№	Признак1: Частота кашля (0-1), где 0 – нет каля, 1 – не прекращается	Признак2: Температура	…	ПризнакK: …	Целевая переменная: Уверенность в том, что человек болен гриппом (0-100)
1	0,2	38,2	…	…	35
2	0,1	39,1	…	…	45
…	…	…	…	…	…
N	0,5	37,3	…	…	18

Регрессия – предсказание значения одной или нескольких целевых переменных на основе набора признаков.

Статистика наблюдений 1 … N – обучающая выборка. В дальнейшем предсказание делается для новых признаков, не представленных в обучающей выборке.

Линейная регрессия:

$$t = y(x, w) + \varepsilon$$

$\varepsilon -$ случайная величина с нормальным распределением

$$p(t|x, w, \sigma^2) = N(t|y(x, w), \sigma^2)$$

$$y(x, w) = w_0 + w_1 x_1 + \ldots + w_K x_K$$

$K -$ число признаков

$(x_1 \ldots x_K)$
$-$ один набор признаков (одна строка из таблицы или новый набор)

$t -$ желаемое значение целевой переменной

для заданного набора признаков

(из статистики)

$y -$ значение целевой переменной

полученное на основе работы модели

(вычисленное по формуле, в лучшем случае совпадает с t)

Линейная комбинация нелинейных функций:

$$y(x, w) = w_0 + \sum_{j=1}^{M} w_j f_j(\mathbf{x})$$

$$\mathbf{x} = (x_1, x_2, \dots, x_K)$$

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots., f_M(\mathbf{x}))$$

$$y(x, w) = \sum_{j=0}^{M} w_j f_j(\mathbf{x})$$

$$\mathbf{x} = (1, x_1, x_2, \dots, x_K)$$

$$\mathbf{w} = (w_0, w_1, w_2, \dots, w_K)$$

$$f_0(\mathbf{x}) = 1$$

$f_j(\mathbf{x}) -$ базисные функции

$f_0(\mathbf{x}) = 1, f_1(\mathbf{x}) = x_1, f_2(\mathbf{x}) = x_2, \dots, f_K(\mathbf{x}) = x_K -$ линейная

$f_0(\mathbf{x}) = 1, f_1(\mathbf{x}) = x_1, f_2(\mathbf{x}) = {x_1}^2, \dots, f_M(\mathbf{x})$
$= {x_1}^M -$ полиномиальная от 1 переменной

Матрица плана:

№	Признак1	Признак2	…	ПризнакK	Значение целевой переменной из статистики	Значение целевой переменной
1	x_{11}	x_{12}	…	x_{1K}	t_1	y_1
2	x_{21}	x_{22}	…	x_{2K}	t_2	y_2
…	…	…	…	…	…	…
N	x_{N1}	x_{N2}	…	x_{NK}	t_N	y_N

$$y(\mathbf{x_n}, w) = \sum_{j=0}^{M} w_j f_j(\mathbf{x_n})$$

$$\mathbf{x_n} = (1, x_{n1}, x_{n2}, \dots, x_{nK})$$

$$\mathbf{f}(\mathbf{x_n}) = (f_1(\mathbf{x_n}), f_2(\mathbf{x_n}), \dots, f_M(\mathbf{x_n}))$$

$$y(\mathbf{x_n}, w) = \mathbf{w}^T\mathbf{f}(\mathbf{x_n})$$

$$\begin{pmatrix} y(\mathbf{x_1}, w) \\ \vdots \\ y(\mathbf{x_N}, w) \end{pmatrix} = \begin{pmatrix} \mathbf{f}^T(\mathbf{x_1})\mathbf{w} \\ \vdots \\ \mathbf{f}^T(\mathbf{x_N})\mathbf{w} \end{pmatrix}$$

$$\mathbf{y}(\mathbf{X}, w) = \begin{pmatrix} y(\mathbf{x_1}, w) \dots y(\mathbf{x_N}, w) \end{pmatrix}$$

$$\mathbf{y}(\mathbf{X}, w) = \mathbf{F}\mathbf{w}$$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x_{11}} & \mathbf{x_{12}} & \cdots & \mathbf{x_{1K}} \\ \mathbf{x_{21}} & \mathbf{x_{22}} & \dots & \mathbf{x_{2K}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x_{N1}} & \mathbf{x_{N2}} & \cdots & \mathbf{x_{NK}} \end{pmatrix}$$

$$\mathbf{F} = \begin{pmatrix} f_0(\mathbf{x_1}) & f_1(\mathbf{x_1}) & \cdots & f_M(\mathbf{x_1}) \\ f_0(\mathbf{x_2}) & f_1(\mathbf{x_2}) & \dots & f_M(\mathbf{x_2}) \\ \vdots & \vdots & \ddots & \vdots \\ f_0(\mathbf{x_N}) & f_1(\mathbf{x_N}) & \cdots & f_M(\mathbf{x_N}) \end{pmatrix}$$

$\mathbf{F}$ – матрица плана

Ошибка:

Предположения: все наблюдения в обучающей выборке независимы; подчиняются нормальному распределению.

$$p(t|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{i=1}^{N} N\,(t_i|y(\mathbf{x_i}, w), \sigma^2)$$

$$N\,(t_i|y(\mathbf{x_i}, w), \sigma^2) = \frac{1}{\sqrt{2\pi\,\sigma^2}} e^{-\frac{(t_i - y(\mathbf{x_i}, w))^2}{2\,\sigma^2}}$$

