Инкрементное и онлайновое обучение

К. В. Воронцов
vokov@forecsys.ru

Этот курс доступен на странице вики-ресурса
http://www.MachineLearning.ru/wiki
«Машинное обучение (курс лекций, К.В.Воронцов)»

МФТИ • 9 апреля 2021

Содержание

Задача онлайнового обучения

Задача обучения с учителем на потоке данных:

$(x_i, y_i)_{i=1}^{\ell}$ — последовательность прецедентов «объект, ответ»

$a(x, w)$ — параметрическая модель зависимости $y(x)$

$\mathscr{L}(a, y)$ — функция потерь

инициализировать параметры модели w_0;

для всех $i = 1, \ldots, \ell$

- получить очередной объект x_i;
- сделать предсказание $a_i := a(x_i, w_{i-1})$;
- получить ответ y_i и оценить потерю $\mathscr{L}_i := \mathscr{L}(a_i, y_i)$;
- обновить модель $w_i := \text{Update}(w_{i-1}, x_i, y_i)$;

$Q(t) = \frac{1}{t} \sum_{i=1}^{t} \mathscr{L}(a_i, y_i)$ — кривая обучения (learning curve)

Steven C. H. Hoi et al. Online learning: a comprehensive survey. 2018

Проблематика инкрементного и онлайнового 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 избежать «катастрофического забывания»?
- Как, добавляя новые объекты, ещё и удалять старые?

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 ≠ 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

Ленивое обучение (lazy learning): методы, хранящие выборку

$U \subseteq X^\ell$ — множество хранимых эталонов (prototypes)
$K_h(x, x_j)$ — ядро ширины h, близость 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 (Надарая-Уотсона):

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

Непараметрическая оценка плотности (Парзена–Розенблатта):

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

Онлайновый отбор эталонов (prototype selection)

B — «бюджет», максимальное число хранимых объектов $|U|$
Δ_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
$\mathscr{L}_{i\setminus j}$ — потеря на объекте x_i при исключении объекта x_j из U

для всех $i = 1, \ldots, \ell$
  получить x_i; вычислить $a(x_i)$; $\Delta_i := 0$; $C_i := 0$;
  для всех $x_j \in U$, близких к x_i: $K_h(x_i, x_j) > K_{\min}$
    $\Delta_i := \Delta_i + (\mathscr{L}_{j\setminus i} - \mathscr{L}_j)$; $C_i := C_i + 1$;
    $\Delta_j := \Delta_j + (\mathscr{L}_{i\setminus j} - \mathscr{L}_i)$; $C_j := C_j + 1$;
  $U := U \cup \{x_i\}$;
  если $|U| > B$ **то** $U := U \setminus \left\{x_j : \frac{\Delta_j}{C_j} \to \min,\ C_j > C_{\min}\right\}$;

Преимущества и недостатки ленивого онлайна

Преимущества:

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

Недостатки:

- хранение выборки — это не настоящий онлайн
- обучение ширины окна h, весов ядер или самих ядер K могут существенно усложнять алгоритм

Наивный байесовский классификатор в общем виде

«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:

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

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

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

Приведение распределений к экспоненциальной форме

Примеры (где $\theta = g(\mu)$ — функции связи, μ — матожидание):

$$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \exp\left(\frac{x\mu-\frac{1}{2}\mu^2}{\sigma^2} - \frac{x^2}{2\sigma^2} - \frac{1}{2}\ln(2\pi\sigma^2)\right)$$

$$\mu^x(1-\mu)^{1-x} = \exp\left(x\ln\frac{\mu}{1-\mu} + \ln(1-\mu)\right)$$

$$C_k^x\mu^x(1-\mu)^{k-x} = \exp\left(x\ln\frac{\mu}{1-\mu} + k\ln(1-\mu) + \ln C_k^x\right)$$

