Обучение с учителем. Классификация. Дискриминантный анализ.

Е. Ларин, Ф. Ежов, И. Кононыхин

1 Обучение с учителем

Рассмотрим задачу обучения с учителем, частным случаем которой являются задачи классификации и регрессии.

Алгоритм в общем виде имеет вид:

- *Вход*: $\boldsymbol{X}$ — выборка $\boldsymbol{\xi}$, случайной величины признаков, $\boldsymbol{y}$ — выборка η, случайной величны «ответов» (принадлежность к классу для классификации, либо значение функции для регрессии). Предполагаем, что существует неизвестное отображение $y^* : \boldsymbol{\xi} \to \eta$ (гипотеза непрерывности или компактности)
- *Задача*: По $\boldsymbol{X}$ и $\boldsymbol{y}$ найти такое отображение $\hat{y}^* : \boldsymbol{\xi} \to \eta$, которое приблизит отображение y^*.
- *Оценка*: Функция потерь $\mathfrak{L}(y^*(x), \hat{y}^*(x))$. Здесь x — реализация $\boldsymbol{\xi}$

1.1 Отступы

Введем понятие отступов. Пусть $\Phi(\boldsymbol{x}, \beta) = sign(g(\boldsymbol{x}, \beta))$ — разделяющий классификатор. Тогда $g(\boldsymbol{x}, \beta)$ — разделяющая функция, а $g(\boldsymbol{x}, \beta) = 0$ — уравнение разделяющей поверхности. Отступом для объекта x_i будем называть значение $M_i(\beta)$, такое что $M_i(\beta) = g(\boldsymbol{x}_i, \beta)y_i$. Если $M_i(\beta) < 0$, тогда алгоритм ошибается на $\boldsymbol{x}_i$, иначе алгоритм классифицировал объект верно. Чем больше значение $|M_i(\beta)|$, тем больше уверенность в правильности или неправильности классификации объекта.

2 Классификация

Перейдём к задаче классификации. Как и в задаче регрессии, данные должны происходить из некоторой генеральной совокупности.

Будем рассматривать выборку признаков и ответов 1

$$\boldsymbol{X} \in \mathbb{R}^{n \times p}, \quad \mathbf{y} \in \mathbb{A}^n. \tag{1}$$

Отметим, что множество $\mathbb{A}^n$ не является непрерывным. Размерность этого множества $k \times n$, где k — количество возможных классов.

Для обоснования применения методов классификации используется **гипотеза компактности**:

«Близкие» объекты, как правило, принадлежат одному классу.

2.1 Классификация: вероятностная постановка

Поставим задачу классификации в терминах генеральных случайный величин.

Дано:

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

Предположение об их зависимости можно записать в виде 2.

$$\eta = \Phi(\boldsymbol{\xi}) \tag{2}$$

Задача: найти Φ

При переходе к выборкам, случайная величина признаков $\boldsymbol{\xi}$ заменяется на матрицу наблюдений $\boldsymbol{X}$, а случайная величина ответов η — на вектор классовой принадлежности $\boldsymbol{y}$.

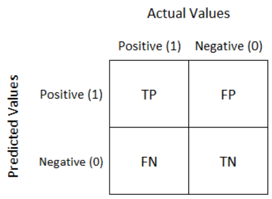
Предположение принимает вид 3.

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

2.2 Классификация: оценка качества

На основе матрицы ошибок 2.2 есть большое количество разных метрик. Приведём некоторые из них:

- $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$,
- $recall = \frac{TP}{TP+FP}$,
- $precision = \frac{TP+TN}{TP+FN}$,
- $F_\beta = (1-\beta^2)\frac{precision \times recall}{(\beta^2 \times precision)+recall}$,
- $ROC–AUC$



2.3 Классификация: этапы обучения модели

- Выбор модели (класс рассматриваемых Φ из 3). Здесь будут рассмотрены модели LDA и QDA.
- Выбор функции потерь. Чаще всего это $\sum_{i=1}^{n}(y_i \neq \widehat{y_i})$.
- Выбор метода обучения. Выбор способа подбора параметров для минимизации функции потерь на обучающем множестве.
- Выбор метода проверки. Выбор оценки качества модели, например, с помощью метрик из раздела 2.2.

2.4 Классификация: общий подход к решению

Как построить функционал Φ?

