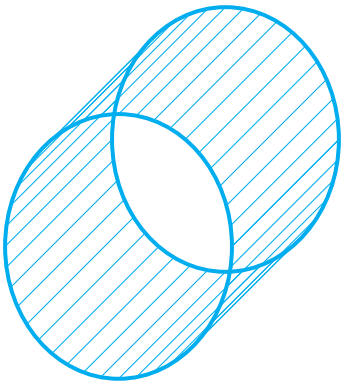
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM

Khoa Toán - Tin Học

BÁO CÁO GIỮA KỲ

PYTHON TRONG KHOA HỌC DỮ LIỆU



Khoa Toán - Tin học

Fac. of Math. & Computer Science

Giảng viên phụ trách: Nguyễn Tấn Trung

TP.Hồ Chí Minh - Tháng 12, 2020

Nhóm sinh viên thực hiện

- Vũ Thiện Nhân - MSSV : 18110171
- Trần Tấn Phong - MSSV : 18110181

Mục lục

1 Định nghĩa

Kỹ thuật hồi qui Bayesian thường được dùng để lấy các tham số chính qui trong quá trình ước lượng : Thông số chính qui không được thiết lập theo ý nghĩa gốc mà được điều chỉnh trong dữ liệu có sẵn.

Điều này có thể thực hiện bằng cách giới thiệu những suy diễn dựa trên các siêu tham số của mô hình. Tham số chính qui l_2 được dùng trong mô hình hồi qui và phân loại Ridge tương đương với việc tìm 1 ước lượng cực đại của phân phối Gaussian với hệ số tương quan w độ tin cậy λ^{-1}. Thay vì đặt Lamda theo cách thủ công, có thể coi nó như là 1 biến ngẫu nhiên được ước lượng từ dữ liệu.

Để có được 1 mô hình xác suất đầy đủ, đầu ra y được giả định là tuân theo phân phối Gaussian và với Xw:

$$p(y|X, w, \alpha) = \mathcal{N}(y|Xw, \alpha)$$

Với alpha là 1 biến ngẫu nhiên được ước lượng từ dữ liệu.

Những ưu điểm của hồi qui Bayesian là :

- Nó thích ứng với dữ liệu đang có
- Nó được sử dụng để chứa những tham số chính quy trong quá trình ước lượng

Những nhược điểm là :

- Việc suy diễn mô hình có thể rất tốn thời gian.

2 Bayesian Ridge Regression

2.1 Định nghĩa

Bayesian Ridge ước lượng 1 mô hình xác suất hồi quy đã mô tả ở trên. Giá trị hệ số tương quan được dựa trên đồ thị hình chuông của phân phối Gaussian:

$$p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1} I_p)$$

Những giá trị cho trước α và λ được chọn từ phân phối Gamma, và liên hợp cho trước của độ tin cậy Gaussian. Kết quả của mô hình được gọi là hồi qui Gaussian và tương tự như hồi qui Ridge cổ điển.

2.2 Ví dụ

2.2.1 Bayesian Ridge Regression

1. Import thư viện

```
        import numpy as np
        import matplotlib.pyplot as plt
        from scipy import stats

        from sklearn.linear_model import BayesianRidge, LinearRegression

