

# Обучение с учителем. Классификация. Дискриминантный анализ.

Е. Ларин, Ф. Ежов, И. Кононыхин

Санкт-Петербургский государственный университет  
Прикладная математика и информатика  
Вычислительная стохастика и статистические модели

# Обучение с учителем

Выборка из генеральной случайной величины

- Для задачи регрессии:  $\mathbf{X} \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{R}^n$
- Для задачи классификации:  $\mathbf{X} \in \mathbb{R}^{n \times p}$ ,  $\mathbf{y} \in \mathbb{A}^n$

# Обучение с учителем: формальная постановка

- *Вход*:  $\mathbf{X}$  — выборка  $\xi$ ,  $\mathbf{y}$  — выборка  $\eta$ . Предполагаем, что существует неизвестное отображение  $y^* : \xi \rightarrow \eta$  (гипотеза непрерывности или компактности)
- *Задача*: По  $\mathbf{X}$  и  $\mathbf{y}$  найти такое отображение  $\hat{y}^* : \xi \rightarrow \eta$ , которое приблизит отображение  $y^*$ .
- *Оценка*: Функция потерь  $\mathcal{L}(y^*(x), \hat{y}^*(x))$ . Здесь  $x$  — реализация  $\xi$

# Классификация

$$\mathbf{X} \in \mathbb{R}^{n \times p}, \quad \mathbf{y} \in \mathbb{A}^n \quad (1)$$

## Гипотеза компактности

«Близкие» объекты, как правило, принадлежат одному классу

Понятие близости может быть формализовано, например, так:

$$\rho(\mathbf{x}_1, \mathbf{x}_2) = \left( \sum_{i=1}^p w_i |x_1^i - x_2^i|^k \right)^{\frac{1}{k}}$$

# Классификация: генеральная постановка

*Дано:*

- $\xi \in \mathbb{R}^p$  — вектор признаков
- $\eta \in \mathbb{A}$  — классовая принадлежность

Предположение об их зависимости можно записать в виде 2.

$$\eta = \Phi(\xi, \varepsilon) \quad (2)$$

Обычно на  $\varepsilon$  накладываются условия

$$E\varepsilon = 0, \quad D\varepsilon = \sigma^2, \quad \xi \perp \varepsilon$$

*Задача:* найти  $\Phi$

# Классификация: выборочная постановка

*Дано:*

- $\mathbf{X} \in \mathbb{R}^{n \times p}$  — матрица признаков
- $\mathbf{y} \in \mathbb{A}^n$  — вектор классовой принадлежности

Предположение имеет вид 3.

$$y_i = \Phi(\mathbf{x}_i, \varepsilon_i), \quad i = 1, \dots, n \quad (3)$$

*Задача:* найти  $\Phi$

# Классификация: оценка качества

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

На основе этой матрицы есть большое количество разных метрик:  
*accuracy*, *recall*, *precision*,  $F_\beta$ , *ROC-AUC*

# Классификация: типы классов

- По количеству классов:
  - ▶ бинарная классификация
  - ▶ многоклассовая классификация
- По пересечению классов
  - ▶ пересекающиеся
  - ▶ непересекающаяся
  - ▶ нечёткие



# Классификация: этапы обучения модели

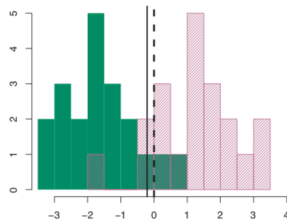
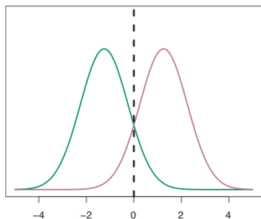
- Выбор модели (класс рассматриваемых  $\Phi$  из 3)
- Выбор метрики
- Выбор метода обучения (способ подбора параметров для минимизации метрики на обучающем множестве)
- Выбор метода проверки (способ оценки качества модели)

# Классификация: задача оптимизации

- $\hat{\beta}$  — параметры модели
- $\Phi(\mathbf{x}, \beta)$  — функционал классификации
- $\mathcal{L}(\Phi(\mathbf{x}, \beta), \mathbf{y})$  — функция потерь (метрика)

$$\hat{\beta} = \arg \min_{\beta} \mathcal{L}(\Phi(\mathbf{x}, \beta), \mathbf{y})$$

# Классификация: отступы



# Классификация: общий подход к решению

Как построить функционал  $\Phi$ ?

