Линейная модель

Шокуров Антон В.
shokurov.anton.v@yandex.ru
http://машинноезрение.рф

18 марта 2017 г.

Версия: 0.10

Аннотация

Линейная регрессия. Показан традиционный способ. Вводится понятие классификации и соответствующая модель – персептрон.

Предварительная версия!

1 Линейная модель

В первом подразделе описывается линейная регрессия. Во втором основа нейронной сети – персептрон.

1.1 Линейная регрессия

Целью является приближение данных линейной функцией.

Линейной пространство Искомое параметрическое множество функций задается как $g(x,\theta) = \sum_{j=1}^{j=m} \theta_j e_j(x)$, $\theta = (\theta_1, ..., \theta_m)$. В качестве функций e_j можно взять любые функции. Например $e_j(x) = x^{j-1}$ или $e_j(x) = sin((j-1)x)$.

Можно двояко интерпретировать эти функции. Можно считать, что они являются частью отмеченного уравнения, т.е. выбраны экспертом для восстановления линейной зависимости.

Также можно считать, что они с самого начала были выбраны экспертом при построении параметрического пространства, т.е. они есть суть функции φ. В последнем случае уравнение сводится просто к линейной функции:

$$g(x,\theta) = \sum_{j=1}^{j=n} \theta_j x_j,$$

где $\theta = (\theta_1, ..., \theta_m)$.

Если подставить обучающее множество в исходное уравнение, то получим систему уравнений: $y_i = \sum_{j=0}^{j=m} \theta_j e_j(x_i)$. Замечу, что выражение $e_j(x_j)$ является числом, поэтому – это система линейных уравнений относительно параметров θ_j.

При определенных условия – что является предметом изучения курса по линейной алгебры – данная система имеет решение.

Если $m = n$, то это задача интерполяции. При интерполяции построенная функция $\tilde{f}$ проходит через исконное множество точек точно. т.е. функция качества будет равна 0. Насколько это важно вопрос спорный. Может быть так, что как раз на тестовом множестве будет достигнут худший вариант.

Если взять $e_j(x) = x^j$, то получим задачу поиска многочлена проходящего через заданные точки. Вместо того, чтобы составлять систему уравнений и её решать, что подразумевает обращение матрицы – задача крайней неблагодарная – можно решить задачу иначе ... можно построить решение явным образом.

Например, используя интерполяционные многочлены Лагранжа. Положим $e_j(x) = \prod_{k \neq j} \frac{x - x_k}{x_j - x_k}$, тогда она будет себя вести как индикаторная функция, а именно – при $j \neq i$ получим $e_j(x_i) = 0$, в тоже время $e_j(x_j) = 1$. Поэтому $\theta = (y_1, ..., y_n)$, т.е. $g(x, \theta) = L(x) = \sum_{j=0}^{m-1} y_j e_j(x)$.

Неудобна...

Формула Ньютона...

Если количество уравнений меньше, чем неизвестных, то появляется неопределенность в выборе параметров. Если же их больше, то система с большой вероятностью будет несовместна.

1.2 Аппроксимация

Традиционно, пусть минимизация будет производится в среднеквадратичном смысле.

Двумерный случай Рассмотрим самый простой случай. Пусть $e_0(x) = a, e_1(x) = ax$, т.е. будем приближать данные линейной функцией. Причем, у нас один единственный признак ($n = 1$). Данными являются пары числе $(x_i, y_i)_{i=1}^{i=l}$, где $x_i, y_i \in \mathbf{R}$. С геометрической точки зрения это точки на плоскости.