```

2. Tạo ra dữ liệu mô phỏng với trọng số Gaussian

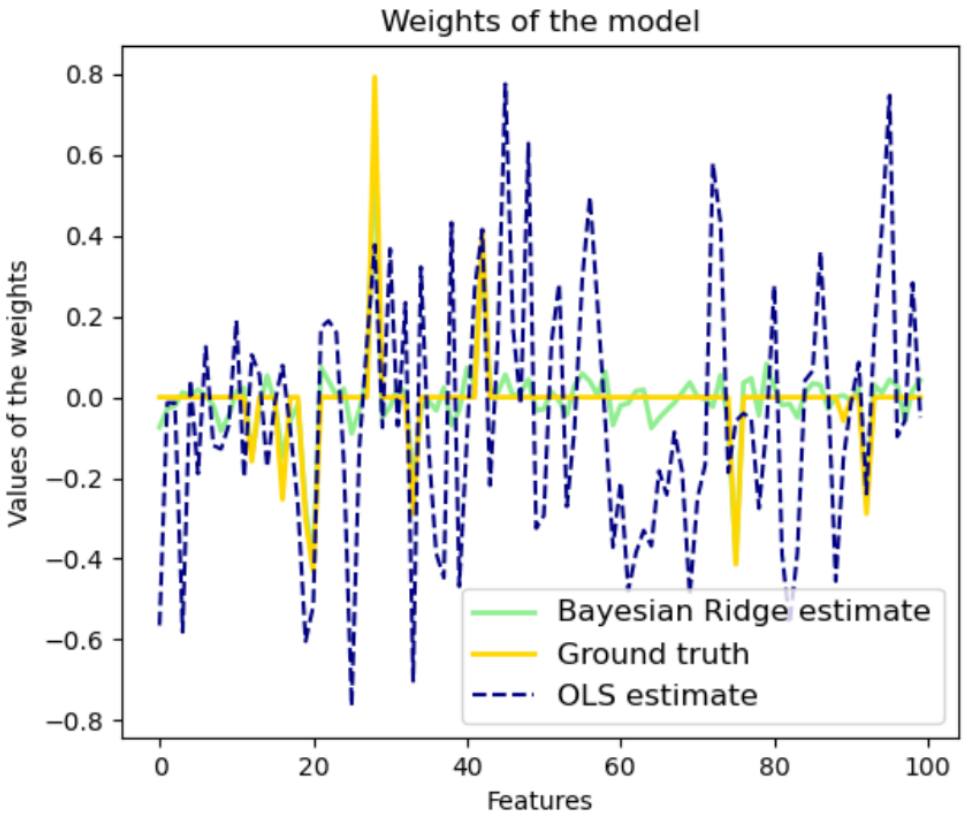
```
        # Tham so cua vi du
        np.random.seed(0)
        n_samples, n_features = 100, 100

        # Tao ra du lieu Gaussian
        X = np.random.randn(n_samples, n_features)

        # Tao ra trong so voi hang so lambda=5
        lambda_ = 4.
        w = np.zeros(n_features)

        # Giu lai 10 trong so
        relevant_features = np.random.randint(0, n_features, 10)
        for i in relevant_features:
            w[i] = stats.norm.rvs(loc=0, scale=1. / np.sqrt(lambda_))

        # Tao ra nhieu voi hang so alpha=50
        alpha_ = 50.
        noise = stats.norm.rvs(loc=0, scale=1. / np.sqrt(alpha_), size=
    n_samples)

        # Tao muc tieu
        y = np.dot(X, w) + noise

```

3. Fit Hồi quy Bayesian Ridge và Bình phương nhỏ nhất (OLS) để so sánh

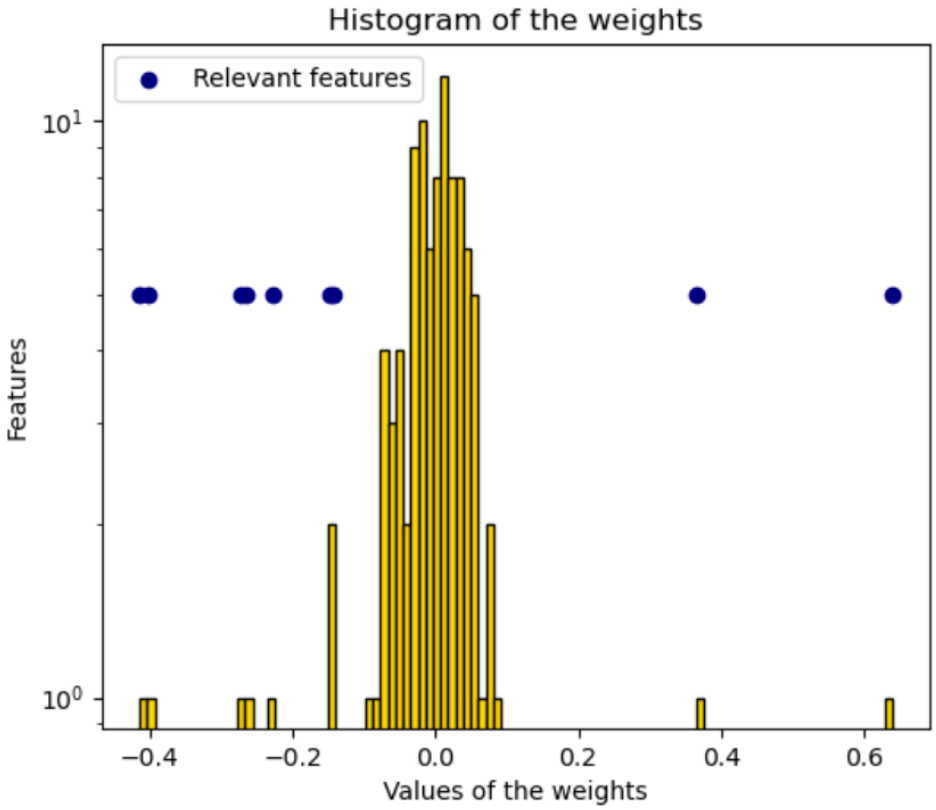
```
        clf = BayesianRidge(compute_score=True)
        clf.fit(X, y)

        ols = LinearRegression()
        ols.fit(X, y)

