Regressione Lineare e Regressione Logistica

Stefano Gualandi
Università di Pavia, Dipartimento di Matematica

email: stefano.gualandi@unipv.it
twitter: @famo2spaghi
blog: http://stegua.github.com

Introduzione

Il problema di *apprendere* dai dati si definisce tramite tre componenti principali:

1. Un generatore di vettori casuali x, estratti a caso da una data distribuzione $P(x)$, non nota.
2. Un **supervisore** che restituisce un vettore di output y per ogni vettore di input x, seguendo una data distribuzione condizionata $P(y|x)$, non nota.
3. Un algoritmo di apprendimento, capace di implementare un insieme di funzioni $f(x, w)$ con $w \in \Lambda$.

Il problema consiste nel determinare una funzione dell'insieme $f(x, w), w \in \Lambda$ che predice la risposta del supervisore nel miglior modo possibile.

Funzione di rischio

In pratica, si deve misurare la **distanza** tra la risposta y del supervisore per un dato vettore di input x e la risposta $f(x, w)$ restituita dall'algoritmo di apprendimento

$$L(y, f(x, w))$$

Questa viene chiamata funzione di LOSS.

Si considera il valore atteso di questa distanza, definendo la funzione di rischio

$$R(w) = \int L(y, f(x, w))\, dP(x, y) \tag{1}$$

L'obbiettivo è trovare la funzione $f(x, w^*)$ che minimizza $R(w)$, avendo a disposizione solo il training set:

$$(x_1, y_1), \ldots, (x_N, y_N). \tag{2}$$

Esempi di funzioni di loss

1. Pattern Recognition (Problema di Classificazione):

$$L(y, f(x, w)) = \begin{cases} 0 & \text{se } y = f(x, w) \\ 1 & \text{se } y \neq f(x, w) \end{cases} \tag{3}$$

2. Stima della funzione di regressione:

$$L(y, f(x, w)) = (y - f(x, w))^2 \tag{4}$$

3. Stime della funzione di densità:

$$L(p(x, w)) = -\log p(x, w) \tag{5}$$

Funzione di rischio empirico

Poiché la distribuzione $P(x, y)$ non è nota si usa la funzione di rischio empirica

$$\hat{R}(w) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i, w)) \tag{6}$$

in cui si calcola la media dei valori della funzione di loss nei punti del training set.

Funzione di rischio empirico

Poiché la distribuzione $P(x, y)$ non è nota si usa la funzione di rischio empirica

$$\hat{R}(w) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i, w)) \tag{6}$$

in cui si calcola la media dei valori della funzione di loss nei punti del training set.

Si deve quindi risolvere il problema di ottimizzazione

$$\hat{w} = \arg\min_{w \in \Lambda} \hat{R}(w) = \arg\min_{w \in \Lambda} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i, w)) \tag{7}$$

Regressione Lineare

Sia $x \in \mathbb{R}^p$ e si consideri la classe di funzioni $f(x, w)$ lineari

$$y = f(x, w) = w_0 + \sum_{i=j}^{p} x_j w_j. \tag{8}$$

Per semplificare la notazione, si include nel vettore x il termine 1, in modo da includere w_0 nel vettore w. Si ottiene quindi

$$y = f(x, w) = x^T w. \tag{9}$$

Regressione Lineare e Minimi quadrati

Dato un insieme di campioni di punti i.i.d. $z_i = (x_i, y_i)$ con $i = 1, \ldots, N$, si vuole minimizzare il rischio empirico

$$\hat{R}(w) = \frac{1}{N} \sum_{i=1}^{N} (y_i - x_i^T w)^2 = (y - Xw)^T (y - Xw), \qquad (10)$$

in cui X è una matrice $N \times (p+1)$ e y è il vettore degli N output del training set.

Regressione Lineare e Minimi quadrati

Dato un insieme di campioni di punti i.i.d. $z_i = (x_i, y_i)$ con $i = 1, \ldots, N$, si vuole minimizzare il rischio empirico

$$\hat{R}(w) = \frac{1}{N} \sum_{i=1}^{N} (y_i - x_i^T w)^2 = (y - Xw)^T (y - Xw), \qquad (10)$$

in cui X è una matrice $N \times (p+1)$ e y è il vettore degli N output del training set.