Логарифмическая функция правдоподобия:

$$L(\mathbf{w}|t, \mathbf{X}) = \ln p(t|\mathbf{X}, \mathbf{w}, \sigma^2) = \sum_{i=1}^{N} \ln N\,(t_i|y(\mathbf{x_i}, w), \sigma^2) =$$

$$= \sum_{i=1}^{N} \ln \frac{1}{\sqrt{2\pi\,\sigma^2}} e^{-\frac{\left(t_i - y(\mathbf{x_i}, w)\right)^2}{2\,\sigma^2}}$$

$$= \sum_{i=1}^{N} \left(\ln \frac{1}{\sqrt{2\pi\,\sigma^2}} + \ln e^{-\frac{\left(t_i - y(\mathbf{x_i}, w)\right)^2}{2\,\sigma^2}} \right) =$$

$$= \sum_{i=1}^{N} \left(\ln \frac{1}{\sqrt{2\pi\,\sigma^2}} - \frac{1}{\sigma^2} \frac{\left(t_i - y(\mathbf{x_i}, w)\right)^2}{2} \right) =$$

$$= \sum_{i=1}^{N} \left(\ln \frac{1}{\sqrt{2\pi\,\sigma^2}} \right) - \frac{1}{\sigma^2} \frac{1}{2} \sum_{i=1}^{N} \left(t_i - y(\mathbf{x_i}, w)\right)^2$$

Для максимизация функцию правдоподобия необходимо минимизировать:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \left(t_i - y(\mathbf{x_i}, w)\right)^2 = \frac{1}{2} \sum_{i=1}^{N} \left(t_i - \mathbf{w}^T \mathbf{f}(\mathbf{x_i})\right)^2$$

Вычисление через псевдообратную матрицу (градиент):

$$\mathbf{t} = \mathbf{F}\mathbf{w}$$

$$\mathbf{t} = (t_1, t_2, \dots, t_N)$$

$$\mathbf{f_i} = (f_i(x_1), f_i(x_2), \dots, f_i(x_N))$$

$$\nabla E(\mathbf{w}) = \begin{pmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_M} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Равенство градиента нулю является необходимым, но не достаточным условием экстремума

$$\begin{pmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_M} \end{pmatrix} = \begin{pmatrix} -\sum_{i=1}^{N} \left(t_i - \mathbf{w}^T \mathbf{f}(\mathbf{x_i})\right) f_0(\mathbf{x_i}) \\ -\sum_{i=1}^{N} \left(t_i - \mathbf{w}^T \mathbf{f}(\mathbf{x_i})\right) f_1(\mathbf{x_i}) \\ \vdots \\ -\sum_{i=1}^{N} \left(t_i - \mathbf{w}^T \mathbf{f}(\mathbf{x_i})\right) f_M(\mathbf{x_i}) \end{pmatrix} =$$

$$= \begin{pmatrix} -\sum_{i=1}^{N} t_i f_0(\mathbf{x_i}) + \sum_{i=1}^{N} \mathbf{w}^T \mathbf{f}(\mathbf{x_i}) f_0(\mathbf{x_i}) \\ -\sum_{i=1}^{N} t_i f_1(\mathbf{x_i}) + \sum_{i=1}^{N} \mathbf{w}^T \mathbf{f}(\mathbf{x_i}) f_1(\mathbf{x_i}) \\ \vdots \\ -\sum_{i=1}^{N} t_i f_M(\mathbf{x_i}) + \sum_{i=1}^{N} \mathbf{w}^T \mathbf{f}(\mathbf{x_i}) f_M(\mathbf{x_i}) \end{pmatrix} =$$

$$= \begin{pmatrix} -\sum_{i=1}^{N} t_i f_0(\mathbf{x_i}) + \mathbf{w}^T \mathbf{f_0}^T \mathbf{F} \\ -\sum_{i=1}^{N} t_i f_1(\mathbf{x_i}) + \mathbf{w}^T \mathbf{f_1}^T \mathbf{F} \\ \vdots \\ -\sum_{i=1}^{N} t_i f_M(\mathbf{x_i}) + \mathbf{w}^T \mathbf{f_M}^T \mathbf{F} \end{pmatrix} =$$

$$= -(\mathbf{t}^T \mathbf{F})^T + (\mathbf{w}^T (\boldsymbol{F}^T \mathbf{F}))^T =>$$

$$\mathbf{t}^T \mathbf{F} = \mathbf{w}^T (\boldsymbol{F}^T \mathbf{F}) =>$$

$$\boldsymbol{w} = (\boldsymbol{F}^T \mathbf{F})^{-1} \boldsymbol{F}^T \boldsymbol{t}$$

$(\boldsymbol{F}^T \mathbf{F})^{-1} \boldsymbol{F}^T$ – один из вариантов вычисления псевдообратной $\boldsymbol{F}^+$

(для полноранговой F, т.е. в том числе при изначальном условии линейной независимости строк и столбцов)

Вычисление через псевдообратную матрицу 2:

$$\mathbf{w} = \mathbf{F}^{+}\mathbf{t}$$

Где $\mathbf{F}^{+}$ псевдообратная матрица и удовлетворяет, в частности (в случае работы с действительными числами):

1. $\mathbf{F}\mathbf{F}^{+}\mathbf{F} = \mathbf{F}$
2. $\mathbf{F}^{+}\mathbf{F}\mathbf{F}^{+} = \mathbf{F}$
3. $(\mathbf{F}^{+}\mathbf{F})^{\mathbf{T}} = \mathbf{F}^{+}\mathbf{F}$
4. $(\mathbf{F}\mathbf{F}^{+})^{\mathbf{T}} = \mathbf{F}\mathbf{F}^{+}$

Существует множество способов ее нахождения. Каждое из них может быть приближенным или иметь ограниченное применением. В предыдущем разделе показан один из способов.

