathcal{L}(a_i, y_i)$ ;

для  $y = y_i$  обновить средние по рекуррентной формуле:

$\bar{x}_{yj} := \frac{1}{\ell_y + 1} x_i^j + \frac{\ell_y}{\ell_y + 1} \bar{x}_{yj}$ ;  $\ell_y := \ell_y + 1$ ;

оценить параметры распределений:

$\theta_{yj} := [c']^{-1}(\bar{x}_{yj})$  и  $\varphi_{yj}$  (в зависимости от типа признака);

обновить коэффициенты линейной модели:

$w_{yj} := \frac{\theta_{yj}}{\varphi_{yj}}$ ;  $b_y := \ln(\lambda_y P(y)) - \sum_{j=1}^n \frac{c(\theta_{yj})}{\varphi_{yj}}$ ;

## Преимущества и недостатки ONB

### Преимущества:

- скорость  $O(nl)$  как в оффлайне, так и в онлайн
- решения онлайн и оффлайна совпадают
- не чувствителен к числу классов и дисбалансу классов
- практически не бывает переобучения
- подходит для разнотипных данных и данных с пропусками
- в задачах классификации текстов качество сопоставимо с SVM (при введении отбора признаков по TF-IDF)
- часто используется в качестве «бейслайна для битья»

### Недостатки:

- в большинстве задач «наивное» предположение о независимости признаков совсем не работает

---

*J.Rennie et al.* Tackling the poor assumptions of Naive Bayes text classifiers. 2003

## Алгоритм Перцептрон для линейного классификатора

Пусть  $x_i \in \mathbb{R}^n$ ,  $y_i \in \{-1, +1\}$ ; модель  $a(x, w) = \text{sign}(x^T w)$ .

Старейший алгоритм онлайн обучения (правило Хебба):

инициализировать параметры модели  $w_0 := 0$ ;

для всех  $i = 1, \dots, \ell$

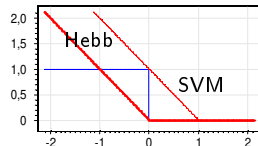
получить  $x_i$ ; предсказать  $a_i := \text{sign}(x_i^T w_{i-1})$ ; получить  $y_i$ ;

если  $a_i \neq y_i$  то

обновить модель  $w_i := w_{i-1} + \eta y_i x_i$ ;

Это эквивалентно градиентному шагу  
с функцией потерь  $\mathcal{L}_i(w) = (-y_i x_i^T w)_+$   
и величиной шага  $\eta$

Вариант с нормализацией:  $w_i := w_{i-1} + \eta y_i \frac{x_i}{\|x_i\|}$



*Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. 1958.*

## Алгоритм Passive-Aggressive для линейного классификатора

$\mathcal{L}_i(w) = (1 - y_i x_i^T w)_+$  — функция потерь как в SVM

Идея:  $w_i$  = проекция  $w_{i-1}$  на множество  $\{w : \mathcal{L}_i(w) = 0\}$

**passive** — если  $\mathcal{L}_i(w_{i-1}) = 0$ , то не менять веса,  $w_i := w_{i-1}$

**aggressive** — сдвинуться как можно дальше к  $w : \mathcal{L}_i(w) = 0$

Задача поиска точки  $w_i$ , с параметром  $C$  и степенью  $p \in \{1, 2\}$ :

$$\|w - w_{i-1}\|^2 + C \mathcal{L}_i^p(w) \rightarrow \min_w$$

Аналитическое решение даёт значение градиентного шага  $\eta_i$ :

$$w_i := w_{i-1} + \eta_i y_i x_i$$

$$\underbrace{\eta_i = \frac{\mathcal{L}_i}{\|x_i\|^2}}_{\text{при } C=0} \quad \text{или} \quad \underbrace{\eta_i = \min \left\{ C, \frac{\mathcal{L}_i}{\|x_i\|^2} \right\}}_{\text{при } p=1} \quad \text{или} \quad \underbrace{\eta_i = \frac{\mathcal{L}_i}{\|x_i\|^2 + \frac{1}{2C}}}_{\text{при } p=2}$$

