collaborator:
B11901040 項達均

5.

$$G(x) = \text{sign}\left(\sum_{t=1}^{2M+1} g_t(\underline{x})\right)$$

$$E_{out}(G) = [\![y \neq \text{sign}\left(\sum_{t=1}^{2M+1} g_t(\underline{x})\right)]\!]$$

at least $M+1$ classifiers $y \neq \text{sign}(g_t(\underline{x}))$ s.t. $E(G) = 1$

$$\Rightarrow \sum_{t=1}^{2M+1} e_t \geq M+1$$

$$\Rightarrow 1 \leq \frac{1}{M+1} \sum_{t=1}^{2M+1} e_t$$

$$\Rightarrow E(G) \leq \frac{1}{M+1} \sum_{t=1}^{2M+1} e_t$$

6.

collaborator: B11901040 項達均

Incorrect: $u_n^{(t+1)} \leftarrow u_n^{(t)} * \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

Correct: $u_n^{(t+1)} \leftarrow u_n^{(t)} / \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

$$U_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)}$$

$$= \sum_{\text{incorrect } n} \left(u_n^{(t)} * \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + \sum_{\text{correct } n} u_n^{(t)} / \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \right)$$

$$= \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \sum_{\text{incorrect } n} u_n^{(t)} + \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} \sum_{\text{correct } n} u_n^{(t)}$$

$$= \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} U_t * \epsilon_t + \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} U_t (1-\epsilon_t)$$

$$= U_t \left(\sqrt{\epsilon_t(1-\epsilon_t)} + \sqrt{\epsilon_t(1-\epsilon_t)} \right)$$

$$= 2\sqrt{\epsilon_t(1-\epsilon_t)}\, U_t$$

$$\therefore \frac{U_{t+1}}{U_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Q.E.D.

7. no collaborators

$$\min_{\alpha} \frac{1}{N}\sum_{n=1}^{N}\left((y_n-s_n)-\alpha g_t(\underline{x_n})\right)^2$$

$$\frac{1}{N}\sum_{n=1}^{N}\left((y_n-s_n)-\alpha g_t(\underline{x_n})\right)^2$$

$$=\frac{1}{N}\sum_{n=1}^{N}\left((y_n-s_n)^2-2\alpha(y_n-s_n)g_t(\underline{x_n})+\alpha^2 g_t^2(\underline{x_n})\right)$$

$$\frac{\partial}{\partial\alpha}\frac{1}{N}\sum_{n=1}^{N}\left((y_n-s_n)^2-2\alpha(y_n-s_n)g_t(\underline{x_n})+\alpha^2 g_t^2(\underline{x_n})\right)=0$$

$$\Rightarrow \sum_{n=1}^{N}\left(-2(y_n-s_n)g_t(\underline{x_n})+2\alpha g_t^2(\underline{x_n})\right)=0$$

$$\Rightarrow 2\alpha\sum_{n=1}^{N}g_t^2(\underline{x_n}) = 2\sum_{n=1}^{N}(y_n-s_n)g_t(\underline{x_n})$$

$$\Rightarrow \alpha=\frac{\sum_{n=1}^{N}(y_n-s_n)g_t(\underline{x_n})}{\sum_{n=1}^{N}g_t^2(\underline{x_n})}$$

From the solution of linear regression, $((y_n-s_n)-g_t(\underline{x_n}))$ are orthogonal to $g_t(\underline{x_n})$

$$\Rightarrow \sum_{n=1}^{N}\left((y_n-s_n)-g_t(\underline{x_n})\right)\cdot g_t(\underline{x_n})=0$$

$$\Rightarrow \sum_{n=1}^{N}(y_n-s_n)g_t(\underline{x_n}) - \sum_{n=1}^{N}g_t^2(\underline{x_n})=0$$

$$\Rightarrow \sum_{n=1}^{N}(y_n-s_n)g_t(\underline{x_n}) = \sum_{n=1}^{N}g_t^2(\underline{x_n})$$

$$\therefore \alpha_1=\frac{\sum_{n=1}^{N}(y_n-s_n)g_t(\underline{x_n})}{\sum_{n=1}^{N}g_t^2(\underline{x_n})}=1$$

Q.E.D.

