5.

$$b_E(\underline{w}_t) = \nabla E(\underline{w}_t) = \frac{1}{N}\sum_{n=1}^{N}\theta(-y_n \underline{w}^T \underline{x}_n)(-y_n \underline{x}_n)$$

for $\theta(z) = \frac{1}{1+e^{-z}}$, $\theta'(z) = \frac{-1}{(1+e^{-z})^2}(-e^{-z})$

$$= \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) = \theta(z)\theta(-z) = \theta(z)(1-\theta(z))$$

$$\Rightarrow A_E(\underline{w}_t) = \nabla b_E(\underline{w}_t)$$

y_n is either 1 or -1

$$\Rightarrow \frac{\partial b_E(\underline{w}_t)}{\partial w_{t,i}} = \frac{1}{N}\sum_{n=1}^{N}\theta(-y_n \underline{w}_t^T \underline{x}_n)\theta(y_n \underline{w}_t^T \underline{x}_n) y_n^2 x_{n,i} \underline{x}_n$$

$$= \frac{1}{N}\sum_{n=1}^{N}\theta(\underline{w}_t^T \underline{x}_n)(1-\theta(\underline{w}_t^T \underline{x}_n)) y_n^2 x_{n,i} \underline{x}_n$$

$$\Rightarrow \nabla b_E(\underline{w}_t) = \frac{1}{N}\sum_{n=1}^{N} \underline{x}_n \underline{x}_n^t h_t(\underline{x}_n)(1-h_t(\underline{x}_n))$$

① $X \in F^{N \times d}$, $X = \begin{bmatrix} \underline{X_1}^T \\ \underline{X_2}^T \\ \vdots \\ \underline{X_N}^T \end{bmatrix}$ $X \in F^{N \times d}$. $X^T D X \in F^{d \times d}$.

② D is N×N diagonal matrix

Given above, we can write $X^T D X$

$$= [\underline{X_1}\ \underline{X_2}\ \cdots\ \underline{X_N}] \begin{bmatrix} d_{11} & & 0 \\ & d_{22} & \\ 0 & & \ddots\ d_{NN} \end{bmatrix} \begin{bmatrix} \underline{X_1}^T \\ \underline{X_2}^T \\ \vdots \\ \underline{X_N}^T \end{bmatrix}$$

we can derive $d_{nn} = h_t(\underline{X_n})(1 - h_t(\underline{X_n}))$

so D is an N by N diagonal matrix

whose element $d_{nn} = h_t(\underline{X_n})(1 - h_t(\underline{X_n}))$

#

6.

$$\ln h_y(x) = \ln e^{\underline{w}_y^T \underline{x}} - \ln\left(\sum_{k=1}^{K} e^{\underline{w}_k^T \underline{x}}\right)$$

$$= \underline{w}_y^T \underline{x} - \ln\left(\sum_{k=1}^{K} e^{\underline{w}_k^T \underline{x}}\right) \quad z$$

$$\frac{\partial \ln z}{\partial \underline{w}_k} = \frac{1}{z}\frac{\partial z}{\partial \underline{w}_k} = \frac{1}{z}\left(e^{\underline{w}_k^T \underline{x}}\right)\underline{x} = h_k(\underline{x})\,\underline{x}$$

$$\Rightarrow \frac{\partial err(W, \underline{x}_n, y_n)}{\partial \underline{w}_k} = \frac{\partial\left(-\ln h_{y_n}(\underline{x}_n)\right)}{\partial \underline{w}_k} = -\left(\frac{\partial\, \underline{w}_{y_n}^T \underline{x}_n}{\partial \underline{w}_k} - \frac{\partial \ln z}{\partial \underline{w}_k}\right)$$

$$= \begin{cases} -\left(\underline{x}_n - h_{y_n}(\underline{x}_n)\underline{x}_n\right) = \underline{x}_n\left(h_{y_n}(\underline{x}_n) - 1\right), & k = y_n \\ h_k(\underline{x}_n)\,\underline{x}_n, & k \neq y_n \end{cases}$$

$$\Rightarrow \frac{\partial err(\underline{W}, \underline{x}_n, y_n)}{\partial \underline{w}_k} = h_k(\underline{x}_n) - [k = y_n]\,\underline{x}_n$$

$$\Rightarrow V = \underline{x}_n \cdot u^T, \quad u \in R^K, \quad u_k = [k = y_n] - h_k(\underline{x}_n)$$

#

7.

