

```
1: import week6
2: import sgd
3: import pandas as pd
4: import matplotlib.pyplot as plt
5: import numpy as np
6: import sys
7:
8: def run_constant(alpha=1, n=5):
9:     T = pd.read_csv("data/T.csv").values
10:    fg = week6.generate_optimisation_functions(
11:        T, minibatch_size=n, seed=None)
12:    o = sgd.StochasticGradientDescent()
13:    o.alg("constant")
14:    o.step_size(alpha)
15:    o.function_generator(fg)
16:    xs = []
17:    fs = []
18:    start = np.array([3, 3])
19:    o.start(start)
20:    xs.append(o._x_value)
21:    for i in range(200):
22:        o.step()
23:        print(f"alpha={alpha}:", o._x_value)
24:        xs.append(o._x_value)
25:        fs.append(week6.f(o._x_value, T))
26:    return {
27:        "x1": [x[0] for x in xs],
28:        "x2": [x[1] for x in xs],
29:        "f": fs,
30:    }
31:
32: np.random.seed(int(sys.argv[1]))
33:
34: T = pd.read_csv("data/T.csv").values
35:
36: x_min, x_max, y_min, y_max = [-5, 5, -5, 5]
37: # Generate data for wireframe plot
38: resolution = 100
39: x_range = np.linspace(x_min, x_max, resolution)
40: y_range = np.linspace(y_min, y_max, resolution)
41: X, Y = np.meshgrid(x_range, y_range)
42:
43: # Plot wireframe
44: fig = plt.figure(figsize=(12, 6))
45: resolution = 100
46: Z_contour = np.zeros_like(X)
47: for i in range(resolution):
48:     for j in range(resolution):
49:         Z_contour[i, j] = week6.f([X[i, j], Y[i, j]], T)
50:
51: # Plot contour
52: ax_contour = fig.add_subplot(122)
53: contour = ax_contour.contourf(X, Y, Z_contour, levels=20, cmap='viridis')
54: plt.colorbar(contour, ax=ax_contour, label='$f_T(x)$')
55: ax_contour.set_xlabel('$x_1$')
56: ax_contour.set_ylabel('$x_2$')
57: ax_contour.set_xlim([-5, 5])
58: ax_contour.set_ylim([-5, 5])
59: plt.suptitle('Stochastic Gradient Descent on $f_T(x)$ with different mini-batch sizes')
60:
61: ax_f = fig.add_subplot(121)
62:
63: for n in [1, 5, 10, 15, 20, 25]:
64:     alpha = 0.5
65:     run = run_constant(alpha, n=n)
66:     label = f"$\\alpha={alpha}$, $n={n}$, final $f_T(x)={run['f'][-1]:.3f}$"
67:     ax_contour.plot(run["x1"], run["x2"], label=label)
68:     ax_f.plot(run['f'], label=label)
69:
70: ax_f.set_yscale('log')
71: ax_f.set_xlabel("iteration $t$")
72: ax_f.set_ylabel("$f_T(x_t)$")
73: ax_f.legend(loc="upper right")
74: plt.savefig("fig/biii.pdf")
75: plt.show()
```