# Онлайн-градиентный спуск (Online Gradient Descent, OGD)

Минимизация аддитивного критерия

$$\sum_{i=1}^{\ell} \mathcal{L}(a(x_i, w), y_i) \rightarrow \max_w$$

Отличие от метода SGD (Stochastic Gradient Descent) в том, что объекты следуют в заданном порядке, а не в случайном:

инициализировать параметры модели  $w_0$ ;

для всех  $i = 1, \dots, \ell$

получить объект  $x_i$ ; предсказать  $a_i := a(x_i, w_{i-1})$ ;

получить ответ  $y_i$ ; оценить потерю  $\mathcal{L}_i := \mathcal{L}(a_i, y_i)$ ;

обновить модель  $w_i := w_{i-1} - \eta_i \nabla_w \mathcal{L}(a(x_i, w_{i-1}), y_i)$ ;

*M. Zinkevich*. Online convex programming and generalized infinitesimal gradient ascent. 2003.

## Напоминание. Многомерная линейная регрессия

**Дано:**  $\{x_1, \dots, x_\ell\}$  — обучающая выборка (training set)

$x_i = (f_1(x_i), \dots, f_n(x_i))$  — вектор признаков объекта  $x_i$

$y_i = y(x_i) \in \mathbb{R}$  — «ответ учителя» на обучающим объекте  $x_i$

**Найти:** параметры  $w \in \mathbb{R}^n$  модели линейной регрессии:

$$a(x, w) = \sum_{j=1}^n w_j f_j(x) = \langle w, x \rangle = w^T x, \quad w, x \in \mathbb{R}^n.$$

Матричные обозначения:

$$F_{\ell \times n} = \begin{pmatrix} f_1(x_1) & \dots & f_n(x_1) \\ \dots & \dots & \dots \\ f_1(x_\ell) & \dots & f_n(x_\ell) \end{pmatrix}, \quad y_{\ell \times 1} = \begin{pmatrix} y_1 \\ \dots \\ y_\ell \end{pmatrix}, \quad w_{n \times 1} = \begin{pmatrix} w_1 \\ \dots \\ w_n \end{pmatrix}.$$

**Критерий** — метод наименьших квадратов (МНК):

$$Q(w, X^\ell) = \sum_{i=1}^{\ell} (a(x_i, w) - y_i)^2 = \|Fw - y\|^2 \rightarrow \min_w$$

# Метод наименьших квадратов. Нормальная система уравнений

Необходимое условие минимума в матричном виде:

$$\frac{\partial Q(w)}{\partial w} = 2F^T(Fw - y) = 0,$$

$F^T F w = F^T y$  — нормальная система задачи МНК

$w^* = (F^T F)^{-1} F^T y = F^+ y$  — решение системы и задачи МНК

$F^+$  — псевдообратная матрица размера  $n \times \ell$ , полного ранга  $n$

## Геометрическая интерпретация МНК:

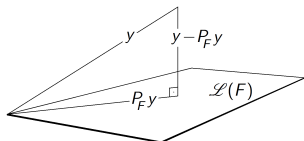
$P_F = FF^+$  — проекционная матрица

$P_F y$  — проекция  $y \in \mathbb{R}^\ell$  на  $\mathcal{L}(F)$ ,

линейную оболочку столбцов матрицы

$F = (f_1, \dots, f_n)$ ,  $f_j \in \mathbb{R}^\ell$ :

$$\mathcal{L}(F) = \left\{ \sum_{j=1}^n w_j f_j \mid w \in \mathbb{R}^n \right\}$$



## Рекурсивный метод наименьших квадратов (RLS)

Пусть  $x_i \in \mathbb{R}^n$ ,  $y_i \in \mathbb{R}$ ; модель регрессии  $a(x, w) = x^T w$ .