```

4. Vẽ biểu đồ trọng số đúng, trọng số ước lượng, histrogram của trong số và dự đoán độ lệch chuẩn

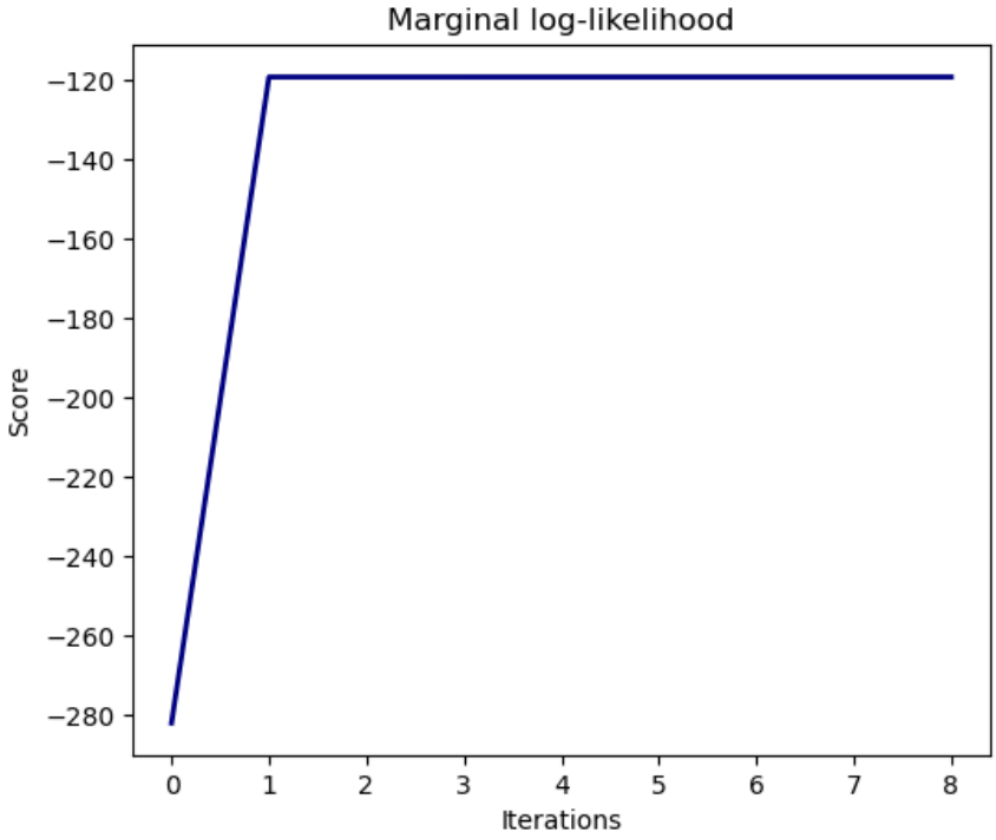
```
        lw = 2
        plt.figure(figsize=(6, 5))
        plt.title("Weights of the model")
        plt.plot(clf.coef_, color='lightgreen', linewidth=lw,
                 label="Bayesian Ridge estimate")
        plt.plot(w, color='gold', linewidth=lw, label="Ground truth")
        plt.plot(ols.coef_, color='navy', linestyle='--', label="OLS
    estimate")
        plt.xlabel("Features")
        plt.ylabel("Values of the weights")
        plt.legend(loc="best", prop=dict(size=12))
        plt.show()

```




```
        plt.figure(figsize=(6, 5))
        plt.title("Histogram of the weights")
        plt.hist(clf.coef_, bins=n_features, color='gold', log=True,
                 edgecolor='black')
        plt.scatter(clf.coef_[relevant_features], np.full(len(
    relevant_features), 5.),
                    color='navy', label="Relevant features")
        plt.ylabel("Features")
        plt.xlabel("Values of the weights")
        plt.legend(loc="upper left")
        plt.show()

