

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:* \mathbf{X} — выборка ξ , случайной величины признаков, \mathbf{y} — выборка η , случайной величины «ответов» (принадлежность к классу для классификации, либо значение функции для регрессии). 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 $\mathfrak{L}(y^*(x), \hat{y}^*(x))$. Здесь x — реализация ξ

1.1 Отступы

Введем понятие отступов. Пусть $\Phi(x, \beta) = \text{sign}(g(x, \beta))$ — разделяющий классификатор. Тогда $g(x, \beta)$ — разделяющая функция, а $g(x, \beta) = 0$ — уравнение разделяющей поверхности. Отступом для объекта x_i будем называть значение $M_i(\beta)$, такое что $M_i(\beta) = g(x_i, \beta)y_i$. Если $M_i(\beta) < 0$, тогда алгоритм ошибается на 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

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

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 случайных величин.

Дано:

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

Предположение об их зависимости можно записать в виде 2.

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

Задача: найти Φ

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

Предположение принимает вид 3.

$$y_i = \Phi(x_i), \quad i = 1, \dots, n \quad (3)$$

2.2 Классификация: оценка качества

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

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

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

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 $\sum_{i=1}^n (y_i \neq \hat{y}_i)$.
- 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 , $i = 1, \dots, K$. Каждая функция $f_i(x)$ показывает меру принадлежности x классу i .

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

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

3 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.

Если известны априорные вероятности получения i -го класса (π_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

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

3.1 LDA

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

Запишем это в виде формулы:

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

Построим классифицирующие функции:

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

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

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

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:

$$P(\xi|\eta = A_i) = N(\mu_i, \Sigma_i)$$

Построим классифицирующие функции:

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

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

$$g_i(\mathbf{x}) = -0.5(\mathbf{x} - \mu_i) \Sigma_i^{-1} (\mathbf{x} - \mu_i)^T - 0.5 \log |\Sigma_i| + \log \pi_i \quad (6)$$

Функции 6 применяются при классификации данным методом.

4 Логистическая регрессия

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

4.1 Подход через минимизацию функции потерь

Воспользуемся линейной моделью для решение задачи классификации:

$$\hat{y} = \Phi(\mathbf{x}, \beta) = \text{sign}\langle \mathbf{x}, \beta \rangle \quad (7)$$

Функция потерь в данном случае будет:

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

Задача минимизация записывается следующим образом:

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

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

$$\mathfrak{L}(\hat{y}, y) = [\langle \mathbf{x}, \beta \rangle y < 0] \leq \hat{\mathfrak{L}}(\langle \mathbf{x}, \beta \rangle y) \quad (10)$$

$$Q(\beta) = \sum_{i=1}^n [\langle \mathbf{x}_i, \beta \rangle y_i < 0] \leq \sum_{i=1}^n \hat{\mathfrak{L}}(\langle \mathbf{x}_i, \beta \rangle y_i) \rightarrow \min_{\beta} \quad (11)$$

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

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

Тогда задача минимизации 11 запишется так:

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

Методы решение задачи минимизации:

- метод стохастического градиента
- метод Ньютона-Рафсона

4.2 Вероятностный подход

Пусть σ — сигмоида и:

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

$$\text{Тогда } P(y_i | \mathbf{x}_i; \beta) = \sigma_{\beta}(M) = \frac{1}{1 + e^{-\langle \mathbf{x}, \beta \rangle y}}$$

Запишем логарифм функции правдоподобия:

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

Заметим, что задача 13 совпадает с задачей 12.

4.3 Регуляризация

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

- **L1 регуляризация:**

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

- **L2 регуляризация:**

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

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

4.4 Многоклассовая логистическая регрессия

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

$$\hat{y} = \arg \max_{y \in \mathbf{Y}} \langle x_i, \beta_y \rangle, x, \beta \in \mathbb{R}^p \quad (14)$$

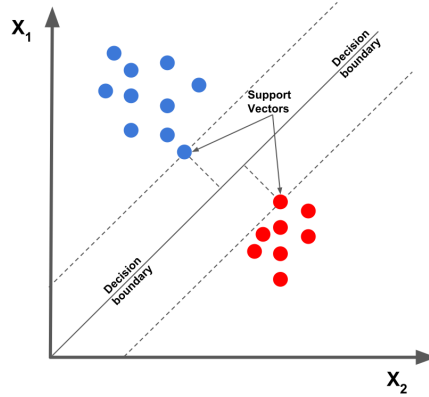
Вероятность того, что объект x_i относится к классу j :

$$P(y = j | x_i; \beta) = \frac{\exp(\langle x_i, \beta_j \rangle)}{\sum_{z \in \mathbf{Y}} \exp(\langle x_i, \beta_z \rangle)} = \frac{e^{\beta_j^T x}}{\sum_{k=1}^K e^{\beta_k^T x}} \quad (15)$$