Другой вариант вычисления (для не полноранговых матриц):

$$\mathbf{F} = \mathbf{F_1}\mathbf{F_2}$$

$$\mathbf{F} - \text{размерность NxM, ранг } k$$

$$\mathbf{F_1} - \text{размерность Nxk, ранг } k$$

$$\mathbf{F_2} - \text{размерность kxM, ранг } k$$

$$\mathbf{F}^{+} = \mathbf{F_2}^{T}\left(\mathbf{F_2}\mathbf{F_2}^{T}\right)^{-1}\left(\mathbf{F_1}^{T}\mathbf{F_1}\right)^{-1}\mathbf{F_1}^{T}$$

Получение скелетного разложения и вывода основной формулы здесь пока пропущено. Может быть найдено просто в поисковике по запросу «матрица Мура — Пенроуза». (например, один из первых результатов: https://scask.ru/a_book_matrix.php?id=7)

Вычисление 3:

Задача $\mathbf{F}\mathbf{w} = \boldsymbol{t}$ может быть «поставлена некорректно и не иметь точного решения» (упрощенное объяснение)

Тогда решение можно искать в виде $w_\delta = R(t_\delta, a)$ (регуляризирующий оператор). При $t_\delta \to t$ и $\delta \to 0$ => $w_\delta \to w$ (упрощенное объяснение)

Дополнительный материал ищется по «регуляризация Тихонова; теорема Тихонова)

Также можем применить метод наименьших квадратов и минимизировать $\|\mathbf{F}\mathbf{w} - \boldsymbol{t}\|^2$

А в соответствии с теоремой Тихонова для «не совсем корректно поставленной задачи» можем минимизировать $\|\mathbf{Fw} - \boldsymbol{t}\|^2 + a\|\mathbf{w}\|^2$ (регуляризация Тихонова)

Регуляризация:

Аналогично для машинного обучения данные статистики могут коррелировать друг с другом. Наблюдается неустойчивость оценок.

$$\boldsymbol{w} = (\boldsymbol{F^T}\mathbf{F})^{-1}\boldsymbol{F^T t}$$

Ограничим $\|\mathbf{w}\|_2^2 \leq l^2$

Тогда аналогично будем минимизировать $\|\mathbf{Fw} - \boldsymbol{t}\|^2 + a\|\mathbf{w}\|^2$

Дополнительный материал ищется по «ridge regression»

Для изначальной функции ошибки при этом будет:

$$E(\mathbf{w}) = \frac{1}{2}\left(\sum_{i=1}^{N}\left(t_i - \mathbf{w}^{\boldsymbol{T}}\mathbf{f}(\mathbf{x}_{\boldsymbol{i}})\right)^2 + a * \sum_{i=1}^{M}|\mathrm{w}_{\boldsymbol{i}}|^2\right)$$

Далее любым из методов (в том числе $\nabla E(\mathbf{w}) = 0$) получаем:

$$\boldsymbol{w} = (\boldsymbol{F^T}\mathbf{F} + a\boldsymbol{I^T})^{-1}\boldsymbol{F^T t}$$

Градиентный спуск:

1. Выбор $\mathbf{w_0}$ – начального вектора весов
2. Корректировка весов
   2.1. $\mathbf{w_1} = \mathbf{w_0} - \gamma\nabla E(\mathbf{w_0})$, где γ - шаг обучения
   2.2. ….
   2.3. $\mathbf{w_l} = \mathbf{w_{l-1}} - \gamma\nabla E(\mathbf{w_{l-1}})$
3. Примеры критериев останова итерационного процесса 2:
   3.1. Количество итераций
   3.2. $\|\mathbf{w_l} - \mathbf{w_{l-1}}\| < \varepsilon$
   3.3. $\|\nabla E(\mathbf{w_{l-1}})\| < \varepsilon$

Варианты нормализации и стандартизации данных:

1. $X = \frac{X - X_{min}}{X_{max} - X_{min}}$
2. $X = a + \frac{X - X_{min}}{X_{max} - X_{min}}(b - a)$

3. $X = \frac{X-\mu}{\sigma}$, $\mu -$ среднее, σ – стандартное отклонение

Может производится для каждого признака по-своему, т.е. для каждого столбца статистики.

Расчет градиента для случая с регуляризацией:

$$E(\mathbf{w}) = \frac{1}{2}\left(\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)^2 + a * \sum_{i=1}^{M}|\mathrm{w_i}|^2\right)$$

$$\nabla E(\mathbf{w}) = \begin{pmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_M} \end{pmatrix}$$

$$= \begin{pmatrix} -\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)\left(-\mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)'_{w_0} + \frac{a}{2} * \sum_{i=1}^{M}(|\mathrm{w_i}|^2)'_{w_0} \\ -\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)\left(-\mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)'_{w_1} + \frac{a}{2} * \sum_{i=1}^{M}(|\mathrm{w_i}|^2)'_{w_1} \\ \vdots \\ -\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)\left(-\mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)'_{w_M} + \frac{a}{2} * \sum_{i=1}^{M}(|\mathrm{w_i}|^2)'_{w_M} \end{pmatrix} =$$

$$= \begin{pmatrix} -\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)f_0(\mathbf{x_i}) + aw_0 \\ -\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)f_1(\mathbf{x_i}) + aw_1 \\ \vdots \\ -\sum_{i=1}^{N}\left(t_i - \mathbf{w}^T\mathbf{f}(\mathbf{x_i})\right)f_M(\mathbf{x_i}) + aw_M \end{pmatrix} =$$

$$= \begin{pmatrix} -\sum_{i=1}^{N} t_i f_0(\mathbf{x_i}) + \sum_{i=1}^{N} \mathbf{w^T f}(\mathbf{x_i}) f_0(\mathbf{x_i}) + a w_0 \\ -\sum_{i=1}^{N} t_i f_1(\mathbf{x_i}) + \sum_{i=1}^{N} \mathbf{w^T f}(\mathbf{x_i}) f_1(\mathbf{x_i}) + a w_1 \\ \vdots \\ -\sum_{i=1}^{N} t_i f_M(\mathbf{x_i}) + \sum_{i=1}^{N} \mathbf{w^T f}(\mathbf{x_i}) f_M(\mathbf{x_i}) + a w_M \end{pmatrix} =$$