```

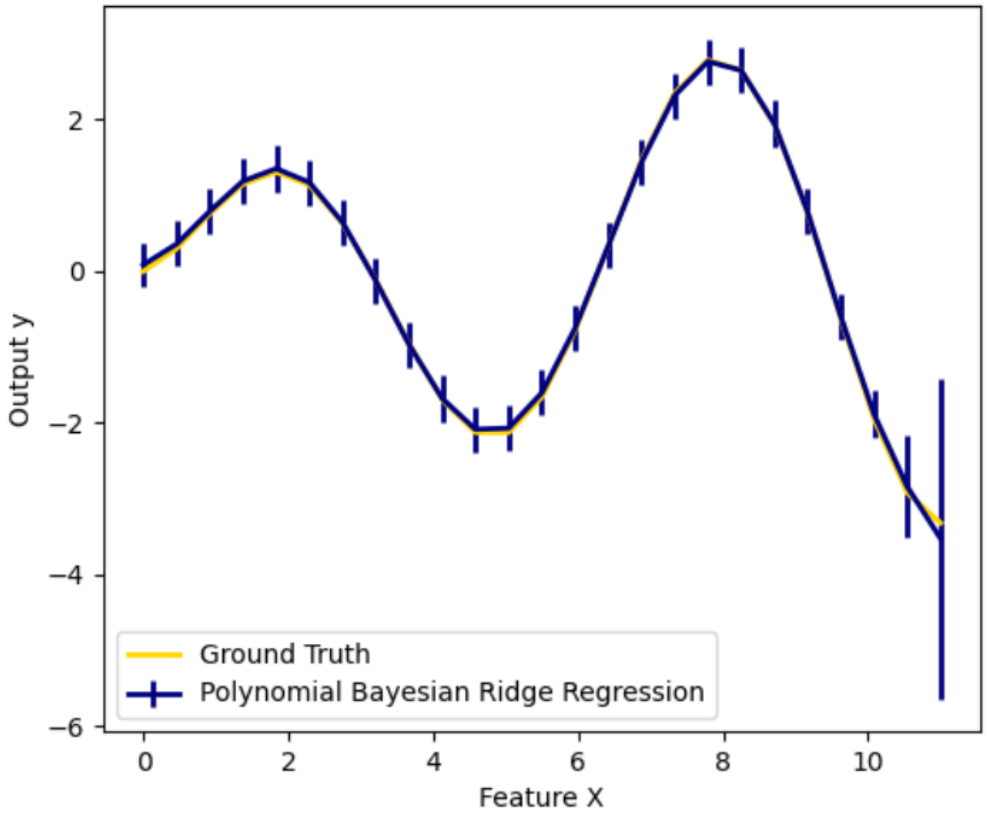



```
        plt.figure(figsize=(6, 5))
        plt.title("Marginal log-likelihood")
        plt.plot(clf.scores_, color='navy', linewidth=lw)
        plt.ylabel("Score")
        plt.xlabel("Iterations")
        plt.show()

```


Marginal log-likelihood
−120
−140
−160
−180
−200
−220
−240
−260
−280
Score
0
1
2
3
4
5
6
7
8
Iterations

5. Vẽ một số biểu đồ để dữ đoán hồi quy đa thức

```
    def f(x, noise_amount):
        y = np.sqrt(x) * np.sin(x)
        noise = np.random.normal(0, 1, len(x))
        return y + noise_amount * noise

    degree = 10
    X = np.linspace(0, 10, 100)
    y = f(X, noise_amount=0.1)
    clf_poly = BayesianRidge()
    clf_poly.fit(np.vander(X, degree), y)

    X_plot = np.linspace(0, 11, 25)
    y_plot = f(X_plot, noise_amount=0)
    y_mean, y_std = clf_poly.predict(np.vander(X_plot, degree),
return_std=True)
    plt.figure(figsize=(6, 5))
    plt.errorbar(X_plot, y_mean, y_std, color='navy',
                 label="Polynomial Bayesian Ridge Regression",
linewidth=lw)
    plt.plot(X_plot, y_plot, color='gold', linewidth=lw,
             label="Ground Truth")
    plt.ylabel("Output y")
    plt.xlabel("Feature X")
    plt.legend(loc="lower left")
    plt.show()

```


Output y
Feature X
Ground Truth
Polynomial Bayesian Ridge Regression
2
0
−2
−4
−6
0
2
4
6
8
10

2.2.2 Curve Fitting with Bayesian Ridge Regression

1. Import thư viện

```
        import numpy as np
        import matplotlib.pyplot as plt

        from sklearn.linear_model import BayesianRidge

```

2. Tạo hàm tính Sin

```
        def func(x):
            return np.sin(2*np.pi*x)

```

3. Tạo bộ dữ liệu hình Sin với nhiễu

```
        size = 25
        rng = np.random.RandomState(1234)
        x_train = rng.uniform(0., 1., size)
        y_train = func(x_train) + rng.normal(scale=0.1, size=size)
        x_test = np.linspace(0., 1., 100)