8.

perform linear regression on $\{(g_t(\underline{x}_n), y_n - s_n)\}_{n=1}^{N}$

collaborator:
B11901040 項達均

$$\min_{\eta} \frac{1}{N} \sum_{n=1}^{N} \left((y_n - s_n) - \eta g_t(\underline{x}_n)\right)^2$$

$$\xRightarrow{\frac{\partial}{\partial \eta}} \frac{1}{N} \sum_{n=1}^{N} 2(-g_t(\underline{x}_n))\left((y_n - s_n) - \eta g_t(\underline{x}_n)\right) = 0$$

$$\Rightarrow \sum_{n=1}^{N} g_t(\underline{x}_n)\left((y_n - s_n) - \eta g_t(\underline{x}_n)\right) = 0$$

$\eta = \alpha_t$

update $s_n \leftarrow s_n + \alpha_t g_t(\underline{x}_n)$

$$\therefore \sum_{n=1}^{N} g_t(\underline{x}_n)\left(y_n - (s_n + \alpha_t g_t(\underline{x}_n))\right) = 0$$

$$\Rightarrow \sum_{n=1}^{N} g_t(\underline{x}_n)(y_n - s_n) = 0$$

Q.E.D.

9.

collaborator :
B11201009 黃勤元

$\because$ one hidden layer including the output neuron

$\therefore L = 2$

$$\begin{cases} s_j^{(2)} = \sum_{i=0}^{d^{(1)}} w_{ij}^{(2)} x_i^{(1)} \\ s_j^{(1)} = \sum_{i=0}^{d^{(0)}} w_{ij}^{(1)} x_i^{(0)} \ , \quad s_{j+1}^{(1)} = \sum_{i=0}^{d^{(0)}} w_{i\,j+1}^{(1)} x_i^{(0)} \\ \delta_j^{(2)} = -2(y_n - s_j^{(2)}) \end{cases}$$

$w_{ij}^{(l)} = 0.5 \ \forall i$ and $1 \le j < d^{(1)}$

$$\delta_j^{(1)} = \sum_k \delta_k^{(2)} w_{jk}^{(2)} \tanh'(s_j^{(1)})$$
$$= \sum_k \delta_k^{(2)} w_{j+1\,k}^{(2)} \tanh'(s_{j+1}^{(1)})$$
$$= \delta_{j+1}^{(1)}$$

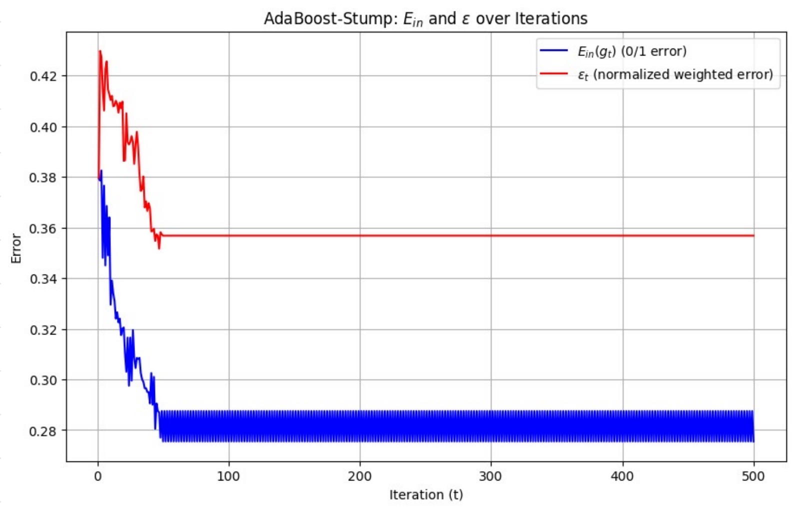
for the backprop algorithm,

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \eta x_i^{(l-1)} \delta_j^{(l)}$$

$$w_{ij}^{(1)} = w_{ij}^{(1)} - \eta x_i^{(0)} \delta_j^{(1)}$$
$$= w_{i\,j+1}^{(1)} - \eta x_i^{(0)} \delta_{j+1}^{(1)}$$
$$= w_{i\,j+1}^{(1)}$$

Q.E.D.