МНК для линейной регрессии с  $L_2$ -регуляризацией:

$$\sum_{i=1}^{\ell} (x_i^T w - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^n w_j^2 = \|Fw - y\|^2 + \frac{\lambda}{2} \|w\|^2 \rightarrow \min_w$$

Решение задачи МНК (гребневая регрессия):

$$w^* = (F^T F + \lambda I_n)^{-1} F^T y$$

Новый объект  $x_i$  добавляется нижней строкой к  $F_{i-1}$ :

$$F_i^T F_i = (F_{i-1}^T \ x_i) \begin{pmatrix} F_{i-1} \\ x_i^T \end{pmatrix} = F_{i-1}^T F_{i-1} + x_i x_i^T$$

Формула Шермана–Моррисона для матрицы  $A = F_{i-1}^T F_{i-1} + \lambda I_n$ :

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}$$

## Рекурсивный метод наименьших квадратов (RLS)

Рекурсивный МНК (Recursive Least Squares):

инициализировать  $w_0 := 0$ ,  $A_0 := (I_n + \lambda I_n)^{-1}$ ;

для всех  $i = 1, \dots, \ell$

получить объект  $x_i$ ; сделать предсказание  $a_i := x_i^T w_{i-1}$ ;

получить ответ  $y_i$ ; оценить потерю  $\mathcal{L}_i := (a_i - y_i)^2$ ;

$$A_i := A_{i-1} - \frac{A_{i-1} x_i x_i^T A_{i-1}}{1 + x_i^T A_{i-1} x_i};$$

$$w_i := w_{i-1} - A_i x_i (a_i - y_i);$$

Сложность  $O(\ell n^2)$ , решение точное, совпадает с оффлайном

Сравнение с OGD:

$$w_i := w_{i-1} - \eta_i x_i (a_i - y_i)$$

Сложность  $O(\ell n)$ , решение приближённое, отличается от оффлайна

## Напоминание. Решающее дерево (Decision Tree)

$F = \{f_1, \dots, f_n\}$  — множество признаков,  $f_j: X \rightarrow Ef_j$ ,  $|Ef_j| < \infty$

Решающее дерево — алгоритм классификации  $a(x)$ , задающийся деревом (связным ациклическим графом):

- 1)  $V = V_{\text{внутр}} \sqcup V_{\text{лист}}$ ,  $v_0 \in V$  — корень дерева;
- 2)  $v \in V_{\text{внутр}}$ : признак  $f_v \in F$  и функция  $S_v: Ef_v \rightarrow V$ ;
- 3)  $v \in V_{\text{лист}}$ : метка класса  $y_v \in Y$ .

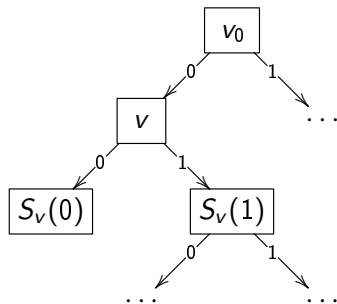
$v := v_0$ ;

**пока** ( $v \in V_{\text{внутр}}$ ):

$v := S_v(f_v(x))$ ;

**вернуть**  $a(x) := y_v$ ;

Если  $Ef_v \equiv \{0, 1\}$  для всех  $v$ , то решающее дерево бинарное



## Напоминание. Алгоритм обучения решающего дерева ID3

$v_0 := \text{TreeGrowing}(X^\ell)$  — функция рекурсивно вызывает себя

**функция**  $\text{TreeGrowing}(U \subseteq X^\ell) \mapsto$  корень дерева  $v$ ;

$f_v := \arg \max_{f \in F} \text{Gain}(f, U)$  — критерий ветвления дерева;

**если**  $\text{Gain}(f_v, U) < G_0$  **то**

└ создать новый лист  $v$ ;  $y_v := \text{Major}(U)$ ; **вернуть**  $v$ ;  
создать новую внутреннюю вершину  $v$  с функцией  $f_v$ ;

**для всех**  $k \in E_{f_v}$ :

└  $U_{vk} := \{x \in U : f_v(x) = k\}$ ;  
└  $S_v(k) := \text{TreeGrowing}(U_{vk})$ ;

**вернуть**  $v$ ;

Мажоритарное правило:  $\text{Major}(U) := \arg \max_{y \in Y} P(y|U)$ .