```

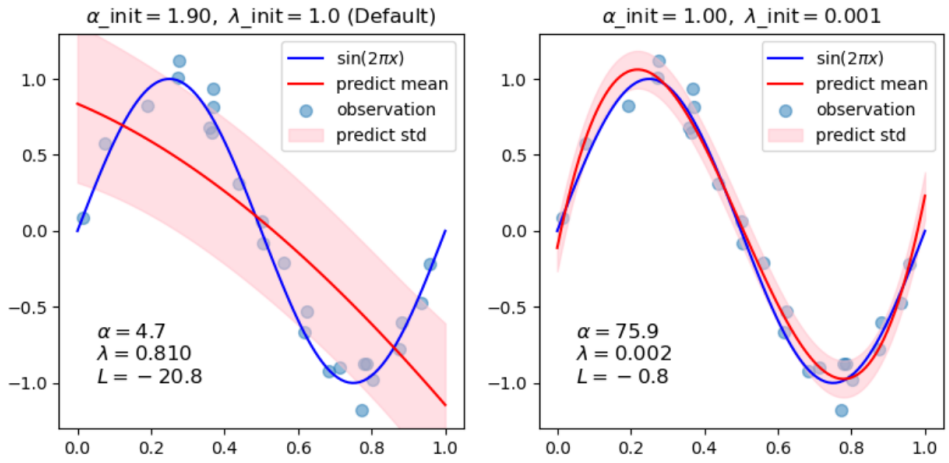
4. Fit đa thức bậc 3

```
        n_order = 3
        X_train = np.vander(x_train, n_order + 1, increasing=True)
        X_test = np.vander(x_test, n_order + 1, increasing=True)

```

5. Vẽ đồ thị các đường cong đúng và dự đoán với độ hợp lý biên

```
        reg = BayesianRidge(tol=1e-6, fit_intercept=False, compute_score
    =True)
        fig, axes = plt.subplots(1, 2, figsize=(8, 4))
        for i, ax in enumerate(axes):
            # Bayesian ridge regression with different initial value
    pairs
            if i == 0:
                init = [1 / np.var(y_train), 1.]  # Default values
```

```
            elif i == 1:
                init = [1., 1e-3]
                reg.set_params(alpha_init=init[0], lambda_init=init[1])
            reg.fit(X_train, y_train)
            ymean, ystd = reg.predict(X_test, return_std=True)

            ax.plot(x_test, func(x_test), color="blue", label="sin($2\\
    pi x$)")
            ax.scatter(x_train, y_train, s=50, alpha=0.5, label="
    observation")
            ax.plot(x_test, ymean, color="red", label="predict mean")
            ax.fill_between(x_test, ymean-ystd, ymean+ystd,
                            color="pink", alpha=0.5, label="predict std"
    )
            ax.set_ylim(-1.3, 1.3)
            ax.legend()
            title = "$\\alpha$_init$={:.2f},\\ \\lambda$_init$={}$".
    format(
                    init[0], init[1])
            if i == 0:
                title += " (Default)"
            ax.set_title(title, fontsize=12)
            text = "$\\alpha={:.1f}$\n$\\lambda={:.3f}$\n$L={:.1f}$".
    format(
                   reg.alpha_, reg.lambda_, reg.scores_[-1])
            ax.text(0.05, -1.0, text, fontsize=12)

        plt.tight_layout()
        plt.show()

```



3 Automatic Relevance Determination - ARD

3.1 Định nghĩa

Hồi qui ARD gần giống như hồi qui Bayesian Ridge, nhưng có thể dẫn tới các hệ số thưa thớt hơn w[1][2]. Hồi qui ARD đặt ra một sự khác biệt trước đó, bằng cách bỏ giả định Gaussian là hình chuông.

Thay vào đó, phân phối của w được giả định là những đường giống hoặc là đường elip với phân phối Gaussian.

Điều này có nghĩa là ứng với mỗi hệ số tương quan w_i được vẽ từ phân phối Gaussian, tập trung vào số 0 với độ tin cậy là λ_i.

$$p(w|\lambda) = \mathcal{N}(w|, 0, A^{-1})$$

với $diag(A) = \lambda = \{\lambda_1,...,\lambda_p\}$

Trái ngược với hồi qui Bayesian Ridge, mỗi tọa độ của w_i đề có những độ lệch chuẩn riêng của nó λ_i. Với λ_i được chọn trong cùng phân phối Gamma được cho bởi các siêu tham số λ_1 và λ_2.

3.2 Ví dụ

1.Import thư viện

```
1        import numpy as np
```

```
    import matplotlib.pyplot as plt
    from scipy import stats

    from sklearn.linear_model import ARDRegression, LinearRegression


```

2. Tạo ra dữ liệu mô phỏng với trọng số Gausian

```
    # Tham so cua vi du
    np.random.seed(0)
    n_samples, n_features = 100, 100

    # Tao ra du lieu Gaussian
    X = np.random.randn(n_samples, n_features)

    # Tao ra trong so voi hang so lambda=5
    lambda_ = 4.
    w = np.zeros(n_features)

    # Giu lai 10 trong so
    relevant_features = np.random.randint(0, n_features, 10)
    for i in relevant_features:
        w[i] = stats.norm.rvs(loc=0, scale=1. / np.sqrt(lambda_))

    # Tao ra nhieu voi hang so alpha=50
    alpha_ = 50.
    noise = stats.norm.rvs(loc=0, scale=1. / np.sqrt(alpha_), size=
n_samples)

    # Tao muc tieu
    y = np.dot(X, w) + noise