In original MLR, we use $h_k(\underline{x})$ to approximate target distribution $P(y|\underline{x})$

that is, $P(y=k|\underline{x}) = h_k(\underline{x})$

In standard binary logistic regression:

$$P(y=1|\underline{x}) = h(\underline{x}) = \frac{1}{1+e^{-\underline{w}^T\underline{x}}}$$

$$y_n' = 2y_n - 3 \Rightarrow \begin{cases} y_n = 1, y_n' = -1 \\ y_n = 2, y_n' = 1 \end{cases}$$

$$P(y_n'=1|\underline{x}) = P(y_n=2|\underline{x}) = h_2(\underline{x})$$

$$\Rightarrow \frac{1}{1+e^{-\underline{w}_1^T\underline{x}}} = \frac{e^{\underline{w}_2^{*T}\underline{x}}}{e^{\underline{w}_2^{*T}\underline{x}} + e^{\underline{w}_1^{*T}\underline{x}}} \quad \begin{matrix} \times e^{-\underline{w}_2^T\underline{x}} \\ \times e^{-\underline{w}_2^T\underline{x}} \end{matrix}$$

$$\Rightarrow \quad \frac{1}{1+e^{-\underline{w}_{lr}^T \underline{x}}} = \frac{1}{1+e^{\left(\underline{w}_1^{*T}\underline{x} - \underline{w}_2^{*T}\underline{x}\right)}}$$

$$\Rightarrow \quad w_{lr} = w_2^* - w_1^* \quad \#$$

8. $(y_1 = f(x_1),\ y_2 = f(x_2))$

$h(x) = w_0 + w_1 x$, $D = \{(x_1, y_1), (x_2, y_2)\}$

$$E_{in} = \left((h(x_1) - y_1)^2 + (h(x_2) - y_2)^2\right)^{\frac{1}{2}}$$

$$= \left((w_0 + w_1 x_1 - y_1)^2 + (w_0 + w_1 x_2 - y_2)^2\right)^{\frac{1}{2}}$$

To minimize E_{in} :

①

$$\frac{\partial E}{\partial w_0} = 0 \ : \ 2(w_0 + w_1 x_1 - y_1) + 2(w_0 + w_1 x_2 - y_2) = 0$$

$$\Rightarrow \quad w_0 = \frac{y_1 + y_2}{2} - \frac{w_1(x_1 + x_2)}{2} \quad \text{— ⓐ}$$

②

$$\frac{\partial E}{\partial w_1} = 0 \ : \ 2x_1(w_0 + w_1 x_1 - y_1) + 2x_2(w_0 + w_1 x_2 - y_2) = 0$$

$$\Rightarrow 2w_0(x_1 + x_2) + 2w_1(x_1^2 + x_2^2) = 2(y_1 x_1 + y_2 x_2)$$

— ⓑ

substitue ⓐ into ⓑ :

$$2\left(\frac{y_1+y_2}{2}-\frac{w_1(x_1+x_2)}{2}\right)(x_1+x_2)+2w_1(x_1^2+x_2^2)=2(y_1x_1+y_2x_2)$$

$$\Rightarrow\ w_1\left(2(x_1^2+x_2^2)-(x_1+x_2)^2\right)=2(y_1x_1+y_2x_2)-(y_1+y_2)(x_1+x_2)$$

$$\Rightarrow\ w_1(x_1-x_2)^2=(y_1-y_2)(x_1-x_2)\ \Rightarrow\ w_1=\frac{y_1-y_2}{x_1-x_2}$$

and $w_0=y_1-w_1x_1$

$$\Rightarrow\begin{cases} w_1=\dfrac{y_1-y_2}{x_1-x_2}=\dfrac{(-2x_1^2)-(-2x_2^2)}{x_1-x_2}=-2(x_1+x_2) \\ w_0=y_1-w_1x_1=(-2x_1^2)+2(x_1+x_2)x_1 \\ \qquad\quad =1+2x_1x_2 \end{cases}$$

$$\Rightarrow\ g(x)=(1+2x_1x_2)-2(x_1+x_2)x$$

$$\Rightarrow\ E_{in}=\frac{1}{2}\left((g(x_1)-f(x_1))^2+(g(x_2)-f(x_2))^2\right)$$

