Linear Regression Classification

August 29, 2019

1 Two-class Linear Regression Classification

Input n data points:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \cdots \begin{pmatrix} x_n \\ y_n \end{pmatrix}, \quad x_i = \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix} \in \mathbb{R}^p, \quad y_i = \pm 1. \tag{1}$$

where y_i are class labels; the class representation are usually

$$\{1, -1\}, \text{or } \{0, 1\}, \text{or } \{1, 2\}. \tag{2}$$

These seemingly different representations are in fact equivalent. The differences are absorbed into the parameter b.

We want to fit the data to the linear function (Curve Fitting):

$$y = f(x) = \beta^T x + b. \tag{3}$$

Error to be minimized:

$$\begin{aligned} \min_{\beta,b} J(\beta, b) &= \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 \\ &= \sum_{i=1}^{n} \left(y_i - \left(\beta^T x_i + b\right)\right)^2 \\ &= \sum_{i=1}^{n} \left(y_i - \tilde{\beta}^T \tilde{x_i}\right)^2 \\ &= \| \, y^T - \tilde{\beta}^T \tilde{X} \, \|^2 \, . \end{aligned} \tag{4}$$

where

$$\tilde{x_i} = \begin{pmatrix} x_i \\ 1 \end{pmatrix},\ \tilde{\beta} = \begin{pmatrix} \beta \\ b \end{pmatrix},\ \tilde{\beta}^T\tilde{x_i} = \beta^T x_i + b,\ y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},\ \tilde{\beta}^T\tilde{X} = \begin{pmatrix} \tilde{\beta}^T\tilde{x}_1 & \tilde{\beta}^T\tilde{x}_2 & \cdots & \tilde{\beta}^T\tilde{x}_n \end{pmatrix}. \tag{5}$$

Here we use the $\tilde{x_i}$ and $\tilde{\beta}$ to eliminate the intercept/bias b. This simplifies the math. This technique is called "padding".

Then J is expanded as

$$\begin{aligned} J &= (y^T - \tilde{\beta}^T\tilde{X})(y^T - \tilde{\beta}^T\tilde{X})^T \\ &= (y^T - \tilde{\beta}^T\tilde{X})(y - \tilde{X}^T\tilde{\beta}) \\ &= y^Ty - 2y^T\tilde{X}^T\tilde{\beta} + \tilde{\beta}^T\tilde{X}\tilde{X}^T\tilde{\beta}. \end{aligned} \tag{6}$$

Set the derivative of J with respect to $\tilde{\beta}$, to zero:

$$\begin{aligned} \frac{\partial J}{\partial \tilde{\beta}} = \ -2\tilde{X}y + 2\tilde{X}\tilde{X}^T\tilde{\beta}^* &= 0 \\ \tilde{X}\tilde{X}^T\tilde{\beta}^* &= \tilde{X}y \\ \tilde{\beta}^* &= (\tilde{X}\tilde{X}^T)^{-1}\tilde{X}y. \end{aligned} \tag{7}$$

Note that for two vectors a, b, the derivative is

$$\frac{\partial}{\partial b_i}a^Tb = \frac{\partial}{\partial b_i}b^Ta = \frac{\partial}{\partial b_i}(\sum_j a_jb_j) = a_i,\ \text{for all } i. \tag{8}$$

We often express the above equality (in element form) as an equality in vector form:

$$\frac{\partial}{\partial b}a^Tb = a. \tag{9}$$

Also, if A is a matrix, b is a vector, then $\frac{\partial}{\partial b}b^TAb = 2Ab$.

Once we obtained the model parameters $(\beta, b) = \tilde{\beta}$, the model training process is done. Now, for a new data object x_t (this is called a test/query object), we use the trained classifier (the trained linear function) to predict (also called classify) the class label

$$y_t = \text{sign}(\tilde{\beta}^Tx_t),\ \text{ assume training class labels are } \{1, -1\} \tag{10}$$

or

$$y_t = \text{sign}(\tilde{\beta}^Tx_t - 1.5),\ \text{ assume training class labels are } \{1, -2\} \tag{11}$$

An useful relation is the following

$$\sum_{i=1}^{n}(y_i-\bar{y})^2=\sum_{i=1}^{n}(y_i-f(x_i))^2+\sum_{i=1}^{n}(f(x_i)-\bar{y})^2 \tag{12}$$

2 Multi-class Linear Regression Classification

Multi-class classification refers to the cases when the number of label classes (we use K) is bigger or equal than 3.

