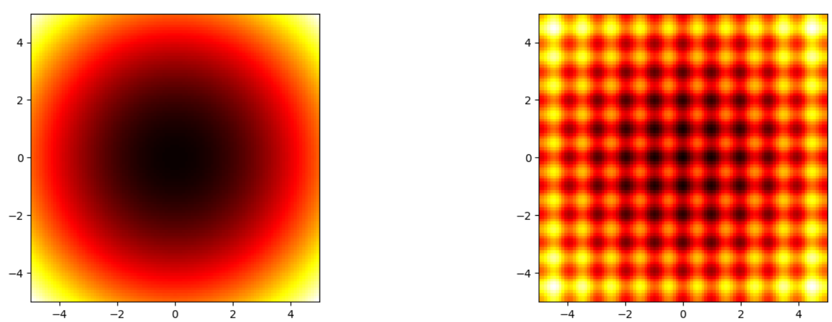
CT5141 Lab Week 5 – Hill-climbing

James McDermott

Solutions are all in `hill_climb_sol.py`.

1. Download `hill_climb.py` and use it to solve `onemax` for $n = 16$, that is the simple bitstring problem where `f = sum`. E.g. the bitstring `0111011101110111` has `onemax` value 12.
2. Our hill-climbing code is inefficient because it recalculates `f(x)` all the time. Can we improve it?
3. Add code to record both the iteration number and current `f` value at each step, and return it as a Numpy array. Use it to make a plot of `f` (on the vertical axis) against iterations during a longer run with $n = 256$. What do we observe about this plot?
4. Try `onemax` for various problem sizes, e.g. $n = 4$, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, with `its = 2000`, and plot the best f value against n. What do we observe?
5. We'll switch now to real-valued optimisation. Use the real-valued versions of `init` and `nbr` in `hill_climb.py` to **minimise** (not maximise!) the `sphere` and `rastrigin` functions. Try $n = 2$, 8, 32. `sphere` should be easy, and `rastrigin` a bit harder! The 2D versions of `sphere` and `rastrigin` are illustrated below. For both of these test problems, the optimum is at the origin.



6. For `onemax`, for small n, we should find that the algorithm reaches the optimum quickly. But it doesn't stop, it just keeps searching for improvements even though no improvement is possible. For `sphere`, it may reach *very close* to the optimum, and bounce around nearby. In both cases, we know the value of f at the optimum. How can we tell the algorithm to *stop* at this point? Bear in mind that we should keep `hill_climb()` **generic**, not problem-specific.
7. Design an objective function on bitstrings where hill-climbing doesn't work, even for a small size such as $n = 16$.
8. Try this code and explain what is happening. (By the way you can press Ctrl-D to quit.)

```python
f = lambda x: float(input(f"Here is my guess: {x}. How many are right? "))
hill_climb(f, lambda: bitstring_init(8), bitstring_nbr, its=20)
```

(We'll look at LAHC in next week's lab.)