$$\frac{1}{x!}e^{-\mu}\mu^x = \exp\left(x\ln(\mu) - \mu - \ln x!\right)$$

распределение	значения	$c(\theta)$	$c'(\theta)$	$[c']^{-1}(z)$	ϕ	$h(x,\phi)$
нормальное	$\mathbb{R}$	$\frac{1}{2}\theta^2$	θ	z	σ^2	$-\frac{x^2}{2\phi} - \frac{\ln(2\pi\phi)}{2}$
Бернулли	$\{0,1\}$	$\ln(1+e^\theta)$	$\frac{1}{1+e^{-\theta}}$	$\ln\frac{1-z}{z}$	1	0
биномиальное	$\{0,\ldots,k\}$	$k\ln(1+e^\theta)$	$\frac{k}{1+e^{-\theta}}$	$\ln\frac{k-z}{z}$	1	$\ln C_k^x$
Пуассона	$\{0,1,\ldots\}$	e^θ	e^θ	$\ln z$	1	$-\ln x!$

Линейный наивный байесовский классификатор

Решение θ_{yj} через среднее значение признака j в классе y:

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

Решение ϕ_{yj} не всегда выражается из уравнения $\frac{\partial L}{\partial \phi_{yj}} = 0$, но для распределений Пуассона, Бернулли, биномиального $\phi_{yj} = 1$; для гауссовского распределения (и если ϕ_{yj} не зависит от y):

$$\phi_{yj} = \frac{1}{\ell} \sum_{i=1}^{\ell} \left(x_i^j - \bar{x}_{y_i j}\right)^2$$

В итоге Naïve Bayes оказывается линейным классификатором:

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

Онлайновый наивный байесовский классификатор (ONB)

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

для всех $i = 1, \ldots, \ell$

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

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 $\mathscr{L}_i := \mathscr{L}(a_i, y_i)$;

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 ϕ_{yj} (в зависимости от типа признака);

обновить коэффициенты модели:
$w_{yj} := \frac{\theta_{yj}}{\phi_{yj}}$; $b_y := b_y - \sum_{j=1}^{n} \frac{c(\theta_{yj})}{\phi_{yj}}$;

Преимущества и недостатки ONB

Преимущества:

- скорость $O(n\ell)$ как в оффлайне, так и в онлайне
- решения онлайна 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 Perceptron для линейного классификатора

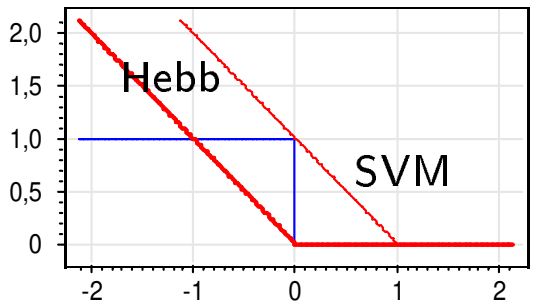
Пусть $x_i \in \mathbb{R}^n$, $y_i \in \{-1, +1\}$; модель $a(x, w) = \text{sign}(x^\mathsf{T} w)$.

Старейший алгоритм онлайнового обучения (правило Хебба):

инициализировать параметры модели $w_0 := 0$;
для всех $i = 1, \ldots, \ell$
  получить x_i; предсказать $a_i := \text{sign}(x^\mathsf{T} w_{i-1})$; 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 $\mathscr{L}_i(w) = (-y_i x_i^\mathsf{T} w)_+$
и величиной шага η

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

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

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

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

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

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

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

Аналитическое решение приводит к выбору градиентного шага:

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

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

K.Crammer et al. Online passive-aggressive algorithms. JMRL, 2006.