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

Таким образом, решение о принадлежности классу принимается при обнаружении классифицирующей функции с наибольшим значением:

$$\Phi(\boldsymbol{x}) = \arg\max_i (f_i(\boldsymbol{x})). \tag{4}$$

3 Дискриминантный анализ

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

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

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

Если известны априорные вероятности получения i-го класса (π_i), применим формулу Байеса

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

Отбросим знаменатель

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

3.1 LDA

Предположим, что искомые классы имеют многомерное нормальное распределение с равными дисперсиями.

Запишем это в виде формулы:

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

Построим классифицирующие функции:

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

Немного упростим (подробнее было изложено в одном из предыдущих курсов):

$$h_i(\boldsymbol{x}) = -0.5\boldsymbol{\mu}_i\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i^{\mathtt{T}} + \boldsymbol{\mu}_i\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \log \pi_i \tag{5}$$

Функции 5 применяются при классификации данным методом.

3.2 QDA

Предположим, что искомые классы имеют многомерное нормальное распределение с различными дисперсиями.

Запишем это в виде формулы:

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

Построим классифицирующие функции:

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

Немного упростим (подробнее было изложено в одном из предыдущих курсов):

$$g_i(\boldsymbol{x}) = -0.5(\boldsymbol{x} - \boldsymbol{\mu}_i)\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i)^{\mathtt{T}} - 0.5 \log|\boldsymbol{\Sigma}_i| + \log \pi_i \tag{6}$$

Функции 6 применяются при классификации данным методом.

4 Логистическая регрессия

Дана выборка $\boldsymbol{X} \in \mathbb{R}^{n \times p}$. Поставим задачу классификации объектов $\boldsymbol{x}_i$ на два класса $\{-1, 1\}$.

4.1 Подход через минимизацию функции потерь

Воспользуемся линейной моделью для решение задачи классификации:

$$\hat{y} = \Phi(\boldsymbol{x}, \beta) = sign\langle \boldsymbol{x}, \beta \rangle \tag{7}$$

Функция потерь в данном случае будет:

$$\mathfrak{L}(\hat{y}, y) = [\hat{y}y < 0] = [\langle \boldsymbol{x}, \beta \rangle y < 0] \tag{8}$$

Задача минимизация записывается следующим образом:

$$Q(\beta) = \sum_{i=1}^{n} [\langle \boldsymbol{x}_i, \beta \rangle y_i < 0] \to \min_{\beta} \tag{9}$$

Перейдем от задачи 9 к другой, заменив в 8 функцию потерь на другую ($\hat{\mathfrak{L}}$) мажорирующую ее, получим:

$$\mathfrak{L}(\hat{y}, y) = [\langle \boldsymbol{x}, \beta \rangle y < 0] \leqslant \hat{\mathfrak{L}}(\langle \boldsymbol{x}, \beta \rangle y) \tag{10}$$

$$Q(\beta) = \sum_{i=1}^{n} [\langle \boldsymbol{x}_i, \beta \rangle y_i < 0] \leqslant \sum_{i=1}^{n} \hat{\mathfrak{L}}(\langle \boldsymbol{x}_i, \beta \rangle y_i) \to \min_{\beta} \tag{11}$$

Видно, что аргумент функции потерь $\hat{\mathfrak{L}}$, это ничто иное как отступ M_i для объекта в $\boldsymbol{x}_i$ введенный в главе 1.1.

Чтобы линейная модель в задаче 11 стала логистической регрессией, достаточно взять функцию потерь $\hat{\mathfrak{L}}(M_i) = \log(1 + e^{-M_i})$. Такая функция потерь называется логарифмической или логистической.

Тогда задача минимизации 11 запишется так:

$$Q(\beta) = \sum_{i=1}^{n} \log(1 + e^{-\langle \boldsymbol{x}_i, \beta \rangle y_i}) \to \min_{\beta} \tag{12}$$

Методы решение задачи минимизации:

- метод стохастического градиента
- метод Ньютона-Рафсона

4.2 Вероятностный подход

Пусть σ — сигмоида и:

- $P(y_i = 1|\boldsymbol{x}_i; \beta) = \sigma_\beta(M_i)$, вероятность объекта $\boldsymbol{x}_i$ принадлежать классу 1.
- $P(y_i = -1|\boldsymbol{x}_i; \beta) = 1 - \sigma_\beta(M_i)$, вероятность объекта $\boldsymbol{x}_i$ принадлежать классу -1.