Общий подход — построить набор  $f_i$ ,  $i = 1, \dots, K$ . Каждая функция  $f_i(\mathbf{x})$  показывает меру принадлежности  $\mathbf{x}$  классу  $i$ .

Таким образом,

$$\Phi(\mathbf{x}) = \arg \max_i (f_i(\mathbf{x})). \quad (4)$$

# Дискриминантный анализ

Примем за функции  $f_i$  из 4 оценку вероятности принадлежности к  $i$ -му классу.

$$\Phi(\mathbf{x}) = \arg \max_i (P(C_i|\mathbf{x})).$$

$C_i$  — класс, состоящий из одного события:  $\mathbf{x}$  принадлежит  $i$ -му классу.

# Дискриминантный анализ

Если известны априорные вероятности получения  $i$ -го класса ( $\pi_i$ ), применим формулу Байеса

$$P(C_i|\mathbf{x}) = \frac{\pi_i P(\mathbf{x}|C_i)}{\sum_{j=1}^K \pi_j P(\mathbf{x}|C_j)}.$$

Отбросим знаменатель

$$f_i = P(C_i|\mathbf{x}) = \pi_i P(\mathbf{x}|C_i).$$

Предположение:

$$P(\mathbf{x}|\eta = A_i) = N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$$

Классифицирующая функция:

$$f_i(\mathbf{x}) = \frac{\pi_i}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)^T\right)$$

После упрощения:

$$h_i(\mathbf{x}) = -0.5\boldsymbol{\mu}_i\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i^T + \boldsymbol{\mu}_i\boldsymbol{\Sigma}^{-1}\mathbf{x} + \log \pi_i$$

Предположение:

$$P(\boldsymbol{\xi}|\eta = A_i) = N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

Классифицирующая функция:

$$f_i(\mathbf{x}) = \frac{\pi_i}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)\boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)^T\right)$$

После упрощения:

$$g_i(\mathbf{x}) = -0.5(\mathbf{x} - \boldsymbol{\mu}_i)\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)^T - 0.5 \log |\boldsymbol{\Sigma}_i| + \log \pi_i$$



# Лог. регрессия

*Зададим модель логистической регрессии следующим образом:*

$$\log \frac{P(\eta = G_i | \boldsymbol{\xi} = \mathbf{x})}{P(\eta = G_K | \boldsymbol{\xi} = \mathbf{x})} = \beta_{i0} + \boldsymbol{\beta}_i^T \mathbf{x}, i = 1, \dots, K - 1.$$

*Перейдем от логитов к вероятностям:*

$$P(\eta = G_i | \boldsymbol{\xi} = \mathbf{x}) = \frac{e^{\beta_{i0} + \boldsymbol{\beta}_i^T \mathbf{x}}}{1 + \sum_{k=1}^{K-1} e^{\beta_{k0} + \boldsymbol{\beta}_k^T \mathbf{x}}}, i = 1, \dots, K - 1,$$

$$P(\eta = G_K | \boldsymbol{\xi} = \mathbf{x}) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_{k0} + \boldsymbol{\beta}_k^T \mathbf{x}}}.$$

# Лог. регрессия: метод максимального правдоподобия

Для оценки параметров воспользуемся методом максимального правдоподобия:

$$l(\theta) = \sum_{i=1}^N \log P(\eta = G_k | \boldsymbol{\xi} = \mathbf{x}; \theta),$$

$$\theta = (\beta_{10}, \beta_1^T, \dots, \beta_{(K-1)0}, \beta_{K-1}^T).$$

Iteratively reweighted least squares (IRLS).

Функция потерь:  $\mathfrak{L}(M_i(\beta)) = \log(1 + e^{-y_i \beta^T x_i})$

Выборка:  $\{x_i, y_i\}_{i=1}^n$ ,  $x_i \in \mathbb{R}^p$ ,  $y_i \in \{-1, 1\}$ .

Задача построить классифицирующее правило.