## Инкрементный алгоритм обучения решающего дерева ID5R

$U_v$  — множество объектов  $(x_i, y_i)$ , дошедших до вершины  $v$ .

$C_v[j, z, y] = \#\{x_i \in U_v : y_i = y, f_j(x_i) = z\}$  — счётчики числа объектов для вычисления критерия ветвления  $\text{Gain}(f_j, U_v)$ .

для всех  $i = 1, \dots, \ell$ :

получить  $x_i$ ; предсказать  $a_i$ ; получить  $y_i$ ;

для всех  $v$  на пути от  $v_0$  до листа, в который попал  $x_i$ :

$C_v[j, f_j(x_i), y_i] += 1$  для всех  $j = 1, \dots, n$ ;

$f'_v := \arg \max_f \text{Gain}(f, U_v)$  — критерий ветвления;

если  $(\text{Gain}(f'_v, U_v) > G_0)$  и  $(v \in V_{\text{лист}})$  то

└ преобразовать  $v$  во внутреннюю вершину;

если  $(f'_v \neq f_v)$  и  $(v \in V_{\text{внутр}})$  то

└  $v := \text{TreeGrowing}(U_v)$ ;  $f_v := f'_v$ ;

## Преимущества и недостатки

### Преимущества:

- хранится не выборка, а счётчики
- дерево растёт постепенно с ростом объёма данных
- решения онлайн (ID5R) и оффлайн (ID3) совпадают
- есть несколько версий более продвинутого алгоритма IDI

### Недостатки:

- большой объём хранимых данных
- из-за этого большой лес из ID5R построить трудно

---

*P.E.Utofff, N.C.Berkman, J.A.Clouse. Decision tree induction based on efficient tree restructuring. 1996*

*P.E.Utofff. An improved algorithm for incremental induction of decision trees. 1994*

## Задачи прогнозирования временных рядов

**Дано:**  $Y_t = (y_0, y_1, \dots, y_t)$  — временной ряд,  $y_i \in \mathbb{R}$

**Найти:**  $\hat{y}_{t+d}(Y_t, w)$  — модель прогноза на момент  $t + d$   
где  $w$  — вектор параметров модели,  
 $d = 1, \dots, D$ ,  $D$  — горизонт прогнозирования

**Критерий:** минимум среднеквадратичной ошибки прогнозов:

$$\sum_{d=1}^D \sum_{t=T_0}^T (\hat{y}_{t+d}(Y_t, w) - y_{t+d})^2 \rightarrow \min_w$$

**Пример:** линейная модель *авторегрессии*. В роли признаков выступают непосредственно  $n$  предыдущих наблюдений ряда:

$$\hat{y}_{t+1}(w) = \sum_{j=1}^n w_j y_{t-j+1}, \quad w \in \mathbb{R}^n$$

Обучающая выборка:  $\ell = t - n + 1$  моментов истории ряда

## Экспоненциальное скользящее среднее (ЭСС)

Простейшая регрессионная модель — константа  $\hat{y}_{t+1} = c$ , наблюдения учитываются с весами, убывающими в прошлое:

$$\sum_{i=0}^t \beta^{t-i} (y_i - c)^2 \rightarrow \min_c, \quad \beta \in (0, 1)$$

Аналитическое решение — формула Надарая-Ватсона:

$$c \equiv \hat{y}_{t+1} = \frac{\sum_{i=0}^t \beta^i y_{t-i}}{\sum_{i=0}^t \beta^i}$$

Запишем аналогично  $\hat{y}_t$ , оценим  $\sum_{i=0}^t \beta^i \approx \sum_{i=0}^{\infty} \beta^i = \frac{1}{1-\beta}$ ,

получим  $\hat{y}_{t+1} = \hat{y}_t \beta + (1 - \beta) y_t$ , заменим  $\alpha = 1 - \beta$ :

$$\hat{y}_{t+1} = \hat{y}_t + \alpha (y_t - \hat{y}_t) = \alpha y_t + (1 - \alpha) \hat{y}_t,$$

$\alpha \in (0, 1)$  называется параметром сглаживания.