Тогда $P(y_i|\boldsymbol{x}_i; \beta) = \sigma_\beta(M) = \dfrac{1}{1 + e^{-\langle x, \beta \rangle y}}$
Запишем логарифм функции правдоподобия:

$$Q(\boldsymbol{X}, \beta) = \log \boldsymbol{L}(\beta) = \log \prod_{i=1}^{n} P(y_i|\boldsymbol{x}_i; \beta) = -\sum_{i=1}^{n} \log(1 + e^{-\langle \boldsymbol{x}_i, \beta \rangle y_i}) \to \max_\beta \tag{13}$$

Заметим, что задача 13 совпадает с задачей 12.

4.3 Регуляризация

В задачу минимизации можно добавить параметры β, тогда мы получим регуляризацию:

- **L1 регуляризация:**

$$Q(\beta) = \sum_{i=1}^{n} \log(1 + e^{-\langle \boldsymbol{x}_i, \beta \rangle y_i}) + \lambda \sum_{j=1}^{p} |\beta_j| \to \min_\beta$$

- **L2 регуляризация:**

$$Q(\beta) = \sum_{i=1}^{n} \log(1 + e^{-\langle \boldsymbol{x}_i, \beta \rangle y_i}) + \frac{\lambda}{2} \sum_{j=1}^{p} \beta_j^2 \to \min_\beta$$

Параметр λ отвечает за «силу» регуляризации и может быть подобран с помощью кросс-валидации.

4.4 Многоклассовая логистическая регрессия

Пусть $\boldsymbol{Y} = \{1, \dots, K\}$, тогда запишем линейный классифкатор следующим образом:

$$\hat{y} = \arg\max_{y \in \boldsymbol{Y}} \langle \boldsymbol{x}_i, \beta_y \rangle, \boldsymbol{x}, \beta \in \mathbb{R}^p \tag{14}$$

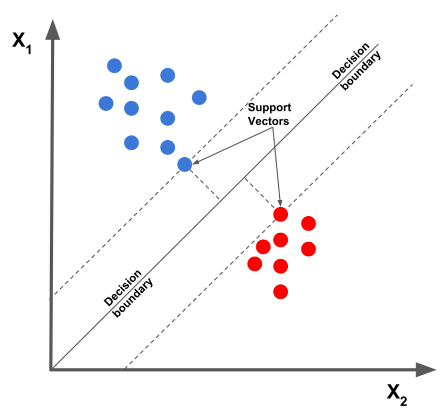
Вероятность того, что объект x_i относится к классу j:

$$P(y = j | x_i; \beta) = \frac{\exp(\langle \boldsymbol{x}_i, \beta_y \rangle)}{\sum\limits_{z \in \boldsymbol{Y}} \exp(\langle \boldsymbol{x}_i, \beta_z \rangle)} = \frac{e^{\beta_j^T \boldsymbol{x}}}{\sum\limits_{k=1}^{K} e^{\beta_k^T \boldsymbol{x}}} \tag{15}$$

Задача максимизации для многоклассового случая получается заменой функции вероятности в 13 на функцию вероятности в 15.

5 Support Vector Machine

Пусть дана выборка $\{x_i, y_i\}_{i=1}^n$, $\boldsymbol{x}_i \in \mathbb{R}^p$, $y_i \in \{-1, 1\}$. Поставим задачу построить классифицирующие правило. Также введем критерий оптимальности: хотим найти такую разделяющую гиперплоскость, что растояние между еще двумя параллельных данной и симметрично расположенных относительно нее имеют максимальное растояние.



5.1 Hard-margin SVM

Предположим что наша выборка линейно разделима. Тогда искомая гиперплоскость:

$$\boldsymbol{x}^T\beta - \beta_0 = 0; \beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R} \tag{16}$$

А классифицирующие правило строится следующим образом:

$$\begin{aligned} g(x) &= \boldsymbol{x}^T\beta - \beta_0, \\ \boldsymbol{\Phi}(x) &= sign(g(x)). \end{aligned} \tag{17}$$

Уравнения пары гиперплоскостей с точностью до нормировки:

$$\begin{aligned} \boldsymbol{x}^T\beta - \beta_0 &= -1, \\ \boldsymbol{x}^T\beta - \beta_0 &= 1. \end{aligned} \tag{18}$$

А расстояние между ними равно: $\dfrac{2}{||\beta||}$

Принадлежность точек обучающей выборки описывается как отступ (19) от разделяющей гипер плоскости. Отступ всегда больше или равен 1, так как наша выборка линейно разделима.

