1. a. $T_1 = a\bar{X} + (1-a) c_* S$

$$E(T_1) = aE(\bar{X}) + (1-a)E(c_* S)$$

$$E(\bar{X}) = E(c_* S) = \gamma$$

$$\therefore E(T_1) = a\gamma + (1-a)\gamma$$

$$= a\gamma + \gamma - a\gamma$$

$$= \gamma$$

$\therefore$ $E(T_1)$ is unbiased for any constant a.

b. $MSE(T_1) = E[(T_1 - \gamma)^2]$

$= E[T_1 - E(T_1)]^2 + (E(T_1) - \gamma)^2$

$= Var(T_1) + Bias(T_1)$

T_1 is unbiased, so

$= Var(T_1)$

$= Var(a\bar{X} + (1-a)c_* S)$

$= a^2 Var(\bar{X}) + (1-a)^2 Var(c_* S)$

$= a^2 \frac{\gamma^2}{n} + (1-a)^2 (c_*^2 (E(S^2) - E(S)^2)$

$= a^2 \frac{\gamma^2}{n} + (1-a)^2 (c_*^2 (\gamma^2 - \frac{\gamma^2}{c_*})$

$= a^2 \frac{\gamma^2}{n} + \gamma^2 (1-a)^2 (c_*^2 - 1)$

Since S^2 is unbiased,

$\hookrightarrow E(S^2) = \gamma^2$

Since $c_* S$ is unbiased,

$\hookrightarrow E(c_* S) = \gamma$

$E(S) = \frac{\gamma}{c_*}$

let $f(a) = a^2 \frac{\gamma^2}{n} + \gamma^2 (1-a)^2 (c_*^2 - 1)$

$f'(a) = 2a \frac{\gamma^2}{n} - 2\gamma^2 (1-a)(c_*^2 - 1)$

$$0 \overset{!}{=} 2a\frac{\gamma^2}{n} - 2\gamma^2(1-a)(c_*^2-1)$$

$$\frac{a}{n} = (c_*^2-1)(1-a)$$

$$a = \frac{n(c_*^2-1)}{n(c_*^2-1)+1}$$

$$\therefore \boxed{T_1^* = \frac{n(c_*^2-1)}{n(c_*^2-1)+1}\bar{X} + \frac{1}{n(c_*^2+1)} c_* S}$$

c. $T_2 = a_1\bar{X} + a_2(c_* S)$

$$MSE(T_2) = E((T_2-\gamma)^2)$$

$$= Var(T_2) + (E(T_2)-\gamma)^2$$

Consider $Var(T_2)$:

$$Var(T_2) = Var(a_1\bar{X} + a_2(c_* S))$$

$$= a_1^2 Var(\bar{X}) + a_2^2 Var(c_* S)$$

from (b),

$$= a_1^2\frac{\gamma}{n} + a_2^2\gamma^2(c_*^2-1)$$

Consider $(E(T_2)-\gamma)^2$

$$= (E(a_1\bar{X} + a_2 c_* S) - \gamma)^2$$

$$= (a_1 E(\bar{X}) + a_2 E(c_* S) - \gamma)^2$$

from (a),

$$= (a_1\gamma + a_2\gamma - \gamma)^2$$

$$= (a_1 + a_2 - 1)^2\gamma^2$$

$$\therefore MSE(T_2) = a_1^2 \frac{\gamma^2}{n} + a_2^2 \gamma^2 (C_*^2 - 1) + (a_1 + a_2 - 1)^2 \gamma^2$$

$$\frac{\partial f}{\partial a_1} = 2a_1 \frac{\gamma^2}{n} + 2(a_1 + a_2 - 1)\gamma^2$$

$$0 = 2a_1 \frac{\gamma^2}{n} + 2(a_1 + a_2 - 1)\gamma^2$$

$$0 = \frac{a_1}{n} + a_1 + a_2 - 1$$

$$n = (n+1)a_1 + na_2$$

$$\frac{\partial f}{\partial a_2} = 2a_2 \gamma^2 (C_*^2 - 1) + 2(a_1 + a_2 - 1)\gamma^2$$