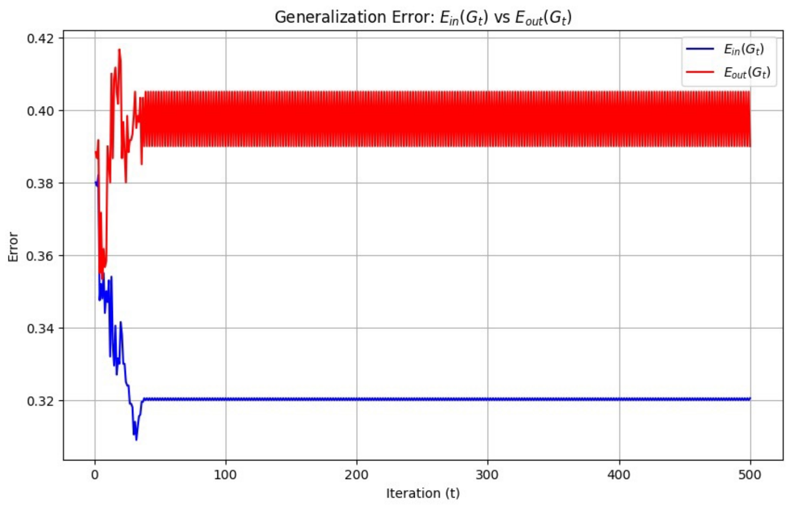
10. collaborators: B11901073 林禹融



The AdaBoost algorithm reduces $E_{in}(g_t)$ and ε_t quickly and remain same value respectively.

```python
class DecisionStump:
    def __init__(self):
        self.feature_index = None
        self.threshold = None
        self.sign = None

    def train(self, X, y, weights):
        _, n = X.shape
        best_error = float('inf')

        # Search for the best dimension, threshold, and sign
        for feature_index in range(n):
            thresholds = np.unique(X[:, feature_index])
            for threshold in thresholds:
                for sign in [-1, 1]:
                    predictions = sign * np.sign(X[:, feature_index] - threshold)
                    error = np.sum(weights * (predictions != y))
                    if error < best_error:
                        best_error = error
                        self.feature_index = feature_index
                        self.threshold = threshold
                        self.sign = sign

        return best_error

    def predict(self, X):
        predictions = self.sign * np.sign(X[:, self.feature_index] - self.threshold)
        return predictions

# AdaBoost Algorithm
def adaBoost(X_train, y_train, X_test, y_test, T=500):
    m = X_train.shape[0]
    weights = np.ones(m) / m
    classifiers = []
    alphas = []

    # Metrics for plotting
    E_in = []
    epsilons = []

    for t in range(T):
        stump = DecisionStump()
        error = stump.train(X_train, y_train, weights)

        # Avoid divide-by-zero
        if error == 0:
            error = 1e-10

        # Compute alpha
        alpha = 0.5 * np.log((1 - error) / error)
        alphas.append(alpha)
        classifiers.append(stump)

        # Update weights
        predictions = stump.predict(X_train)
        weights *= np.exp(-alpha * y_train * predictions)

        # Compute 0/1 error (E_in) and epsilon
        E_in.append(zero_one_loss(y_train, np.sign(sum(alpha * clf.predict(X_train) for clf, alpha in zip(classifiers, alphas)))))
        epsilons.append(error)

        print(f"Iteration {t+1}: Error = {error:.4f}, Alpha = {alpha:.4f}, E_in = {E_in[-1]:.4f}")

    # Final testing error
    final_predictions = np.sign(sum(alpha * clf.predict(X_test) for clf, alpha in zip(classifiers, alphas)))
    E_out = zero_one_loss(y_test, final_predictions)

    print(f"Final Test Error (E_out): {E_out:.4f}")

    return E_in, epsilons
```

11. collaborators: B11901073 林禹融

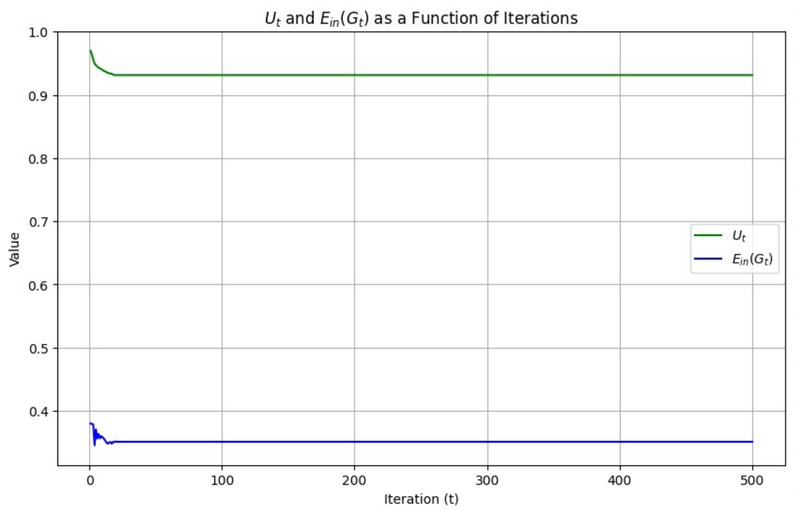


① Although $E_{in}(G_t)$ is relatively high initially, it decreases fast and remains around 0.32 after about 50 iterations.

② $E_{out}(G_t)$ oscillates between about 0.39 and 0.41 after about 50 iterations. Each new weak classifier added to G_t contributes a small correction based on its α_t. If the weak classifier's corrections are small and do not generalize well to the test data, they may slightly shift the decision boundary of G_t in an inconsistent manner. However, the mean value of the oscillation remains about 0.395 after about 50 iterations.

① & ② indicate that after a certain number of iterations, the model reaches its generalization capacity.

12. no collaborators



Ut decreases steeply in the first few iterations and remains at about 0.93, which suggest that AdaBoost algorithm quickly adjusts the sample weights and coverges efficiently. However, it doesn't reach perfect accuracy.