Онлайновый градиентный спуск (Online Gradient Descent, OGD)

Минимизация аддитивного критерия

$$\sum_{i=1}^{\ell} \mathscr{L}\big(a(x_i, w), y_i\big) \to \max_w$$

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 $i = 1, \ldots, \ell$
  получить объект x_i; предсказать $a_i := a(x_i, w_{i-1})$;
  получить ответ y_i; оценить потерю $\mathscr{L}_i := \mathscr{L}(a_i, y_i)$;
  обновить модель $w_i := w_{i-1} - \eta_i \nabla_w \mathscr{L}\big(a(x_i, w_{i-1}), y_i\big)$;

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

Рекурсивный метод наименьших квадратов (RLS)

Пусть $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$; модель регрессии $a(x, w) = x^\mathsf{T} w$.
Метод наименьших квадратов (МНК) для линейной регрессии:

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

Решение задачи МНК (гребневая регрессия):

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

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

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

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

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

Рекурсивный метод наименьших квадратов (RLS)

Рекурсивный МНК (Recursive Least Squares):

инициализировать $w_0 := 0$, $A_0 := (I_n + \lambda I_n)^{-1}$;
для всех $i = 1, \ldots, \ell$
- получить объект x_i; сделать предсказание $a_i := x_i^{\mathsf{T}} w_{i-1}$;
- получить ответ y_i; оценить потерю $\mathscr{L}_i := (a_i - y_i)^2$;
- $A_i := A_{i-1} - \dfrac{A_{i-1} x_i x_i^{\mathsf{T}} A_{i-1}}{1 + x_i^{\mathsf{T}} A_{i-1} x_i}$;
- $w_i := w_{i-1} - A_i x_i (a_i - y_i)$;

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

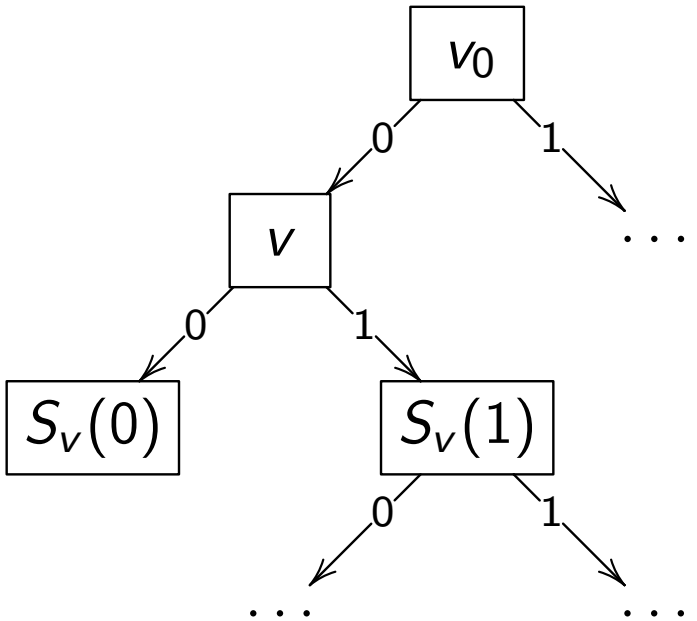
Напоминание. Решающее дерево (Decision Tree)

$\mathscr{F} = \{f_1, \ldots, f_n\}$ — множество признаков, $f_j : X \to Ef_j$, $|Ef_j| < \infty$

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 $f_v \in \mathscr{F}$ и функция $S_v : Ef_v \to V$;
3) $v \in V_{\text{лист}}$: метка класса $y_v \in Y$.

$v := v_0$;
пока $(v \in V_{\text{внутр}})$:
  $v := S_v(f_v(x))$;
выход $a(x) := y_v$;



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;
$f_v := \arg\max\limits_{f \in \mathscr{F}} \text{Gain}\,(f, U)$ — критерий ветвления дерева;
если $\text{Gain}\,(f_v, U) < G_0$ **то**
  создать новый лист v; $y_v := \text{Major}\,(U)$; **выход** v;
создать новую внутреннюю вершину v с функцией f_v;
для всех $k \in Ef_v$:
  $U_{vk} := \{x \in U \colon f_v(x) = k\}$;
  $S_v(k) := \text{TreeGrowing}\,(U_{vk})$;
выход v;

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

John Ross Quinlan. Induction of Decision Trees // Machine Learning, 1986.