Class Numeral Representation

For 2-class classification problems, Eq.2 lists several ways to represent class labels.

For multi-class classification problems, one intuitive and natural way to represent class labels is to set

$$y_i \in \{1,2,\cdots,K\} \tag{13}$$

This simple method is the most commonly used one when discussing classification.

Using this representation, we can do multi-class classification by using the above 2-class linear regression classification, and the class prediction formula Eq.(8) is modified to

$$y_t=\begin{cases}1, & \text{if } f(x_t)<1.5\\ 2, & \text{if } 1.5\le f(x_t)<2.5\\ 3, & \text{if } f(x_t)\ge 2.5\end{cases} \tag{14}$$

Here we assume $K=3$ and $f(x)=\tilde{\beta}^T x_t$.

However, this simple multi-class classification has a significant flaw. The flaw is in class label representation. In this class label representation, the distances between different classes are different.

For example, assume x_i, x_j, x_k belong to class 1, 2, 3 respectively. Thus $y_i=1, y_j=2, y_k=3$. The class distance can be computed using their class label representations y_i, y_j, y_k.

Thus the distance between y_i and y_k (the distance between class 1 and class 3) can be computed as

$$\parallel y_i-y_k \parallel = \mid 1-3 \mid = 2 \tag{15}$$

Similarly, the distance between y_i and y_j (the distance between class 1 and class 2) is computed as is large than distance(class 1 , class 3):

$$\| y_i - y_j \| = |1 - 2| = 1 \tag{16}$$

If the data has 26 classes for 26 English letters

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

the distances between 'a' and 'z' will be 25, and the distance between 'a' and 'b' will be 1.

Clearly, the distances between different classes in this representation do not correctly capture the difference of various classes in human perception. For example, distance("u","v")=1, and distance("u","n") = 8. But in real English, "uv" never occur, but "un" occurs in "united", "universe" etc. In other words, the smaller distance of (u,v) does not imply they appear close-by in English words. Thus the distance between classes is not correctly captured by the representation. We therefore conclude that the class representation Eq.(13) is not appropriate for linear regression classification. (We will discuss class distance later.)

Class Indicator Representation

In this representation, we use a K-dimensional vector to represent the class. For K = 3, they are

$$y_i^{c_1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad y_j^{c_2} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad y_k^{c_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{17}$$

Using this indicator represention, the distance between classes can be computed as

$$\| y_i{}^{c_1} - y_j{}^{c_2} \| = \left\| \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\| = \left\| \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \right\| = \sqrt{1^2 + (-1)^2} = \sqrt{2}.$$

$$\| y_i{}^{c_1} - y_j{}^{c_3} \| = \left\| \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\| = \left\| \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \right\| = \sqrt{1^2 + (-1)^2} = \sqrt{2}.$$

Thus, class label distances between any two classes are equal.

2.1 Class Label Distance

There is no clear theory on what is the best way to specify the class label distance.

One way to approach this question is to compute the distances between feature vectors in different classes. There exits many different way to define this distances. They are generally called **class separation index**.

However, different class are different concepts in human perception. Class separation index are distances in feature vector space, which depends on the representation of objects. The same object can be represented by different features; Thus distances in feature space are not inherent properties. Hence it is not clear why distances between different concepts should be related to feature representations.

To avoid these deeper questions, the general approach is to assume we have prior information on concept distances; in this situation, the most natural solution is that different concepts/classes have equal distance: from class c_1's point of view, all other classes are equally distant. No more. No less. For this reason, class indicator representation is the natural choice.