В таком случае мы хотим минимизировать:

$$\mathcal{L}(X, Y; a, b) = \sum_{i=1}^{i=l} (a + bx_i - y_i)^2.$$

Таким образом мы используем среднеквадратичное расстояние. Известно, что его можно решить точно. Более того это можно сделать используя школьный уровень,

а именно –

$$\mathcal{L}(X,Y;a,b) = \sum_{i=1}^{i=l} a^2 + \sum_{i=1}^{i=l}(bx_i)^2 + \sum_{i=1}^{i=l} y_i^2 + 2\sum_{i=1}^{i=l} abx_i - 2\sum_{i=1}^{i=l} ay_i - 2\sum_{i=1}^{i=l} bx_iy_i =$$

$$= na^2 + b^2\sum_{i=1}^{i=l} x_i^2 + \sum_{i=1}^{i=l} y_i^2 + 2ab\sum_{i=1}^{i=l} x_i - 2a\sum_{i=1}^{i=l} y_i - 2b\sum_{i=1}^{i=l} x_iy_i.$$

Пусть

$$\bar{x} = \frac{\sum_{i=1}^{i=l} x_i}{l}, \bar{y} = \frac{\sum_{i=1}^{i=l} y_i}{l}, \bar{z} = \frac{\sum_{i=1}^{i=l} x_iy_i}{l}$$

и

$$\hat{x} = \frac{\sum_{i=1}^{i=l} x_i^2}{l}, \hat{y} = \frac{\sum_{i=1}^{i=l} y_i^2}{l}.$$

Тогда

$$\mathcal{L}(X,Y;a,b) = la^2 + b^2l\hat{x} + l\hat{y} + 2abl\bar{x} - 2al\bar{y} - 2bl\bar{z}.$$

Следовательно

$$\frac{\mathcal{L}(X,Y;a,b)}{l} = a^2 + b^2\hat{x} + \hat{y} + 2ab\bar{x} - 2a\bar{y} - 2b\bar{z}.$$

Дополнив выражение константами можно выделить полный квадрат относительно каждой из переменных. Последнее позволит минимизировать выражение относительно двух переменных.

Относительно a (считаем b константой):

$$(a + b\bar{x} - \bar{y})^2.$$

Последнее означает, что если для параметра b уже найдено оптимальное выражение, то a нужно выбрать минимизировав данный квадрат, т.е. потребовав

$$a + b\bar{x} - \bar{y} = 0.$$

По аналогии, относительно b (считаем a константой):

$$\hat{x}(b + \frac{a\bar{x} - \bar{z}}{\hat{x}})^2$$

и соответственно:

$$b + \frac{a\bar{x} - \bar{z}}{\hat{x}} = 0.$$

Последние два требования сводятся к линейной системе

$$\begin{cases} a + \bar{x}b - \bar{y} = 0 \\ \hat{x}b + \bar{x}a - \bar{z} = 0. \end{cases}$$

Решив систему, находим значения a и b.

Общий случай Есть точки в n-мерном пространстве. Есть значение ассоциированное с каждой из точек. Необходимо построить линейную функцию приближающую эти данные.

На математическом языке, есть уравнение: $Ax = b \iff Ax - b = 0$. Хотим решить в среднеквадратичном смысле, т.е. минимизировать $(Ax-b)^T(Ax-b) = 0$.

Напомним правило взятия производной у матриц....

Возьмем производную по x: $A^T(Ax-b)+A^T(Ax-b) = 2C$, где $C = A^T(Ax-b)$. Тогда из равенства производной 0 следует, что $2C = 0 \impliedby C = 0$. Следовательно верно:

$$A^T(Ax-b) = 0 \iff A^TAx - A^Tb = 0 \iff A^TAx = A^Tb.$$

Тогда $x = (A^TA)^{-1}A^Tb$. Пусть $A^+ = (A^TA)^{-1}A^T$, искомый ответ представляется в виде $x = A^+b$. Напомню, что для обычных уравнений ответ записывается в виде: $x = A^{-1}b$. В этом смысле матрица A^+ считается псевдо обратной к A.