$$= \begin{pmatrix} -\sum_{i=1}^{N} t_i f_0(\mathbf{x_i}) + \mathbf{w^T f_0}^{\boldsymbol{T}} \mathbf{F} + a w_0 \\ -\sum_{i=1}^{N} t_i f_1(\mathbf{x_i}) + \mathbf{w^T f_1}^{\boldsymbol{T}} \mathbf{F} + a w_1 \\ \vdots \\ -\sum_{i=1}^{N} t_i f_M(\mathbf{x_i}) + \mathbf{w^T f_M}^{\boldsymbol{T}} \mathbf{F} + a w_M \end{pmatrix} =$$

$$= -(\mathbf{t^T F})^T + (\mathbf{w^T} (\boldsymbol{F^T} \mathbf{F}))^T + a \mathbf{w^T}$$

Метрики классификации:

N – размер выборки

TP – количество объектов класса 1, отнесенных к классу 1

TN – количество объектов класса 0, отнесенных к классу 0

FP – количество объектов класса 0, отнесенных к классу 1

FN – количество объектов класса 1, отнесенных к классу 0

alpha = FP / (TN + FP)

beta = FN / (TP + FN)

Accuracy = (TP + TN) / N

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F1 = 2 * (Precision * Recall) / (Precision + Recall)

FPR = FP / (FP + TN) = FP / $Count_0$

TPR = TP / (TP + FN) = TP / $Count_1$

ROC кривая

ROC кривая - график (FPR, TPR) при изменении порога классификации.

AUC – площадь под ROC-кривой

Варианты расчета:

1. Менять порог и считать

2. Использовать один из алгоритмов. Например, отсортировать значения, полученные из классификатора до применения порога по убыванию. Далее перебираем их считаем порогом. Т.е. для каждого следующего значения считаем, что он предсказан как класс 1, смотрим «желаемый класс» и если он 0, то увеличиваем FPR на 1/(FP + TN), а если он 1, то TPR на 1/(TP + FN).

AUC в этом случае считается как сумма площадей получившихся прямоугольников. Можно увеличивать каждый раз когда увеличивается FPR на TPR / (FP + TN)

3. и т.д.

Метрики классификации для множества классов:

T_i – количество объектов класса i, отнесенных к классу i

F_{ij} – количество объектов класса j, отнесенных к классу i

Accuracy = (T_1+ T_2 +…+ T_n) / N

Confusion matrix:

$$\begin{pmatrix} T_1 & F_{12} & \cdots & F_{1n} \\ F_{21} & T_2 & \cdots & F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ F_{n1} & F_{n2} & \cdots & T_n \end{pmatrix}$$

Логистическая регрессия:

Логистическая функция (сигмоида): $g(a) = \frac{1}{1+e^{-a}}$

Логистическая регрессия: $y(x, w) = \frac{1}{1+e^{-w^T f(x)}}$

Функция потерь: $E(w) = -\sum_{i=1}^{N}(t_i \ln y(x_i, w) + (1 - t_i)\ln(1 - y(x_i, w)))$, где t_i {0,1}

$$\nabla E(w) = \sum_{i=1}^{N}(y(x_i, w) - t_i)f(x_i)$$

Softmax:

Уверенность в принадлежности классу: $g(a)_c = \frac{e^{a_c}}{\sum_{i=1}^{K} e^{a_i}}$

Для 2 классов $g(a)_1 = \frac{e^{a_1}}{\sum_{i=1}^{2} e^{a_i}} = \frac{e^{a_1}}{e^{a_1} + e^{a_2}} = \frac{1}{1 + e^{-(a_1 - a_2)}}$

Функция потерь: $E(x, w) = -\sum_{i=1}^{N}\sum_{c=1}^{C} t_{ic} \ln (y_c(x_i, w))$, где C – число классов, c – текущий класс (опциально $*\frac{1}{N}$)

Доп. обозначение для входов X – матрица NxM, где N – число примеров, M – число признаков.

Доп обозначение весов W – матрица MxC, где M – число признаков, C – число классов. Коэффициенты для каждого класса по стобцам матрицы W.