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 $i = 1, \ldots, \ell$:
- 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 $j = 1, \ldots, n$;
  - $f'_v := \arg\max_f \text{Gain}(f, U_v)$ — 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**
    - реструктурировать всё поддерево S_v; $f_v := f'_v$;

Paul E. Utgoff. Incremental Induction of Decision Trees. 1989

Преимущества и недостатки

Преимущества:

- хранится не выборка, а счётчики
- дерево растёт постепенно с ростом объёма данных
- решения онлайна (ID5R) и оффлайна (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.Utogff, N.C.Berkman, J.A.Clouse. Decision tree induction based on efficient tree restructuring. 1996

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

Онлайновое обучение ансамбля: алгоритм Hedge(β)

$b_t(x) \in [0,1]$, $t = 1, \ldots, T$ — [обучаемые] базовые предикторы
$\mathscr{L}(b, y) \in [0,1]$ — выпуклая по b функция потерь
$\beta \in (0,1)$ — параметр основания степени (β^z убывает по z)
В бустинге фиксируем ℓ, наращиваем T, а в Hedge — наоборот!

инициализировать веса предикторов $w_{0t} = \frac{1}{T}$, $t = 1, \ldots, T$;
для всех $i = 1, \ldots, \ell$
- получить очередной объект x_i;
- сделать предсказания $b_t(x_i)$, $t = 1, \ldots, T$;
- получить ответ y_i и оценить потери $\mathscr{L}_{it} := \mathscr{L}(b_t(x_i), y_i)$;
- обновить веса предикторов: $w_{it} := \underset{t}{\text{norm}}(w_{i-1,t}\beta^{\mathscr{L}_{it}})$;
- дообучить предикторы b_t, $t = 1, \ldots, T$ 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(β)

Задача портфельного инвестора (Online Portfolio Selection):
b_t — финансовые инструменты или стратегии (equity)
$\mathscr{L}_{it}$ — потеря от инструмента t в момент времени i
w_{it} — доля капитала в инструменте t в момент времени i
$\mathscr{L}_i = \sum_{t=1}^{T} w_{it}\mathscr{L}_{it}$ — потеря по всему портфелю в момент i

Теорема

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

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

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

Свойства алгоритма Hedge(β)

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

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

- Чем меньше β, тем быстрее обучается ансамбль
- Можно оценить β, минимизировав более точную оценку:

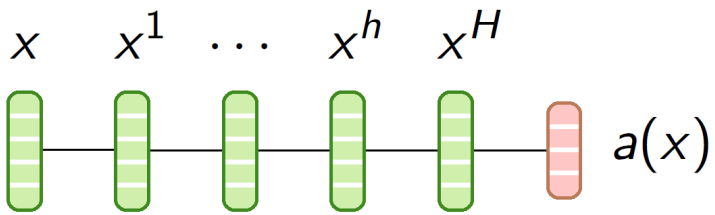
$$\sum_i \mathscr{L}_i \leqslant \frac{-L \ln \beta + \ln T}{1-\beta} \to \min_\beta, \text{ где } L = \min_t \sum_i \mathscr{L}_{it} \leqslant \ell$$

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, $a = (a_y)_{y \in Y}$:

$$x^0 = x$$
$$x^h = \sigma(W^h x^{h-1})$$
$$a(x) = \mathrm{SoftMax}_y(V x^H)$$



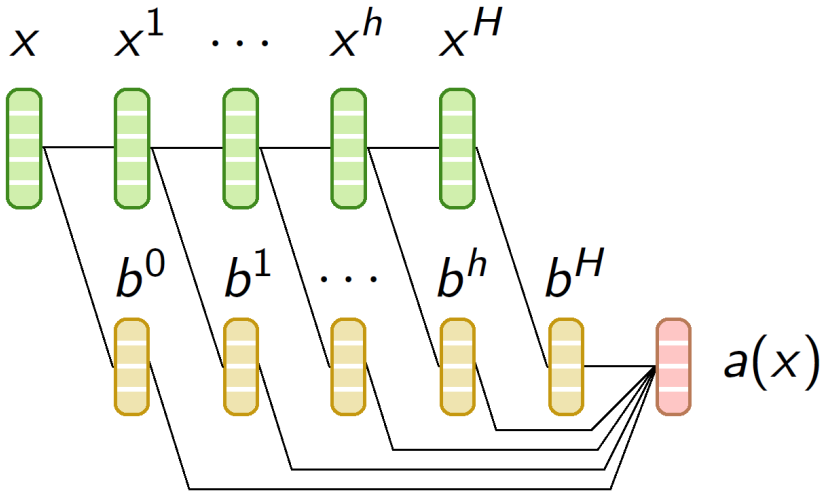
Проблема: как обучить все слои, пока данных мало?