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

5 Support Vector Machine

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



5.1 Hard-margin SVM

Предположим что наша выборка линейно разделима. Тогда искомая гиперплоскость:

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

А классифицирующее правило строится следующим образом:

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

Уравнения пары гиперплоскостей с точностью до нормировки:

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

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

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

$$M_i = (\mathbf{x}^T \boldsymbol{\beta} - \beta_0) y_i \geq 1 \quad (19)$$

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

$$\begin{cases} \frac{1}{2} \|\beta\|_2^2 \rightarrow \min_{\beta, \beta_0} \\ M_i \geq 1 \end{cases} \quad (20)$$

5.2 Soft-margin SVM

Перед тем как перейти к случаю, когда выборка линейно не разделима, тогда задача 20 не имеет решения, так как система неравенств несовместна. Перепишем задачу 20, добавив небольшие изменения.

$$\begin{cases} \frac{1}{2} \|\beta\|_2^2 + C \sum \xi_i \rightarrow \min_{\beta, \beta_0, \xi} \\ M_i \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \quad (21)$$

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

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

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

λ_i — переменные, двойственные к ограничениям $M_i \geq 1 - \xi_i$,

η_i — переменные, двойственные к ограничениям $\xi_i \geq 0$.

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

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

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

$$\begin{aligned}\frac{\partial L}{\partial \beta} &= \beta - \sum_{i=1}^n \lambda_i y_i x_i = 0 \implies \beta = \sum_{i=1}^n \lambda_i y_i x_i \\ \frac{\partial L}{\partial \beta_0} &= - \sum_{i=1}^n \lambda_i y_i = 0 \implies \sum_{i=1}^n \lambda_i y_i = 0 \\ \frac{\partial L}{\partial \xi_i} &= \lambda_i - \eta_i - C = 0 \implies \lambda_i + \eta_i = C, i = 1, \dots, n;\end{aligned}\quad (24)$$

Из этих условий можем записать типизацию объектов:

1. $\lambda_i = 0; \eta_i = C; \xi = 0; M_i \geq 1$ - 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:

$$\begin{cases} -L(\lambda) = - \sum_{i=1}^n \lambda_i + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j (x_i, x_j) \rightarrow \min_{\lambda}; \\ 0 \leq \lambda_i \leq C, i = 1, \dots, n; \\ \sum_{i=1}^n \lambda_i y_i = 0. \end{cases} \quad (25)$$

Решение прямой задачи выражается через решение двойственной:

$$\begin{cases} \beta = \sum_{i=1}^n \lambda_i y_i x_i; \\ \beta_0 = \langle \beta, x_i \rangle - y_i, \text{ для любого } i : \lambda_i > 0, M_i = 1. \end{cases} \quad (26)$$

Тогда

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

$$\Phi(x) = \text{sign} \left(\sum_{i=1}^n \lambda_i y_i \langle x, x_i \rangle - \beta_0 \right) \quad (27)$$

5.3 Минимизация эмпирического риска

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

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

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

Подставляя функцию потерь и отступ, получаем задачу минимизации:

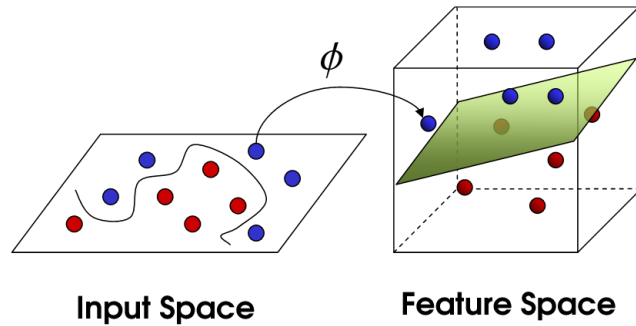
$$Q(\beta) = \sum_{i=1}^n \max(0, 1 - (x_i^T \beta - \beta_0) y_i) \rightarrow \min_{\beta} \quad (28)$$

5.4 Нелинейное обобщение

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

Определение: Функция $K : X \times X \rightarrow \mathbb{R}$ — ядро, если $K(x, x') = \langle \phi(x), \phi(x') \rangle$ при некотором $\phi : X \rightarrow H$, 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: $K(u, v) = \phi(u)\phi(v), \phi : x \rightarrow \mathbb{R}$
- 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:

$$f : \mathbb{R} \rightarrow \mathbb{R}$$

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 $\leq d$): $K(x, x') = (\langle x - x' \rangle + 1)^d$

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 $\mathcal{T} = \mathcal{T}^t \cup \mathcal{T}^{N-t}$.

Train	Test
-------	------

После чего решается задача оптимизации:

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

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.

