Bayesian Logistic Regression

Sargur Srihari, Tianhang Zheng
University at Buffalo, State University of New York

Tasks

- Train the Bayesian Logistic Regression and acquire the hyper parameters
- Prediction based on the hyper parameters

Hyper parameters acquisition

- Randomly choose two priors: m_0, S_0

$$p(\boldsymbol{w}) = N(\boldsymbol{w}|\boldsymbol{m}_0, \mathrm{S}_0)$$

- Gaussian approximation to posterior

$$q(\boldsymbol{w}) = N(\boldsymbol{w} \mid \boldsymbol{w}_{\mathrm{map}}, S_N)$$

Hyper parameters acquisition

- Coding

```
def hyper_para_bayes_logistic(m_0, S_0, Theta, y):
    import numpy as np
    w_map = m_0
    S_N = np.linalg.inv(S_0)

    Theta = Theta.T


    for i in range(Theta.shape[0]):
        S_N = S_N + y[i]*(1-y[i])*np.matmul(Theta[i].T, Theta[i])

    return w_map, S_N
```

Prediction

- Predict the label: $\int \sigma(w^T \phi) q(w) dw$

- Denote $a = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}$

$$\int \sigma(a) p(a) da \quad \Rightarrow \quad \int \sigma(a) N\left(a \mid \mu_a, \sigma_a^2\right) da$$

- Approximation

$$\sigma(\kappa(\sigma_a^2)\mu_a)$$

Prediction

- Compute $\sigma(\kappa(\sigma_a^2)\mu_a)$

$$\mu_a = \mathrm{E}[a] = \int p(a)da = \int q(\boldsymbol{w})\boldsymbol{w}^T\boldsymbol{\phi}\,d\boldsymbol{w} = \boldsymbol{w}_{map}^T\boldsymbol{\phi}$$

$$\sigma_a^2 = \mathrm{var}[a] = \int p(a)\left\{a^2 - \mathrm{E}[a]^2\right\}da$$

$$= \int q(\boldsymbol{w})\left\{(\boldsymbol{w}^T\boldsymbol{\phi})^2 - (m_N^T\boldsymbol{\phi})^2\right\}d\boldsymbol{w} = \boldsymbol{\phi}^T S_N\boldsymbol{\phi}$$

$$\kappa(\sigma^2) = (1 + \pi\sigma^2/8)^{-1/2}$$

Prediction

- Coding

```
def pred_bayes_logistic(w_map, S_N, theta):
    import numpy as np
    import math

    mu_a = np.dot(w_map.T, theta)

    var_a = np.dot(np.dot(theta.T, S_N), theta)

    kappa_var = (1 + math.pi*var_a/8)^(-0.5)

    x = kappa_var*mu_a

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

Multi-class classification

- How to extend it into multi-class case

0 or rest

1 or rest

2 or rest

Multi-class classification

- Coding

```
class logistic_sigmoid_model:
    def _init(self):

        self.hyper_para_bayes_logistic = hyper_para_bayes_logistic

        self.pred_bayes_logistic = pred_bayes_logistic

    def training(self, Theta, y):
        import numpy as np
        self.w0 = np.random.normal(0, 1)
        self.S0 = np.diag(np.random.normal(0, 1, y.shape[0]))

        # Theta n*m (n samples, m features), y n*1
        self.w_map, self.S_N = self.hyper_para_bayes_logistic(self.w0, self.S0,
Theta, y)

    def predict(self, theta):
        return self.pred_bayes_logistic(self.w_map, self.S_N, theta)
```

Multi-class classification

- Coding

```
def multiclass_sigmoid_logistic(Theta, Y):

    n_class = Y.shape[1]

    models = []

    for i in range(n_class):
        models.append(logistic_sigmoid_model())
        models[i].training(Theta, Y[:, i])


    return models
```

Multi-class classification

- Prediction

```
def pred_multiclass_sigmoid_logistic(theta, Theta, Y):

    models = multiclass_sigmoid_logistic(Theta, Y)

    props = []

    for i in range(len(models)):
        props.append(models[i].predict(theta))

    retrun max(props)
```