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

$$x^0 = x$$
$$x^h = \sigma(W^h x^{h-1})$$
$$b^h(x) = \mathrm{SoftMax}_y(V^h x^h)$$

линейный ансамбль с весами w_h:

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



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

Обратное распространение ошибки: Hedge BackProp

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

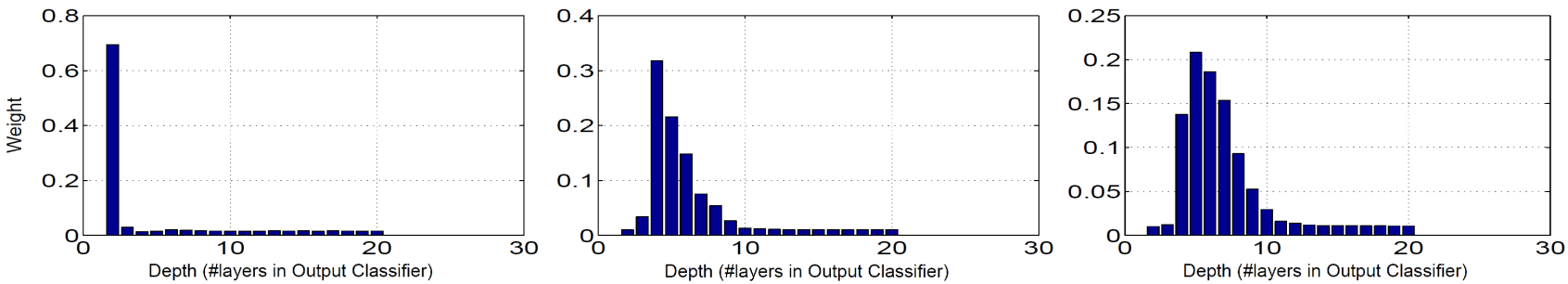
$$\mathscr{L}_i(a(x_i), y_i) = \sum_{y \in Y} [y_i = y] \ln a_y(x_i) + [y_i \neq y] \ln\big(1 - a_y(x_i)\big)$$

- Особенности градиентных шагов в OGD:

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

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

- Hedge включает слои постепенно с ростом объёма данных:

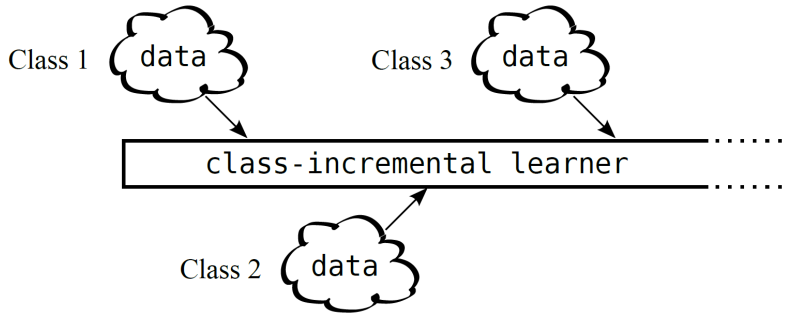


(a) First 0.5% of Data (b) 10-15% of Data (c) 60-80% of Data

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

Задача инкрементного обучения новым классам

$X^1, X^2, \ldots, X^y, \ldots$ — последовательные выборки новых классов
$X^y = \{x_1, \ldots, x_{n_y}\}$ — объекты класса $y \in Y$