Si deriva la (10) rispetto w e si definisce l'**equazione normale**

$$X^T (y - Xw) = 0 \qquad (11)$$

Se la matrice X non è singolare, allora l'unica soluzione è

$$\hat{w} = (X^T X)^{-1} X^T y. \qquad (12)$$

Plot

Esercizio 1

Implementare una classe chiamata `RegressioneLineare` che ha due metodi:

1. `fit(x,y)`: trova i parametri w risolvendo l'equazione normale
2. `predict(x)`: predice l'output y per il dato vettore di input x

Regressione Logistica con due sole classi

Quando ci sono solo due classi, poniamo

1. $y_i = 1$, quando il campione i è nella prima classe
2. $y_i = 0$, quando è nella seconda classe

Abbiamo quindi

$$\begin{aligned} p_1(x, w) &= p(x, w) = \frac{e^{x^T w}}{1 + e^{x^T w}} \\ p_2(x, w) &= 1 - p(x, w) = \frac{1}{1 + e^{x^T w}} \end{aligned}$$

Regressione Logistica: Funzione di Loss

La funzione di rischio empirico diventa

$$
\begin{aligned}
\hat{R}(w) &= -\sum_{i=1}^{N} \left(y_i \log p(x_i, w) + (1 - y_i) \log(1 - p(x_i, w)) \right) \\
&= -\sum_{i=1}^{N} \left(y_i \log \frac{e^{x_i^T w}}{1 + e^{x_i^T w}} + (1 - y_i) \log \frac{1}{1 + e^{x_i^T w}} \right) \\
&= -\sum_{i=1}^{N} \left(y_i x_i^T w - \log(1 + e^{x^T w}) \right)
\end{aligned}
$$

Regressione Logistica: Funzione di Loss

La funzione di rischio empirico diventa

$$\begin{aligned}
\hat{R}(w) &= -\sum_{i=1}^{N} (y_i \log p(x_i, w) + (1 - y_i) \log(1 - p(x_i, w))) \\
&= -\sum_{i=1}^{N} \left(y_i \log \frac{e^{x_i^T w}}{1 + e^{x_i^T w}} + (1 - y_i) \log \frac{1}{1 + e^{x_i^T w}} \right) \\
&= -\sum_{i=1}^{N} \left(y_i x_i^T w - \log(1 + e^{x^T w}) \right)
\end{aligned}$$

Per trovare il valore $\hat{w}$ che dia il minimo della funzione precedente dobbiamo risolvere le $p + 1$ equazioni non lineari in w

$$\frac{\partial \hat{R}(w)}{\partial w} = -\sum_{i=1}^{N} x_i \left(y_i - \frac{e^{x_i^T w}}{1 + e^{x_i^T w}} \right) = 0. \qquad (13)$$

Minimizzare la Funzione di Loss

Per risolvere le equazioni non lineari usiamo l'algoritmo di Newton Raphson che calcola il valore otteniamo w in maniera iterativa:

$$w^{j+1} = w^j - \left(\frac{\partial^2 \hat{R}(w)}{\partial w \partial w^T} \right)^{-1} \frac{\partial \hat{R}(w)}{\partial w}, \tag{14}$$

in cui tutte le derivate sono valutate in w^j.

Minimizzare la Funzione di Loss

In forma matriciale, si può dimostrare che

$$\frac{\partial \hat{R}(w)}{\partial w} = -X^T(y - p) \tag{15}$$

$$\frac{\partial^2 \hat{R}(w)}{\partial w \partial w^T} = X^T W X \tag{16}$$

In cui p è il vettore delle probabilità calcolate per l'i-esimo campione di dati $p(x_i, w^j)$, e W la matrice $N \times N$ con l'i-esimo elemento diagonale pari a $p(x_i, w^j)(1 - p(x_i, w^j))$.