$$0 = 2a_2 \gamma^2 (C_*^2 - 1) + 2(a_1 + a_2 - 1)\gamma^2 \quad ②$$

$$0 = a_2(C_*^2 - 1) + a_1 + a_2 - 1$$

$$1 = C_*^2 a_2 + a_1 \quad ②$$

① and ②:

$$\left(\begin{array}{cc|c} 1 & c_*^2 & 1 \\ n+1 & n & n \end{array}\right)$$

$R_2 \to R_2 - (n+1)R_1$

$$\left(\begin{array}{cc|c} 1 & c_*^2 & 1 \\ 0 & n - nc_*^2 - c_*^2 & -1 \end{array}\right)$$

$R_2 \to \dfrac{R_2}{n - nc_*^2 - c_*^2}$

$$\left(\begin{array}{cc|c} 1 & c_*^2 & 1 \\ 0 & 1 & \dfrac{1}{n(c_*^2 - 1) + c_*^2} \end{array}\right)$$

$R_1 \to R_1 - c_*^2 R_2$

$$\left(\begin{array}{cc|c} 1 & 0 & 1 - \dfrac{c_*^2}{n(c_*^2 - 1) + c_*^2} \\ 0 & 1 & \dfrac{1}{n(c_*^2 - 1) + c_*^2} \end{array}\right)$$

$$a_1 = 1 - \frac{c_*^2}{n(c_*^2 - 1) + c_*^2}$$

$$a_2 = \frac{1}{n(c_*^2 - 1) + c_*^2}$$

a. $MSE(T_2) = \frac{\gamma^2}{n} a_1^2 + \gamma^2 (c_*^2 - 1) a_2^2 + \gamma (a_1 + a_2 - 1)^2$

$$= \frac{\gamma^2}{\cancel{n}} \left(\frac{\cancel{n}(c_*^2 - 1)}{n(c_*^2-1)+c_*^2} \right) + \gamma^2 (c_*^2 - 1) \left(\frac{1}{n(c_*^2-1)+c_*^2} \right)^2 + \gamma \left(\frac{n(c_*^2-1)+1}{n(c_*^2-1)+c_*^2} - 1 \right)^2$$

Let $n(c_*^2-1)+c_*^2 = A$

$$= \gamma^2 \left(\frac{c_*^2 - 1}{A} \right) + \gamma^2 \left(\frac{c_*^2 - 1}{A^2} \right) + \gamma \left(\left(\frac{n(c_*^2-1)+1}{A} \right)^2 - 2 \left(\frac{n(c_*^2-1)+1}{A} \right) + 1 \right)$$

Let $c_*^2 - 1 = B$

$$= \gamma^2 \frac{B}{A} + \gamma^2 \frac{B}{A^2} + \gamma \frac{n^2B^2 + 2nB + 1}{A^2} - 2\frac{B}{A} + 1$$

$$= \gamma^2 \frac{B}{A} + \gamma^2 \frac{B}{A^2} + n^2 \gamma \frac{B^2}{A^2} + 2n \frac{B}{A^2} + \frac{1}{A^2} - 2\frac{B}{A} + 1$$

$$= \frac{\gamma^2 (c_*^2 - 1)}{n(c_*^2 - 1) + c_*^2}$$

$$MSE(T_1) = \frac{\gamma^2 (c_*^2 - 1)}{n(c_*^2 - 1) + 1}$$

Since $c_*^2 > 1$

$$n(c_*^2 - 1) + c_*^2 > n(c_*^2 - 1) + 1$$

$$\therefore \quad MSE(T_2^*) = \frac{\gamma^2 (c_*^2 - 1)}{n(c_*^2 - 1) + c_*^2} < \frac{\gamma^2 (c_*^2 - 1)}{n(c_*^2 - 1) + 1} = MSE(T_1)$$

e. $MSE(V_+) = E[(\max(0, T_2^*) - \gamma)^2]$

$MSE(T_2^*) = E[(T_2^* - \gamma)^2]$

if $T_2^* \geq 0$,

$\max(0, T_2^*) = T_2^*$

$\therefore E((\max(0, T_2^*) - \gamma)^2) = E((T_2^* - \gamma)^2)$

if $T_2^* < 0$,

$\max(0, T_2^*) = 0$

$\therefore E((\max(0, T_2^*) - \gamma)^2) = E(\gamma^2)$

$\therefore$ in both cases, $E[(\max(0, T_2^*) - \gamma)^2] \leq E[(T_2^* - \gamma)^2]$

so $MSE(V_+) \leq MSE(T_2^*)$.