$$M_i = (\boldsymbol{x}^T\beta - \beta_0)y_i \geqslant 1 \tag{19}$$

Задача построения разделяющей гиперплоскости сводится к задаче квадратичного программирования с линейными ограничениями:

$$\begin{cases} \frac{1}{2}||\beta||_2^2 \to \min\limits_{\beta,\beta_0} \\ M_i \geqslant 1 \end{cases} \tag{20}$$

5.2 Soft-margin SVM

Передем к случаю когда выборка линейно не разделима, тогда задача 20 не имеет решения, так как система неравенств несовместна. Перепишем задачу 20, добавим небольшие изменения.

$$\begin{cases} \frac{1}{2}||\beta||_2^2 + C\sum\xi_i \to \min\limits_{\beta,\beta_0,\xi} \\ M_i \geqslant 1 - \xi_i \\ \xi_i \geqslant 0 \end{cases} \tag{21}$$

Теперь мы разрешили объектам нарушать границы разделяющей гиперплоскости. Так же мы добавили сумму этих нарушений ξ с весом C в минимизируемый функцианал, чтобы подобрать такую гиперплоскость в которой объектов «нарушителей» будет наименьшим.

Применим условия Каруши-Куна-Таккера к задаче 21. Запишем функцию Лагранжа:

$$L(\beta, \beta_0, \xi; \lambda, \eta) = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{n}(M_i - 1) - \sum_{i=1}^{n}\xi_i(\lambda_i + \eta_i - C) \tag{22}$$

λ_i — переменные, двойственные к ограничениям $M_i \geqslant 1 - \xi_i$,
η_i — переменные, двойственные к ограничениям $\xi_i \geqslant 0$.

Также запишем условия для первых производных функции Лагранжа, исходные ограничения, ограничения к двойственным переменным и условиями дополняющей нежесткости.

$$\begin{cases} \frac{\partial L}{\partial \beta} = 0, \frac{\partial L}{\partial \beta_0} = 0, \frac{\partial L}{\partial \xi} = 0; \\ \xi_i \geqslant 0, \lambda_i \geqslant 0, \eta_i \geqslant 0, i = 1, \dots, n; \\ \lambda_i = 0 \text{ либо } M_i = 1 - \xi_i, i = 1, \dots, n; \\ \eta_i = 0 \text{ либо } \xi_i = 0, i = 1, \dots, n; \end{cases} \tag{23}$$

Распишем производные поподробнее, получаем необходимые условия седловой точки функции Лагранжа:

$$\begin{aligned} \frac{\partial L}{\partial \beta} = \beta - \sum_{i=1}^{n}\lambda_i y_i \boldsymbol{x}_i = 0 \Longrightarrow \beta = \sum_{i=1}^{n}\lambda_i y_i \boldsymbol{x}_i \\ \frac{\partial L}{\partial \beta_0} = -\sum_{i=1}^{n}\lambda_i y_i = 0 \Longrightarrow \sum_{i=1}^{n}\lambda_i y_i = 0 \\ \frac{\partial L}{\partial \xi_i} = \lambda_i - \eta_i - C = 0 \Longrightarrow \lambda_i + \eta_i = C, i = 1, \dots, n; \end{aligned} \tag{24}$$

Из этих условий можем записать типизацию объектов:

1. $\lambda_i = 0; \eta_i = C; \xi = 0; M_i \geqslant 1$ - неинформативные объекты.
2. $0 < \lambda_i < C; 0 < \eta_i < C; \xi = 0; M_i = 1$ - опорные граничные объекты.
3. $\lambda_i = C; \eta_i = 0; \xi > 0; M_i < 1$ - опорные-нарушители.

Из формул 24 видно, что объекты отступ M_i которых больше 1 не участвуют в построении разделяющей гиперплоскости.

Отсюда можно дать определение опорным объектам. Объект x_i называется опорным, если $\lambda_i \neq 0$.

