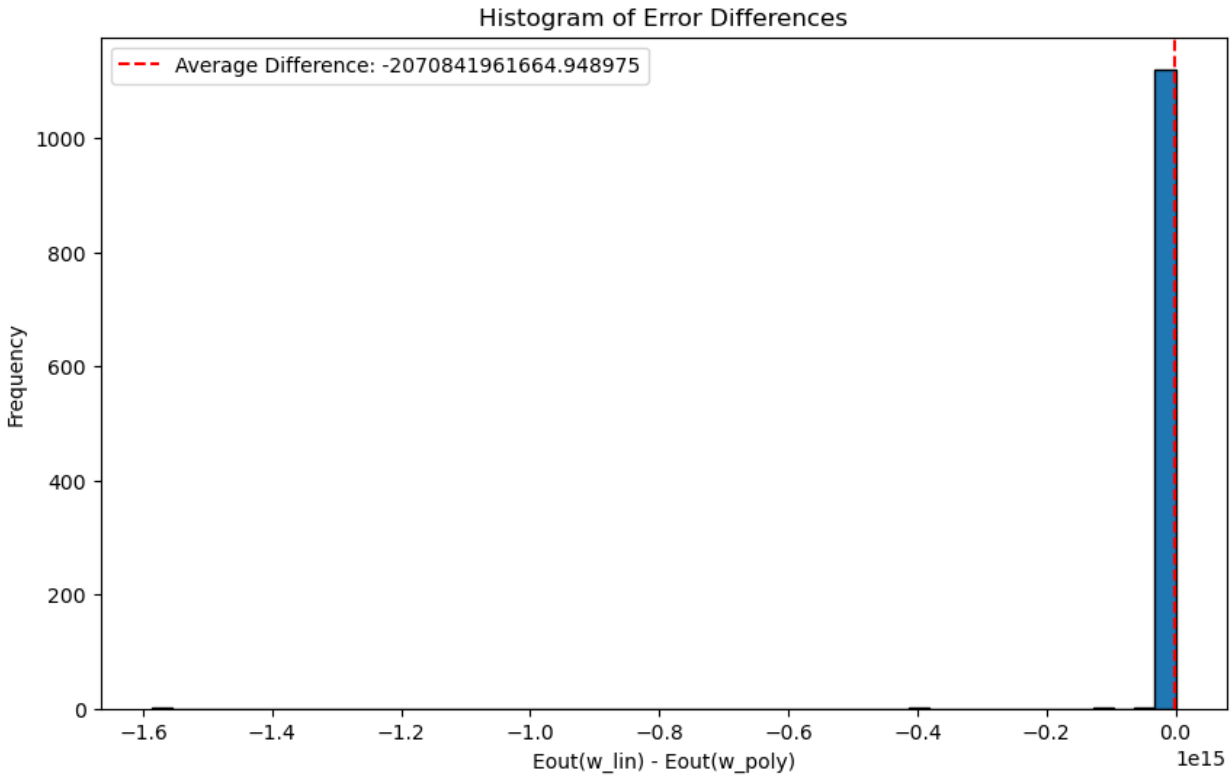
ML homework 4: question 12



Meaning:

From the plot we can see that the differences are all negative, indicating that we're getting way more bigger $\mathrm{E}_{\mathrm{out}}(\mathrm{w}_{\mathrm{poly}})$ than $\mathrm{E}_{\mathrm{out}}(\mathrm{w}_{\mathrm{LIN}})$, this may due to we're using too many features to fit our data. The original data we have might lie in a smaller space, so overfitting occurs, hence generating enormous out of sample error.

Code:

transform function $\Phi()$

We use this function to map the orignal data into a higher dimensional space, and save the augmented input vectors in `X_aug_arr`. Which means that each of the element in `X_aug_arr` is a 37-dimensional vector (12 features + 1 constant + 12 squared features + 12 cubed features).

```
[41]  ✓ 0.0s
```

```python
def Phi(X_arr):
    X_aug_arr = []
    for arr in X_arr:
        Phi_arr = np.concatenate((np.array([1]), arr, arr**2, arr**3))
        X_aug_arr.append(Phi_arr)
    return np.array(X_aug_arr)
```

get weight vector

+ Code + Markdown

```
def weight_vector(X_in_sample_mat_lin, y_in_sample_array, X_in_sample_mat_poly):

    w_lin = np.linalg.inv(X_in_sample_mat_lin.T @ X_in_sample_mat_lin) @ X_in_sample_mat_lin.T @ y_in_sample_array
    w_poly = np.linalg.inv(X_in_sample_mat_poly.T @ X_in_sample_mat_poly) @ X_in_sample_mat_poly.T @ y_in_sample_array
    return w_lin , w_poly

```

[44] ✓ 0.0s

out of sample error

```
def out_of_sample_error(X_out_of_sample_mat_lin, X_out_of_sample_mat_poly, y_out_of_sample_array, w_lin, w_poly):

    out_of_sample_error_lin = np.mean((X_out_of_sample_mat_lin @ w_lin - y_out_of_sample_array) ** 2)
    out_of_sample_error_poly = np.mean((X_out_of_sample_mat_poly @ w_poly - y_out_of_sample_array) ** 2)
    return out_of_sample_error_lin, out_of_sample_error_poly


```

[45] ✓ 0.0s

```
for experiment in range(1126):

    seed = experiment
    random_sample_indices = generate_random_sample(seed)

    X_sample = [X[i] for i in random_sample_indices]
    y_sample = [y[i] for i in random_sample_indices]

    X_in_sample_mat_lin = np.array(convert_dtype(X_sample))
    X_in_sample_mat_poly = Phi(convert_dtype(X_sample))

    y_in_sample_array = np.array(y_sample)

    out_ind = generate_out_of_sample_ind(random_sample_indices)

    X_out_of_sample = [X[i] for i in out_ind]
    y_out_of_sample = [y[i] for i in out_ind]

    X_out_of_sample_mat_lin = np.array(convert_dtype(X_out_of_sample))
    X_out_of_sample_mat_poly = Phi(convert_dtype(X_out_of_sample))

    y_out_of_sample_array = np.array(y_out_of_sample)

    w_lin, w_poly = weight_vector(X_in_sample_mat_lin, y_in_sample_array, X_in_sample_mat_poly)
    out_of_sample_error_lin, out_of_sample_error_poly = out_of_sample_error(X_out_of_sample_mat_lin, X_out_of_sample_mat_poly, y_out_of_sample_array, w_lin, w

    out_sample_error_lin.append(out_of_sample_error_lin)
    out_sample_error_poly.append(out_of_sample_error_poly)
    lin_sub_poly_error.append(out_of_sample_error_lin - out_of_sample_error_poly)
    avg_difference = np.mean(lin_sub_poly_error)


plt.figure(figsize=(10, 6))
plt.hist(lin_sub_poly_error, bins=50, edgecolor='black')
plt.axvline(x=avg_difference, color='r', linestyle='--',
            label=f'Average Difference: {avg_difference:.6f}')
plt.xlabel('Eout(w_lin) - Eout(w_poly)')
plt.ylabel('Frequency')
plt.title('Histogram of Error Differences')
plt.legend()
plt.show()
```