Hamming distance. More general, we can view the class variable as a **categorical** variable, like weather variable whose values are 'sunny', 'cloudy', 'rainly'. Typically, we use Hamming distance on categorical variables. In this point of view, the distance between different classes are the same value. In Hamming distance, this value is 1. Using indicator vector, this value is $\sqrt{2}$. The exact value of this number is un-important. The importance is that Hamming distance is consistent with the indicator vector representation.

2.2 Multi-class Linear Regression derivation

Function of $f(x)$ is redefined as:

$$f(x_i) = \begin{bmatrix} \beta_1^T x_i + b_1 \\ \beta_2^T x_i + b_2 \\ \vdots \\ \beta_K^T x_i + b_K \end{bmatrix} = (\beta_1\, \beta_2\, \cdots\, \beta_K)^T x_i + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix} = \left[\begin{pmatrix} \beta_1 \\ b_1 \end{pmatrix} \cdots \begin{pmatrix} \beta_K \\ b_K \end{pmatrix} \right]^T \begin{pmatrix} x_i \\ 1 \end{pmatrix} = \tilde{B}^T \tilde{x}_i$$

Error to be minimized:

$$\begin{aligned} J &= \sum_{k=1}^{K}\sum_{i=1}^{n}\left(y_i^k - (\beta_k^T x_i + b_k)\right)^2 \\ &= \sum_{i=1}^{n}\left(y_i - f(x_i)\right)^2 \\ &= \| \underbrace{(y_1\ y_2\ \cdots\ y_n)}_{K\times n\ Matrix} - \underbrace{(f(x_1)\ f(x_2)\ \cdots\ f(x_n))}_{K\times n\ Matrix} \| \\ &= \left|\left|\ Y - \left(\tilde{B}^T\tilde{x_1}\ \tilde{B}^T\tilde{x_2}\ \cdots\ \tilde{B}^T\tilde{x_n}\right)\right|\right|_F^2 \\ &= \left|\left|\ Y - \tilde{B}^T\tilde{X}\right|\right|_F^2 \end{aligned}$$

Note that $\|A\|_F^2 = trace(AA^T) = \sum_{ij} A_{ij}^2$.

Then J is expanded as

$$J = trace\left(Y^TY - 2Y^T(\tilde{B}^T\tilde{X}) + \tilde{B}^T\tilde{X}\tilde{X}^T\tilde{B}\right). \tag{18}$$

Setting the derivative of J with respect to $\tilde{B}$ to zero:

$$\begin{aligned} \frac{\partial J}{\partial \tilde{B}} = \ -2\tilde{X}Y^T + 2\tilde{X}\tilde{X}^T\tilde{B}^* &= 0 \\ \tilde{X}\tilde{X}^T\tilde{B}^* &= \tilde{X}Y^T \\ \tilde{B}^* &= (\tilde{X}\tilde{X}^T)^{-1}\tilde{X}Y^T. \end{aligned} \tag{19}$$

Note that

$$\frac{\partial}{\partial A_{ij}} trace(AB) = \frac{\partial}{\partial A_{ij}}\left(\sum_{k\ell} A_{k\ell}B_{\ell k}\right) = B_{ji}. \tag{20}$$

This element-form equality can be equivalently written as the matrix-form equality: $\frac{\partial}{\partial A} trace(AB) = B^T$. Note also $\frac{\partial}{\partial A} trace(A^TCA) = 2CA$.

Once the classifier is trained, i.e., B is obtained, the classifier classifies a new query data object x_t using the rule

$$y_t = \arg\max_{1\le k\le K}\ \tilde{\beta}_k^T\tilde{x}_t = B^T\tilde{x}_t \tag{21}$$

Matlab implementation. The expression $(\tilde{X}\tilde{X}^T)^{-1}\tilde{X}$ is the pseudo inverse of $\tilde{X}$ and computed as pinv($\tilde{X}$) in matlab.

2.3 Regularization

$$\begin{aligned} J_{reg} &= \sum_{k=1}^{K} \Big[\sum_{i=1}^{n} \big(y_i^k - (\beta_k^T x_i + b_k)\big)^2 + c\|\beta_k\|_2^2 \Big] \\ &= \big|\big|\ Y - B^T X - be^T \big|\big|_F^2 + c\|B\|_F^2 \end{aligned}$$

where $e = (1 \cdots 1)^T$

Setting the derivative of J with respect to b to zero: we obtain

$$b = (Y - B^T X)e/n = \bar{y} - B^T \bar{x},\ \bar{y} = Ye/n,\ \bar{x} = Xe/n$$

where $\bar{x}$ is the mean of X, $\bar{y}$ is the mean of Y.