f. $T_3 = a_1 \bar{x} + a_2 c_* S + a_3$

T_3 is unbiased, so

$MSE(T_3) = Var(T_3)$

$= Var(a_1 \bar{x} + a_2 c_* S + a_3)$

$= a_1^2 Var(\bar{x}) + a_2^2 Var(c_* S)$

$= a_1^2 \frac{\gamma^2}{n^2} + a_2^2 \gamma^2 (c_*^2 - 1)$

Let $f(a_1, a_2) = a_1^2 \frac{\gamma^2}{n^2} + a_2^2 \gamma^2 (c_*^2 - 1)$

$\frac{\partial f}{\partial a_1} = 2a_1 \frac{\gamma^2}{n^2}$

$$0 = 2a_1 \frac{\gamma^2}{n^2}$$

$$a_1 = 0$$

$$\frac{\partial f}{\partial a_2} = 2a_2 \gamma^2 (c_*^2 - 1)$$

$$0 = 2a_2 \gamma^2 (c_*^2 - 1)$$

$$a_2 = 1$$

we know T_3 is unbiased.

$$\therefore E(T_3) = \gamma$$

$$E(a_1 \bar{x} + a_2 c_* s + a_3) = \gamma$$

$$E(a_3) = \gamma$$

$$a_3 = \gamma$$

2. a. $\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$

Show: $r_{t,i} = y_i - f_{t-1}(x_i)$

GC1:

$$r_{t,i} = -\frac{\partial}{\partial f(x_i)} \sum_{j=1}^{n} \frac{1}{2}(y_j - f(x_j))^2 \Big|_{f=f_{t-1}}$$

$$= -\sum_{j=1}^{n} (y_j - f_{t-1}(x_j))\delta_{ij}$$

$$= y_i - f_{t-1}(x_i)$$

GC2:

$$h_t = \arg\min_{h \in F} \sum_{i=1}^{n} \frac{1}{2}(r_{t,i} - h(x_i))$$

b. $\alpha_t = \arg\min_{\alpha} \sum_{i=1}^{n} \frac{1}{2}(y_i - (f_{t-1}(x_i) + \alpha h_t(x_i)))^2$

$$\frac{d}{d\alpha} = \sum_{i=1}^{n} (y_i - f_{t-1}(x_i) - \alpha h_t(x_i))(-h_t(x_i))$$