Доп обозначение выходов Y – матрица NxC уверенностей от 0 до 1

Доп обозначение желаемых T – матрица NxC принадлежностей к классу или 0 или 1

$$\nabla E(w) = X^T(Y - T)$$

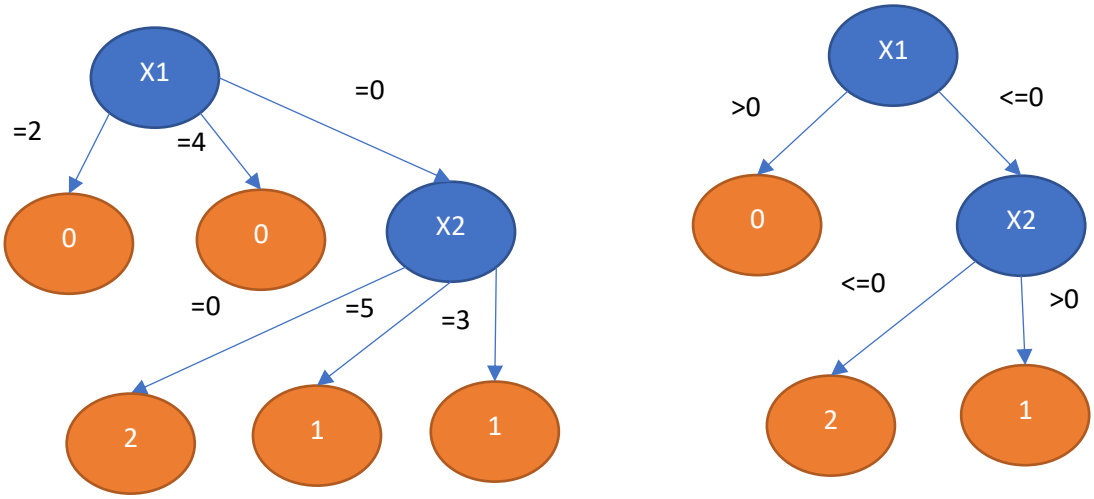
Деревья решений

Рассматривается упрощенный синтетический пример классификации. Вход: трехмерный вектор (x1, x2, x3). Выходные классы (3 класса): первая компонента больше 0 (метка 0), вторая больше 0 (метка 1), третья больше 0 (метка 2).

Обучающая выборка:

(0, 0, 4) -> 2 | (2, 0, 0) -> 0 | (4, 0, 0) -> 0 | (0, 5, 0) -> 1 | (0, 0, 1) -> 2 | (0, 3, 0) -> 1

Рассмотрим 2 варианта дерева, которые можем получить:

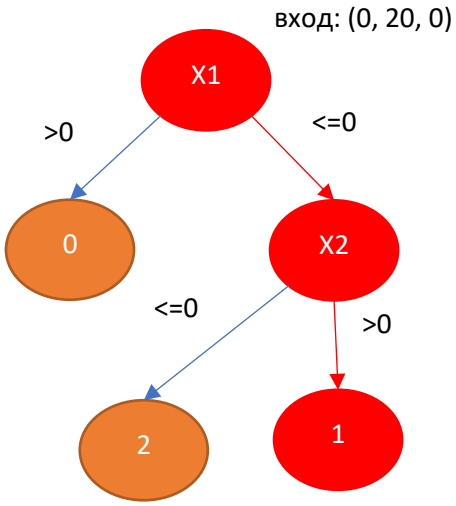


Узлы оранжевый – терминальные узлы (в подписи номер класса), синие – узлы разделения (в подписи имя атрибута, значение которого сравниваем, чтобы идти дальше). В подписях к ребрам условия, с которым нужно сравнить атрибут узла, из которого выходит ребро. Далее идем по ребру, с условием которого совпадает атрибут. Не может быть удовлетворено более 1 условия.

Для разбивания используются атрибуты. Тут их 3 (каждый компонент вектора). В случае картинки 8x8 один из вариантов рассматривать 64 атрибута.

Левая картинка больше применима к перечисляемым значениям атрибутов (условия на равенство; к перечисляемым, т.к. можем сказать, что нового значения не появится и сравнения останутся такими же), правая к непрерывным (условия в виде неравенств и как правило бинарное дерево). Очевидно, что в нашем случае правый вариант подходит лучше.

Рассмотрим работу дерева для классификации вектора (0, 20, 0). На рисунке ниже путь выделен красным. Получаем класс с меткой 1 (как и нужно).



При автоматическом построении для бинарного дерева остаются проблемы: какой атрибут выбрать для разбиения, какое число для сравнения выбрать в неравенстве. Для выбора атрибута можно использовать различные метрики. Тут будет на примере энтропии и прироста информации.

Корень дерева (рассматриваем всю выборку):

$H = -\sum_{i=1}^{n} \frac{N_i}{N} log_2 \frac{N_i}{N}$ - *энтропия (n – число классов, N – элементов в выборке,* N_i – элементов i класса)

$H = -\frac{\text{2 (количество элементов класса 0)}}{\text{6 (элементов в выбокре)}} log_2 \frac{2}{6} - \frac{\text{2 (класса 1)}}{6} log_2 \frac{2}{6} - \frac{\text{2 (класса 2)}}{6} log_2 \frac{2}{6} =$ *1.6*

Далее нужно выбрать атрибут и параметр разделения. Для этого пробуем все возможные варианты разбиения и смотрим у какого больше прирост информации.

$I = H - \sum_{j=1}^{m} \frac{N_j}{N} (-\sum_{i=1}^{n} \frac{N_{ji}}{N_j} log_2 \frac{N_i}{N_j})$ - *прирост информации (H – энтропия разбиваемого узла, прямо сейчас корня, m – число потомков разбиваемого узла, в бинарном 2,* N_j – элементов выборки, достигших j потомка текущего разбиваемого узла, N_{ji} – элементов выборки класса i, достигших j потомка текущего разбиваемого узла *, остальное аналогично предыдущему)*

Начнем считать для атрибута x1. Также выберем для него k вариантов разделяющего параметра. Тогда только для x1 нужно будет посчитать I k раз. Выбор разделяющего параметра можно сделать по-разному. Например, рандом. Но, рассмотрим другой вариант. Рассмотрим все варианты значений

x1: 0 2 4 0 0 0. Упорядочим их и оставим только уникальные: 0 2 4. Далее выберем в качестве параметров разделения среднее между двумя соседними: 1 3. Для нашей небольшой выборки эти параметры покроют все варианты разбивки по x1.

