UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Machine learning

Logistic regression notes

Helder Daniel

hdaniel@ualg.pt

v2024/5 1.0

Variable definition

- We assume that we have 2 input variables **X** and **y**

	X				Y
	x1	**x2**	**…**	**xn**	
sample 0	**1**	**3**		**-4**	**2**
sample 1	**3**	**2**		**1**	**1**
…					
sample m	**5**	**-2**		**4**	**0**

- **X** is an *m x n* matrix with features
- **y** is an *m x 1* vector with targets

- *m* is the number of samples (or data points)
- *n* is the number of features

Variable definition

- We assume that we have 2 input variables **X** and **y**

$$\mathbf{X}_{m\times(1+n)} = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(n)} \\ 1 & x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(n)} \\ & & \vdots & & \\ 1 & x_m^{(1)} & x_m^{(2)} & \dots & x_m^{(n)} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

- **X** is an *m x n* matrix with features
- **y** is an *m x 1* vector with targets

- *m* is the number of samples (or data points)
- *n* is the number of features

Gradient descent update with regularization

Gradient descent update with regularization

$$\theta_0 = \theta_0 \qquad\qquad - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \qquad , \quad j = 0$$

$$\theta_j = \theta_j \left(1 - \alpha \frac{\lambda}{m}\right) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad , \quad j = 1, 2, \ldots, n$$

- Note that θ_0 is not updated

- θ_0 coefficient sets the distance from the axis

Gradient descent update with regularization

$$\theta_0 = \theta_0 \qquad - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \qquad , \quad j = 0$$

$$\theta_j = \theta_j \left(1 - \alpha \frac{\lambda}{m}\right) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad , \quad j = 1, 2, \ldots, n$$

- We need to multiply $\theta_{j>0}$ by: $\left(1 - \alpha \frac{\lambda}{m}\right)$

Gradient descent update with regularization

$$\theta_0 = \theta_0 \qquad\qquad - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)} \qquad , \quad j = 0$$

$$\theta_j = \theta_j \left(1 - \alpha \frac{\lambda}{m}\right) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad , \quad j = 1, 2, \ldots, n$$

- The vectorial form without regularization is:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \frac{1}{m} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T \mathbf{X}$$

Gradient descent update with regularization

- To avoid updating θ_0 we can form matrix **R:**

$$\mathbf{R}_{(n+1)\times(n+1)} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & \left(1-\alpha\frac{\lambda}{m}\right) & \dots & 0 \\ & & \vdots & \\ 0 & 0 & \dots & \left(1-\alpha\frac{\lambda}{m}\right) \end{bmatrix}$$

and update with:

$$\boldsymbol{\theta} = \mathbf{R}\boldsymbol{\theta} - \alpha\frac{1}{m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T\mathbf{X}$$

Gradient descent update with regularization

To compute in Python use Numpy specialized functions

$$\boldsymbol{\theta} = \mathbf{R}\boldsymbol{\theta} - \alpha \frac{1}{m} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T \mathbf{X}$$

np.add

np.multiply

np.dot
…

Gradient descent update with regularization

This translates to Python as below:

$$\boldsymbol{\theta} = \mathbf{R}\boldsymbol{\theta} - \alpha \frac{1}{m} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T \mathbf{X}$$

```
θ = R.dot(θ)-(α/m)*(X.dot(θ)-y).T.dot(X))
```

Gradient descent update with regularization

Or we can:

1) save θ_0

2)multiply all θ

3) replace θ_0 with the saved value

Gradient descent update with regularization

This translates to Python as below:

$$\boldsymbol{\theta} = \mathbf{R}\boldsymbol{\theta} - \alpha \frac{1}{m} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T \mathbf{X}$$

```
theta0 = θ[0]
θ = θ * (1-α *λ/m) - (α/m)*(X.dot(θ)-y).T.dot(X))
θ[0] = theta0
```

Logistic Regression

Gradient descent update
with regularization

Gradient descent update with regularization for Logistic Regression

to apply the logistic function:

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

Gradient descent update with regularization for Logistic Regression

to apply the logistic function:

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

first compute: **z = X θ**

Gradient descent update with regularization for Logistic Regression

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

to apply the logistic function:

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

first compute: **z = X θ**

Then: **gz** = sigmoid(z)

Gradient descent update with regularization for Logistic Regression

to apply the logistic function:

$$h_\theta(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

first compute: **z = X θ**

Then: **gz** = sigmoid(z)

and replace **X θ** by **gz** in:

$$\boldsymbol{\theta} = \mathbf{R}\boldsymbol{\theta} - \alpha\frac{1}{m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T\mathbf{X}$$

Logistic Regression

with

feature expansion

Logistic Regression with feature expansion

- Before **fit(Xe, y)** expand **X** matrix with:

  Xe = mapFeatures(X1, X2, degree)

- before **predict(xpe)** expand **xp** vector with:

  xpe = mapFeatures(xp[:,0], xp[:,1], degree)