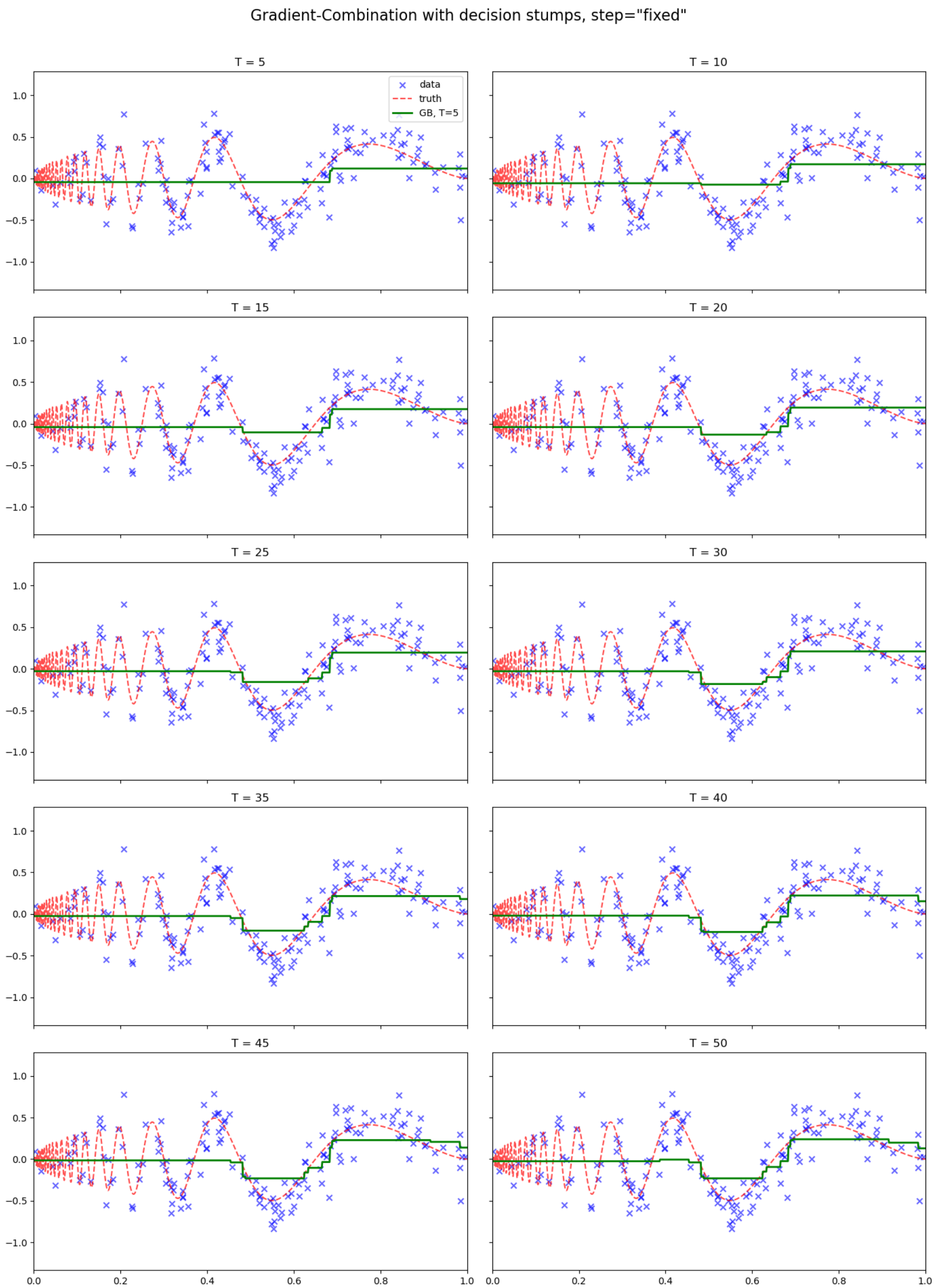
$$= \sum_{i=1}^{n} (r_{t,i} - \alpha h_t(x_i)) h_t(x_i)$$