## Рекуррентная формула для среднего арифметического

Экспоненциальное скользящее среднее (ЭСС):

$$\hat{y}_{t+1} = \hat{y}_t + \alpha(y_t - \hat{y}_t)$$

Среднее арифметическое:

$$\hat{y}_{t+1} = \frac{1}{t+1} \sum_{i=0}^t y_i = \hat{y}_t + \frac{1}{t+1}(y_t - \hat{y}_t)$$

При  $\alpha_t = \frac{1}{t+1}$  имеем среднее арифметическое

При  $\alpha_t = \text{const}$  имеем экспоненциальное скользящее среднее

Условие сходимости к среднему (для стационарных задач):

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

ЭСС применяется к нестационарным временным рядам

## Подбор параметра сглаживания

Экспоненциальное скользящее среднее (ЭСС):

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha) \hat{y}_t$$

Чем больше  $\alpha$ , тем больше вес последних точек,  
при  $\alpha \rightarrow 1$  прогноз стремится к тривиальному  $\hat{y}_{t+1} = y_t$ .

Чем меньше  $\alpha$ , тем сильнее сглаживание,  
при  $\alpha \rightarrow 0$  прогноз стремится к тривиальному  $\hat{y}_{t+1} = \bar{y}$ .

Оптимальное  $\alpha^*$  находим по скользящему контролю:

$$Q(\alpha) = \sum_{t=t_0}^T (\hat{y}_t(\alpha) - y_t)^2 \rightarrow \min_{\alpha}$$

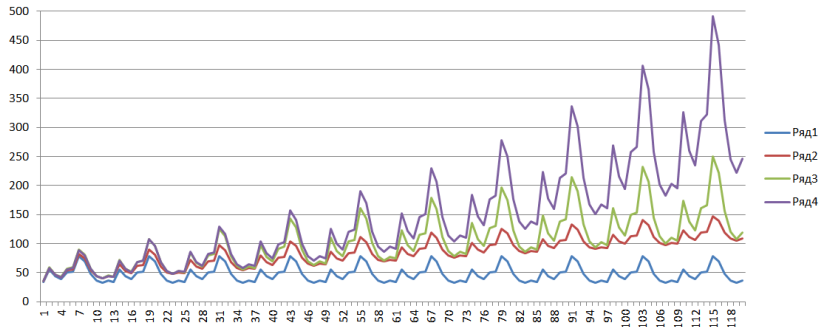
Эмпирические правила:

если  $\alpha^* \in (0, 0.3)$ , то ряд стационарен, ЭСС работает;

если  $\alpha^* \in (0.3, 1)$ , то ряд нестационарен, нужна модель тренда.

## Временные ряды с трендом и сезонностью

Пример. Сочетания тренда и сезонности (модельные данные)



Ряд 1 — сезонность без тренда

Ряд 2 — линейный тренд, аддитивная сезонность

Ряд 3 — линейный тренд, мультипликативная сезонность

Ряд 4 — экспоненциальный тренд, мультипликативная сезонность

## Аддитивный тренд и сезонность

Модель Хольта:  $\hat{y}_{t+d} = a_t + b_t d$

Рекуррентные формулы для параметров линейного тренда:

$$a_t = \alpha_1 y_t + (1 - \alpha_1)(a_{t-1} + b_{t-1})$$

$$b_t = \alpha_2 (a_t - a_{t-1}) + (1 - \alpha_2) b_{t-1}$$

Модель Тейла–Вейджа:  $\hat{y}_{t+d} = (a_t + b_t d) + \theta_{t+d-s}$ ,  $d \leq s$