Избавимся от всех переменных, кроме двойственных в 22, получим:

$$\begin{cases} -L(\boldsymbol{\lambda}) = -\sum\limits_{i=1}^{n} \lambda_i + \frac{1}{2} \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} \lambda_i \lambda_j y_i y_j (x_i, x_j) \to \min\limits_{\boldsymbol{\lambda}}; \\ 0 \leqslant \lambda_i \leqslant C, i = 1, \ldots, n; \\ \sum\limits_{i=1}^{n} \lambda_i y_i = 0. \end{cases} \tag{25}$$

Решение прямой задачи выражается через решение двойственной:

$$\begin{cases} \beta = \sum\limits_{i=1}^{n} \lambda_i y_i x_i; \\ \beta_0 = \langle \beta, \boldsymbol{x}_i \rangle - y_i, \text{ для любого } i : \lambda_i > 0, M_i = 1. \end{cases} \tag{26}$$

Тогда

Линейный классификатор с признаками $f_i(x) = \langle x, x' \rangle$:

$$\boldsymbol{\Phi}(x) = sign(\sum_{i=1}^{n} \lambda_i y_i \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle - \beta_0) \tag{27}$$

5.3 Минимизация эмпирического риска

Как и в случае задачи минимизации 11 (логистической регрессии), можем записать задачу минимизации в общей форме:

$$Q(\beta) = \sum_{i=1}^{n} \hat{\mathfrak{L}}(M_i) \to \min_{\beta}$$

Чтобы получить SVM достаточно взять $\hat{\mathfrak{L}}(M) = \max(0, 1 - M)$. В случае SVM $M_i = g(x_i, \beta) y_i = (\boldsymbol{x}_i^T \beta - \beta_0) y_i$.

Подставляя функцию потерь и отступ, получаем задачу минимизации:

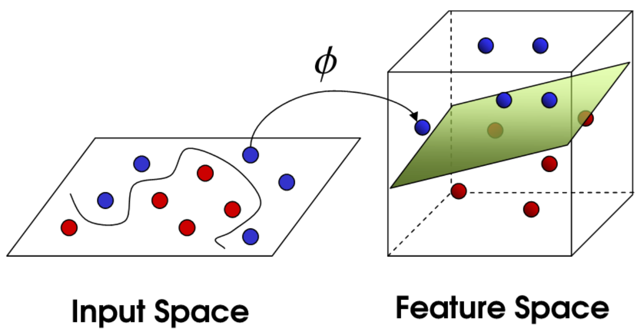
$$Q(\beta) = \sum_{i=1}^{n} \max(0, 1 - (\boldsymbol{x}_i^T \beta - \beta_0) y_i) \to \min_{\beta} \tag{28}$$

5.4 Нелинейное обобщение

Можно заменить $\langle x, x' \rangle$ в 28 на нелинейную функцию $K(x, x')$.

Определение: Функция $K : X \times X \to \mathbb{R}$ — ядро, если $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$ при некотором $\phi : X \to H$, где H — гильбертово пространство.

Данное соображение позволяет применять SVM к данным в достаточной степени линейно разделимым в некотором гильбертовом пространстве (в том числе — бесконечномерном); в том числе — без предъявления в явном виде отображения из исходного пространства в данное.



Так как непосредственная проверка положительной определённости ядра представляет собой проблему, можно воспользоваться следующими операциями, приводящими к получению новых ядер:

- Скалярное произведение в векторном пространстве
- Положительная константа
- Произведение ядер: $K(u, v) = K_1(u, v)K_2(u, v)$
- Произведение отображений: $K(u, v) = \phi(u)\phi(v), \phi : x \to \mathbb{R}$
- Линейная комбинация с положительными коэффициентами: $K(u, v) = \alpha_1 K_1(u, v) + \alpha_2 K_2(u, v), \alpha_{1,2} > 0$
- Композиция ядра и отображения: $K(u, v) = K_1(\phi(u), \phi(v))$
- Степенной ряд:
$$f : \mathbb{R} \to \mathbb{R}$$

сходящийся степенной ряд с положительными коэффициентами, тогда
$$K(u, v) = f(K_1(u, v))$$
является ядром.

Часто используемые ядра:

- RBF (radial basis functions): $K(x, x') = e^{-\gamma||x-x'||_2^2}$
- Полиномиальное (степеней $\leqslant d : K(x, x') = (\langle x - x' \rangle + 1)^d$)

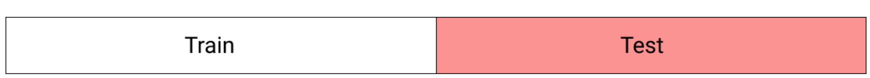
6 Кросс-валидация

Кросс-валидация (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. Среди видов перекрестной проверки встречаются следующие:

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

6.1 Кросс-валидация: Hold-Out Validation

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



После чего решается задача оптимизации:

$$HO(\mu, \boldsymbol{T}^t, \boldsymbol{T}^{N-t}) = \mathfrak{L}(\mu(\boldsymbol{T}^t), \boldsymbol{T}^{N-t}) \to min$$

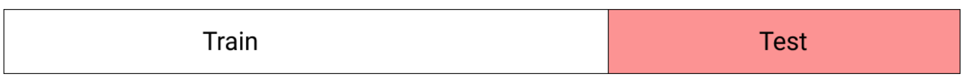
.