Предположим, что данные - разделимы гиперплоскостью,

$$\mathbf{x}^T \beta - \beta_0 = 0; \beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R},$$

$$g(x) = \mathbf{x}^T \beta - \beta_0,$$

$$h(x) = \text{sign}(g(x)).$$

Критерий оптимальности: максимальное расстояние между двумя гиперплоскостями, параллельных данной и симметрично расположенных относительно нее.

Эта пара гиперплоскостей может быть описана парой уравнений:

$$\mathbf{x}^T \boldsymbol{\beta} - \beta_0 = -1,$$

$$\mathbf{x}^T \boldsymbol{\beta} - \beta_0 = 1.$$

Расстояние между ними:  $\frac{2}{\|\boldsymbol{\beta}\|}$ .

Принадлежность точек обучающей выборки полупространства описывается

$$(\mathbf{x}^T \boldsymbol{\beta} - \beta_0) y_i \geq 1$$

Задача сводится к задаче квадратичного программирования с линейными ограничениями:

$$\begin{cases} \frac{1}{2} \|\boldsymbol{\beta}\|_2^2 \rightarrow \min_{\boldsymbol{\beta}, \beta_0} \\ y_i (x_i^T \boldsymbol{\beta} - \beta_0) \geq 1 \end{cases}$$

Принадлежность точек обучающей выборки полупространства описывается

$$(\mathbf{x}^T \beta - \beta_0) y_i \geq 1$$

Задача сводится к задаче квадратичного программирования с линейными ограничениями:

$$\begin{cases} \frac{1}{2} \|\beta\|_2^2 \rightarrow \min_{\beta, \beta_0} \\ y_i (x_i^T \beta - \beta_0) \geq 1 \end{cases}$$

# Кросс-валидация

Кросс-валидация (aka перекрестная проверка, скользящий контроль) — процедура эмпирического оценивания обобщающей способности алгоритмов.

С помощью кросс-валидации "эмулируется" наличие тестовой выборки, которая не участвует в обучении модели, но для которой известны правильные ответы.

# Кросс-валидация: виды

- Валидация на отложенных данных (Hold-Out Validation);
- Полная кросс-валидация (Complete Cross-Validation);
- k-fold Cross-Validation;
- $t \times k$ -fold Cross Validation;
- Кросс-валидация по отдельным объектам (Leave-One-Out);
- Случайные разбиения (Random Subsampling).



# Кросс-валидация: Обозначения

Введем обозначения:

- $\mathbf{X}$  — матрица признаков, описывающие таргеты  $\mathbf{y}$ ;
- $\mathbf{X}' = (x_i, y_i)_{i=1}^I$ ,  $x_i \in \mathbf{X}$ ,  $y_i \in \mathbf{y}$  — обучающая выборка;
- $\mathcal{L}$  — функция потерь (мера качества);
- $A$  — исследуемая модель;
- $\mu : (\mathbf{X} \times \mathbf{y}) \rightarrow A$  — алгоритм обучения.

# Кросс-валидация: Hold-Out Validation

Обучающая выборка один раз случайным образом разбивается на две части  $\mathbf{T} = \mathbf{T}^t \cup \mathbf{T}^{l-t}$ .

Train, $T^t$	Test, $T^{l-t}$
--------------	-----------------

После чего решается задача оптимизации:

$$HO(\mu, \mathbf{T}^t, \mathbf{T}^{l-t}) = \mathcal{L}(\mu(\mathbf{T}^t), \mathbf{T}^{l-t}) \rightarrow \min.$$

Метод Hold-out применяется в случаях больших датасетов, т.к. требует меньше вычислительных мощностей по сравнению с другими методами кросс-валидации.

Недостатком метода является то, что оценка существенно зависит от разбиения, тогда как желательно, чтобы она характеризовала только алгоритм обучения.

# Кросс-валидация: Complete cross-validation

- Выбирается значение  $t$ ;
- Выборка разбивается всевозможными способами на две части  $\mathbf{T} = \mathbf{T}^t \cup \mathbf{T}^{l-t}$ .