Линейный тренд и аддитивная сезонность с периодом  $s$ :

$$a_t = \alpha_1 (y_t - \theta_{t-s}) + (1 - \alpha_1)(a_{t-1} + b_{t-1})$$

$$b_t = \alpha_2 (a_t - a_{t-1}) + (1 - \alpha_2) b_{t-1}$$

$$\theta_t = \alpha_3 (y_t - a_t) + (1 - \alpha_3) \theta_{t-s}$$

$\theta_0, \dots, \theta_{s-1}$  — сезонный профиль периода  $s$

$\alpha_1, \alpha_2, \alpha_3$  — параметры сглаживания

## Мультипликативный тренд и сезонность

**Модель Уинтерса:**  $\hat{y}_{t+d} = (a_t + b_t d) \cdot \theta_{t+d-s}$ ,  $d \leq s$

Линейный тренд и мультипликативная сезонность периода  $s$ :

$$a_t = \alpha_1(y_t/\theta_{t-s}) + (1 - \alpha_1)(a_{t-1} + b_{t-1});$$

$$b_t = \alpha_2(a_t - a_{t-1}) + (1 - \alpha_2)b_{t-1};$$

$$\theta_t = \alpha_3(y_t/a_t) + (1 - \alpha_3)\theta_{t-s};$$

**Модель Уинтерса:**  $\hat{y}_{t+d} = a_t \cdot r_t^d \cdot \theta_{t+d-s}$ ,  $d \leq s$

Мультипликативные тренд и сезонность периода  $s$ :

$$a_t = \alpha_1(y_t/\theta_{t-s}) + (1 - \alpha_1)a_{t-1}r_{t-1};$$

$$r_t = \alpha_2(a_t/a_{t-1}) + (1 - \alpha_2)r_{t-1};$$

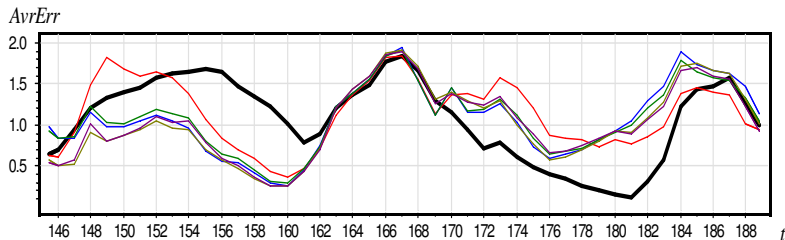
$$\theta_t = \alpha_3(y_t/a_t) + (1 - \alpha_3)\theta_{t-s};$$

$\theta_0, \dots, \theta_{s-1}$  — сезонный профиль периода  $s$

$\alpha_1, \alpha_2, \alpha_3$  — параметры сглаживания

## Идея адаптивной селекции моделей

**Пример:** Динамика ЭСС ошибок прогнозов  $|\varepsilon_t|$  для 6 моделей (по реальным данным объёмов продаж в супермаркете):



**Идея:** кажется, можно успевать включать наиболее удачные модели и отключать менее удачные...

*Разладка* — момент времени, когда временной ряд переключается с одной модели поведения на другую

## Адаптивная селективная модель

Пусть имеется  $k$  моделей прогнозирования,

$\hat{y}_{j,t+d}$  — прогноз  $j$ -й модели на момент  $t + d$

$\varepsilon_{jt} = y_t - \hat{y}_{jt}$  — ошибка прогноза  $j$ -й модели в момент  $t$

$\tilde{\varepsilon}_{jt} = \gamma|\varepsilon_{jt}| + (1 - \gamma)\tilde{\varepsilon}_{j,t-1}$  — ЭСС модуля ошибки

Лучшая модель в момент времени  $t$ :

$$j_t^* = \arg \min_{j=1, \dots, k} \tilde{\varepsilon}_{jt}$$

Адаптивная селективная модель — прогноз по лучшей модели:

$$\hat{y}_{t+d} := \hat{y}_{j_t^*, t+d}$$

Требуется подбор  $\gamma$ , рекомендация:  $\gamma = 0.01 \dots 0.1$ .

---

Ю.П.Лукашин. Адаптивные методы краткосрочного прогнозирования временных рядов. 2003.