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.

6.2 Кросс-валидация: Complite cross-validatio

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



После чего решается задача оптимизации:

$$CVV_t = \frac{1}{C_N^{N-t}} \sum_{\boldsymbol{T}^N = \boldsymbol{T}^t \cup \boldsymbol{T}^{N-t}} \mathfrak{L}(\mu(\boldsymbol{T}^t), \boldsymbol{T}^{N-t}) \to min$$

.

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

6.3 Кросс-валидация: k-fold Cross-Validation

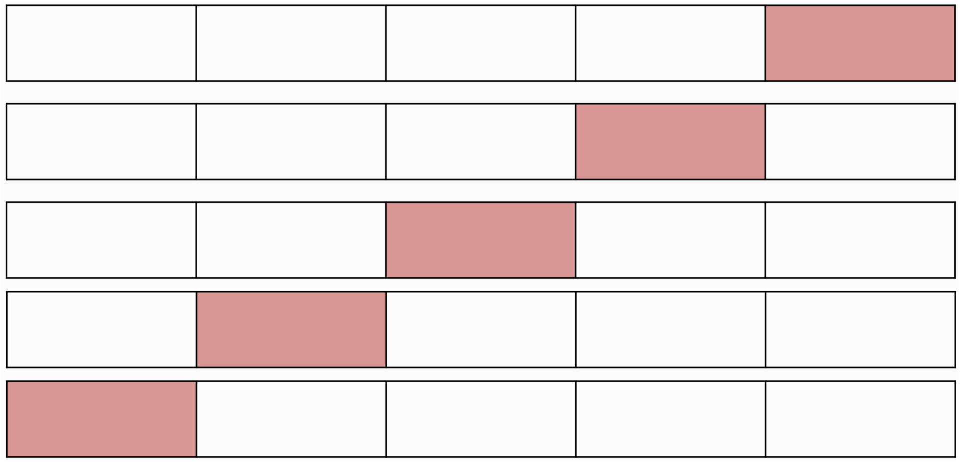
- $\boldsymbol{T}^l$ разбивается на $\boldsymbol{T_1} \cup \dots \cup \boldsymbol{T_k}, |\boldsymbol{T_i}| \approx \frac{1}{k}$ частей;
- Производится k итераций:
  - Модель обучается на $k-1$ части обучающей выборки;
  - Модель тестируется на части обучающей выборки, которая не участвовала в обучении.

Каждая из k частей единожды используется для тестирования.

После чего решается задача оптимизации:

$$CV_k = \frac{1}{k} \sum_{i=1}^{k} \mathfrak{L}(\mu(\boldsymbol{T}^N \setminus \boldsymbol{T}_i), \boldsymbol{T}_i) \to min$$

.



6.4 Кросс-валидация: t×k-fold Cross Validation

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

Разбиение: $\boldsymbol{T}^N = \boldsymbol{T}_{(1,1)} \cup \dots \cup \boldsymbol{T}_{(k,1)} = \boldsymbol{T}_{(1,t)} \cup \dots \cup \boldsymbol{T}_{(k,t)}$, $|\boldsymbol{T}_{(i,j)}| \approx \frac{N}{k}$,
Задача оптимизации:

$$CV_{t\times k} = \frac{1}{tk} \sum_{j=1}^{t} \sum_{i=1}^{k} \mathfrak{L}(\mu(\boldsymbol{T}^N \setminus \boldsymbol{T}_{(i,j)}), \boldsymbol{T}_{(i,j)}) \to min$$

6.5 Кросс-валидация: Leave-One-Out

Выборка разбивается на $N-1$ и 1 объект N раз. $LOO = \frac{1}{N} \sum\limits_{i=1}^{N} \mathfrak{L}(\mu(\boldsymbol{T}^N \setminus p_i), p_i) \to min$, 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.