Общий Сначала для общего случае. Тогда справедливо:

$$\min_\theta \sum_{i=1}^{i=l}(y_i - g(x_i,\theta))^2 \Longrightarrow \sum_{i=1}^{i=l} 2(y_i - g(x_i,\theta))\frac{\partial g(x_i,\theta)}{\partial \theta_j} = 0.$$

Если

$$\frac{\partial g(x_i,\theta)}{\partial \theta_j} \neq const$$

то тогда описанный данный подход не применим. Нужно действовать иначе: градиент.

В более общем случае не получится:

1.3 Классификация

В первой лекции было указано, что признак соответствующий классам задаются конечным множеством элементов. Например: $\{1, ..., N\}$. Такой подход неизбежно приводит к тому, что классы обрабатываются не равнозначно/равносильно. Так, 1 самое маленькое число, а N – большое. У крайних чисел чисел (1 и N) по одному непосредственному соседу, а у других чисел (классов) их 2. Можно применить к классам операцию больше и меньше, в частности, можно выделить два суперкласса: $\{1...k\}$ и $\{k+1...N\}$, что опять же будет препятствовать честному обучению.

Итог. Лучше заменить данный признак на N бинарных, каждый их которых способен отделить друг от друга только два класса. Фактически дать ответ на вопрос Да/Нет относительно принадлежности данному классу. Конечно может быть неоднозначность, когда два или более бинарных признаков дадут ответ Да.

Итого, классификация сводится к изучению бинарных классификаторов. Без ограничения общности, можно считать, что обучающая выборка будет содержать y из множества $\{1, -1\}$ и в соответствии с этим при $f(x) > 0$ ответ Да, а при $f(x) < 0$ ответ Нет. Тогда нетрудно видеть, что для правильно спрогнозированного ответа верно:$yf(x) > 0$.

1.4 Персептрон

Модель: семейство функций и алгоритм поиска минимума. Из биологии.

В данном случае мы придерживаемся идеи о том, что мозг может решить сложные задачи, в частности, задачи распознавания. Поэтому, если позаимствовать механизмами, которые лежат в его устройстве, есть шанс достичь высоких результатов. Подчеркну, что цели нет в создании точной модели мозга. Цель воспользоваться идеями его устройства для создания собственной модели распознавания.

Опишем устройство мозга в рамках вышесказанного. Мозг состоит из так называемых нейронов – узлов, которые передают друг другу электро-импульсы. Вся суть заключается в топологи – устройстве соединения нейронов друг с другом – и способе преобразования входных импульсов в выходные.

Специфика топологии заключается в том, что нейроны имеют много входов и один единственный выход. Так, нейрону передается на вход много импульсов, а на выходе он выдает один выходной импульс. Но, естественно, этот один выход можно, продублировав, подать на вход целому множеству нейронов, а не только одному единственному. Важной характеристикой самого соединения заключается в том, что интенсивность передачи электро-импульса показывает степень активности нейрона.

Внутреннее устройство нейрона является черной коробкой, но считается крайне простым. Считается, что нейрон комбинируя входные нейроны выстреливает свой сигнал в выходное.

Обучение за счет соединений, точнее адаптации силы подсоединения к другим нейронам. В биологии все устроено сложнее, но нам это не важно как уже было сказано ранее.

Пусть мы решаем задачу бинаризации, т.е. задачу отделения одного множества объектов от другого. Для нас нейрон сводится к функции от n-переменных:

$$f(x_1, x_2, ..., x_n).$$

А в учитывая то, что мы хотим получить максимально простую функцию, то нет ничего проще линейной функции:

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{i=n} \omega_i x_i + \omega_0,$$

где $\omega = (\omega_0, \omega_1, ..., \omega_n)$ – параметр характеризующий функцию. Ответ нейрона мы будет интерпретировать знаком функции. В случае положительного знака считаем, что нейрон относит данный объект к + объектам, в случае отрицательного – к - объектам. При равенстве функции – считаем что имеет место неопределенность.