Minimizzare la Funzione di Loss

Il passo di Newton Raphson diventa quindi

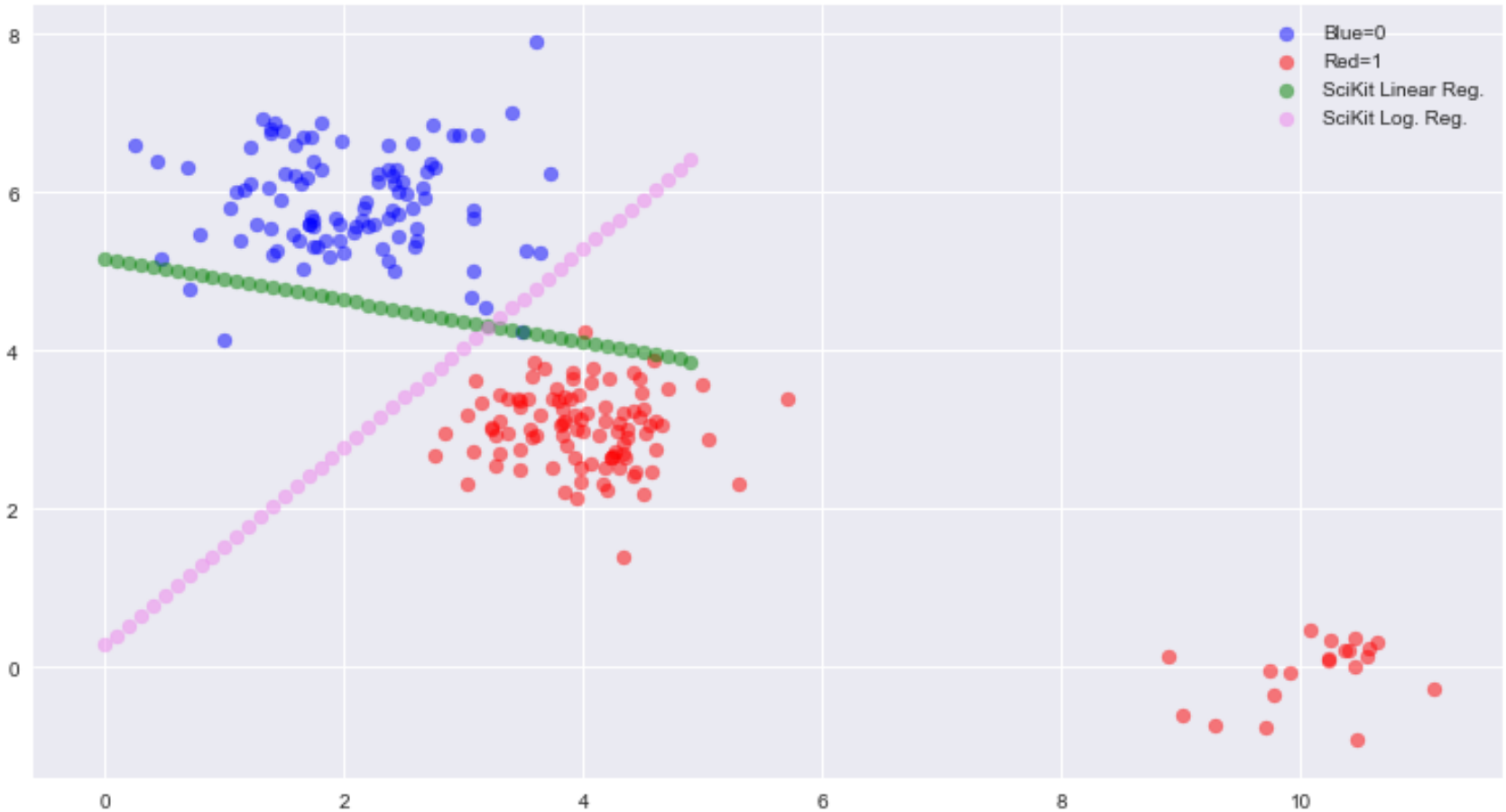
$$\begin{aligned} w^{j+1} &= w^j + (X^T W X)^{-1} X^T (y - p) \\ &= (X^T W X)^{-1} X^T W (X w^j + W^{-1}(y - p)) \\ &= (X^T W X)^{-1} X^T W z, \end{aligned}$$

in cui, per semplificare la notazione, abbiamo prima moltiplicato w^j per $(X^T W X)^{-1}(X^T W X)$ e poi abbiamo moltiplicato $(y - p)$ per $W^{-1}W$. In pratica abbiamo riformulato il passo di Newton Raphson come un passo di minimi quadrati pesati, con risposta

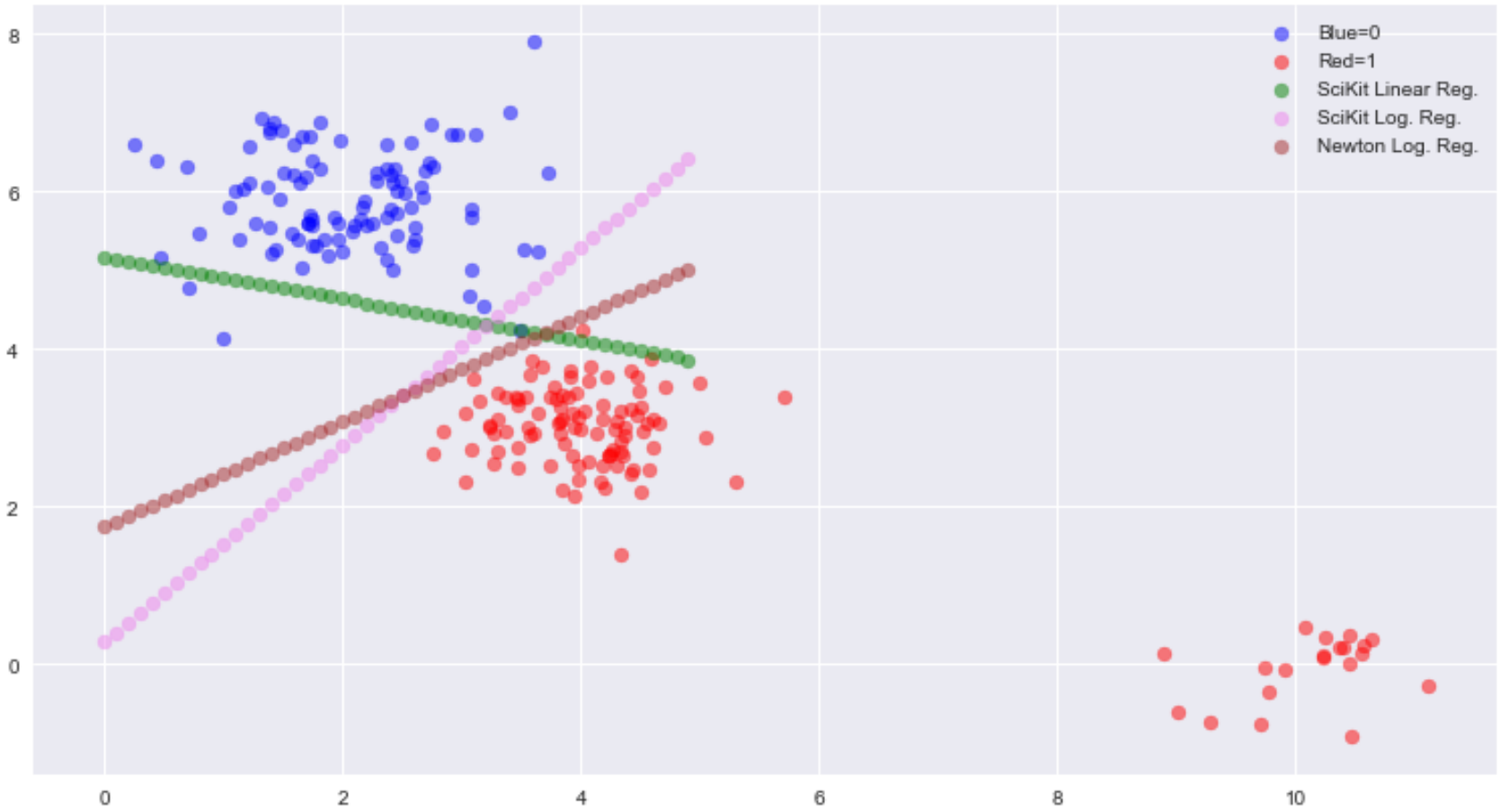
$$z = X w^j + W^{-1}(y - p).$$

Plot

Plot



Plot


Blue=0
Red=1
SciKit Linear Reg.
SciKit Log. Reg.
Newton Log. Reg.
8
6
4
2
0
0
2
4
6
8
10

Esercizio 2

Implementare una classe chiamata `RegressioneLogistica` che ha due metodi:

1. `fit(x,y)`: trova i parametri w utilizzando il metodo di Newton Raphson
2. `predict(x)`: predice l'output y per il dato vettore di input x

ATTENZIONE: Si ponga un numero massimo di iterazioni, e si controlli che il valore della funzione di loss non diventi troppo piccolo (rischio di instabilità numerica!)

Funzione di Numpy da usare

- `from numpy import` **matmul**: moltiplicazioni tra matrici
- `from numpy import` **transpose**: calcola la trasposta di una matrice
- `from numpy.linalg import` **inv**: calcola l'inversa di una matrice
- `from numpy.linalg import` **pinv**: calcola la pseudo inversa
- Altre funzioni di cui si deve consultare la documentazione:

`numpy.array`, `numpy.matrix`, `numpy.append`,
`numpy.diag`, `numpy.vectorize`