Substitute b into above equation, we have

$$J_{reg} = \big|\big|\ \bar{Y} - B^T \bar{X} \big|\big|_F^2 + c\|B\|_F^2,\ \bar{Y} = Y - \bar{y}e^T,\ \bar{X} = X - \bar{x}e^T,$$

The optimal solution for B is

$$B_{reg}^* = (\bar{X}\bar{X}^T + cI)^{-1}\bar{X}\bar{Y}^T.$$

where I is the p-by-p identity matrix. Once B_{reg}^*, b are obtained, the class label of a query data point x_t, is

$$y_t = \arg\max_{1 \le k \le K} f_k(x_t) = b_{k,reg}^T x_t + b_k. \tag{22}$$

The parameter b here is the same as in un-regularized case. The regression coefficients are usually suppressed (become smaller) from the un-regularized case.

2.4 Semi-supervised transductive Learning with Regularization

Suppose the input data contains a small labeled data L and a large unlabeled data U. Our task is to learn the class labels of the unlabeled data.

Let the data be represented as $X = [X_L\ X_U]$ and $Y = [Y_L\ Y_U]$. Here X_L, X_U, Y_L are known; $Y_U]$ is unknown and to be predicted. The transductive learning is to solve the model

$$\min_{Y_U, B}\ J_{td}(Y_U, B) = \big|\big|\ [Y_L\ Y_U] - B^T[X_L\ X_U] \big|\big|_F^2 + c\|B\|_F^2 \tag{23}$$

This is a convex optimization and there is a unique global solution for (Y_U, B).

We solve this optimization problem using a EM type algorithm. Note that

$$J_{td}(Y_U, B) = \left|\left|\ Y_L - B^T X_L\right|\right|_F^2 + \left|\left|\ Y_U - B^T X_U\right|\right|_F^2 + c\|B\|_F^2. \tag{24}$$

The algorithm to solve this problem is the following.

Step 0. Learn B using labeled data only.

$B = \arg\min_B \ \left|\left|\ Y_L - B^T X_L\right|\right|_F^2 + c\|B\|_F^2$.

Do iteration $t = 1, 2, 3, \cdots$ until converge. In each iteration, do

Step E: Set $Y_U = \text{discretize}(B^T X_U)$;

Step M: Solve for B on all data while fixing Y_U.

This completes the algorithm. Here "discretize" operation on a vector means setting the largest element to 1, and the rest to 0. An example is

$$\text{discretize} \begin{bmatrix} 0.1 \\ 0.4 \\ 0.34 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Another choice is to replace "discretize" by "probabilistic" using a softmax function:

$$\text{probabilistic} \begin{bmatrix} 0.1 \\ 0.4 \\ 0.34 \\ -0.5 \end{bmatrix} = \begin{bmatrix} e^{0.1}/s \\ e^{0.4}/s \\ e^{0.35}/s \\ e^{-0.5}/s \end{bmatrix} = \begin{bmatrix} 0.275 \\ 0.371 \\ 0.353 \\ 0.002 \end{bmatrix}$$

where $s = e^{0.1} + e^{0.4} + e^{0.35} + e^{-0.5} = 4.023$. The result is a probability distribution and the sum of all elements is equal to 1. For a matrix $A = (a_1\ a_2\ \cdots\ a_n)$, the discretize or probabilistic operations are applied to every column independently.

The objective J_{td} in Eq.24 decreases in step E because the second term vanishes and other terms remain fixed. J_{td} in Eq.23 decrease because B is the global optimal solution when Y_U is fixed.

Thus the objective J_{td} in decrease monotonically during the iterations. Also J_{td} is bounded from below (it is greater than 0). These two properties ensure the iteration will converge. Since the optimization is a convex problem, the iteration will converge to the unique global solution.