Train, $T^t$	Test, $T^{l-t}$
--------------	-----------------

После чего решается задача оптимизации:

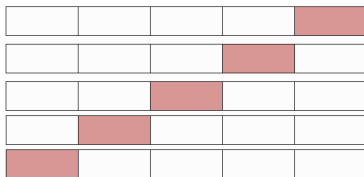
$$CVV_t = \frac{1}{C_l^{l-t}} \sum_{\mathbf{x}^l = \mathbf{T}^t \cup \mathbf{T}^{l-t}} \mathcal{L}(\mu(\mathbf{T}^t), \mathbf{T}^{l-t}) \rightarrow \min.$$

Здесь число разбиений  $C_l^{l-t}$  становится слишком большим даже при сравнительно малых значениях  $t$ , что затрудняет практическое применение данного метода.

# Кросс-валидация: k-fold Cross-Validation

- $\mathbf{X}'$  разбивается на  $\mathbf{F}_1 \cup \dots \cup \mathbf{F}_k$ ,  $|\mathbf{F}_i| \approx \frac{1}{k}$  частей;
- Производится  $k$  итераций:
  - ▶ Модель обучается на  $k - 1$  части обучающей выборки;
  - ▶ Модель тестируется на части обучающей выборки, которая не участвовала в обучении.

Каждая из  $k$  частей единожды используется для тестирования.



После чего решается задача оптимизации:

$$CV_k = \frac{1}{k} \sum_{i=1}^k \mathcal{L}(\mu(\mathbf{X}' \setminus \mathbf{F}_i), \mathbf{F}_i) \rightarrow \min.$$

# Кросс-валидация: $t \times k$ -fold Cross Validation

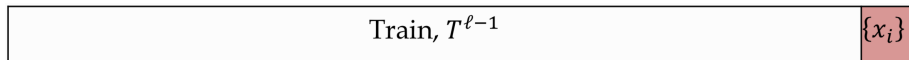
*Как  $k$ -fold Cross-Validation, только  $t$  раз.*

Разбиение:  $\mathbf{X}^I = \mathbf{F}_{(1,1)} \cup \dots \cup \mathbf{F}_{(k,1)} = \mathbf{F}_{(1,t)} \cup \dots \cup \mathbf{F}_{(k,t)}$ ,  $|\mathbf{F}_{(i,j)}| \approx \frac{I}{k}$ ,

Задача оптимизации:  $CV_{t \times k} = \frac{1}{tk} \sum_{j=1}^t \sum_{i=1}^k \mathcal{L}(\mu(\mathbf{X}^I \setminus \mathbf{F}_{i,j}), \mathbf{F}_{i,j}) \rightarrow \min$

# Кросс-валидация: Leave-One-Out

Выборка разбивается на  $l - 1$  и 1 объект  $l$  раз.



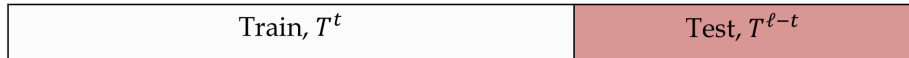
$$LOO = \frac{1}{l} \sum_{i=1}^l \mathcal{L}(\mu(\mathbf{X}^l \setminus p_i), p_i) \rightarrow \min, \text{ где } p_i = (x_i, y_i).$$

Преимущества LOO в том, что каждый объект ровно один раз участвует в контроле, а длина обучающих подвыборок лишь на единицу меньше длины полной выборки.

Недостатком LOO является большая ресурсоёмкость, так как обучаться приходится  $L$  раз.

## Кросс-валидация: Random subsampling (Monte-Carlo cross-validation)

Выборка разбивается в случайной пропорции. Процедура повторяется несколько раз.



# Кросс-валидация: Выбор лучшей модели

Не переобученный алгоритм должен показывать одинаковую эффективность на каждой части.



Стохастический градиентный спуск - оптимизационный алгоритм, при котором градиент оптимизируемой функции считается на каждой итерации от случайно выбранного элемента выборки.

## Вспомним про GD: обозначения

- $\mathbf{X} \in \mathbb{R}^{n \times p}$  — матрица признаков;
- $y^* : \mathbf{X} \rightarrow \mathbf{y}$  — целевая зависимость, известная на  $\mathbf{X}^I$ ;
- $\mathbf{y} = y^*(\mathbf{X}) \in A^n$  — вектор классовой принадлежности;
- $\mathbf{X}^I$  — обучающая выборка, состоящая из пар  $(x_i, y_i)_{i=1}^I$ ;
- $a(x, \omega)$  — семейство алгоритмов с параметром вектора весов  $\omega$ ;
- $\mathcal{L}(a, y)$  — функция потерь, выбранная из класса  $C_1$ .