6.2 Кросс-валидация: Complete cross-validation

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

Train	Test
-------	------

После чего решается задача оптимизации:

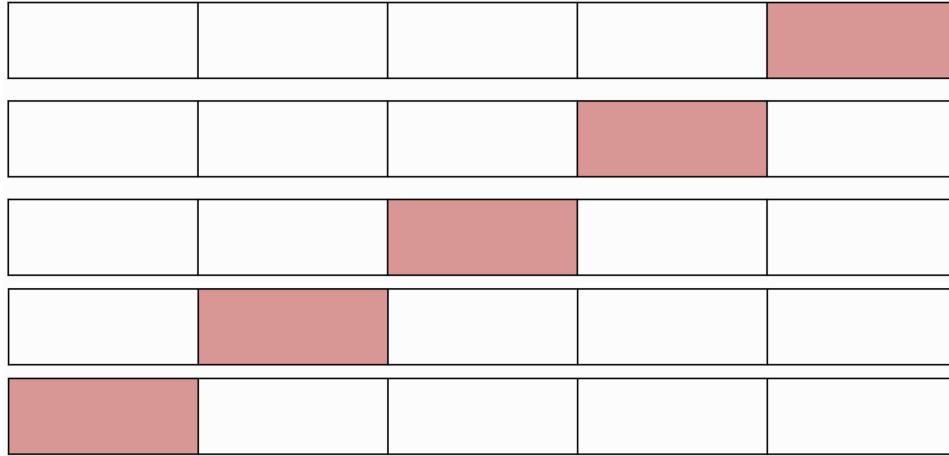
$$CVV_t = \frac{1}{C_N^{N-t}} \sum_{\mathcal{T}^N = \mathcal{T}^t \cup \mathcal{T}^{N-t}} \mathcal{L}(\mu(\mathcal{T}^t), \mathcal{T}^{N-t}) \rightarrow \min$$

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

- T^I разбивается на $T_1 \cup \dots \cup T_k, |T_i| \approx \frac{1}{k}$ 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:

$$CV_k = \frac{1}{k} \sum_{i=1}^k \mathcal{L}(\mu(T^N \setminus T_i), T_i) \rightarrow \min$$

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

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

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

$$CV_{t \times k} = \frac{1}{tk} \sum_{j=1}^t \sum_{i=1}^k \mathcal{L}(\mu(T^N \setminus T_{(i,j)}), T_{(i,j)}) \rightarrow \min$$

6.5 Кросс-валидация: Leave-One-Out

Выборка разбивается на $N - 1$ и 1 объект N раз. $LOO = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(\mu(T^N \setminus p_i), p_i) \rightarrow \min$, 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.

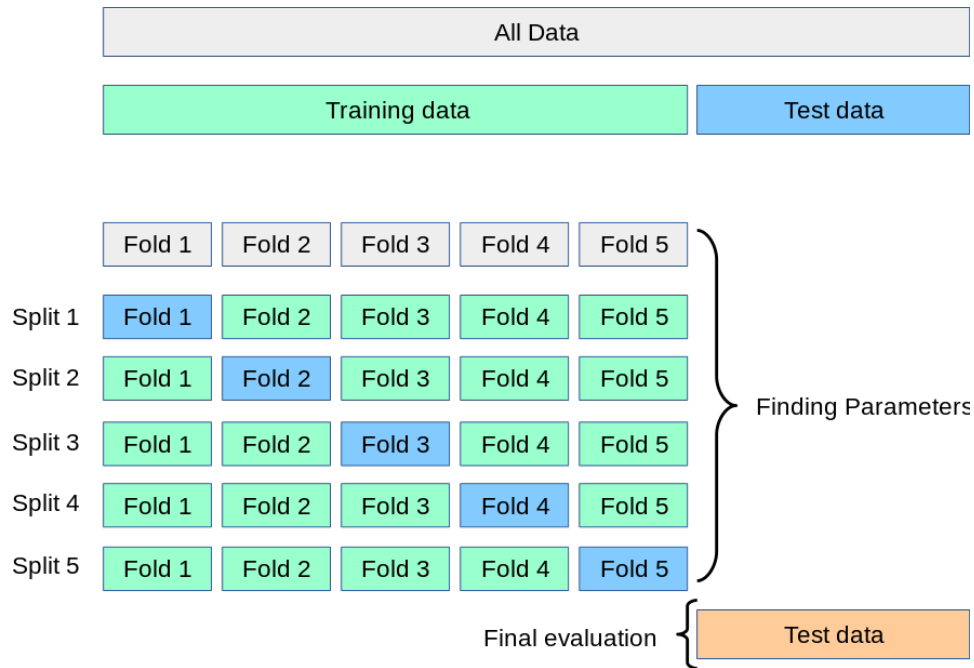
Исходя из полученных результатов принимается окончательное решение - искать другие параметры или остановиться на текущей модели и ее конфигурации.

7 SGD

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

Введем дополнительные обозначения:

- \mathbf{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:

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

где $\phi(z)$ - функция активации.

Решаемая задача оптимизации:

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

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:

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

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

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

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

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.