$$=\frac{1}{2}(0+0)=0$$

$$E_{out} = \int_0^1 (g(x) - f(x))^2 dx$$

$$= \int_0^1 (2x_1x_2 - 2(x_1+x_2)x + 2x^2)^2 dx$$

$$= \int_0^1 \Big(4x_1^2x_2^2 - 8x_1x_2(x_1+x_2)x + 8x_1x_2x^2 + 4(x_1+x_2)^2x^2 - 8(x_1+x_2)x^3 + 4x^4\Big) dx$$

$$= 4x_1^2x_2^2 - 4x_1x_2(x_1+x_2) + \frac{8}{3}x_1x_2 + \frac{4}{3}(x_1+x_2)^2 - 2(x_1+x_2) + \frac{4}{5}$$

$$= 4x_1^2x_2^2 - 4x_1^2x_2 - 4x_1x_2^2 + \frac{8}{3}x_1x_2 + \frac{4}{3}x_1^2 + \frac{8}{3}x_1x_2 + \frac{4}{3}x_2^2 - 2x_1 - 2x_2 + \frac{4}{5}$$

$$\Rightarrow E_D\big(|E_{in}(g) - E_{out}(g)|\big) = E_D\big(|E_{out}(g)|\big)$$

$$= \iint |E_{out}(g)|\, p(x_1,x_2)\, dx_1 dx_2$$

$$= \int_0^1\int_0^1 |E_{out}(x_1,x_2)|\, dx_1 dx_2 \quad \text{since } x_1, x_2 \text{ sample uniformly from } [0,1]$$

$$= \int_0^1 \left(\frac{4}{3}x_2^2 - \frac{4}{3}x_2 - 2x_2^2 + \frac{4}{3}x_2 + \frac{4}{9} + \frac{4}{3}x_2 + \frac{4}{3}x_2^2 - 1 - 2x_2 + \frac{4}{5} \right) dx_2$$

$$= \frac{4}{9} - \cancel{\frac{2}{3}} - \cancel{\frac{2}{3}} + \cancel{\frac{2}{3}} + \frac{4}{9} + \cancel{\frac{2}{3}} + \frac{4}{9} - 1 - 1 + \frac{4}{5}$$

$$= \frac{2}{15} \quad \#$$

9.

$\varepsilon \in R^{d+1}$

$\varepsilon\varepsilon^T \in R^{(d+1)\times(d+1)}$

$$X_h^T X_h = \sum_{n=1}^{N} \underline{x}_n \underline{x}_n^T + \sum_{n=1}^{N} \tilde{\underline{x}}_n \tilde{\underline{x}}_n^T$$

$$\tilde{\underline{x}}_n \tilde{\underline{x}}_n^T = (\underline{x}_n + \underline{\varepsilon})(\underline{x}_n + \underline{\varepsilon})^T$$

$$= \underline{x}_n \underline{x}_n^T + \underline{x}_n \underline{\varepsilon}^T + \underline{\varepsilon} \underline{x}_n^T + \underline{\varepsilon}\underline{\varepsilon}^T$$

Since $\varepsilon \sim N(0_{d+1}, \sigma^2 I_{d+1})$, it indicates that the mean of each variable in ε is 0, and each individual variable has variance σ^2, while each pair of them has covariance 0, indicating they are independent of each other.

$\Rightarrow E[\varepsilon_i] = 0,\ E[\varepsilon_i^2] = \sigma^2,\ i \in \{0, 1, 2, \ldots d+1\}$

Given above, we can derive that

① $E(\underline{X}_n \varepsilon^T) = \underline{X}_n E(\varepsilon^T) = \bigcirc$

② $E(\varepsilon \underline{X}_n^T) = E(\varepsilon)\underline{X}_n^T = \bigcirc$

③ $E(\varepsilon\varepsilon^T) = \sigma^2 I_{d+1}$

$$\Rightarrow E(X_h^T X_h) = E\left(\sum_{n=1}^{N} \underline{X}_n \underline{X}_n^T\right) + E\left(\sum_{n=1}^{N} (\underline{X}_n \underline{X}_n^T + \sigma^2 I_{d+1})\right)$$

$$= 2X^T X + N\sigma^2 I_{d+1}$$

$$\Rightarrow \alpha = 2, \ \beta = N \quad \#$$

Problem 10

First we can observe that according to $\mathbf{w}_{lin}$, it has good E_{in} and bad E_{out}, which means that we may match the sampled data too much, hence it isn't general enough. Next consider the $\mathbf{w}_t$, we can see that E_{in} decreases continuously, but E_{out} reaches its smallest value at the early stage of iteration, and start increasing slightly after keep iterating, indicate it may start overfitting. I think it can reflect the core concept of SGD algorithm : select an appropriate large t and stop. When we find that E_{out} seems to stop decreasing, we should just stop and return $\mathbf{w}_t$.