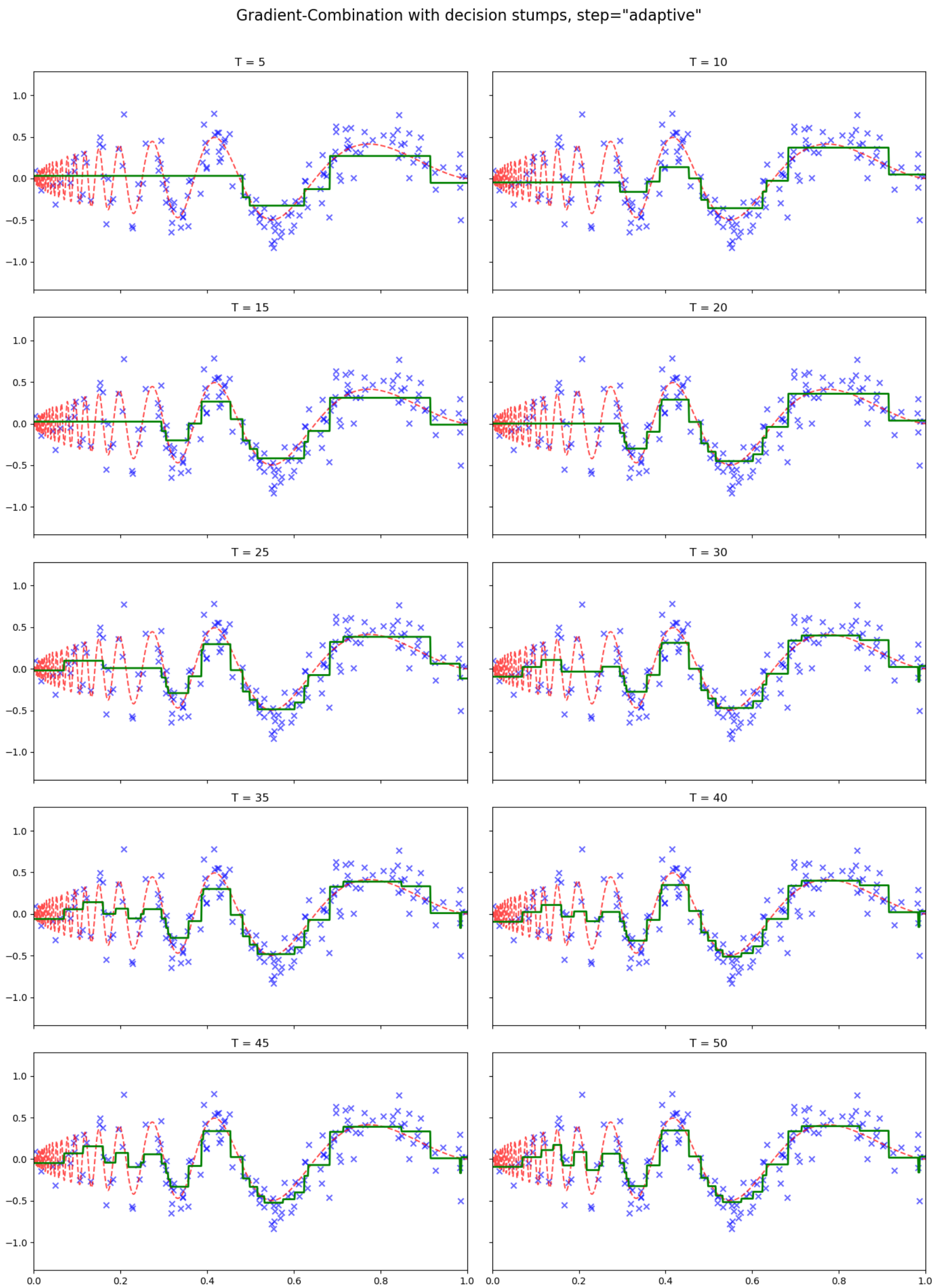
$$\frac{d}{d\alpha} = 0$$

$$0 = \sum_{i=1}^{n} (r_{t,i} - \alpha h_t(x_i)) h_t(x_i)$$

$$\alpha_t = \frac{\sum_{i=1}^{n} r_{t,i} h_t(x_i)}{\sum_{i=1}^{n} h_t(x_i)^2} = \frac{\sum (y_i - f_{t-1}(x_i)) h_t(x_i)}{\sum h_t(x_i)^2}$$

c.


Gradient-Combination with decision stumps, step="fixed"
T = 5
T = 10
T = 15
T = 20
T = 25
T = 30
T = 35
T = 40
T = 45
T = 50
data
truth
GB, T=5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0