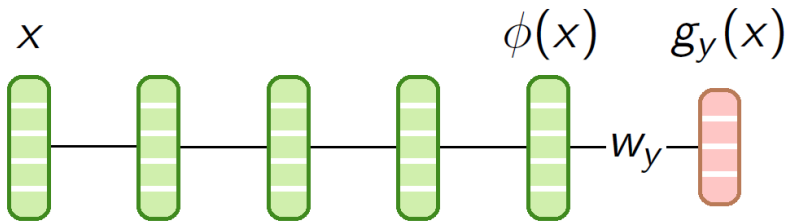


- выборка объектов не случайна: классы следуют пакетами, появляются новые классы, их число не ограничено
- алгоритм должен быть готов (any-time) классифицировать объекты тех классов, по которым он уже обучился
- память для хранения экземпляров выборки ограничена

S.-A.Rebuffi et al. iCaRL: Incremental classifier and representation learning. 2017
M.Masana et al. Class-incremental learning: survey and performance evaluation. 2020

Архитектура сети и проблема катастрофического забывания

x — исходные объекты (в частности, изображения)
$\phi(x)$ — эмбеддинг объекта, формируемый глубокой нейросетью
$g_y(x) = \sigma\left(w_y^{\mathsf{T}}\phi(x)\right)$ — вероятностная модель классификации



Проблема катастрофического забывания (catastrophic forgetting):

- постепенно происходит рассогласование эмбеддингов $\phi(x)$ и модели классификации w_y старых классов
- **идея 1:** фиксировать классы с помощью эталонов
- **идея 2:** классифицировать объекты не вероятностной моделью, а сравнивая их эмбеддинги с эмбеддингами эталонов

S.-A.Rebuffi et al. iCaRL: Incremental classifier and representation learning. 2017

Классификатор ближайшего среднего эталона

$U^y \subseteq X^y$ — подвыборка, множество эталонов класса $y \in Y$

Классификатор ближайшего среднего (nearest-mean) в пространстве векторных представлений (эмбедингов):

$$y(x) = \arg\min_{y \in Y} \left\| \phi(x) - c_y \right\|$$

где c_y — центр класса y в пространстве эмбедингов, вычисленный только по множеству эталонов класса y:

$$c_y = \frac{1}{|U^y|} \sum_{u \in U^y} \phi(u)$$

Далее рассмотрим, как формируются множества эталонов U^y

S.-A.Rebuffi et al. iCaRL: Incremental classifier and representation learning. 2017

Стратегия добавления и удаления эталонов

$U^y \subseteq X^y$ — подвыборка, множество эталонов класса $y \in Y$

B — «бюджет», максимальное число эталонов всех классов

$b = B/t$ — текущий бюджет на каждый из t классов

$\mu_y = \frac{1}{|X^y|} \sum\limits_{x \in X^y} \phi(x)$ — центр класса y в пространстве эмбедингов, вычисленный по всем обучающим объектам класса y.

Добавление эталонов: последовательно для всех $i = 1, \ldots, b$:

$$u_i = \arg\min_{u \in X^y} \left\| \mu_y - \frac{1}{i} \left(\phi(u) + \sum_{j=1}^{i-1} \phi(u_j) \right) \right\|$$

В результате эталоны ранжируются по убыванию значимости.

Удаление эталонов: когда при увеличении числа классов t уменьшается бюджет b, остаются только b первых эталонов.

S.-A.Rebuffi et al. iCaRL: Incremental classifier and representation learning. 2017

Инкрементное обучение сети

$X^s, \ldots, X^t$ — новый пакет данных с новыми классами $s, \ldots, t$.

$p_{yi} := g_y(x_i)$ — оценки вероятностей классов $P(y|x_i)$, полученные с помощью сети, ранее обученной на $X^1, \ldots, X^{s-1}$.

$D = \{U^1, \ldots, U^{s-1}, X^s, \ldots, X^t\}$ — объединённая выборка объектов, по которой будет дообучаться сеть.