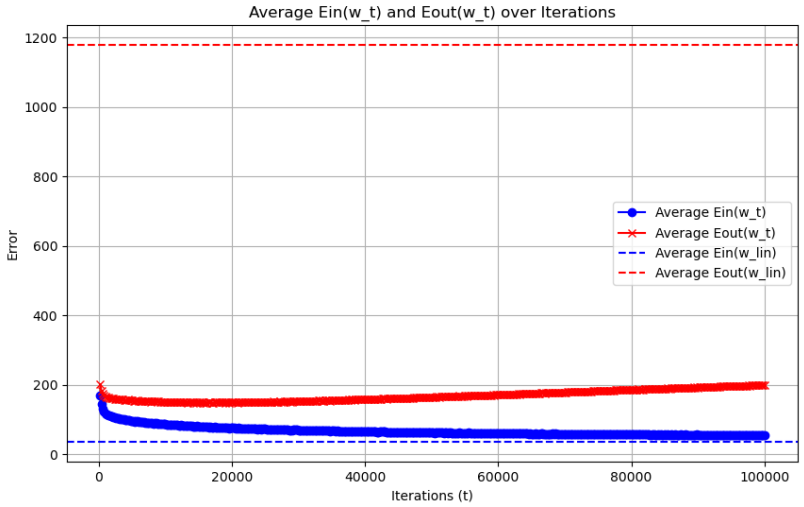

Figure 1:

```python
def sgd_linear_regression(train_data, train_target, validation_data, validation_target, eta=0.01, iterations=100000, interval=200):

    # N = 64, d = 13
    size, d = train_data.shape

    w_t = np.zeros(d)
    ein_sgd_records = []
    eout_sgd_records = []

    for t in range(1, iterations + 1):

        i = np.random.choice(np.arange(size))
        xi = train_data[i]
        yi = train_target[i]

        gradient = 2 * (yi - xi @ w_t) * xi
        w_t += eta * gradient

        if t % interval == 0:
            ein_t = np.mean((train_data @ w_t - train_target) ** 2)
            eout_t = np.mean((validation_data @ w_t - validation_target) ** 2)
            ein_sgd_records.append(ein_t)
            eout_sgd_records.append(eout_t)

    return ein_sgd_records, eout_sgd_records
```

Figure 2: SGD for linear regression

```python
def sgd_exepriments(X, y, N, times):
    Ein_wlin_list = []
    Eout_wlin_list = []
    Ein_sgd_records_list = []
    Eout_sgd_records_list = []
    length = len(X)

    for _ in tqdm(range(times), desc="Running Linear Regression"):

        # training data
        train_indices = np.random.choice(np.arange(length), size=N, replace=False)
        X_train, y_train = X[train_indices], y[train_indices]

        # testing data
        test_indices = li (parameter) X: Any  - set(train_indices))
        X_test, y_test = X[test_indices], y[test_indices]

        # w_lin
        w_lin = linear_regression(X_train, y_train)
        # Ein(w_lin), Eout(w_out)
        Ein = mean_squared_error(X_train, y_train, w_lin)
        Eout = mean_squared_error(X_test, y_test, w_lin)
        Ein_wlin_list.append(Ein)
        Eout_wlin_list.append(Eout)

        # SGD
        Ein_sgd_records, Eout_sgd_records = sgd_linear_regression(X_train, y_train, X_test, y_test)
        Ein_sgd_records_list.append(Ein_sgd_records)
        Eout_sgd_records_list.append(Eout_sgd_records)

    Ein_wlin_mean = np.mean(Ein_wlin_list)
    Eout_wlin_mean = np.mean(Eout_wlin_list)
    Ein_sgd_mean_list = np.mean(Ein_sgd_records_list, axis=0)
    Eout_sgd_mean_list = np.mean(Eout_sgd_records_list, axis=0)

    return Ein_wlin_mean, Eout_wlin_mean, Ein_sgd_mean_list, Eout_sgd_mean_list
```

