modules/plots.py

```python
# -*- coding: utf-8 -*-
"""function for plot."""
import matplotlib.pyplot as plt
import numpy as np
from grid_search import get_best_parameters


def prediction(w0, w1, mean_x, std_x):
    """Get the regression line from the model."""
    x = np.arange(1.2, 2, 0.01)
    x_normalized = (x - mean_x) / std_x
    return x, w0 + w1 * x_normalized


def base_visualization(grid_losses, w0_list, w1_list,
                       mean_x, std_x, height, weight):
    """Base Visualization for both models."""
    w0, w1 = np.meshgrid(w0_list, w1_list)

    fig = plt.figure()

    # plot contourf
    ax1 = fig.add_subplot(1, 2, 1)
    cp = ax1.contourf(w0, w1, grid_losses.T, cmap=plt.cm.jet)
    fig.colorbar(cp, ax=ax1)
    ax1.set_xlabel(r'$w_0$')
    ax1.set_ylabel(r'$w_1$')
    # put a marker at the minimum
    loss_star, w0_star, w1_star = get_best_parameters(w0_list, w1_list, grid_losses)
    ax1.plot(w0_star, w1_star, marker='*', color='r', markersize=20)

    # plot f(x)
    ax2 = fig.add_subplot(1, 2, 2)
    ax2.scatter(height, weight, marker=".", color='b', s=5)
    ax2.set_xlabel("x")
    ax2.set_ylabel("y")
    ax2.grid()

    return fig


def grid_visualization(grid_losses, w0_list, w1_list,
                       mean_x, std_x, height, weight):
    """Visualize how the trained model looks like under the grid search."""
    fig = base_visualization(grid_losses, w0_list, w1_list, mean_x, std_x, height, weight)

    loss_star, w0_star, w1_star = get_best_parameters(w0_list, w1_list, grid_losses)
    # plot prediction
    x, f = prediction(w0_star, w1_star, mean_x, std_x)
    ax2 = fig.get_axes()[2]
    ax2.plot(x, f, 'r')

```

```
    return fig


def gradient_descent_visualization(gradient_losses, gradient_ws,
                                   grid_losses, grid_w0, grid_w1,
                                   mean_x, std_x, height, weight, n_iter=None):
    """Visualize how the loss value changes until n_iter."""
    fig = base_visualization(grid_losses, grid_w0, grid_w1, mean_x, std_x,
height, weight)

    ws_to_be_plotted = np.stack(gradient_ws)
    if n_iter is not None:
        ws_to_be_plotted = ws_to_be_plotted[:n_iter]

    ax1, ax2 = fig.get_axes()[0], fig.get_axes()[2]
    ax1.plot(ws_to_be_plotted[:, 0], ws_to_be_plotted[:, 1],
             marker='o', color='w', markersize=10)
    pred_x, pred_y = prediction(ws_to_be_plotted[-1, 0], ws_to_be_plotted[-1, 1],
                                mean_x, std_x)
    ax2.plot(pred_x, pred_y, 'r')

    return fig
```