## Адаптивная композиция моделей

Пусть имеется  $k$  моделей прогнозирования,

$\hat{y}_{j,t+d}$  — прогноз  $j$ -й модели на момент  $t + d$

$\varepsilon_{jt} = y_t - \hat{y}_{jt}$  — ошибка прогноза  $j$ -й модели в момент  $t$

$\tilde{\varepsilon}_{jt} = \gamma|\varepsilon_{jt}| + (1 - \gamma)\tilde{\varepsilon}_{j,t-1}$  — ЭСС модуля ошибки

Линейная (выпуклая) комбинация моделей:

$$\hat{y}_{t+d} = \sum_{j=1}^k w_{jt} \hat{y}_{j,t+d}, \quad \sum_{j=1}^k w_{jt} = 1, \quad \forall t.$$

Адаптивный подбор весов [Лукашин, 2003]:

$$w_{jt} = \frac{(\tilde{\varepsilon}_{jt})^{-1}}{\sum_{s=1}^k (\tilde{\varepsilon}_{st})^{-1}}.$$

Требуется подбор  $\gamma$ , рекомендация:  $\gamma = 0.01 \dots 0.1$ .

---

Ю.П.Лукашин. Адаптивные методы краткосрочного прогнозирования временных рядов. 2003.

Онлайновое обучение ансамбля: алгоритм Hedge( $\beta$ )

$b_t(x) \in [0, 1]$ ,  $t = 1, \dots, T$  — [обучаемые] базовые предикторы

$\mathcal{L}(b, y) \in [0, 1]$  — выпуклая по  $b$  функция потерь

$\beta \in (0, 1)$  — параметр основания степени ( $\beta^z$  убывает по  $z$ )

В бустинге фиксируем  $\ell$ , наращиваем  $T$ , а в Hedge — наоборот!

инициализировать веса предикторов  $w_{0t} = \frac{1}{T}$ ,  $t = 1, \dots, T$ ;

**для всех**  $i = 1, \dots, \ell$

получить очередной объект  $x_i$ ;

сделать предсказания  $b_t(x_i)$ ,  $t = 1, \dots, T$ ;

получить ответ  $y_i$  и оценить потери  $\mathcal{L}_{it} := \mathcal{L}(b_t(x_i), y_i)$ ;

**обновить веса предикторов:**  $w_{it} := \underset{t}{\text{norm}}(w_{i-1,t} \beta^{\mathcal{L}_{it}})$ ;

**дообучить предикторы**  $b_t$ ,  $t = 1, \dots, T$  на  $(x_i, y_i)$ ;

Yoav Freund, Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. 1997

## Финансовая интерпретация алгоритма Hedge( $\beta$ )

**Задача портфельного инвестора (Online Portfolio Selection):**

$b_t$  — финансовые инструменты или стратегии (equity)

$\mathcal{L}_{it}$  — потеря от инструмента  $t$  в момент времени  $i$

$w_{it}$  — доля капитала в инструменте  $t$  в момент времени  $i$

$\mathcal{L}_i = \sum_{t=1}^T w_{it} \mathcal{L}_{it}$  — потеря по всему портфелю в момент  $i$

### Теорема

Для любых  $\mathcal{L}_{it} \in [0, 1]$  потеря ансамбля не сильно хуже потери лучшего из предикторов и стремится к ней при  $\ell \rightarrow \infty$ :

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}_i \leq \min_t \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}_{it} + \sqrt{\frac{2 \ln T}{\ell}} + \frac{\ln T}{\ell}$$

**Интерпретация:** доходность портфеля стремится к доходности лучшего из инструментов при  $\ell \rightarrow \infty$  со скоростью  $O(\sqrt{\frac{\ln T}{\ell}})$

## Свойства алгоритма Hedge( $\beta$ )

- Теорема справедлива для любых  $\mathcal{L}_{it} \in [0, 1]$ , без каких-либо вероятностных предположений
- Та же оценка верна и для средних потерь ансамбля в силу выпуклости функции потерь и нормировки  $w_{it}$ :

$$\mathcal{L}\left(\sum_{t=1}^T w_{it} b_t(x_i), y_i\right) \leq \sum_{t=1}^T w_{it} \mathcal{L}(b_t(x_i), y_i) = \mathcal{L}_i$$