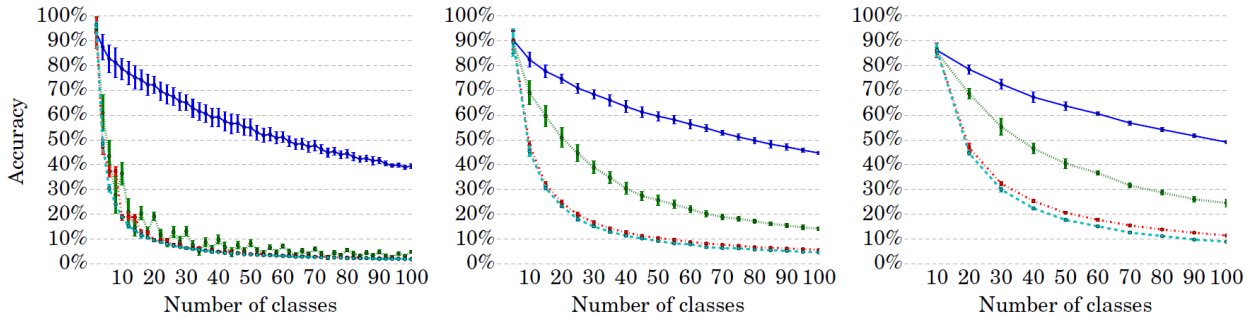
Два критерия дообучения сети на объединённой выборке D:
1) классификация всех объектов по новым классам
2) дистилляция (сохранение) старой модели на старых классах

$$\sum_{(x_i,y_i)\in D} \left(\sum_{y=s}^{t} [y = y_i] \ln g_y(x_i, W) + [y \neq y_i] \ln(1 - g_y(x_i, W)) + \right.$$
$$\left. + \sum_{y=1}^{s-1} p_{yi} \ln g_y(x_i, W) + (1 - p_{yi}) \ln(1 - g_y(x_i, W)) \right) \to \max_W$$

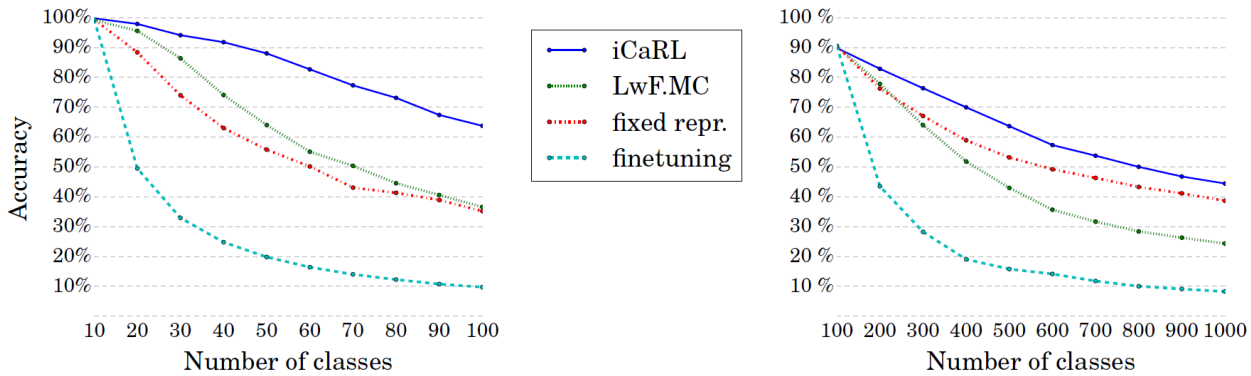
S.-A.Rebuffi et al. iCaRL: Incremental classifier and representation learning. 2017

Точность многоклассовой классификации изображений

Точность iCaRL деградирует с ростом числа классов существенно медленнее по сравнению с другими методами:



Данные iCIFAR-100: 2 / 5 / 10 классов в каждом пакете



Данные ilLSVRC-small и ilLSVRC-full

S.-A.Rebuffi et al. iCaRL: Incremental classifier and representation learning. 2017

Резюме

- Потоковых 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 онлайновый
- Инкрементные методы могут быть
  — онлайновые 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

Steven C. H. Hoi et al. Online learning: a comprehensive survey. 2018