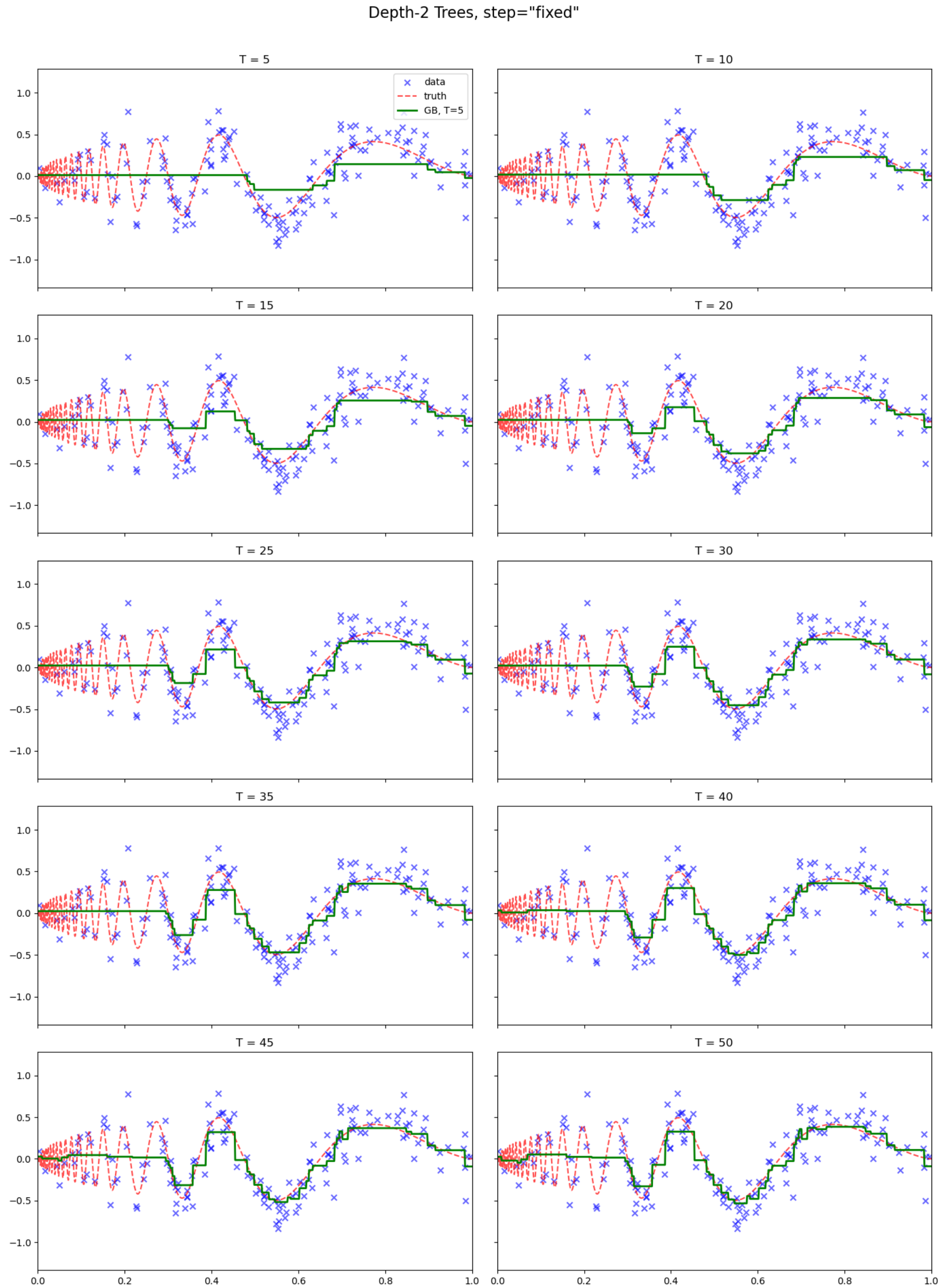


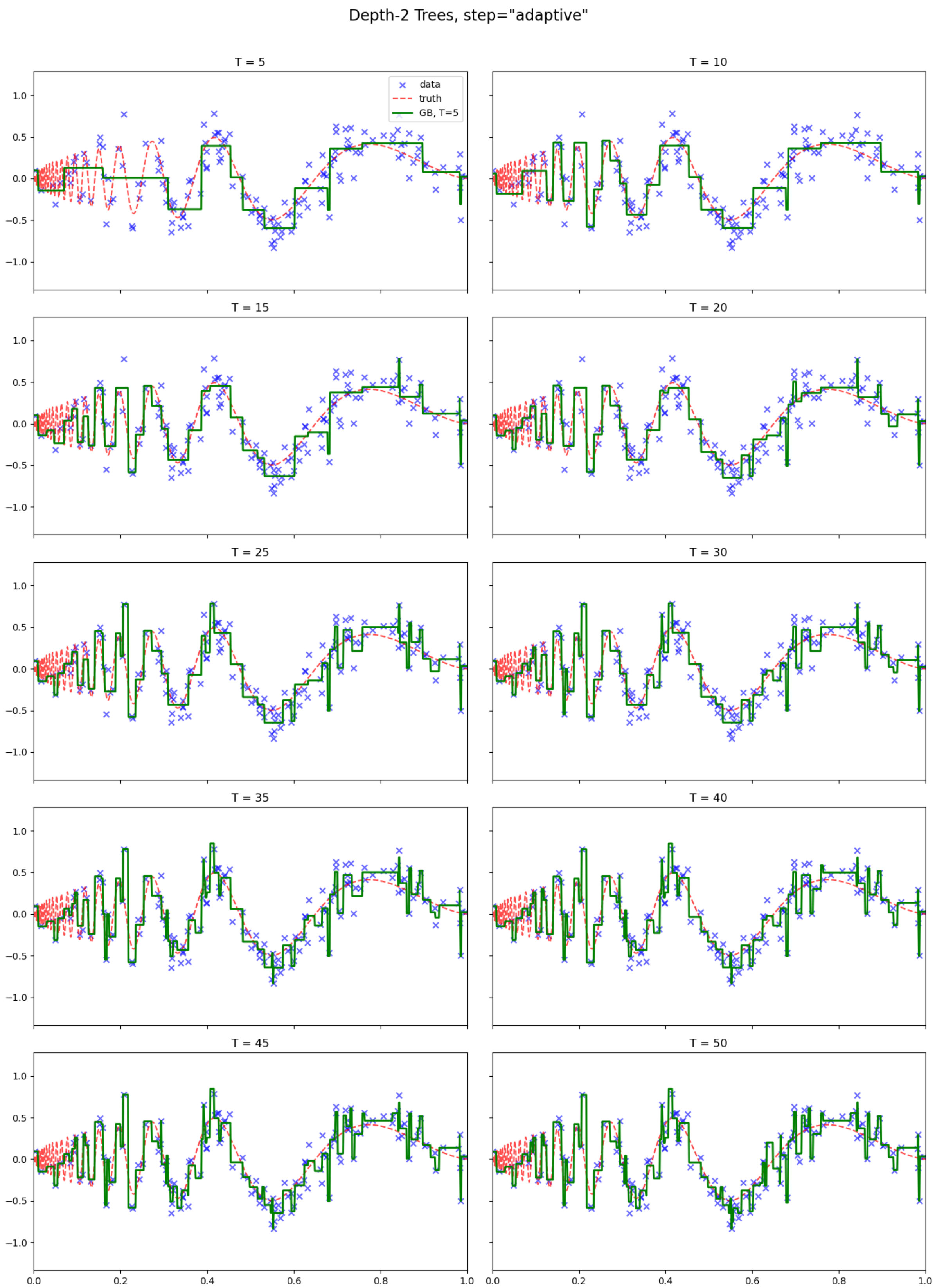
With the fixed step size, it underfits significantly, even at high T. The adaptive step size converges much faster and also fits the data much better.

```python
yy_true = f(xx.flatten())

def gradient_combination(X, y, xx, T, max_depth, step='fixed', alpha_fixed=0.1):
    n = X.shape[0]
    f_pred = np.zeros(n)
    f_grid = np.zeros_like(xx.flatten())
    models = []
    alphas = []

    for t in range(T):
        r = y - f_pred

        stump = DecisionTreeRegressor(max_depth=max_depth)
        stump.fit(X, r)
        h_train = stump.predict(X)
        h_grid = stump.predict(xx.reshape(-1, 1))

        if step == 'fixed':
            alpha = alpha_fixed
        else:
            alpha = np.dot(r, h_train) / np.dot(h_train, h_train)

        f_pred += alpha * h_train
        f_grid += alpha * h_grid

        models.append(stump)
        alphas.append(alpha)

    return f_grid

T_list = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

for step in ['fixed', 'adaptive']:
    fig, axes = plt.subplots(5, 2, figsize=(14, 20), sharex=True, sharey=True)
    fig.suptitle(f'Gradient-Combination with decision stumps, step="{step}"', fontsize=16)

    for ax, T in zip(axes.flat, T_list):
        yb = gradient_combination(X, y, xx, T=T, max_depth=1,
                                  step=step, alpha_fixed=0.1)

        ax.scatter(X, y, marker='x', color='blue', alpha=0.6, label='data')
        ax.plot(xx, yy_true, 'r--', alpha=0.7, label='truth')
        ax.plot(xx, yb, 'g-', lw=2, label=f'GB, T={T}')
        ax.set_xlim(0, 1)
        ax.set_ylim(np.min(y) - 0.5, np.max(y) + 0.5)
        ax.set_title(f'T = {T}')
        if (T == T_list[0] and step=='fixed'):
            ax.legend(loc='upper right')

    plt.tight_layout(rect=[0, 0.03, 1, 0.97])
    plt.show()
```

8] ✓ 3.0s

d.


Depth-2 Trees, step="fixed"
T = 5
T = 10
T = 15
T = 20
T = 25
T = 30
T = 35
T = 40
T = 45
T = 50
data
truth
GB, T=5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0



for a fixed alpha = 0.1, fit is piecewise constant but tracks the curvature well. adaptive alpha: each tree is stronger, so adaptive boosting zooms in on residuals very quickly. therefore, it ends up overfitting the data.

```python
T_list = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

for step in ['fixed', 'adaptive']:
    fig, axes = plt.subplots(5, 2, figsize=(14, 20), sharex=True, sharey=True)
    fig.suptitle(f'Depth-2 Trees, step="{step}"', fontsize=16)

    for ax, T in zip(axes.flat, T_list):
        yb = gradient_combination(X, y, xx, T=T, max_depth=2,
                                  step=step, alpha_fixed=0.1)
        ax.scatter(X, y, marker='x', color='blue', alpha=0.6, label='data')
        ax.plot(xx.flatten(), yy_true, 'r--', alpha=0.7, label='truth')
        ax.plot(xx.flatten(), yb, 'g-', lw=2, label=f'GB, T={T}')
        ax.set_xlim(0, 1)
        ax.set_ylim(np.min(y) - 0.5, np.max(y) + 0.5)
        ax.set_title(f'T = {T}')
        if T == T_list[0]:
            ax.legend(loc='upper right')

    plt.tight_layout(rect=[0, 0.03, 1, 0.97])
    plt.show()
```

c. $l(y, \hat{y}) = \log(1 + e^{-y\hat{y}})$ $\quad y \in \{-1, 1\}$

$$r_{t,i} = -\frac{\partial}{\partial f(x_i)} \sum_{j=1}^{n} \log(1 + e^{-y_j f(x_j)}) \Big|_{f=f_{t-1}} = -\left(-\frac{y_i e^{-y_i f_{t-1}(x_i)}}{1 + e^{-y_i f_{t-1}(x_i)}}\right) = \frac{y_i}{1 + e^{y_i f_{t-1}(x_i)}}$$

$h \in F$

$$h_t = \underset{h \in F}{\arg\min} \sum_{i=1}^{n} \log(1 + e^{-r_{t,i} h(x_i)})$$

f. choose:

$$\alpha_t = \arg\min_{\alpha} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \{ f_{t-1}(x_i) + \alpha h_t(x_i) \}\right)\right)$$

derivative and set to zero:

$$0 = \sum_{i=1}^{n} \frac{-y_i h_t(x_i)\, e^{-y_i(f_{t-1}(x_i) + \alpha h_t(x_i))}}{1 + e^{-y_i(f_{t-1}(x_i) + \alpha h_t(x_i))}}$$

so in α,

$$\sum_i y_i h_t(x_i) \cdot \frac{1}{1 + e^{y_i(f_{t-1}(x_i) + \alpha h_t(x_i))}} = 0$$

No closed form exists because the logistic loss terms for different i couple α in separate exponentials.

9. we can't solve for α_t directly.

but we can approximate it using a 1D line search each boosting round $\nearrow$ cost of $O(Kn)$ extra per round.

↳ Backtracking line search → Armijo rule

↳ Brent's method or golden section search

- $\phi(\alpha) = \sum_i l(y_i, f_{t-1}(x_i) + \alpha h_t(x_i))$

- use univariate optimizer to find minimum on $\alpha \in [0, A]$

↳ Newton Raphson on $\phi'(\alpha) = 0$

- iterate $\alpha_{k+1} = \alpha_k - \dfrac{\phi'(\alpha_k)}{\phi''(\alpha_k)}$