- Чем меньше  $\beta$ , тем быстрее обучается ансамбль
- Можно оценить  $\beta$ , минимизировав более точную оценку:

$$\sum_i \mathcal{L}_i \leq \frac{-L \ln \beta + \ln T}{1 - \beta} \rightarrow \min_{\beta} \text{, где } L = \min_t \sum_i \mathcal{L}_{it} \leq \ell$$

---

Yoav Freund, Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. 1997

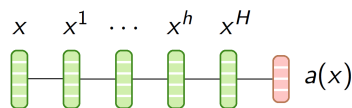
## Онлайновое глубокое обучение

Сеть для многоклассовой классификации с  $H$  слоями,  $a = (a_y)_{y \in Y}$ :

$$x^0 = x$$

$$x^h = \sigma(W^h x^{h-1})$$

$$a(x) = \text{SoftMax}_y(Vx^H)$$



**Проблема:** как обучить все слои, пока данных мало?

После каждого слоя  $h$  будем строить классификатор  $b^h(x)$ :

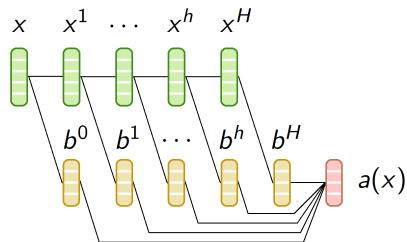
$$x^0 = x$$

$$x^h = \sigma(W^h x^{h-1})$$

$$b^h(x) = \text{SoftMax}_y(V^h x^h)$$

линейный ансамбль с весами  $w_h$ :

$$a(x) = \sum_{h=0}^H w_h b^h(x^h)$$



## Обратное распространение ошибки: Hedge BackProp

- Веса ансамбля  $w_h$  вычисляются алгоритмом Hedge( $\beta$ )
- Функция потерь — многоклассовый log-loss:

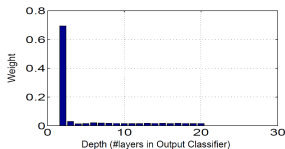
$$\mathcal{L}_i(a(x_i), y_i) = \sum_{y \in \mathcal{Y}} [y_i = y] \ln a_y(x_i) + [y_i \neq y] \ln(1 - a_y(x_i))$$

- Особенности градиентных шагов в OGD:

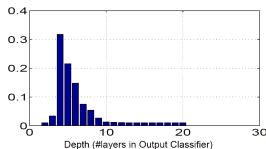
$$V^h := V^h - \eta w_h \nabla_{V^h} \mathcal{L}_i(b^h(x_i), y_i)$$

$$W^h := W^h - \eta \sum_{j=h}^H w_j \nabla_{W^j} \mathcal{L}_i(b^j(x_i), y_i)$$

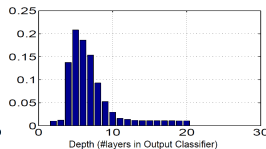
- Hedge включает слои постепенно с ростом объёма данных:



(a) First 0.5% of Data



(b) 10-15% of Data



(c) 60-80% of Data

D.Sahoo et al. Online deep learning: learning deep neural networks on the fly. 2018

- Поточковых данных становится всё больше, в перспективе всё машинное обучение может стать инкрементным
- Инкрементные модификации существуют для большинства методов машинного обучения
- Не существует универсальных рецептов, как из обычного (оффлайнового) метода сделать онлайнный
- Инкрементные методы могут быть
  - онлайнные или пакетные
  - точные или приближённые в сравнении с оффлайном
  - с изменяемой или неизменной сложностью модели
  - с хранением части выборки (эталонов) или без него
  - с теоретическими гарантиями или без них
  - с возможностью декремента или без него
- Deep Online Learning — активное развивающееся новое направление, охотно заимствующее идеи старых методов