Для упрощения записи функции, можно считать, что например есть фиктивный признак (x_0) всегда равный 1, тогда формулу можно записать в виде:

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{i=n} \omega_i x_i + \omega_0 x_0 = \sum_{i=0}^{i=n} \omega_i x_i =< \omega, x >,$$

где последнее является скалярным произведением векторов $x = (x_0, x_1, ..., x_n)$ и $(\omega_0, \omega_1, ..., \omega_n)$. Иногда будет использована такая форма для сокращения записи.

Семейство функций есть. Как же устроен алгоритм обучения. Их может быть много. Рассмотрим традиционный.

Как было уже написано выше, знак величины

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{i=n} \omega_i x_i + \omega_0,$$

определяет принадлежность к определенному классу. Если взять пример пары (x^j, y^j), где x^j – вектор признаков j-примера, а y^j – ответ для j-примера (т.е. значение $+1$ или -1). Тогда согласно ранее описанному в случае, если персептрон не ошибается будет истинно:

$$f(x_1^j, x_2^j, ..., x_n^j) y^j > 0.$$

Соответственно в случае ошибки, величина будет меньше или равна нулю. Тогда мы должны проверить персептрон на всех примерах данных (обучающая выборка), и если он даст правильный ответ то ничего не делать. А вот в случае ошибки потребуется некая модификация вектора ω. В деталях последнего действия и заключается суть алгоритма. Для традиционного алгоритма она такова: Данные

Algorithm 1 Шаг персептрона

$a \leftarrow \sum_{i=1}^{i=n} \omega_i x_i^j + \omega_0^j$
if $ay^j \leq 0$ **then**
  $\omega \leftarrow w + y^j x^j$
  $\omega_0 \leftarrow \omega_0 + y^j$
end if

алгоритм вызывается для всех точек. Порядок пока будем считать не важным.

Покажем, что при таком обучении действительно можно ожидать настройки параметров под модель. Без ограничения общности можно считать, что нашим примером является $(x, 1)$. Тогда $a < 0$. Что будет если посчитать значение этого примера на обновленных параметрах? Будет следующее:

$$\acute{a} = \sum_{i=1}^{i=n} \acute{\omega}_i x_i + \acute{\omega}_0 = \sum_{i=1}^{i=n} (\omega_i + x_i) x_i + \omega_0 + 1 = \sum_{i=1}^{i=n} \omega_i x_i + \omega_0 + \sum_{i=1}^{i=n} x_i^2 + 1 =$$

$$= a + \sum_{i=1}^{i=n} x_i^2 + 1 > a.$$

Последнее показывает, что $\acute{a}$ увеличивается как минимум на 1 относительно a. Поэтому есть шанс того, что оно может стать больше 0 и тогда нейрон будет правильно классифицировать данный пример. Но конечно не ясно, какой будет результат давать нейрон при новых параметрах на других примерах, в частности тех, на которых ранее давался правильный ответ.

Граница Важным аспектом задания семейства функций – это то, какая "мощь"у этого семейства. Так, крайней важным является разделяющая граница множеств класса. Иначе говоря хотелось бы взглянуть на семейство функций с точки зрения геометрии.

Пусть пока $w_0 = 0$. Рассмотрим некое $\omega = (\omega_1, ..., \omega_n)$. Построим множество $x = (x_1, ..., x_n)$ов:

$$\mathcal{B} = \{x | \sum_{i=1}^{i=n} \omega_i x_i = 0\}.$$

Учитывая, что $\sum_{i=1}^{i=n} \omega_i x_i$ есть $< w, x >$, и вспоминая смысл равенства скаляра 0 следует, что множество $\mathcal{B}$ состоит из векторов перпендикулярных вектору ω:

$$\mathcal{B} \perp \omega, \mathcal{B} = \{x | x \perp \omega\}.$$

В двумерном случае, $\mathcal{B}$ есть вектор перпендикулярный вектору ω, в трехмерном (как и выше) это уже называется плоскостью (гиперплоскостью).