```

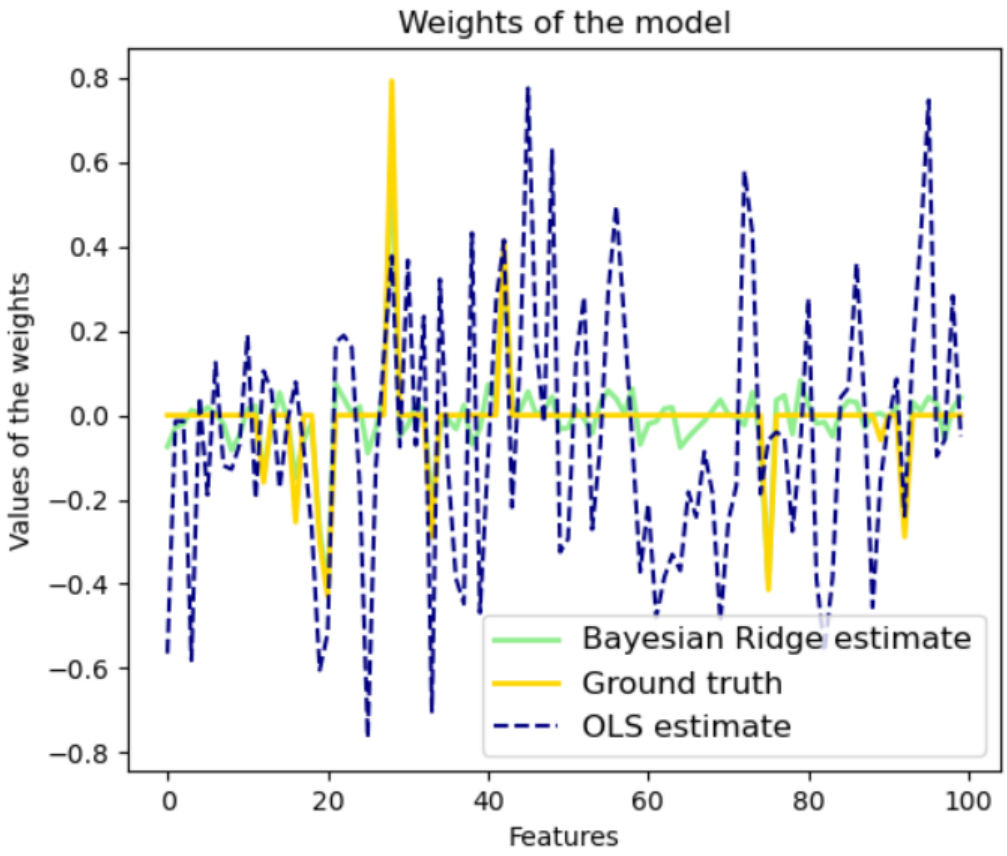
3. Fit mô hình hồi quy ARD

```
    import numpy as np
    import matplotlib.pyplot as plt
    from scipy import stats
```

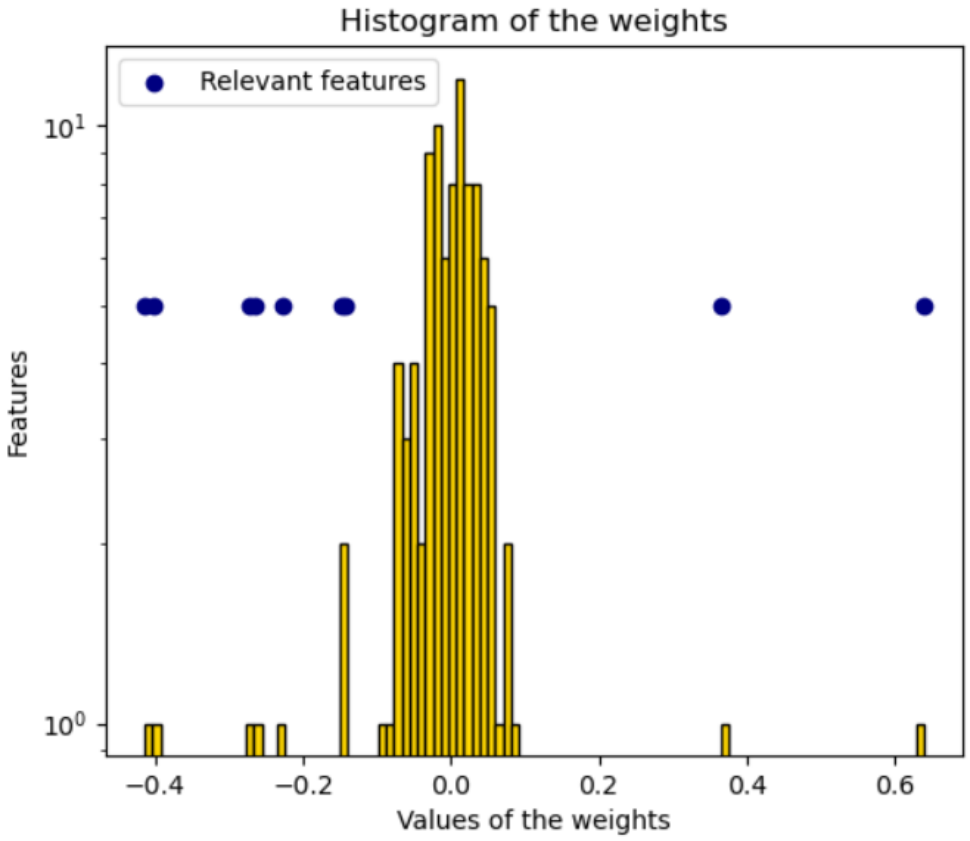
```
4
5       from sklearn.linear_model import ARDRegression, LinearRegression
6
7
```

