Обзор курса. Оптимизационные задачи машинного обучения

Воронцов Константин Вячеславович
vokov@forecsys.ru
http://www.MachineLearning.ru/wiki?title=User:Vokov

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

Видеолекции: http://shad.yandex.ru/lectures

МФТИ • 22, 29 ноября 2019

Содержание

Оптимизационная задача восстановления регрессии

Обучающая выборка: $X^\ell = (x_i, y_i)_{i=1}^\ell, \quad x_i \in \mathbb{R}^n, \quad y_i \in \mathbb{R}$

1. Фиксируется модель регрессии, например, *линейная*:
$$a(x, w) = \langle x, w \rangle = \sum_{j=1}^{n} w_j f_j(x), \qquad w \in \mathbb{R}^n$$
2. Фиксируется функция потерь, например, *квадратичная*:
$$\mathscr{L}(a, y) = (a - y)^2$$
3. Метод обучения — *метод наименьших квадратов*:
$$Q(w) = \sum_{i=1}^{\ell} \big(a(x_i, w) - y_i\big)^2 \to \min_w$$
4. Проверка по тестовой выборке $X^k = (\tilde{x}_i, \tilde{y}_i)_{i=1}^k$:
$$\bar{Q}(w) = \frac{1}{k} \sum_{i=1}^{k} \big(a(\tilde{x}_i, w) - \tilde{y}_i\big)^2$$

Оптимизационная задача обучения классификация

Обучающая выборка: $X^\ell = (x_i, y_i)_{i=1}^\ell, \quad x_i \in \mathbb{R}^n, \quad y_i \in \{-1, +1\}$

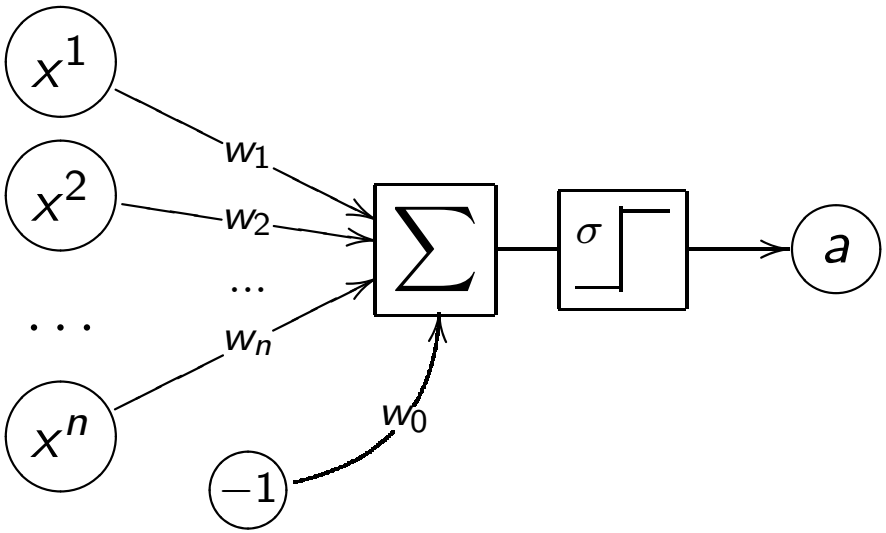
1. Фиксируется модель классификации, например, *линейная*:
$$a(x, w) = \text{sign}\langle x, w\rangle = \text{sign} \sum_{j=1}^n w_j f_j(x)$$
2. Функция потерь — пороговая или её верхняя оценка:
$$\mathscr{L}(a, y) = \big[ay < 0\big] = \big[\langle x, w\rangle y < 0\big] \leqslant \mathscr{L}\big(\langle x, w\rangle y\big)$$
3. Метод обучения — *минимизация эмпирического риска*:
$$Q(w) = \sum_{i=1}^\ell \big[\langle x_i, w\rangle y_i < 0\big] \leqslant \sum_{i=1}^\ell \mathscr{L}\big(\langle x_i, w\rangle y_i\big) \to \min_w$$
4. Проверка по тестовой выборке $X^k = (\tilde{x}_i, \tilde{y}_i)_{i=1}^k$:
$$\bar{Q}(w) = \frac{1}{k} \sum_{i=1}^k \big[\langle \tilde{x}_i, w\rangle \tilde{y}_i < 0\big]$$

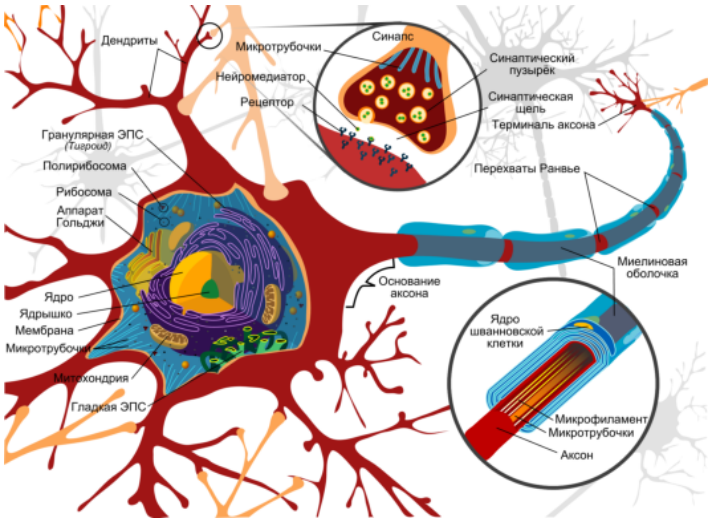
Линейный классификатор — математическая модель нейрона

Линейная модель нейрона МакКаллока-Питтса [1943]:

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

$\sigma(z)$ — функция активации (например, sign, tanh, sigmoid),
w_j — весовые коэффициенты синаптических связей,
w_0 — порог активации,
$w, x \in \mathbb{R}^{n+1}$, если ввести константный признак $f_0(x) \equiv -1$



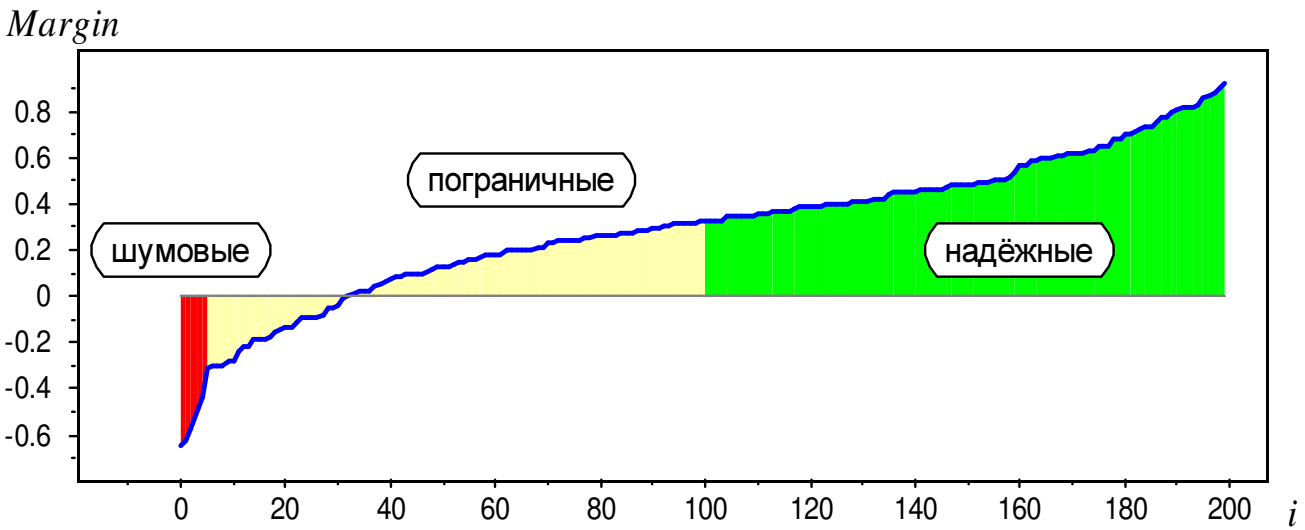


Понятие отступа для разделяющих классификаторов

Разделяющий классификатор: $a(x, w) = \operatorname{sign} g(x, w)$
$g(x, w)$ — *разделяющая (дискриминантная) функция*
$g(x, w) = 0$ — уравнение разделяющей поверхности