## Вспомним про GD: постановка задачи

Найдем алгоритм  $a(x, \omega)$ , аппроксимирующий зависимость  $y^*$ .

Например, в случае линейного классификатора искомый алгоритм имеет вид:

$$a(x, \omega) = \phi\left(\sum_{j=1}^n \omega_j x^j - \omega_0\right),$$

где  $\phi(z)$  - функция активации.

*Решаемая задача оптимизации:*

$$Q(\omega) = \sum_{i=1}^I \mathfrak{L}(a(x_i, \omega), y_i) \rightarrow \min_{\omega}$$

*Обновление весов:*

$$\omega^{(t+1)} = \omega^{(t)} - h \nabla Q(\omega^{(t)})$$

# Вспомним про GD: Проблемы

Проблема GD — чтобы определить новое приближение вектора весов необходимо вычислить градиент от каждого элемента выборки, что может сильно замедлять алгоритм. Идея ускорения заключается в использовании либо одного элемента (SGD), либо некоторой подвыборки (Batch GD) для получения нового приближения весов.

# Ускоряем GD - SGD

*Веса, при подходе SGD, обновляются следующим образом:*

$$\omega^{(t+1)} = \omega^{(t)} - h \nabla \mathcal{L}(a(x_i, \omega^{(t)}), y_t),$$

где  $i$  — случайно выбранный индекс.

Т.к. направление изменения  $\omega$  будет определяться за  $O(1)$ , подсчет  $Q$  на каждом шаге будет слишком дорогостоящим. Для ускорения оценки, будем использовать одну из рекуррентных формул:

- Среднее арифметическое:

$$\overline{Q}_m = \frac{1}{m} \mathcal{L}(a(x_m, \omega), y_m) + (1 - \frac{1}{m}) \overline{Q}_{m-1};$$

- Экспоненциальное скользящее среднее:

$$\overline{Q}_m = \lambda \mathcal{L}(a(x_m, \omega), y_m) + (1 - \lambda) \overline{Q}_{m-1}, \text{ где } \lambda \text{ — темп забывания "предыстории" ряда.}$$

## SGD: А что с инициализацией весов?

Существует несколько способов:

- $\omega = 0$ ;
- $\omega_j = \text{random}(-\frac{1}{2n}, \frac{1}{2n})$ ;
- $\omega_j = \frac{\langle y, x_j \rangle}{\langle x_j, x_j \rangle}$ , если признаки независимы, функция активации линейна, а функция потерь квадратична.

## SGD: А что со сходимостью?

Установлено, что если скорость обучения убывает при увеличении числа итераций SGD сходится почти наверняка к глобальному минимуму в случае выпуклой или псевдовыпуклой функции  $Q(\omega)$ , в противном случае метод сходится почти наверняка к локальному минимуму.

# SGD: Достоинства и недостатки

## Достоинства:

- Легко реализуется;
- Функция потерь и семейство алгоритмов могут быть любыми (если функция потерь не дифференцируема, ее можно аппроксимировать дифференцируемой);
- Легко добавить регуляризацию;
- Возможно потоковое обучение;
- Подходит для задач с большими данными, иногда можно получить решение даже не обработав всю выборку.

## Недостатки:

- Нет универсального набора эвристик, их нужно выбирать для конкретной задачи отдельно;
- На практике остаются проблемы с локальными экстремумами.



## Модификации (или о чем можно почитать на досуге)

- Не выраженные явно изменения (ISGD) — градиент пересчитывается на следующей итерации;
- Метод импульса — запоминает  $\Delta w$  на каждой итерации и определяет следующее изменение в виде линейной комбинации градиента и предыдущего изменения;
- Усреднённый стохастический градиентный спуск;
- AdaGrad — модификация SGD с отдельной для каждого параметра скоростью обучения;
- RMSProp — это метод, в котором скорость обучения настраивается для каждого параметра. Идея заключается в делении скорости обучения для весов на скользящие средние значения недавних градиентов для этого веса;
- Adam — это обновление оптимизатора RMSProp. Здесь используются скользящие средние как градиентов, так и вторых моментов градиентов.
- Kalman-based SGD.