Figure 3: Experiments of problem 10

```python
def plot(Ein_wlin_mean, Eout_wlin_mean, Ein_sgd_mean_list, Eout_sgd_mean_list):
    # t = 200, 400, 600, ..., 100000
    t_values = np.arange(200, 200 * (len(Ein_sgd_mean_list) + 1), 200)

    plt.figure(figsize=(10, 6))

    plt.plot(t_values, Ein_sgd_mean_list, label="Average Ein(w_t)", color="blue", marker="o")
    plt.plot(t_values, Eout_sgd_mean_list, label="Average Eout(w_t)", color="red", marker="x")

    plt.axhline(y=Ein_wlin_mean, color="blue", linestyle="--", label="Average Ein(w_lin)")
    plt.axhline(y=Eout_wlin_mean, color="red", linestyle="--", label="Average Eout(w_lin)")

    plt.xlabel("Iterations (t)")
    plt.ylabel("Error")
    plt.title("Average Ein(w_t) and Eout(w_t) over Iterations")
    plt.legend()

    # # Show plot
    # plt.grid()
    # plt.show()

    # Save plot
    plt.grid()
    plt.savefig("./10_sgo.png")
    plt.close()
```

Figure 4:

Problem 11

We can see that the value of $E_{in}(\mathbf{w}_{lin})$ - $E_{in}(\mathbf{w}_{poly})$ is positive, which indicate that after transform, it has better E_{in}

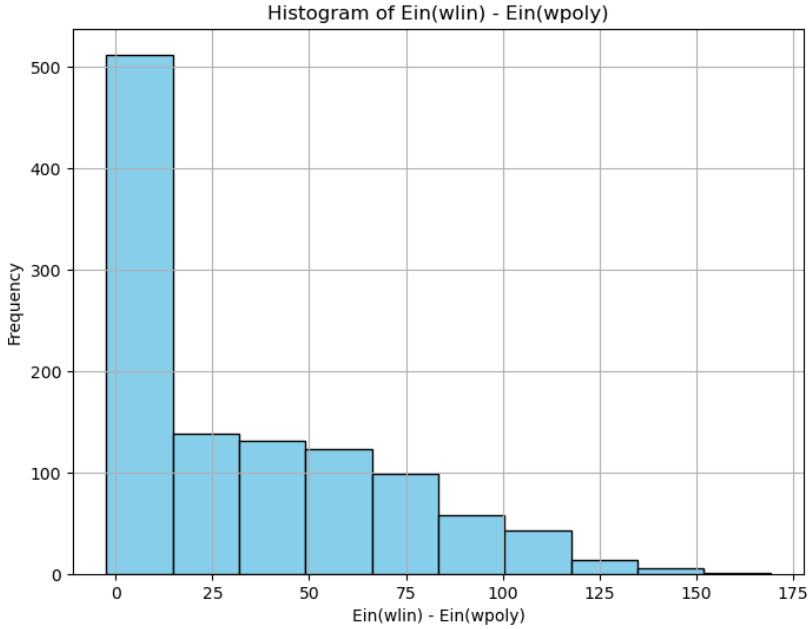

Figure 5:

```
Codeium: Refactor | Explain | Generate Docstring | ×
def polynomial_transform(x, Q):
    poly_features = [x**q for q in range(1, Q + 1)]
    return np.hstack([np.ones((x.shape[0], 1))] + poly_features)
```

Figure 6: polynomial transform

```python
def exepriments(X, y, N, times):

    Ein_list = []
    Eout_list = []

    length = len(X)

    for _ in tqdm(range(times), desc="Running Linear Regression"):

        # randomly selected examples
        train_indices = np.random.choice(np.arange(length), size=N, replace=False)
        X_train, y_train = X[train_indices], y[train_indices]

        # construct the remaining examples
        test_indices = list(set(range(len(X))) - set(train_indices))
        X_test, y_test = X[test_indices], y[test_indices]

        w_lin = linear_regression(X_train, y_train)

        Ein = mean_squared_error(X_train, y_train, w_lin)
        Eout = mean_squared_error(X_test, y_test, w_lin)

        Ein_list.append(Ein)
        Eout_list.append(Eout)

    return Ein_list, Eout_list
```