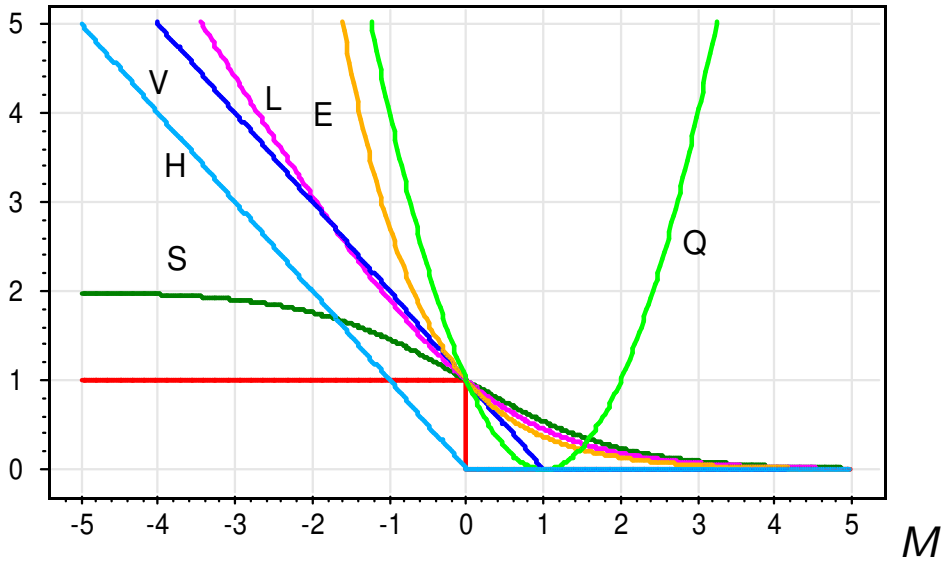
$M_i(w) = g(x_i, w)y_i$ — *отступ* (margin) объекта x_i
$M_i(w) < 0 \iff$ алгоритм $a(x, w)$ ошибается на x_i

Ранжирование объектов по возрастанию отступов $M_i(w)$:



Непрерывные верхние оценки пороговой функции потерь

Часто используемые непрерывные функции потерь $\mathscr{L}(M)$:



$[M < 0]$ — пороговая функция потерь

$V(M) = (1 - M)_+$ — кусочно-линейная (SVM)

$H(M) = (-M)_+$ — кусочно-линейная (Hebb's rule)

$L(M) = \log_2(1 + e^{-M})$ — логарифмическая (LR)

$Q(M) = (1 - M)^2$ — квадратичная (FLD)

$S(M) = 2(1 + e^{M})^{-1}$ — сигмоидная (ANN)

$E(M) = e^{-M}$ — экспоненциальная (AdaBoost)

Алгоритм SG (Stochastic Gradient)

Вход: выборка X^ℓ, темп обучения h, темп забывания λ;
Выход: вектор весов w;

инициализировать веса w_j, $j = 0, \ldots, n$;
инициализировать оценку функционала:
$\bar{Q} := \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}_i(w)$;
повторять
  выбрать объект x_i из X^ℓ случайным образом;
  вычислить потерю: $\varepsilon_i := \mathscr{L}_i(w)$;
  сделать градиентный шаг: $w := w - h\nabla\mathscr{L}_i(w)$;
  оценить функционал скользящим средним:
  $\bar{Q} := \lambda\varepsilon_i + (1-\lambda)\bar{Q}$;
пока значение $\bar{Q}$ и/или веса w не сойдутся;

Robbins, H., Monro S. A stochastic approximation method // Annals of Mathematical Statistics, 1951, 22 (3), p. 400–407.

Проблема мультиколлинеарности в линейных моделях

Мультиколлинеарность (линейная зависимость признаков):

- пусть построен классификатор $a(x, w) = \text{sign}\langle w, x\rangle$
- мультиколлинеарность: $\exists u: \ \forall x \ \langle u, x\rangle = 0$
- неединственность решения: $\forall \gamma \ a(x, w) = \text{sign}\langle w + \gamma u, x\rangle$

Проявления мультиколлинеарности:

- слишком большие веса $|w_j|$ разных знаков
- неустойчивость модели $\langle w, x\rangle$ к шуму в x
- переобучение: $Q(X^\ell) \ll Q(X^k)$

Устранение мультиколлинеарности и переобучения:

- регуляризация: $\|w\| \to \min$;
- отбор признаков: $f_1, \ldots, f_n \to f_{j_1}, \ldots, f_{j_m}, \ \ m \ll n$.
- преобразование признаков: $f_1, \ldots, f_n \to g_1, \ldots, g_m, \ \ m \ll n$;

Регуляризаторы, штрафующие сложность модели

Регуляризатор — аддитивная добавка к основному критерию:

$$Q(w) = \sum_{i=1}^{\ell} \mathscr{L}\big(\langle x_i, w\rangle, y_i\big) + \tau\, \text{штраф}(w) \;\to\; \min_w$$

где τ — коэффициент регуляризации

L_2-регуляризация (гребневая регрессия, SVM):

$$\text{штраф}(w) = \|w\|_2^2 = \sum_{j=1}^{n} w_j^2.$$

L_1-регуляризация (LASSO, ElasticNet):

$$\text{штраф}(w) = \|w\|_1 = \sum_{j=1}^{n} |w_j|.$$

L_0-регуляризация (критерии Акаике AIC, байесовский BIC):

$$\text{штраф}(w) = \|w\|_0 = \sum_{j=1}^{n} \big[w_j \neq 0\big].$$

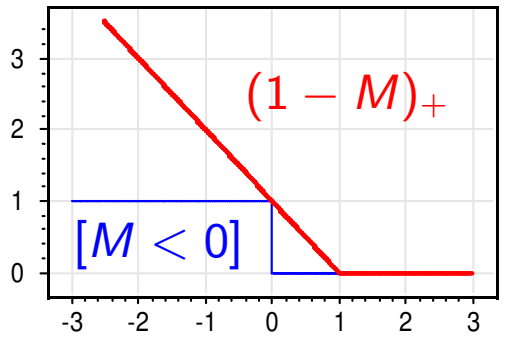
Метод опорных векторов SVM (двухклассовый)

$M_i(w, w_0) = y_i\big(\langle w, x_i\rangle - w_0\big)$ — отступ в линейной модели

Кусочно-линейная функция потерь:

$$\sum_{i=1}^{\ell}\big(1 - M_i(w, w_0)\big)_+ + \frac{1}{2C}\|w\|^2 \;\to\; \min_{w, w_0}$$

- *Функция потерь* штрафует объекты за приближение к границе классов
- *Регуляризация* максимизирует зазор между классами и штрафует мультиколлинеарность



Важнейшие свойства SVM:

- Задача выпуклого программирования, решение единственно
- Решение *разрежено* — зависит только от *опорных объектов*
- Обобщение на нелинейные модели: $\langle x, x_i\rangle \to K(x, x_i)$

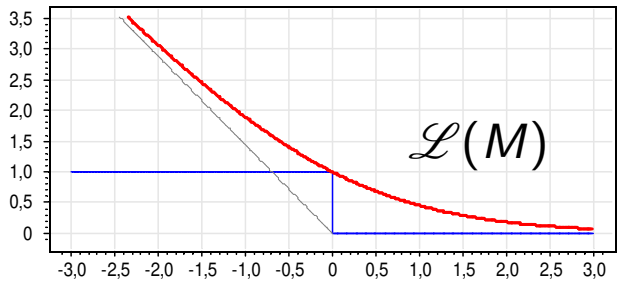
Логистическая регрессия (двухклассовая)

Логарифмическая функция потерь:

$$\sum_{i=1}^{\ell} \ln\bigl(1 + \exp(-\langle w, x_i\rangle y_i)\bigr) + \frac{\tau}{2}\|w\|^2 \;\to\; \min_w$$

Логарифмическая функция потерь:

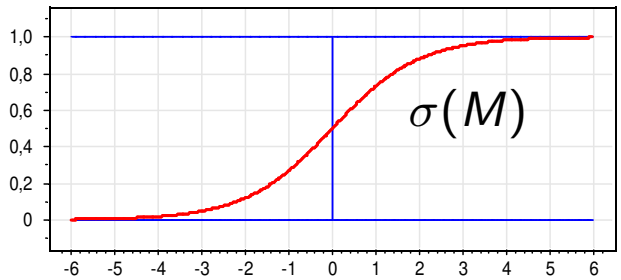
$$\mathscr{L}(M) = \ln\bigl(1 + e^{-M}\bigr)$$



Модель условной вероятности:

$$P(y|x,w) = \sigma(M) = \frac{1}{1+e^{-M}},$$

где $\sigma(M)$ — сигмоидная функция



Логистическая регрессия (многоклассовая)

Линейный классификатор при произвольном числе классов $|Y|$:

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

Вероятность того, что объект x относится к классу y:

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

где SoftMax: $\mathbb{R}^Y \to \mathbb{R}^Y$ переводит произвольный вектор в нормированный вектор дискретного распределения.

Максимизация правдоподобия (log-loss) с регуляризацией:

$$-\sum_{i=1}^{\ell} \ln P(y_i|x_i, w) + \frac{\tau}{2} \sum_{y \in Y} \|w_y\|^2 \to \min_w.$$

Связь правдоподобия и аппроксимации эмпирического риска

Пусть $X \times Y$ — в.п. с плотностью $p(x, y|w) = P(y|x, w)p(x)$.
Пусть X^ℓ — простая (i.i.d.) выборка: $(x_i, y_i)_{i=1}^\ell \sim p(x, y|w)$

- *Максимизация правдоподобия (Maximum Likelihood, ML):*

$$\sum_{i=1}^{\ell} \ln P(y_i|x_i, w) \to \max_w.$$

- *Минимизация аппроксимированного эмпирического риска:*

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

Эти два принципа эквивалентны, если приравнять

$$-\ln P(y_i|x_i, w) = \mathscr{L}\big(y_i g(x_i, w)\big).$$

$$\boxed{\text{модель } P(y|x, w)} \rightleftarrows \boxed{\text{модель } g(x, w) \text{ и } \mathscr{L}(M)}.$$

Вероятностный смысл регуляризации

$P(y|x,w)$ — вероятностная модель данных;
$p(w;\gamma)$ — априорное распределение параметров модели;
γ — вектор *гиперпараметров*;

Теперь не только появление выборки X^ℓ,
но и появление модели w также полагается стохастическим.

Совместное правдоподобие данных и модели:

$$p(X^\ell, w) = p(X^\ell|w)\, p(w;\gamma).$$

Принцип максимума апостериорной вероятности
(Maximum a Posteriori Probability, MAP):

$$\ln p(X^\ell, w) = \sum_{i=1}^{\ell} \ln P(y_i|x_i,w) + \underbrace{\ln p(w;\gamma)}_{\text{регуляризатор}} \to \max_w$$

Квантильная регрессия

Функция потерь:

$$\mathscr{L}(\varepsilon) = \begin{cases} C_+|\varepsilon|, & \varepsilon > 0 \\ C_-|\varepsilon|, & \varepsilon < 0; \end{cases}$$

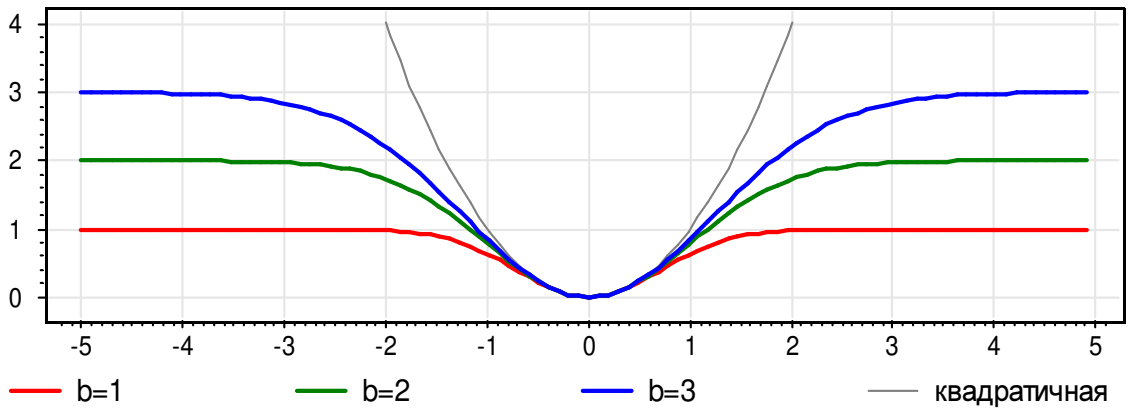
Модель регрессии: линейная $a(x_i) = \langle x_i, w \rangle$.

Сведение к задаче линейного программирования:
замена переменных $\varepsilon_i^+ = \big(a(x_i) - y_i\big)_+$, $\varepsilon_i^- = \big(y_i - a(x_i)\big)_+$;

$$\begin{cases} Q = \sum\limits_{i=1}^{\ell} C_+\varepsilon_i^+ + C_-\varepsilon_i^- \to \min\limits_w; \\ \langle x_i, w \rangle - y_i = \varepsilon_i^+ - \varepsilon_i^-; \\ \varepsilon_i^+ \geqslant 0; \quad \varepsilon_i^- \geqslant 0. \end{cases}$$

Робастная регрессия

Функция Мешалкина: $\mathscr{L}(\varepsilon) = b\big(1 - \exp\big(-\frac{1}{b}\varepsilon^2\big)\big), \quad \varepsilon = f - y$



Модель регрессии: $a(x) = f(x, w)$

Постановка оптимизационной задачи:

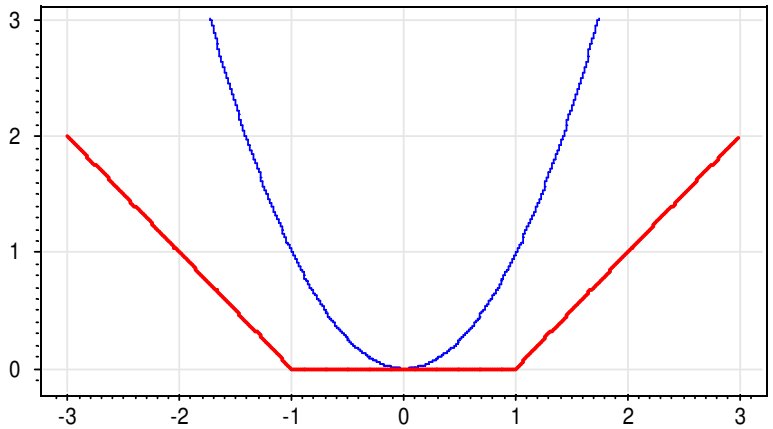
$$\sum_{i=1}^{\ell} \exp\Big(-\frac{1}{b}\big(f(x_i, w) - y_i\big)^2\Big) \to \max_w$$

Численное решение методом Ньютона-Рафсона

SVM-регрессия

Модель регрессии: $a(x) = \langle x, w \rangle - w_0, \quad w \in \mathbb{R}^n, \quad w_0 \in \mathbb{R}.$

Функция потерь: $\mathscr{L}(\varepsilon) = \big(|\varepsilon| - \delta\big)_+$



Постановка оптимизационной задачи:

$$\sum_{i=1}^{\ell} \big(|\langle w, x_i \rangle - w_0 - y_i| - \delta\big)_+ + \frac{1}{2C}\|w\|^2 \to \min_{w, w_0}.$$

Сводится к выпуклой задаче квадратичного программирования

Определения и обозначения

X — множество объектов
$X^\ell = \{x_1, \ldots, x_\ell\}$ — обучающая выборка
$i \prec j$ — правильный порядок на парах $(i, j) \in \{1, \ldots, \ell\}^2$

Задача:
построить ранжирующую функцию $a\colon X \to \mathbb{R}$ такую, что

$$i \prec j \quad \Rightarrow \quad a(x_i) < a(x_j)$$

Линейная модель ранжирования:

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

где $x \mapsto \big(f_1(x), \ldots, f_n(x)\big) \in \mathbb{R}^n$ — вектор признаков объекта x

Пример. Задача ранжирования поисковой выдачи

D — коллекция текстовых документов (documents)
Q — множество запросов (queries)
$D_q \subseteq D$ — множество документов, найденных по запросу q
$X = Q \times D$ — объектами являются пары «запрос, документ»:

$$x \equiv (q, d),\ q \in Q,\ d \in D_q$$

Y — упорядоченное множество рейтингов
$y : X \to Y$ — оценки релевантности, поставленные асессорами:
чем выше оценка $y(q, d)$, тем релевантнее документ d запросу q

Правильный порядок определён только между документами, найденными по одному и тому же запросу q:

$$(q, d) \prec (q, d') \quad \Leftrightarrow \quad y(q, d) < y(q, d')$$

Основные подходы к ранжированию

- Point-wise — поточечный
- Pair-wise — попарный
- List-wise — списочный

Переход к гладкому функционалу качества ранжирования:

$$Q(w) = \sum_{i \prec j} \Big[\underbrace{a(x_j) - a(x_i)}_{\text{Margin}(i,j)} < 0\Big] \leqslant \sum_{i \prec j} \mathscr{L}\big(a(x_j, w) - a(x_i, w)\big) \to \min_w$$

где $a(x, w)$ — модель ранжирования с вектором параметров w;
$\mathscr{L}(M)$ — убывающая непрерывная функция отступа $\text{Margin}(i, j)$:

- $\mathscr{L}(M) = (1 - M)_+$ — RankSVM
- $\mathscr{L}(M) = \exp(-M)$ — RankBoost
- $\mathscr{L}(M) = \ln(1 + e^{-M})$ — RankNet

RankNet: логистическая регрессия для ранжирования

RankNet: линейная модель ранжирования:

$$Q(w) = \sum_{i \prec j} \mathscr{L}\big(\langle x_j, w\rangle - \langle x_i, w\rangle\big) \quad \to \quad \min_w$$

где $\mathscr{L}(M) = \ln(1 + e^{-\sigma M})$ — логарифмическая функция потерь.

Метод стохастического градиента:
выбираем на каждой итерации случайную пару $i \prec j$:

$$w := w + \eta \cdot \frac{\sigma}{1 + \exp(\sigma\langle x_j - x_i, w\rangle)} \cdot (x_j - x_i)$$

где η — градиентный шаг, σ — параметр функции потерь

Christopher J.C. Burges From RankNet to LambdaRank to LambdaMART: An Overview // Microsoft Research Technical Report MSR-TR-2010-82. 2010.

От RankNet до LambdaRank

Метод стохастического градиента:

$$w := w + \eta \cdot \underbrace{\frac{\sigma}{1 + \exp(\sigma\langle x_j - x_i, w\rangle)}}_{\lambda_{ij}} \cdot (x_j - x_i);$$

Оказывается, для оптимизации негладких функционалов MAP, NDCG, pFound достаточно домножить λ_{ij} на изменение данного функционала при перестановке местами $x_i \leftrightarrows x_j$.

LambdaRank: домножение на изменение NDCG при $x_i \leftrightarrows x_j$ приводит к оптимизации NDCG:

$$w := w + \eta \cdot \frac{\sigma}{1 + \exp(\sigma\langle x_j - x_i, w\rangle)} \cdot |\Delta NDCG_{ij}| \cdot (x_j - x_i);$$

Christopher J.C. Burges From RankNet to LambdaRank to LambdaMART: An Overview // Microsoft Research Technical Report MSR-TR-2010-82. 2010.

Бустинг для задачи классификации с двумя классами

Взвешенное голосование базовых классификаторов:

$$a(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t b_t(x)\right), \quad x \in X,$$

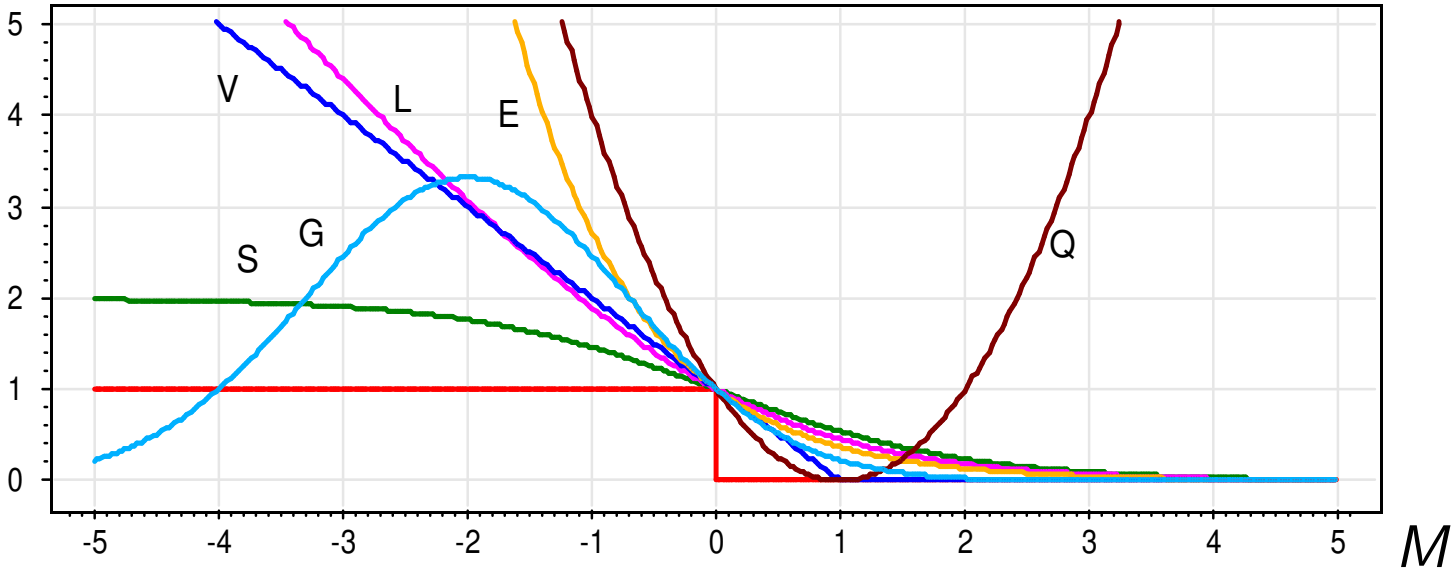
где $b_t \colon X \to \{\pm 1, 0\}$, $b_t(x) = 0$ — отказ от классификации
Функционал качества композиции — число ошибок на X^ℓ:

$$Q(\alpha_1, b_1, \ldots, \alpha_T, b_T) = \sum_{i=1}^{\ell} \left[y_i \sum_{t=1}^{T} \alpha_t b_t(x_i) < 0 \right] \to \min_{\{\alpha_t, b_t\}}$$

Две основные эвристики бустинга:

- фиксация $\alpha_1 b_1(x), \ldots, \alpha_{t-1} b_{t-1}(x)$ при добавлении $\alpha_t b_t(x)$;
- гладкая аппроксимация пороговой функции потерь $[M \leqslant 0]$.

Гладкие аппроксимации пороговой функции потерь $[M < 0]$



$E(M) = e^{-M}$ — экспоненциальная (AdaBoost);
$L(M) = \log_2(1 + e^{-M})$ — логарифмическая (LogitBoost);
$Q(M) = (1 - M)^2$ — квадратичная (GentleBoost);
$G(M) = \exp\big(-cM(M + s)\big)$ — гауссовская (BrownBoost);
$S(M) = 2(1 + e^{M})^{-1}$ — сигмоидная;
$V(M) = (1 - M)_+$ — кусочно-линейная (из SVM);

Градиентный бустинг для произвольной функции потерь

Линейная (выпуклая) комбинация базовых алгоритмов:

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in \mathbb{R}_+.$$

Функционал качества с произвольной функцией потерь $\mathscr{L}(a, y)$:

$$Q(\alpha, b; X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\Bigg(\underbrace{\sum_{t=1}^{T-1} \alpha_t b_t(x_i)}_{f_{T-1,i}} + \alpha b(x_i), y_i\Bigg) \to \min_{\alpha, b}.$$

$f_{T-1} = \big(f_{T-1,i}\big)_{i=1}^{\ell}$ — вектор текущего приближения
$f_T = \big(f_{T,i}\big)_{i=1}^{\ell}$ — вектор следующего приближения

Friedman G. Greedy Function Approximation: A Gradient Boosting Machine. 1999.

Параметрическая аппроксимация градиентного шага

Градиентный метод минимизации $Q(f) \to \min$, $f \in \mathbb{R}^\ell$:

$$f_0 := \text{начальное приближение};$$
$$f_{T,i} := f_{T-1,i} - \alpha g_i, \quad i = 1, \ldots, \ell;$$

$g_i = \mathscr{L}'\big(f_{T-1,i},\, y_i\big)$ — компоненты вектора градиента,
α — градиентный шаг.

Наблюдение: это очень похоже на одну итерацию бустинга!

$$f_{T,i} := f_{T-1,i} + \alpha b(x_i), \quad i = 1, \ldots, \ell$$

Идея: будем искать такой базовый алгоритм b_T, чтобы вектор $\big(b_T(x_i)\big)_{i=1}^\ell$ приближал вектор антиградиента $\big(-g_i\big)_{i=1}^\ell$:

$$b_T := \arg\min_b \sum_{i=1}^{\ell} \big(b(x_i) + g_i\big)^2$$

Алгоритм градиентного бустинга (Gradient Boosting)

Вход: обучающая выборка X^ℓ; параметр T;

Выход: базовые алгоритмы и их веса $\alpha_t b_t$, $t = 1, \ldots, T$;

1: инициализация: $f_i := 0$, $i = 1, \ldots, \ell$;

2: **для всех** $t = 1, \ldots, T$

3: базовый алгоритм, приближающий антиградиент:

$$b_t := \arg\min_b \sum_{i=1}^{\ell} \big(b(x_i) + \mathscr{L}'(f_i, y_i)\big)^2;$$

4: задача одномерной минимизации:

$$\alpha_t := \arg\min_{\alpha>0} \sum_{i=1}^{\ell} \mathscr{L}\big(f_i + \alpha b_t(x_i), y_i\big);$$

5: обновление вектора значений на объектах выборки:

$f_i := f_i + \alpha_t b_t(x_i)$; $i = 1, \ldots, \ell$;

Принцип максимума правдоподобия

Дано: i.i.d. выборка $X^m = \{x_1, \ldots, x_m\}$

Найти: параметрическую модель плотности

$$p(x) = \varphi(x; \theta),$$

где θ — параметр, φ — фиксированная функция.

Критерий — принцип максимума правдоподобия:

$$L(\theta; X^m) = \sum_{i=1}^{m} \ln \varphi(x_i; \theta) \to \max_{\theta}.$$

Необходимое условие оптимума:

$$\frac{\partial}{\partial \theta} L(\theta; X^m) = \sum_{i=1}^{m} \frac{\partial}{\partial \theta} \ln \varphi(x_i; \theta) = 0,$$

где функция $\varphi(x; \theta)$ достаточно гладкая по параметру θ.

Классический пример

Многомерное нормальное распределение

$$\mathcal{N}(x;\mu,\Sigma) = \frac{e^{-\frac{1}{2}(x-\mu)^{\mathsf{T}}\Sigma^{-1}(x-\mu)}}{\sqrt{(2\pi)^n \det\Sigma}},$$

μ — вектор матожидания, Σ — ковариационная матрица

Теорема

Оценки максимального правдоподобия по выборке:

$$\hat{\mu} = \frac{1}{m}\sum_i x_i;$$

$$\hat{\Sigma} = \frac{1}{m}\sum_i (x_i - \hat{\mu})(x_i - \hat{\mu})^{\mathsf{T}}.$$

Задача восстановления смеси распределений

Порождающая модель смеси распределений:

$$p(x) = \sum_{j=1}^{k} w_j \varphi(x, \theta_j), \qquad \sum_{j=1}^{k} w_j = 1, \qquad w_j \geqslant 0,$$

k — число компонент смеси;
$\varphi(x, \theta_j) = p(x|j)$ — *модель плотности* j-й компоненты;
$w_j = P(j)$ — *априорная вероятность* j-й компоненты.

Задача 1: при фиксированном k,
имея простую выборку $X^m = \{x_1, \ldots, x_m\} \sim p(x)$,
оценить вектор параметров $(w, \theta) = (w_1, \ldots, w_k, \theta_1, \ldots, \theta_k)$.

Задача 2: оценить ещё и k.

Максимизация правдоподобия и EM-алгоритм

Задача максимизации логарифма правдоподобия

$$\ln \prod_{i=1}^{m} p(x_i) = \sum_{i=1}^{m} \ln \sum_{j=1}^{k} w_j \varphi(x_i, \theta_j) \to \max_{w,\theta}.$$

при ограничениях $\sum_{j=1}^{k} w_j = 1;\ \ w_j \geqslant 0.$

Итерационный алгоритм Expectation–Maximization:

1: начальное приближение параметров (w, θ);
2: **повторять**
3: оценка скрытых переменных $G = (g_{ij}),\ \ g_{ij} = P(j|x_i)$:
$G := \text{E-шаг}\,(w, \theta)$;
4: максимизация правдоподобия, отдельно по компонентам:
$(w, \theta) := \text{M-шаг}\,(w, \theta, G)$;
5: **пока** w, θ и G не стабилизируются.

EM-алгоритм как способ решения системы уравнений

Теорема (необходимые условия экстремума)

Точка $(w_j, \theta_j)_{j=1}^k$ локального экстремума правдоподобия удовлетворяет системе уравнений относительно w_j, θ_j и g_{ij}:

$$\text{E-шаг:}\quad g_{ij} = \frac{w_j \varphi(x_i, \theta_j)}{\sum_{s=1}^k w_s \varphi(x_i, \theta_s)}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, k;$$

$$\text{M-шаг:}\quad \theta_j = \arg\max_\theta \sum_{i=1}^m g_{ij} \ln \varphi(x_i, \theta), \quad j = 1, \ldots, k;$$

$$w_j = \frac{1}{m} \sum_{i=1}^m g_{ij}, \quad j = 1, \ldots, k.$$

EM-алгоритм — это метод простых итераций для её решения

Вероятностная интерпретация

E-шаг — это формула Байеса:

$$g_{ij} = \mathsf{P}(j|x_i) = \frac{P(j)p(x_i|j)}{p(x_i)} = \frac{w_j\varphi(x_i,\theta_j)}{p(x_i)} = \frac{w_j\varphi(x_i,\theta_j)}{\sum_{s=1}^{k} w_s\varphi(x_i,\theta_s)}.$$

Очевидно, выполнено условие нормировки: $\sum_{j=1}^{k} g_{ij} = 1$.

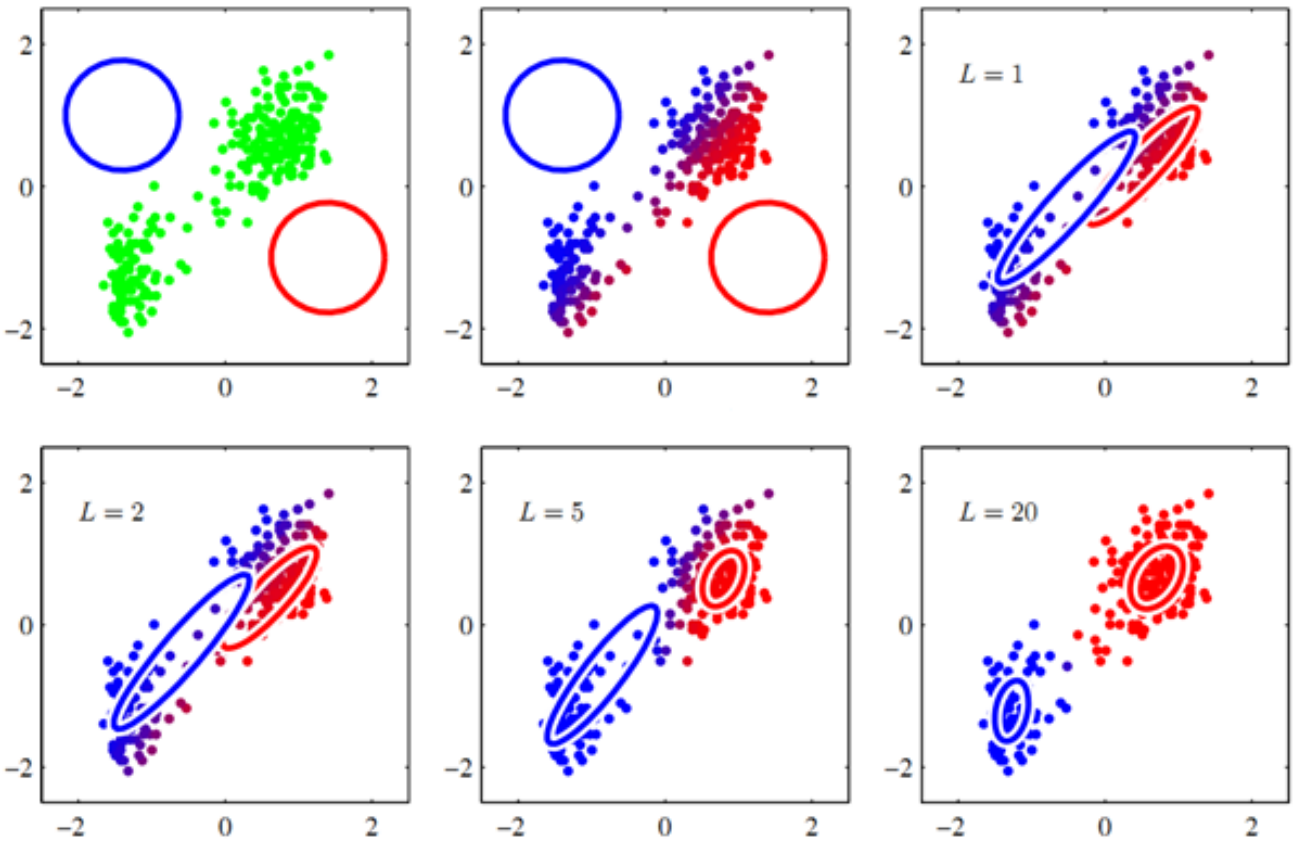
M-шаг — это максимизация взвешенного правдоподобия, с весами объектов g_{ij} для j-й компоненты смеси:

$$\theta_j = \arg\max_{\theta} \sum_{i=1}^{m} g_{ij} \ln \varphi(x_i,\theta),$$

$$w_j = \frac{1}{m}\sum_{i=1}^{m} g_{ij}.$$

Пример

Две гауссовские компоненты $k = 2$ в пространстве $X = \mathbb{R}^2$.
Расположение компонент в зависимости от номера итерации L:



Постановка задачи кластеризации

Дано:

X — пространство объектов;

$X^\ell = \{x_1, \ldots, x_\ell\}$ — обучающая выборка;

$\rho\colon X \times X \to [0, \infty)$ — функция расстояния между объектами.

Найти:

Y — множество кластеров,

$a\colon X \to Y$ — алгоритм кластеризации,

такие, что:

— каждый кластер состоит из близких объектов;

— объекты разных кластеров существенно различны.

Это задача *обучения без учителя* (unsupervised learning).

Постановка задачи частичного обучения (SSL)

Дано:

множество объектов X, множество классов Y;

$X^k = \{x_1, \ldots, x_k\}$ — размеченные объекты (labeled data);

$\{y_1, \ldots, y_k\}$

$U = \{x_{k+1}, \ldots, x_\ell\}$ — неразмеченные объекты (unlabeled data).

Два варианта постановки задачи:

- *Частичное обучение* (semi-supervised learning):
  построить алгоритм классификации $a\colon X \to Y$.
- *Трансдуктивное обучение* (transductive learning):
  зная все $\{x_{k+1}, \ldots, x_\ell\}$, получить метки $\{a_{k+1}, \ldots, a_\ell\}$.

Типичные приложения:

классификация и каталогизация текстов, изображений, и т. п.

Метод K-средних (K-means) для кластеризации

Минимизация суммы квадратов внутрикластерных расстояний:

$$\sum_{i=1}^{\ell} \|x_i - \mu_{a_i}\|^2 \to \min_{\{a_i\},\,\{\mu_a\}}, \quad \|x_i - \mu_a\|^2 = \sum_{j=1}^{n} \big(f_j(x_i) - \mu_{aj}\big)^2$$

Алгоритм Ллойда

Вход: X^ℓ, $K = |Y|$. **Выход:** центры кластеров μ_a, $a \in Y$

1: $\mu_a :=$ начальное приближение центров, для всех $a \in Y$;
2: **повторять**
3: отнести каждый x_i к ближайшему центру:
$a_i := \arg\min_{a \in Y} \|x_i - \mu_a\|, \quad i = 1, \ldots, \ell$;
4: вычислить новые положения центров:
$\mu_a := \dfrac{\sum_{i=1}^{\ell} [a_i = a] x_i}{\sum_{i=1}^{\ell} [a_i = a]}, \quad a \in Y$;
5: **пока** a_i не перестанут изменяться;

Метод K-средних (K-means) для частичного обучения

Модификация алгоритма Ллойда

при наличии размеченных объектов $\{x_1, \ldots, x_k\}$

Вход: X^ℓ, $K = |Y|$. **Выход:** центры кластеров μ_a, $a \in Y$

1: $\mu_a :=$ начальное приближение центров, для всех $a \in Y$;
2: **повторять**
3: отнести каждый $x_i \in U$ к ближайшему центру:
$a_i := \arg\min_{a \in Y} \|x_i - \mu_a\|$, $i = k+1, \ldots, \ell$;
4: вычислить новые положения центров:
$$\mu_a := \frac{\sum_{i=1}^{\ell} [a_i = a] x_i}{\sum_{i=1}^{\ell} [a_i = a]}, \quad a \in Y;$$
5: **пока** a_i не перестанут изменяться;

Метод K-средних — упрощение EM-алгоритма для GMM

EM-алгоритм: максимизация правдоподобия для разделения смеси гауссиан (GMM, Gaussian Mixture Model)

1: начальное приближение w_a, μ_a, Σ_a для всех $a \in Y$;
2: **повторять**
3: E-шаг: отнести каждый x_i к ближайшим центрам:

$$g_{ia} := P(a|x_i) \equiv \frac{w_a p_a(x_i)}{\sum_y w_y p_y(x_i)}, \quad a \in Y, \quad i = 1, \ldots, \ell;$$

$$a_i := \arg\max_{a \in Y} g_{ia}, \quad i = 1, \ldots, \ell;$$

4: M-шаг: вычислить новые положения центров:

$$\mu_{ad} := \frac{1}{\ell w_a} \sum_{i=1}^{\ell} g_{ia} f_d(x_i), \quad a \in Y, \quad d = 1, \ldots, n;$$

$$\sigma^2_{ad} := \frac{1}{\ell w_a} \sum_{i=1}^{\ell} g_{ia} (f_d(x_i) - \mu_{ad})^2, \quad a \in Y, \quad d = 1, \ldots, n;$$

$$w_a := \frac{1}{\ell} \sum_{i=1}^{\ell} g_{ia}, \quad a \in Y;$$

5: **пока** a_i не перестанут изменяться;

Метод главных компонент: постановка задачи

$f_1(x), \ldots, f_n(x)$ — исходные числовые признаки;
$g_1(x), \ldots, g_m(x)$ — новые числовые признаки, $m \leqslant n$;

Требование: старые признаки должны линейно восстанавливаться по новым:

$$\hat{f}_j(x) = \sum_{s=1}^{m} g_s(x) u_{js}, \quad j = 1, \ldots, n, \quad \forall x \in X,$$

как можно точнее на обучающей выборке $x_1, \ldots, x_\ell$:

$$\sum_{i=1}^{\ell} \sum_{j=1}^{n} \big(\hat{f}_j(x_i) - f_j(x_i)\big)^2 \to \min_{\{g_s(x_i)\}, \{u_{js}\}}$$

Матричные обозначения

Матрицы «объекты–признаки», старая и новая:

$$\underset{\ell\times n}{F} = \begin{pmatrix} f_1(x_1) & \ldots & f_n(x_1) \\ \ldots & \ldots & \ldots \\ f_1(x_\ell) & \ldots & f_n(x_\ell) \end{pmatrix}; \quad \underset{\ell\times m}{G} = \begin{pmatrix} g_1(x_1) & \ldots & g_m(x_1) \\ \ldots & \ldots & \ldots \\ g_1(x_\ell) & \ldots & g_m(x_\ell) \end{pmatrix}.$$

Матрица линейного преобразования новых признаков в старые:

$$\underset{n\times m}{U} = \begin{pmatrix} u_{11} & \ldots & u_{1m} \\ \ldots & \ldots & \ldots \\ u_{n1} & \ldots & u_{nm} \end{pmatrix}; \qquad \hat{F} = GU^{\mathsf{T}} \overset{\text{хотим}}{\approx} F.$$

Найти: и новые признаки G, и преобразование U:

$$\sum_{i=1}^{\ell}\sum_{j=1}^{n}\big(\hat{f}_j(x_i) - f_j(x_i)\big)^2 = \big\|GU^{\mathsf{T}} - F\big\|^2 \to \min_{G,U},$$

Основная теорема метода главных компонент

Теорема

Если $m \leqslant \operatorname{rk} F$, то минимум $\left\|GU^{\mathsf{T}} - F\right\|^2$ достигается, когда столбцы U — это с.в. матрицы $F^{\mathsf{T}}F$, соответствующие m максимальным с.з. $\lambda_1, \ldots, \lambda_m$, а матрица $G = FU$.

При этом:

1. матрица U ортонормирована: $U^{\mathsf{T}}U = I_m$;
2. матрица G ортогональна: $G^{\mathsf{T}}G = \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$;
3. $U\Lambda = F^{\mathsf{T}}FU$; $G\Lambda = FF^{\mathsf{T}}G$;
4. $\left\|GU^{\mathsf{T}} - F\right\|^2 = \|F\|^2 - \operatorname{tr}\Lambda = \sum\limits_{j=m+1}^{n} \lambda_j$.

Обобщение. Задачи низкорангового матричного разложения

- Понижение размерности в задачах регрессии
- Понижение размерности в задачах классификации
- Формирование сжатого представления данных

Дано: матрица $Z = \|z_{ij}\|_{n\times m}$, $(i,j) \in \Omega \subseteq \{1..n\} \times \{1..m\}$

Найти: матрицы $X = \|x_{it}\|_{n\times k}$ и $Y = \|y_{tj}\|_{k\times m}$ такие, что

$$\|Z - XY\|^2 = \sum_{(i,j)\in\Omega} \left(z_{ij} - \sum_t x_{it} y_{tj}\right)^2 \to \min_{X,Y}$$

Дополнительные ограничения, вынуждающие отказаться от SVD:

- неквадратичная функция потерь
- неотрицательное матричное разложение: $x_{it} \geqslant 0$, $y_{tj} \geqslant 0$
- разреженные данные: $|\Omega| \ll nm$

Примеры прикладных задач матричного разложения

1. **Выявление интересов в рекомендательных системах (recommender systems, collaborative filtering)**

$$z_{iu} = \sum_t p_{it} q_{tu}$$

дано: z_{iu} — рейтинги товаров i, поставленные пользователем u;
найти: p_{it} — профиль интересов товара i;
q_{tu} — профиль интересов пользователя u.

2. **Разделение смеси химических веществ по данным жидкостной хроматографии**

$$z_{t\lambda} = \sum_i x_{ti} y_{i\lambda}$$

дано: $z_{t\lambda}$ — выход сканирующего УФ-детектора;
найти: x_{ti} — хроматограмма i-го вещества, t — время;
$y_{i\lambda}$ — спектр i-го вещества, λ — длина волны.

Примеры прикладных задач матричного разложения

3. **Латентный семантический анализ коллекций текстов (тематическое моделирование)**

$$z_{wd} = \sum_t \varphi_{wt}\theta_{td}$$

дано: $z_{wd} = p(w|d)$ — частоты слов w в документах d;
найти: $\varphi_{wt} = p(w|t)$ — распределения слов w в темах t,
$\theta_{td} = p(t|d)$ — распределения тем t в документах d.

4. **Оценивание экспрессии генов по данным ДНК-микрочипов с учётом кросс-гибридизации**

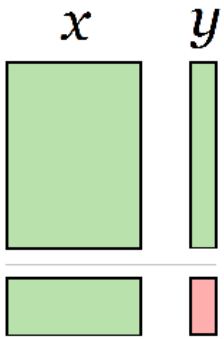
$$z_{pk} = \sum_g a_{pg} c_{gk}$$

дано: z_{pk} — интенсивность свечения p-й пробы на k-м чипе;
найти: a_{pg} — коэффициент сродства p-й пробы g-му гену,
c_{gk} — концентрация g-го гена на k-м чипе.

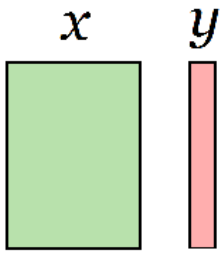
Обучение с использованием привилегированной информации

LUPI — Learning Using Priveleged Information

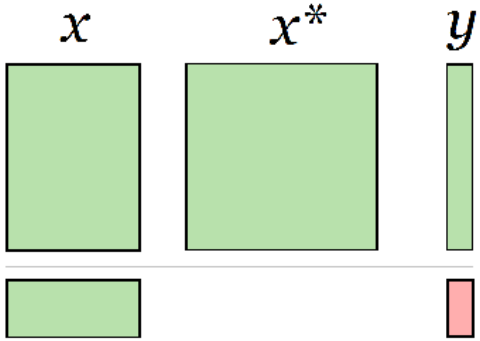
с учителем



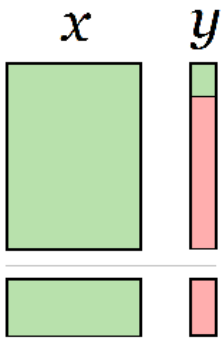
без учителя



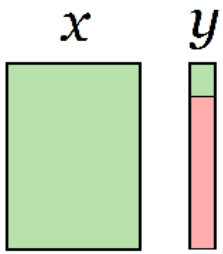
привилегированное (LUPI)



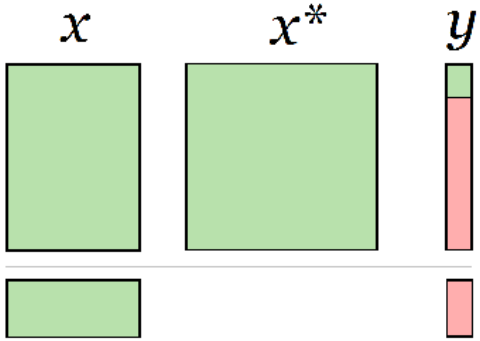
частичное



трансдуктивное



частичное LUPI



V.Vapnik, A.Vashist. A new learning paradigm: Learning Using Privileged Information // Neural Networks. 2009.

Примеры задач с привилегированной информацией x^*

- x — первичная (1D) структура белка
  x^* — третичная (3D) структура белка
  y — иерархическая классификация
- x — предыстория временного ряда
  x^* — информация о будущем поведении ряда
  y — прогноз следующей точки ряда
- x — данные баллистокардиографии
  x^* — данные ЭКГ (мониторирование по Холтеру)
  y — диагноз
- x — документ
  x^* — выделенные ключевые слова или фразы
  y — категория документа
- x — пара (запрос, документ)
  x^* — выделенные асессором ключевые слова или фразы
  y — асессорская оценка релевантности

Задача обучения с привилегированной информацией

Раздельное обучение модели-ученика и модели-учителя:

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

Модель-ученик обучается повторять ошибки модели-учителя:

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

Совместное обучение модели-ученика и модели-учителя:

$$\sum_{i=1}^{\ell} \mathscr{L}\big(a(x_i, w), y_i\big) + \lambda \mathscr{L}\big(a(x_i^*, w^*), y_i\big) + \\ + \mu \mathscr{L}\big(a(x_i, w), a(x_i^*, w^*)\big) \to \min_{w, w*}$$

D.Lopez-Paz, L.Bottou, B.Scholkopf, V.Vapnik. Unifying distillation and privileged information. 2016.

Задача о многоруком бандите (multi-armed bandit)

A — множество возможных *действий*
$p(r|a)$ — неизвестное распределение *премии* $r \in \mathbb{R}$ для $a \in A$
$\pi_t(a)$ — *стратегия* (policy) агента в момент t, распределение на A

Игра агента со средой:

инициализация стратегии $\pi_1(a)$;
для всех $t = 1, \ldots, T, \ldots$
- агент выбирает действие $a_t \sim \pi_t(a)$;
- среда генерирует премию $r_t \sim p(r|a_t)$;
- агент корректирует стратегию $\pi_{t+1}(a)$;

$$R_t(a) = \frac{\sum_{i=1}^{t} r_i[a_i = a]}{\sum_{i=1}^{t}[a_i = a]} \quad \text{— средняя премия в } t \text{ раундах}$$

$$R^*(a) = \lim_{t\to\infty} R_t(a) \to \max_{a\in A} \quad \text{— } \textit{ценность действия } a$$

Примеры прикладных задач

- Рекомендация новостных статей пользователям
- Показ рекламы в Интернете
- Управление технологическими процессами
- Управление роботами
- Управление ценами и ассортиментом в сетях продаж
- Игра на бирже
- Маршрутизация в телекоммуникационных сетях
- Маршрутизация в беспроводных сенсорных сетях
- Стратегические игры: шахматы, го, Dota2, StarCraft2, ...

Задача о многоруком бандите впервые рассмотрена в статье
H. Robbins. Some aspects of the sequential design of experiments.
Bulletin of the American Mathematics Society, 58:527–535, 1952.

Метод UCB (upper confidence bound)

Выбор действия с максимальной верхней оценкой ценности:

$$a_t = \arg\max_{a \in A} \left(R_t(a) + \delta \sqrt{\frac{2 \ln t}{k_t(a)}} \right),$$

где $k_t(a) = \sum_{i=1}^{t} [a_i = a]$, δ — параметр exr/ext-компромисса.

Компромисс «Exploration–Exploitation»:

- чем меньше $k_t(a)$, тем менее исследована стратегия, тем выше должна быть вероятность выбрать a;
- чем больше δ, тем стратегия более исследовательская.

P. Auer, N. Cesa-Bianchi, P. Fischer. Finite-time analysis of the multiarmed bandit problem, Machine Learning, 2002.

Постановка задачи в случае, когда агент влияет на среду

A — конечное множество возможных *действий* (action)

S — конечное множество состояний среды (state)

Игра агента со средой:

инициализация стратегии $\pi_1(a \mid s)$ и состояния среды s_1;

для всех $t = 1, \ldots, T, \ldots$

- агент выбирает действие $a_t \sim \pi_t(a \mid s_t)$;
- среда генерирует премию $r_t \sim p(r \mid a_t, s_t)$
- и новое состояние $s_{t+1} \sim p(s \mid a_t, s_t)$;
- агент корректирует стратегию $\pi_{t+1}(a \mid s)$;

Q-learning: жадная стратегия принятия решений

$$a_t = \arg\max_{a \in A} Q(a, s_t),$$

$Q(a, s)$ — оценка будущей ценности действия a в состоянии s

Градиентная оптимизация стратегии (policy gradient)

Обобщение:

- $p(x|\theta) = \pi(a \,|\, s, \theta)$ — параметризованная стратегия агента
- $x = (a, s)$ — вектор признаков *действия в состоянии*
- $Q(x)$ — оценка будущей ценности *действия в состоянии*

Задача: максимизировать среднюю Q по параметрам θ:

$$\mathsf{E}_\pi Q(x) \equiv \mathsf{E}_{x\sim p(x|\theta)} Q(x) \equiv \mathsf{E}_{x|\theta} Q(x) \to \max_\theta$$

Градиентный метод: $\theta^{(t+1)} := \theta^{(t)} + \eta \nabla_\theta \mathsf{E}_{x|\theta^{(t)}} Q(x)$;

$$\nabla_\theta \mathsf{E}_{x|\theta} Q(x) = \nabla_\theta \sum_x Q(x) p(x|\theta) = \sum_x Q(x) \nabla_\theta p(x|\theta) =$$

$$= \sum_x Q(x) p(x|\theta) \frac{\nabla_\theta p(x|\theta)}{p(x|\theta)} = \mathsf{E}_{x|\theta}\big(Q(x) \nabla_\theta \ln p(x|\theta)\big)$$

Градиентная оптимизация стратегии (policy gradient)

Шаги с градиентом $Q_t(a_t, s_t)\nabla_\theta \ln \pi(a_t \mid s_t, \theta)$ похожи на шаги Stochastic Gradient для максимизации log-правдоподобия:

$$\sum_t Q_t(a_t, s_t) \ln \pi(a_t \mid s_t, \theta) \;\to\; \max_\theta$$

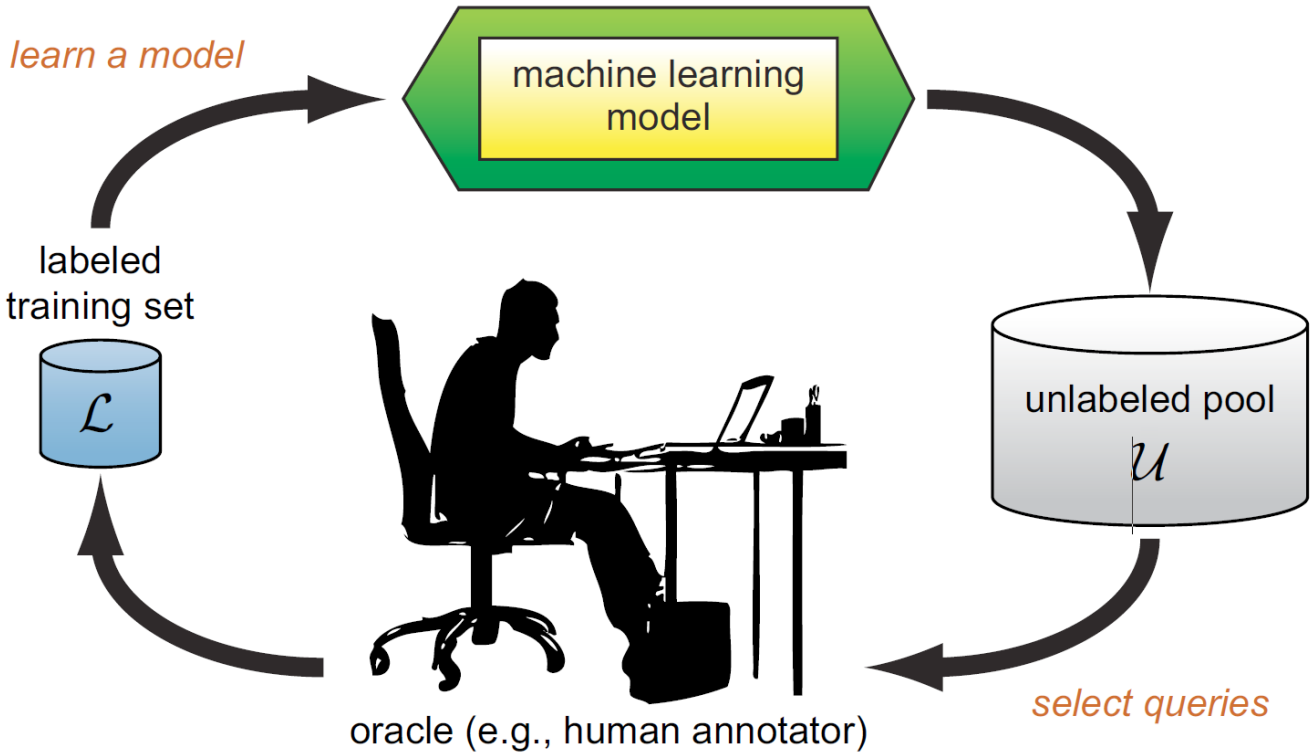
где $Q_t(a, s)$ — текущая (вычисленная в раунде t) оценка будущей ценности действия a в состоянии s

Отличия от SG-максимизации log-правдоподобия:

- в роли классов выступают действия a_t
- в роли объектов — «действия в состоянии» (a_t, s_t)
- объекты выбираются не случайно, а в порядке поступления
- вместо бинарных классификаций вещественные $Q_t(a_t, s_t)$

Постановка задачи активного обучения

Задача: обучение предсказательной модели $a: X \to Y$
по выборке (x_i, y_i), когда получение ответов y_i стоит дорого.



Burr Settles. Active Learning Literature Survey. 2010.

Постановка задачи активного обучения

Задача: обучение предсказательной модели $a\colon X \to Y$
по выборке (x_i, y_i), когда получение ответов y_i стоит дорого.

Вход: начальная размеченная выборка $X^\ell = (x_i, y_i)_{i=1}^{\ell}$;
Выход: модель a и размеченная выборка $(x_i, y_i)_{i=\ell+1}^{\ell+k}$;

обучить модель a по начальной выборке $(x_i, y_i)_{i=1}^{\ell}$;
пока остаются неразмеченные объекты
- выбрать неразмеченный объект x_i;
- узнать для него y_i;
- дообучить модель a ещё на одном примере (x_i, y_i);

Цель активного обучения:
достичь как можно лучшего качества модели a,
использовав как можно меньше дополнительных примеров k.

Примеры приложений активного обучения

- сбор асессорских данных для информационного поиска, анализа текстов, сигналов, речи, изображений, видео
- *планирование экспериментов* в естественных науках (пример — комбинаторная химия)
- оптимизация трудно вычислимых функций (пример — поиск в пространстве гиперпараметров)
- управление ценами и ассортиментом в торговых сетях
- выбор товара для проведения маркетинговой акции

Сэмплирование по неуверенности (uncertainty sampling)

Идея: выбирать x_i с наибольшей неопределённостью $a(x_i)$.

Задача многоклассовой классификации:

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

$p_k(x)$, $k = 1 \ldots |Y|$ — ранжированные по убыванию $P(y|x)$, $y \in Y$.

- Принцип *наименьшей достоверности* (least confidence):

$$x_i = \arg\min_{u \in X^k} p_1(u)$$

- Принцип *наименьшей разности отступов* (margin sampling):

$$x_i = \arg\min_{u \in X^k} \big(p_1(u) - p_2(u)\big)$$

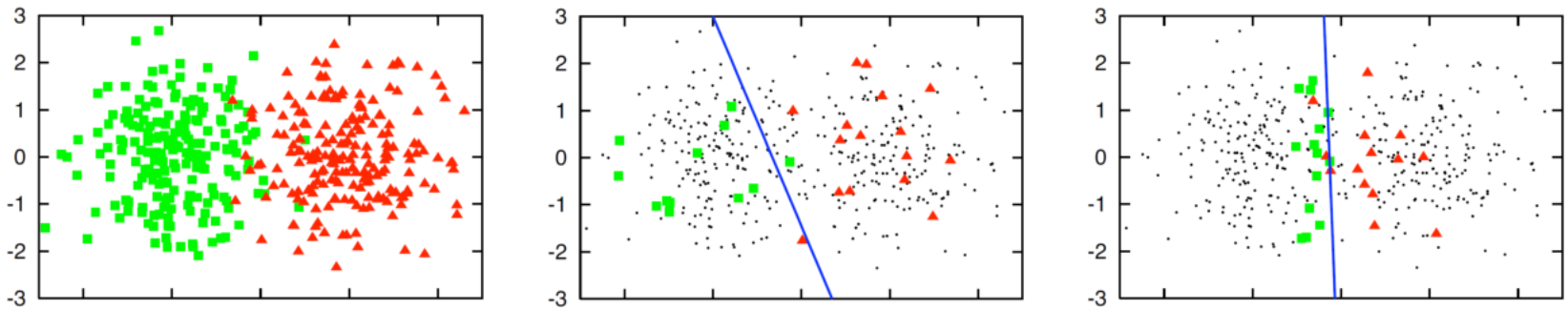
- Принцип *максимума энтропии* (maximum entropy):

$$x_i = \arg\min_{x \in X^k} \sum_k p_k(u) \ln p_k(u)$$

Почему активное обучение быстрее пассивного

Пример 1. Синтетические данные: $\ell = 30$, $\ell + k = 400$;

(a) два гауссовских класса;
(b) логистическая регрессия по 30 случайным объектам;
(c) логистическая регрессия по 30 объектам, отобранным с помощью активного обучения.



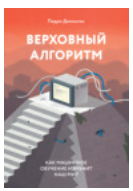
(a) (b) (c)

Обучение по смещённой неслучайной выборке требует меньше данных для построения алгоритма сопоставимого качества.

Burr Settles. Active Learning Literature Survey. 2010.

Основные школы машинного обучения

1. символизм (поиск логических закономерностей)
   - Decision Tree, Rule Induction, Association Rules
2. коннекционизм (нейронные сети)
   - BackPropagation, Deep Belief Nets, Deep Learning
3. эволюционизм (генетические алгоритмы)
   - Genetic Algorithms, Genetic Programming
4. байесионизм (оценивание апостериорных распределений)
   - Naive Bayes, Bayesian Networks, Graphical Models
5. аналогизм (гипотезы непрерывности и компактности)
   - kNN, RBF, SVM, Kernel Regression, Kernel Density Estimation
6. + композиционизм
   - Voting, Boosting, Bagging, Stacking, RF, MatrixNet, CatBoost



Домингос П. Верховный алгоритм. 2016. 336 с.

Типология задач машинного обучения

1. Предварительная обработка (data preparation)
   - извлечение признаков (feature extraction)
   - отбор признаков (feature selection)
   - восстановление пропусков (missing values)
   - фильтрация выбросов (outlier detection)
2. Обучение с учителем (supervised learning)
   - классификация (classification)
   - регрессия (regression)
   - ранжирование (learning to rank)
   - прогнозирование (forecasting)
3. Обучение без учителя (unsupervised learning)
   - кластеризация (clustering)
   - поиск ассоциативных правил (association rule learning)
   - восстановление плотности (density estimation)
   - одноклассовая классификация (anomaly detection)
4. Частичное обучение (semi-supervised learning)
   - трансдуктивное обучение (transductive learning)
   - обучение с положительными примерами (PU-learning)

Типология задач машинного обучения

5. Обучение представлений (representation learning)
   - обучение признаков (feature learning)
   - обучение многообразий (manifold learning)
   - матричные разложения (matrix factorization)
6. Обучение близости/связей (similarity/relational learning)
7. Обучение структуры модели (structure learning)
8. Глубокое обучение (deep learning)
9. Состязательное обучение (adversarial learning)
10. Обучение с подкреплением (reinforcement learning)
11. Привилегированное обучение (privileged learning)
12. Динамическое обучение (online/incremental learning)
13. Активное обучение (active learning)
14. Мета-обучение (meta-learning)
15. Перенос обучения (transfer learning)
    - inductive transfer, learning to learn — просто синонимы
    - многозадачное обучение (multitask learning)