Рассмотрим разбивку по x1 и 1 (x1 > 1, x1 <= 1). В этом случае выборка разбивается на 2 подмножества:

x1>1: (2, 0, 0) -> 0 | (4, 0, 0) -> 0

и

x1<=1: (0, 0, 4) -> 2 | (0, 5, 0) -> 1 | (0, 0, 1) -> 2 | (0, 3, 0) -> 1

Соответственно расчеты будут следующими:

$$I(x1,1) = H - \frac{2\ (\text{число элементов } x1 > 1)}{6\ (\text{число элементов в разбиваемом узле})} \times$$

$$\times \left(- \frac{2\ (\text{число класса 0 в подмножестве } x1 > 1}{2\ (\text{число элементов в } x1 > 1)} log_2 \frac{2}{2} \right)$$

$$- \frac{4\ (\text{число элементов } x1 \leq 1)}{6\ (\text{число элементов в разбиваемом узле})} \times$$

$$\times \left(- \frac{2\ (\text{число класса 1 в подмножестве } x1 \leq 1}{4\ (\text{число элементов в } x1 \leq 1)} log_2 \frac{2}{4} \right.$$

$$\left. - \frac{2\ (\text{число класса 2 в подмножестве } x1 \leq 1}{4} log_2 \frac{2}{4} \right) = H - 0.67$$

$I(x1,3) = H - 1.27$

Для разбивки по x1 вариант I(x1,1) больше. Соответственно, если выберем x1, то с параметром разбивки 1.

Для x2 (разбивка 1.5, 4), x3 (разбивка 0.5, 2.5) в данном случае будут такие же результаты.

Продолжим для разбивки по x1 и 1. Тогда узел с подмножеством (2, 0, 0) -> 0 | (4, 0, 0) -> 0 станет листом по условию энтропии 0.

Для подмножества (0, 0, 4) -> 2 | (0, 5, 0) -> 1 | (0, 0, 1) -> 2 | (0, 3, 0) -> 1 произведем разбивку далее.

I(x2,1.5)=

$$H2 - \frac{2\ (\text{число элементов } x2 > 1.5)}{4\ (\text{число элементов в разбиваемом узле})} \times$$

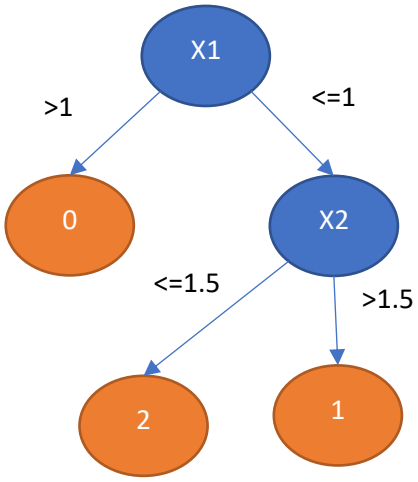
$$\times\left(-\frac{2\ (\text{число класса 1 в подмножестве } x2>1.5}{2\ (\text{число элементов в } x2>1.5)}\log_2\frac{2}{2}\right)$$
$$-\frac{2\ (\text{число элементов } x2\le 1.5)}{4\ (\text{число элементов в разбиваемом узле})}\times$$
$$\times\left(-\frac{2\ (\text{число класса 2 в подмножестве } x2\le 1.5}{2\ (\text{число элементов в } x2\le 1.5)}\log_2\frac{2}{2}\right)$$
$$=H2-0$$

I(x2,4)=….

В любом случае I(x2,1.5) меньше. Причем при такой разбивки оба дочерних узла имеют энтропию 0.

Условия останова (объявлении узла терминальным) могут быть по энтропии менее порога; по глубине узла; по числу примеров, достигших узла, менее порога; и т.д.

Получаем следующее после нашего обучения:



Возможно, полезное по python:

Стандартные numpy, matplotlib:

```
import numpy as np
import matplotlib.pyplot as plt

# Создать вектор
```

```
print(
   np.array((1, 2))
)

>>>
[1 2]

# Вектор в матрицу из одной строки (новая ось)
print(
   np.array((1, 2))[np.newaxis, :]
)

>>>
[[1 2]]

# Вектор в матрицу из одного столбца (новая ось)
print(
   np.array((1, 2))[:, np.newaxis]
)

>>>
[[1]
 [2]]

# Добавить в матрицу строку
print(
   np.concatenate(
      (
         np.matrix([
            [1, 2],
            [3, 4]
         ]),
         np.array([5, 6])[np.newaxis, :]
      )
   )
)

>>>
[[1 2]
 [3 4]
 [5 6]]
```

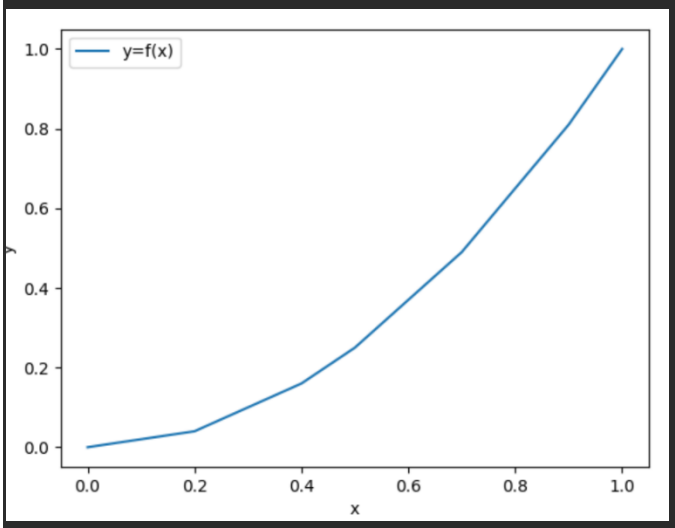
```
# Добавить в матрицу столбец
print(
    np.concatenate(
        (
            np.matrix([
                [1, 2],
                [3, 4]
            ]),
            np.array([5, 6])[:, np.newaxis]
        ),
        axis=1
    )
)

>>>
[[1 2 5]
 [3 4 6]]

# Сгенерировать вектор из N значений между a, b, включая a,b
print(
    np.linspace(0, 10, 4)
)

>>>
[ 0.         3.33333333  6.66666667 10.        ]

# Повторить N раз каждую строку матрицы
print(
    np.repeat(
        np.matrix([
            [1, 2],
            [3, 4]
        ]),
        2,
        axis=0  # Важно указывать даже для 0, иначе матрица преобразуется в
массив
    )
)

>>>
[[1 2]
 [1 2]
```

```
 [3 4]
 [3 4]]

# Повторить N раз каждый столбец матрицы
print(
    np.repeat(
        np.matrix([
            [1, 2],
            [3, 4]
        ]),
        2,
        axis=1
    )
)

>>>
[[1 1 2 2]
 [3 3 4 4]]

# Возвести каждый столбец матрицы в соответствующую степень
print(
    np.power(
        np.matrix([
            [2, 2, 2],
            [2, 2, 2]
        ]),
        [1, 2, 3]
    )
)

[[2 4 8]
 [2 4 8]]

# Обратная матрица
print(
    np.linalg.inv(
        np.matrix([
            [1, 2],
            [3, 4]
        ]),
    )
)
```

```
>>>
[[-2.   1. ]
 [ 1.5 -0.5]]

# Транспонированная матрица
print(
    np.transpose(
        np.matrix([
            [1, 2],
            [3, 4]
        ]),
    )
)

>>>
[[1 3]
 [2 4]]

# Единичная матрица определенной рамерности
print(
    np.eye(5)
)

>>>
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]

# Матрица из нулей
print(
    np.zeros((2, 3))
)

>>>
[[0. 0. 0.]
 [0. 0. 0.]]

# Матрица из единиц
print(
```

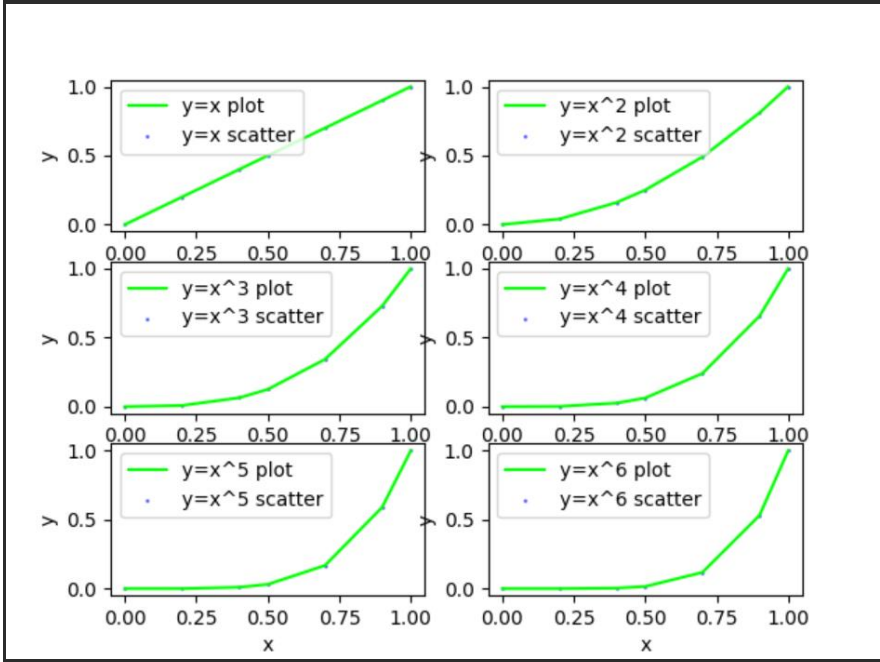
```
    np.ones((2, 3))
)

>>>
[[1. 1. 1.]
 [1. 1. 1.]]

# Псевдообратная матрица
print(
    np.linalg.pinv(
        np.matrix([
            [1, 2, 3],
            [4, 5, 6]
        ]),
    )
)

>>>
[[-0.94444444  0.44444444]
 [-0.11111111  0.11111111]
 [ 0.72222222 -0.22222222]]

# решение матричного уравнения Ax=b (найти x)
A = np.matrix([
    [1, 2],
    [3, 4]
])
b = np.array([3, 2])
print(
    np.linalg.solve(
        A, b
    )
)

>>>
[-4.   3.5]

# умножение матриц и векторов
print(
    np.dot(
        np.matrix([
            [1, 2],
```

```
        [3, 4]
      ]),
      np.matrix([
        [1, 2],
        [3, 4]
      ]),
   )
)

>>>
[[ 7 10]
 [15 22]]

# простейший график y = f(x)
x = np.array((0, 0.2, 0.4, 0.5, 0.7, 0.9, 1))
y = x ** 2  # y = x^2
plt.figure()  # новая фигура/окно
plt.plot(x, y, label="y=f(x)")
plt.legend(loc="upper left")  # где выводить label="y=f(x)"
plt.xlabel("x")  # подпись оси x
plt.ylabel("y")  # подпись оси y
plt.show()  # показать графики
```