Figure 7: Experiments of problem 10

```python
def plot_Ein_gain(Ein_wlin_list, Ein_wpoly_list):

    gain = np.array(Ein_wlin_list) - np.array(Ein_wpoly_list)
    mean_gain = np.mean(gain)

    plt.figure(figsize=(8, 6))
    plt.hist(gain, bins=10, color='skyblue', edgecolor='black')
    plt.xlabel("Ein(wlin) - Ein(wpoly)")
    plt.ylabel("Frequency")
    plt.title("Histogram of Ein(wlin) - Ein(wpoly)")

    # # show plot
    # plt.grid(True)
    # plt.show()

    # save plot
    plt.grid()
    plt.savefig("./11_poly_transfrom_Ein.png")
    plt.close()

    print(f"Average gain: {mean_gain}")
```

Figure 8:

Problem 12

The value of $E_{out}(\mathbf{w}_{lin})$ - $E_{out}(\mathbf{w}_{poly})$ is negative, which indicate that after transform, it has worse E_{out}, combining the observation above, after transform, we have better E_{in} and worse E_{out}, which indicate that transform may result in overfitting.

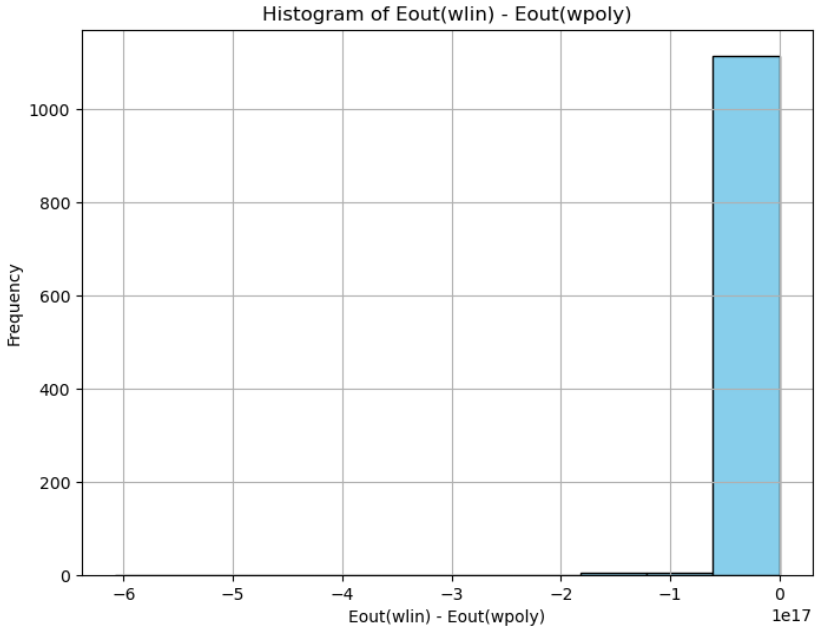

Figure 9:

```python
def plot_Eout_change(Eout_wlin_list, Eout_wpoly_list):

    change = np.array(Eout_wlin_list) - np.array(Eout_wpoly_list)
    mean_change = np.mean(change)

    plt.figure(figsize=(8, 6))
    plt.hist(change, bins=10, color='skyblue', edgecolor='black')
    plt.xlabel("Eout(wlin) - Eout(wpoly)")
    plt.ylabel("Frequency")
    plt.title("Histogram of Eout(wlin) - Eout(wpoly)")

    # # show plot
    # plt.grid(True)
    # plt.show()

    # save plot
    plt.grid()
    plt.savefig("./12_poly_transfrom_Eout.png")
    plt.close()

    print(f"Average gain: {mean_change}")
```

Figure 10:

```python
# Q-order polynomial transform
X_poly = polynomial_transform(dense_data, 3)

Ein_wlin_list, Eout_wlin_list = exepriments(X, labels, N, times)
Ein_wpoly_list, Eout_wpoly_list = exepriments(X_poly, labels, N, times)


# problem 11
plot_Ein_gain(Ein_wlin_list, Ein_wpoly_list)


# problem 12
plot_Eout_change(Eout_wlin_list, Eout_wpoly_list)
```

Figure 11: