Data Mining: LR lab

Jakramate Bootkrajang

LR Algorithm

- A discriminative model for classification
  - p(y=1|x) = 1/(1+exp(-w*x))
  - p(y=0|x) = 1 - p(y=1|x)
- Need to estimate weight vector from data
- Weight vector can be estimated using iterative optimisation algorithm such as the Newton’s method

Psuedo code

Pseudo Code

1. $\mathbf{w} \leftarrow \mathbf{0}$
2. Make sure class label vector $\mathbf{y}$ is in $\{0, 1\}$ format
3. Compute $\mathbf{p}_1$ by setting its elements to

$$p(y = 1|\mathbf{x}_i; \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}_i}}{1 + e^{\mathbf{w}^T \mathbf{x}_i}}$$

4. Compute diagonal matrix $\mathbf{Q}$ by setting its diagonal elements to $p(y = 1|\mathbf{x}_i; \mathbf{w})(1 - p(y = 1|\mathbf{x}_i; \mathbf{w}))$
5. $\mathbf{w}^{new} = \mathbf{w}^{old} + \eta(\mathbf{X}^T \mathbf{Q} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{p}_1)$
6. Until stopping criteria is met (usually $|\mathbf{w}^{new} - \mathbf{w}^{old}| < \epsilon$)

Loading packages

- Data is in CSV format
- Loading all packages needed

```
using CSV, LinearAlgebra, Statistics
```

Defining sigmoid function

In [ ]:
```
sigmoid(w, x) = exp.(x*w)./(1 .+ exp.(x*w))
```

Loading data

- Load `ionosphere.csv’

```
d = CSV.read("ionosphere.csv")
```

- You should get

351 rows × 35 columns

	1	2	3	4	5	6	
	Int64	Int64	Float64	Float64	Float64	Float64	Floa
1	1	0	0.99539	-0.05889	0.85243	0.02306	0.8
2	1	0	1.0	-0.18829	0.93035	-0.36156	-0.1
3	1	0	1.0	-0.03365	1.0	0.00485	
4	1	0	1.0	-0.45161	1.0	1.0	0.7
5	1	0	1.0	-0.02401	0.9414	0.06531	0.9
6	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.0
7	1	0	0.97588	-0.10602	0.94601	-0.208	0.9

About the data

- d is a dataframe (excel-like table)
- Col 1 to 34 are features / col 35 is label

```
d[:,35]
```

```
351-element Array{Union{Missing, String},1}:
 "g"
 "b"
 "g"
 "b"
 "g"
 "b"
 "g"
 "b"
 "g"
 "b"
 "g"
 "b"
 "g"
```

Data preprocessing

- We will convert dataframe to data matrix by selecting only all except the 2nd component

```
X = convert(Array ,[d[1] d[3:34]])
```

- Also perform z-score normalisation

```
X = convert(Array ,[d[1] d[3:34]])
X = (X - repeat(mean(X,dims=1),351,1))
      ./ repeat(std(X,dims=1), 351, 1)
```

Data preprocessing

- Adding bias term to X

```
X = [ones(size(X,1)) X]
```

Preprocessing label vector

- Convert from dataframe to array
- Convert “b” to 0 and “g” to 1

```
y = convert(Array, ones(size(d[:,end])))
y[d[:,end].== "b"] .= 0
```

The learning [1]

Pseudo Code

1. $\mathbf{w} \leftarrow \mathbf{0}$

- Initialising weight vector to zero vector (can be random vector)
- And fix learning rate (eta) to 0.3

```
# initialising weight vector
w = zeros(size(X,2))
η = 0.3
```

- Type ‘\eta + Tab’ to get the Greek letter

The learning [2]

Pseudo Code

1. $\mathbf{w} \leftarrow \mathbf{0}$
2. Make sure class label vector $\mathbf{y}$ is in $\{0, 1\}$ format

- Already did !

The learning [3]

Pseudo Code

1. $\mathbf{w} \leftarrow \mathbf{0}$
2. Make sure class label vector $\mathbf{y}$ is in $\{0, 1\}$ format
3. Compute $\mathbf{p}_1$ by setting its elements to

$$p(y = 1|\mathbf{x}_i; \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}_i}}{1 + e^{\mathbf{w}^T \mathbf{x}_i}}$$

```
p1 = sigmoid(w,X)
```

The learning [4]

```
Q = Diagonal(sigmoid(w,X).*(1 .- sigmoid(w,X)))
```

4. Compute diagonal matrix $\mathbf{Q}$ by setting its diagonal elements to $p(y = 1|\mathbf{x}_i; \mathbf{w})(1 - p(y = 1|\mathbf{x}_i; \mathbf{w}))$
5. $\mathbf{w}^{new} = \mathbf{w}^{old} + \eta(\mathbf{X}^T\mathbf{Q}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{p}_1)$
6. Until stopping criteria is met (usually $|\mathbf{w}^{new} - \mathbf{w}^{old}| < \epsilon$)

The learning [5]

```
w_new = w + (η * inv(X'*Q*X .+ 1) * X' * (y-p1))
```

4. Compute diagonal matrix $\mathbf{Q}$ by setting its diagonal elements to $p(y = 1|\mathbf{x}_i; \mathbf{w})(1 - p(y = 1|\mathbf{x}_i; \mathbf{w}))$
5. $\mathbf{w}^{new} = \mathbf{w}^{old} + \eta(\mathbf{X}^T\mathbf{Q}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{p}_1)$
6. Until stopping criteria is met (usually $|\mathbf{w}^{new} - \mathbf{w}^{old}| < \epsilon$)

The learning [6]

```
while true
    p1 = sigmoid(w,X)
    Q = Diagonal(sigmoid(w,X).*(1 .- sigmoid(w,X)))
    w_new = w + (η * inv(X'*Q*X .+ 1) * X' * (y-p1))
    if sum(abs.(w_new.-w)) < 0.1
        break
    else
        w = w_new
    end
end
```

6. Until stopping criteria is met (usually $|\mathbf{w}^{new} - \mathbf{w}^{old}| < \epsilon$)

Using optimised w to predict x_q

- Calculate $p(y=1|x_q) = \text{sigmoid}(w, x_q)$
- If $p(y=1|x_q) > 0.5$
  - Predict $y_q = 1$
- Else
  - Predict $y_q = 0$

The code for predicting training data

```
yh = sigmoid(w,X)
yh[yh .<0.5] .= 0
yh[yh .>0.5] .= 1
err = sum(yh .!= y)
println(err/length(y))
```

The End

- Thanks !
- Questions ?