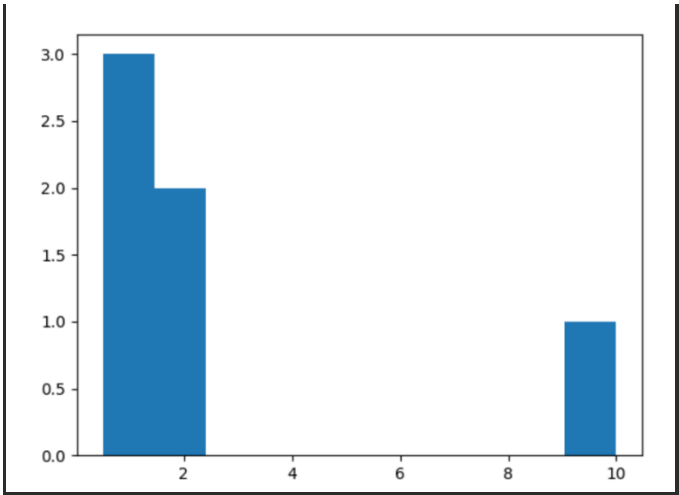



```
# в одном окне несколько графиков
# fig - окно, axs - массив из 6 указателей на графики
fig, axs = plt.subplots(3, 2)  # 3 строки, 2 столбца, 6 графиков
# просто для удобства именования
# axs = [[1, 2], [3, 4], [5, 6]] потому преобразуем в [1, 2, 3, 4, 5, 6] через np.flat
dict_axs = dict(zip(["y=x", "y=x^2", "y=x^3", "y=x^4", "y=x^5", "y=x^6"],
np.array(axs).flat))
```

```
# данные
x = np.array((0, 0.2, 0.4, 0.5, 0.7, 0.9, 1))
y = x
# графики
for (key, ax) in dict_axs.items():
    # график непрерывный зеленого цвета
    ax.plot(x, y, color=(0, 1, 0, 1), label=(key + ' plot'))
    # график точками размера 1, синего цвета, полупрозрачный
    ax.scatter(x, y, s=1, color=(0, 0, 1, 0.5), label=(key + ' scatter'))
    ax.legend(loc="upper left")
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    y = y * x  # поэлементно умножаем (воводим в следующую степень)
plt.show()
```




```
# гистограмма
plt.figure()
plt.hist([1, 2, 2, 1, 10, 0.5])
plt.show()
```




```
# 5 случайных чисел из нормального распределения N(1, 2^2)
# матрицы аналогично размерность (2, 3) вместо 5
print(
   2 * np.random.randn(5) + 1
)

>>>
[-2.67640222  2.81043519 -2.18850503  3.67893342 -1.68334377]

# сумма элементов матрицы/вектора
print(
   np.sum(np.array((1, 2, 3)))
)

>>>
6

# среднее по столбцам
print(
   np.mean(
      np.matrix(
         [
            [1, 2, 3],
            [2, 3, 3]
         ]
      ),
      axis=0
   )
)
```

```
>>>
[[1.5 2.5 3. ]]

# отклонение по столбцам
print(
   np.std(
      np.matrix(
         [
            [1, 2, 3],
            [2, 3, 3]
         ]
      ),
      axis=0
   )
)

>>>
[[0.5 0.5 0. ]]

#
```

Загрузка датасета:

```
from sklearn.datasets import fetch_california_housing

# только x -> t
x, t = fetch_california_housing(return_X_y=True)
print(x)
print(t)

>>>
[   8.3252      41.          6.98412698 ...    2.55555556
   37.88      -122.23      ]
 [   8.3014      21.          6.23813708 ...    2.10984183
   37.86      -122.22      ]
 [   7.2574      52.          8.28813559 ...    2.80225989
   37.85      -122.24      ]
 ...
 [   1.7         17.          5.20554273 ...    2.3256351
   39.43      -121.22      ]
 [   1.8672      18.          5.32951289 ...    2.12320917
```

```
   39.43      -121.32     ]
 [  2.3886     16.          5.25471698 ...    2.61698113
   39.37      -121.24     ]]
[4.526 3.585 3.521 ... 0.923 0.847 0.894]

# вся информация
print(
   fetch_california_housing()
)

>>>
{'data': array([[   8.3252    ,   41.      ,    6.98412698, ...,    2.55555556,
        37.88     , -122.23     ],
       [   8.3014    ,   21.      ,    6.23813708, ...,    2.10984183,
        37.86     , -122.22     ],
       [   7.2574    ,   52.      ,    8.28813559, ...,    2.80225989,
        37.85     , -122.24     ],
       ...,
       [   1.7       ,   17.      ,    5.20554273, ...,    2.3256351 ,
        39.43     , -121.22     ],
       [   1.8672    ,   18.      ,    5.32951289, ...,    2.12320917,
        39.43     , -121.32     ],
       [   2.3886    ,   16.      ,    5.25471698, ...,    2.61698113,
        39.37     , -121.24     ]]), 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847,
0.894]), 'frame': None, 'target_names': ['MedHouseVal'], 'feature_names':
['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
'Latitude', 'Longitude'], 'DESCR': '.. _california_housing_dataset:\n\nCalifornia
Housing dataset\n--------------------------\n\n**Data Set Characteristics:**\n\n
:Number of Instances: 20640\n\n    :Number of Attributes: 8 numeric, predictive
attributes and the target\n\n    :Attribute Information:\n        - MedInc        median
income in block group\n        - HouseAge      median house age in block group\n
- AveRooms      average number of rooms per household\n        - AveBedrms
average number of bedrooms per household\n        - Population    block group
population\n        - AveOccup      average number of household members\n        -
Latitude      block group latitude\n        - Longitude     block group longitude\n\n
:Missing Attribute Values: None\n\nThis dataset was obtained from the StatLib
repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html\n\n
The target variable is the median house value for California districts,\nexpressed
in hundreds of thousands of dollars ($100,000).\n\nThis dataset was derived from
the 1990 U.S. census, using one row per census\nblock group. A block group is the
smallest geographical unit for which the U.S.\nCensus Bureau publishes sample
data (a block group typically has a population\nof 600 to 3,000 people).\n\nAn
```

```
household is a group of people residing within a home. Since the
average\nnumber of rooms and bedrooms in this dataset are provided per
household, these\ncolumns may take surpinsingly large values for block groups
with few households\nand many empty houses, such as vacation resorts.\n\nIt
can be downloaded/loaded using
the\n:func:`sklearn.datasets.fetch_california_housing` function.\n\n.. topic::
References\n\n    - Pace, R. Kelley and Ronald Barry, Sparse Spatial
Autoregressions,\n      Statistics and Probability Letters, 33 (1997) 291-297\n'}

#digits
from sklearn.datasets import load_digits
digits = load_digits()
print(digits)
```