Заметим, что $\mathcal{B} \perp \omega \Leftrightarrow \mathcal{B} \perp 2\omega$, т.е. определение множества $\mathcal{B}$ не зависит от масштаба вектора ω. Поэтому для определенности можно положить его норму равной 1:$\|\omega\| = 1$.

Если $\|\omega\| = 1$, то $a =< x, \omega >$ есть величина проекции вектора x на ω. Тогда все вектора x, в частности, все входные данные подвергаются данной проекции. Фактически мы строи новый признак равный некоторой взвешенной сумме имеющихся признаков. В таком случае задачу разделения двух множеств будет сводится к поиску точки раздела на оси ω. Эта точка и задается величиной w_0.

Сходимость алгоритма Сходимость зависит от данных. Если в принципе такой разделяющей плоскости провести нельзя, то не алгоритм сойдется никогда.

Крайние случаи.

Пример 1. Признаков нет. Тогда можно менять только ω_0, которая и будет задавать знак. Если все-таки объекты принадлежать к разным классам, то такое b, чтобы эти два класса разделить, подобрать не удастся.

Пример 2. Нельзя разделить прямой объекты:

$$+-$$

$$-+$$

Таким образом для успешной гипотетической сходимости алгоритма должна существовать линейная разделимость данных, т.е. они в принципе должны быть разделимы гиперплоскостью.

$$\delta(X,w) = \begin{cases} \min_{j=1}^{j=l} y^j(< w, x^j > +\omega_0), & \text{если } \omega \text{ линейно разделяет } X. \\ -\infty, & \text{иначе.} \end{cases}$$

$$\delta = \delta(X) = \sup_{\omega} \delta(X,w).$$

Ясно, что если X не разделима, то $\delta = -\infty$.

Теорема. Если $\delta > 0$, $\forall x \in X \Rightarrow \|x\| \leq 1$, то предложенный ранее алгоритм сойдется за не более чем за $\frac{1}{\delta^2}$ обновлений.

Док-во. Раз $\delta > 0$, то $\exists x^*$ и ω^*, которые реализуют δ, т.е.

$$\forall x \in X, w \Rightarrow < w, x > +\omega_0 \geq \delta =< w^*, x^* > +\omega_0.$$

Пусть $\omega^{(0)}$ начальное значение вектора параметра итерационного процесса алгоритма. Тогда на k шаге раз $w^{(k)}$ обновили, то была ошибка на прошлом шаге на неком примере x, т.е. $y < w^{(k-1)}, x >< 0$. По построению алгоритма $w^{(k)} = w^{(k-1)} + yx$. Вычислим

$$< w^*, w(k) >=< w^*, w^{(k-1)} + yx >=< w^*, w^{(k-1)} > + < w^*, yx >\geq$$

$$\geq < w^*, w^{(k-1)} > +\delta,$$

т.е. проекция вектора w^k на правильный вектор постепенно увеличивается:

$$< w^*, w^{(k)} >> k\delta.$$

Теперь необходимо понять вектор $w^{(k)}$ увеличивает свой размер вдоль вектора w^* или нет. Для этого оценим скорость роста самого вектора $\|\omega^{(k)}\|$.

$$\|w^{(k)}\|^2 = \|\omega^{(k-1)} + yx\|^2 = \|\omega^{(k-1)}\|^2 + y^2\|x\|^2 + 2y < \omega^{(k-1)}, x >$$

учитывая, что $|y| = 1$, $\|x\| \leq 1$ и вспоминая, что $y < \omega^{(k-1)}, x >< 0$:

$$\leq \|\omega^{(k-1)}\|^2 + 1 + 0.$$

Откуда $\|\omega^{(k)}\|^2 \leq k$.

Тогда

$$\sqrt{k} \geq \|\omega^{(k)}\| \geq< w^*, w^{(k)} >\geq k\delta.$$

Откуда $\sqrt{k} \geq k\delta$. Следовательно $k \leq \frac{1}{\delta^2}$. ЧТД.