2. Automatic Relevance Determination

A. [5p] Implement the EM version of ARD (inference for the β, update σy and all σβm) for the wine data set as regression problem.

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
```

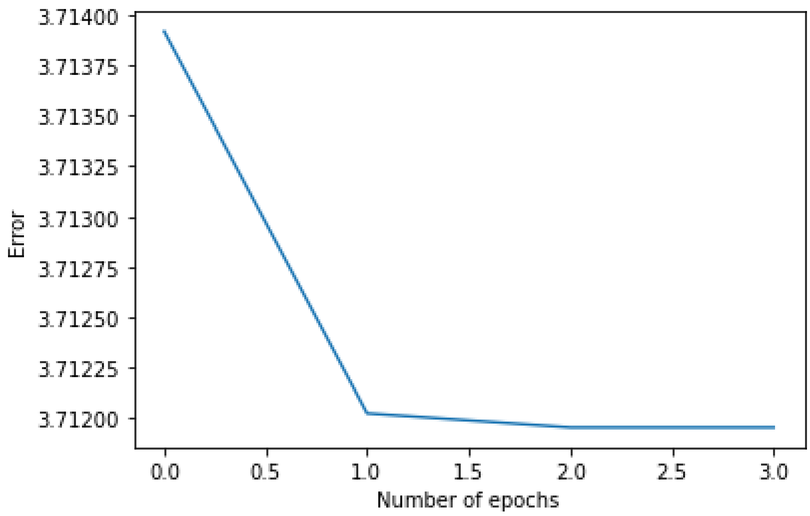
In [7]:
```
from sklearn.datasets import load_wine
wine_data = load_wine()

X = np.array(wine_data.data)
Y = np.array(wine_data.target).reshape(len(wine_data.target),1)
```

In [20]:
```
def ARD(X, Y, a, b, c, d, epochs):
    N = X.shape[0]
    sigma2_y = 1
    sigma2_beta = np.ones((X.shape[1],1))
    old_loss = 0
    losses = []
    for i in range(epochs):
        print("\nEpoch "+str(i+1)+"\n")
        if i > 0:
            old_loss = np.sqrt((Y - X @ mu_beta).T @ (Y - X @ mu_beta))

        Sigma_beta = np.diag(sigma2_beta[:,0])

        C_beta = np.linalg.pinv(((1/sigma2_y) * X.T @ X) + np.linalg.pinv(Sigma_beta

        mu_beta = ((1/sigma2_y) * C_beta) @ (X.T @ Y)

        sigma2_beta = (2 * b + np.square(mu_beta) + np.diag(C_beta).reshape(C_beta.s

        sigma2_y = (2 * d + (Y - X @ mu_beta).T @ (Y - X @ mu_beta) + np.sum(np.diag

        Y_hat = X @ mu_beta

        loss = np.round(float(np.sqrt((Y - Y_hat).T @ (Y - Y_hat))), 5)
        print("loss:\n", loss)
        losses.append(loss)

        accuracy = np.round((np.sum(Y == np.array(Y_hat, dtype=int)) / N) * 100, 2)
        print("accuracy:\n", accuracy)

        if np.abs(old_loss - loss) < 1e-5:
            print("\nModel Converged in "+str(i+1)+" epochs")
            break
    plt.plot(np.arange(i+1), losses)
    plt.xlabel("Number of epochs")
    plt.ylabel("Error")
    plt.show()
    return C_beta, mu_beta, sigma2_beta, sigma2_y
```

In [21]:
```
a, b, c, d = 1, 1, 1, 1
n_epochs = 100
```

```
C_beta, mu_beta, sigma2_beta, sigma2_y = ARD(X, Y, a, b, c, d, n_epochs)
```

```
Epoch 1

loss:
 3.71392
accuracy:
 61.8

Epoch 2

loss:
 3.71202
accuracy:
 61.8

Epoch 3

loss:
 3.71195
accuracy:
 61.8

Epoch 4

loss:
 3.71195
accuracy:
 61.8

Model Converged in 4 epochs
```



B. [4p] What information does C_β carry? How can you interpret the values and how does C_β relate to last weeks ${\tau_\beta}^2$?

$C_ß$ are the variances of the normal distribution with mean $\mu_ß$. Hence, these help us in estimation of parameter ß. Moreover, these are computed by taking into account the fact that each has different variance. In the last tutorial τ^2, were the latent variables that were the variances of the Gaussian distributions and were

exponentially distributed. These also allowed us to estimate parameters ß given the regularization constants λ .

C. [3p] Sample different model configurations from the learned distribution β ~ N (μβ,Cβ) and qualitatively compare their predictions via a loss measure and an accuracy measure.

In [19]:

```
import pandas as pd
models = []
for j in range(10):
    betas = np.random.multivariate_normal(mu_beta.reshape(mu_beta.shape[0]), C_beta,
    betas = betas.reshape(-1,1)
    Y_hat = X @ betas
    loss = np.round(float(np.sqrt((Y - X @ betas).T @ (Y - X @ betas))), 5)
    accuracy = np.round((np.sum(Y == np.array(Y_hat, dtype=int)) / X.shape[0]) * 100
    models.append(["Model "+str(j+1), loss, accuracy])
models_df = pd.DataFrame(models, columns=['Model', 'loss', 'accuracy'])
models_df = models_df.set_index('Model')
display(models_df)
print("Lowest loss: ", models_df.loss.min(), " - Model No: ", models_df.loss.argmin(
print("Best accuracy: ", models_df.accuracy.max(), "% - ", "Model No: ", models_df.a
```

	loss	accuracy
Model		
Model 1	3.93883	63.48
Model 2	4.02858	63.48
Model 3	3.85821	56.74
Model 4	3.82295	58.43
Model 5	3.92338	57.30
Model 6	3.88100	58.99
Model 7	3.78160	64.04
Model 8	3.88462	62.36
Model 9	3.78552	60.67
Model 10	3.99599	61.80

```
Lowest loss:  3.7816  - Model No:  7
Best accuracy:  64.04 % -  Model No:  7
```