4. Vẽ biểu đồ trọng số đúng, trọng số ước lượng, histrogram của trong số và dự đoán độ lệch chuẩn

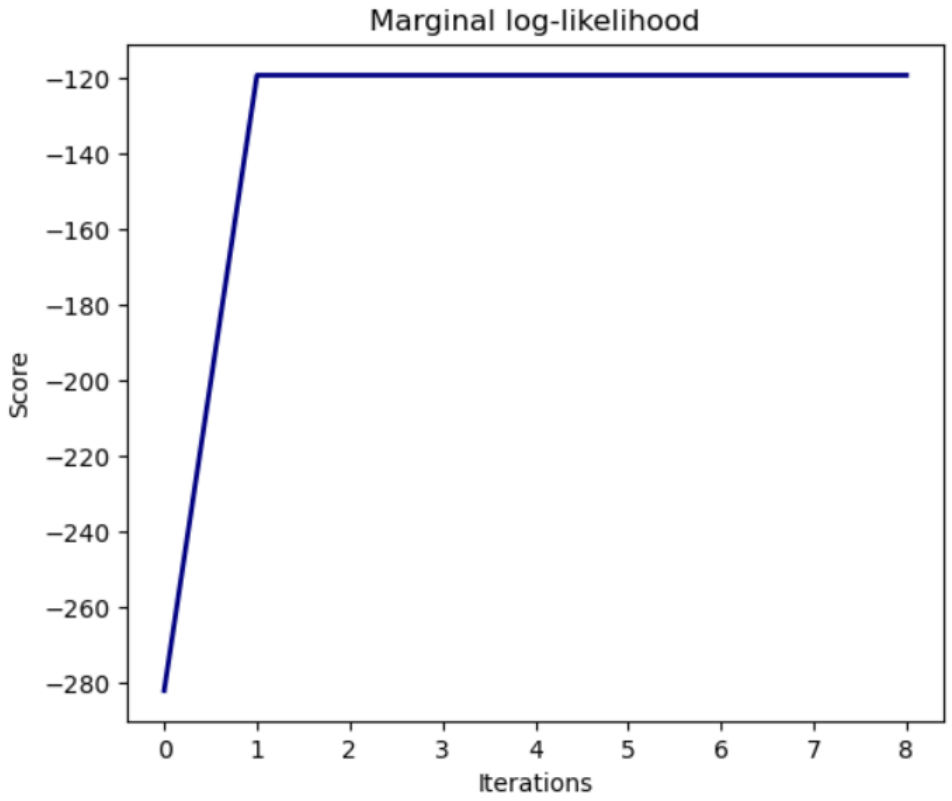
```
1       plt.figure(figsize=(6, 5))
2       plt.title("Weights of the model")
3       plt.plot(clf.coef_, color='darkblue', linestyle='-', linewidth
    =2,
4                label="ARD estimate")
5       plt.plot(ols.coef_, color='yellowgreen', linestyle=':',
    linewidth=2,
6                label="OLS estimate")
7       plt.plot(w, color='orange', linestyle='-', linewidth=2, label="
    Ground truth")
8       plt.xlabel("Features")
9       plt.ylabel("Values of the weights")
10      plt.legend(loc=1)
11      plt.show()
12
```


Weights of the model
Values of the weights
Features
Bayesian Ridge estimate
Ground truth
OLS estimate
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
20
40
60
80
100

```
        plt.figure(figsize=(6, 5))
        plt.title("Histogram of the weights")
        plt.hist(clf.coef_, bins=n_features, color='navy', log=True)
        plt.scatter(clf.coef_[relevant_features], np.full(len(
    relevant_features), 5.),
                    color='gold', marker='o', label="Relevant features")
        plt.ylabel("Features")
        plt.xlabel("Values of the weights")
        plt.legend(loc=1)
        plt.show()