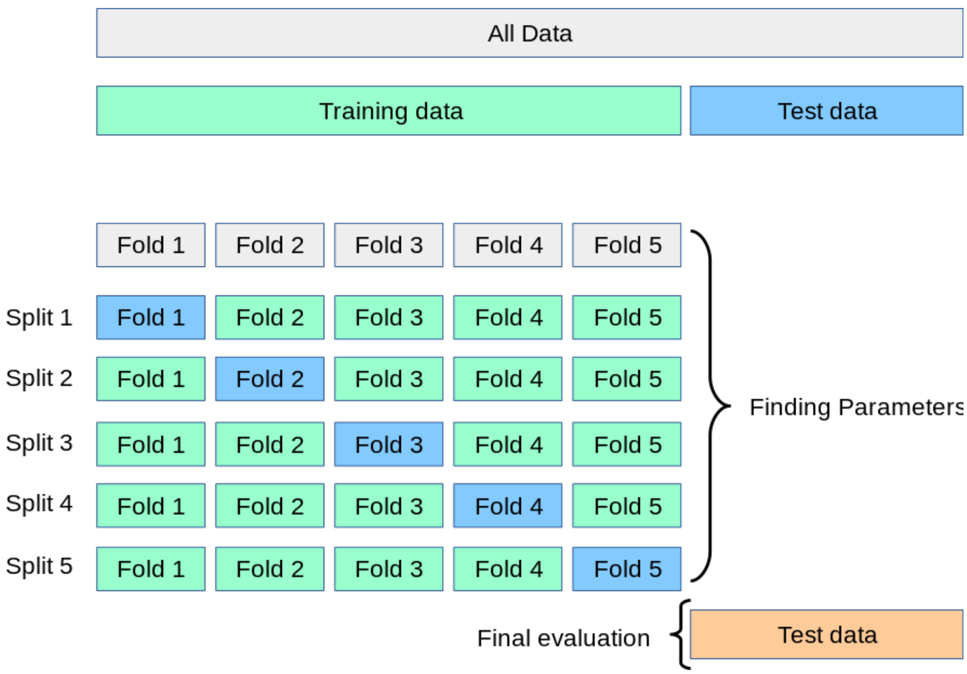
Недостатком LOO является большая ресурсоёмкость, так как обучаться приходится N раз.

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

Выборка разбивается в случайной пропорции. Процедура повторяется несколько раз.

6.7 Кросс-валидация: Выбор лучшей модели

Обычно, с помощью кросс-валидации находят лучшие параметры модели в рассматриваемой задаче. Весь набор данных делят на тренировочную и тестовую выборки, после чего эмпирически находят лучшие параметры средством перекрестной проверки на тренировочном наборе и оценивают эффективность модели на тестовом множестве.



Исходя из полученных результатов принимается окончательное решение - искать другие параметры или остановиться на текущей модели и ее конфигурации.

7 SGD

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

Введем дополнительные обозначения:

- $\boldsymbol{X}^l$ — обучающая выборка, состоящая из пар $(x_i, y_i)_{i=1}^l$;
- $a(x, \omega)$ — семейство алгоритмов с параметром вектора весов ω;

Так как стохастический градиентный спуск является модификацией всеми известного градиентного спуска, начнем с последнего.

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

Например, в случае линейного классификатора искомый алгоритм имеет вид:

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

где $\phi(z)$ - функция активации.

Решаемая задача оптимизации:

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

Обновление весов:

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

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.

Веса, при подходе SGD, обновляются следующим образом:

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

где i — случайно выбранный индекс.

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

- Среднее арифметическое: $\overline{Q}_m = \frac{1}{m} \mathfrak{L}(a(x_{i_m}, \omega^{(m)}), y_{i_m}) + (1 - \frac{1}{m}) \overline{Q}_{m-1}$;
- Экспоненциальное скользящее среднее: $\overline{Q}_m = \lambda \mathfrak{L}(a(x_{i_m}, \omega^{(m)}), y_{i_m}) + (1 - \lambda) \overline{Q}_{m-1}$, где λ — темп забывания "предыстории"ряда.

Так как алгоритм итеративный, нужно задать начальные значения весов оптимизируемой модели. Существует несколько способов инициализации весов, например:

- $\omega_0 = 0$;
- $\omega_0 = random(-\frac{1}{2n}, \frac{1}{2n})$;

Говоря о сходимости, 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.

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:

- Не выраженные явно изменения (ISGD) — градиент пересчитывается на следующей итерации;
- Метод импульса — запоминает $\Delta\omega$ 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.