```

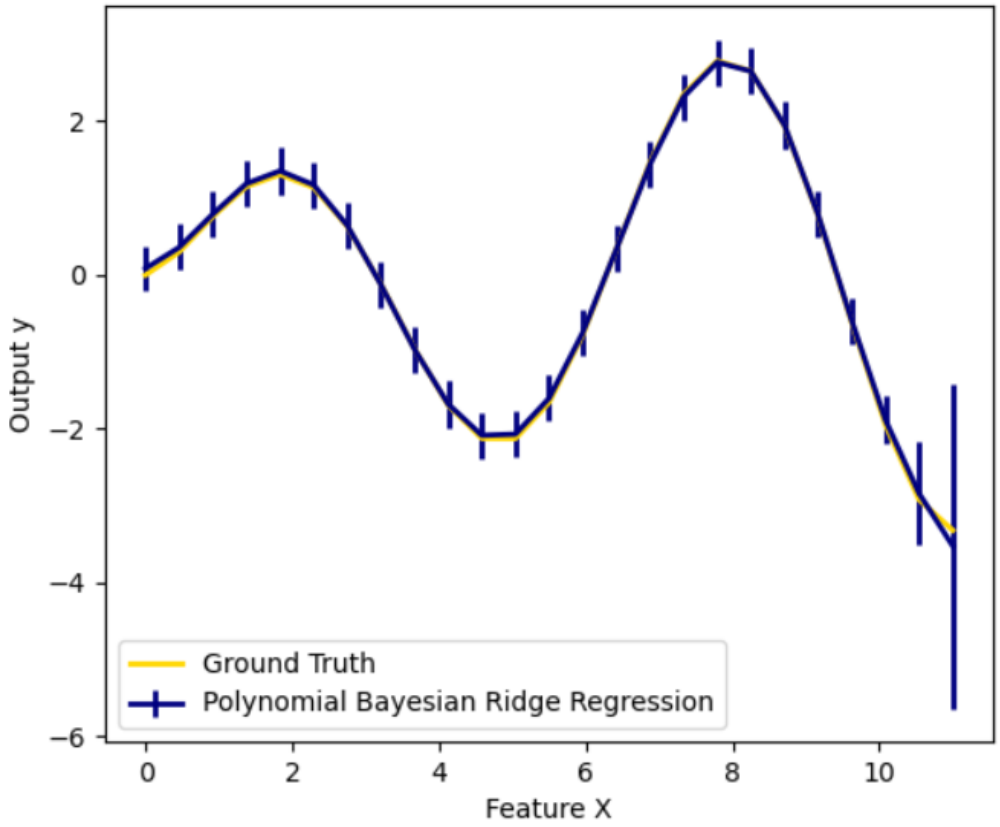


```
        plt.figure(figsize=(6, 5))
        plt.title("Marginal log-likelihood")
        plt.plot(clf.scores_, color='navy', linewidth=2)
        plt.ylabel("Score")
        plt.xlabel("Iterations")
        plt.show()

```



5. Vẽ một số biểu đồ để dữ đoán hồi quy đa thức

```
        def f(x, noise_amount):
            y = np.sqrt(x) * np.sin(x)
            noise = np.random.normal(0, 1, len(x))
            return y + noise_amount * noise


        degree = 10
        X = np.linspace(0, 10, 100)
        y = f(X, noise_amount=1)
        clf_poly = ARDRegression(threshold_lambda=1e5)
        clf_poly.fit(np.vander(X, degree), y)

        X_plot = np.linspace(0, 11, 25)
        y_plot = f(X_plot, noise_amount=0)
        y_mean, y_std = clf_poly.predict(np.vander(X_plot, degree),
    return_std=True)
        plt.figure(figsize=(6, 5))
        plt.errorbar(X_plot, y_mean, y_std, color='navy',
                     label="Polynomial ARD", linewidth=2)
        plt.plot(X_plot, y_plot, color='gold', linewidth=2,
                 label="Ground Truth")
        plt.ylabel("Output y")
        plt.xlabel("Feature X")
        plt.legend(loc="lower left")
        plt.show()

```


Output y
Feature X
Ground Truth
Polynomial Bayesian Ridge Regression
2
0
−2
−4
−6
0
2
4
6
8
10

Tài liệu tham khảo

1. www.scikit-learn.org

2. David J. C. MacKay, Bayesian Interpolation, 1992

3. Michael E. Tipping, Sparse Bayesian Learning and the Relevance Vector Machine, 2001.