Lecture 24

The Gradient and Max-Min Problems

The gradient of a function of multiple variables

If $f(x_1, x_2, \ldots, x_n)$ is a real-valued function of n variables, i.e. x_1, x_2, $\ldots$, x_n, then the *gradient* of f is the vector:

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \frac{\partial f}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{x}) \end{pmatrix},$$

where we use the vector notation $\mathbf{x} = (x_1, x_2, \ldots, x_n)'$. For example, if $f(x_1, x_2) = x_1 e^{-x_1^2 - x_2^2 + \sin(x_1 x_2)}$, then

$$\nabla f(\mathbf{x}) = \begin{pmatrix} (1 + x_1(-2x_1 + \cos(x_1 x_2) x_2)) e^{-x_1^2 - x_2^2 + \sin(x_1 x_2)} \\ x_1(-2x_2 + \cos(x_1 x_2) x_1) e^{-x_1^2 - x_2^2 + \sin(x_1 x_2)} \end{pmatrix}.$$

Note that the gradient is a vector-valued function of multiple variables. This type of function is often called a *vector field*.

The gradient vector has important features with geometric meaning.

- The gradient vector points in the direction in which f is increasing the most.
- The length of the gradient vector is equal to the derivative of f in that direction.
- The gradient vector is perpendicular to the level curve through that point.

We can visualize a gradient (or other vector field) using the Matlab command `quiver` (makes arrows).

```
≫ [x1,x2] = meshgrid(-2:0.2:2);
≫ z = x1.*exp(-x1.^2-x2.^2+sin(x1.*x2)); % function then partial derivatives
≫ dz_dx1 = (1+x1.*(-2*x1+cos(x1.*x2).*x2)).*exp(-x1.^2-x2.^2+sin(x1.*x2));
≫ dz_dx2 = x1.*(-2*x2+cos(x1.*x2).*x1).*exp(-x1.^2-x2.^2+sin(x1.*x2));
≫ contour(x1,x2,z,10)
≫ hold on
≫ quiver(x1,x2,dz_dx1,dz_dx2)
≫ hold off
```

Compare this to the surface plot:

```
≫ figure()
≫ surf(x1,x2,z)
```

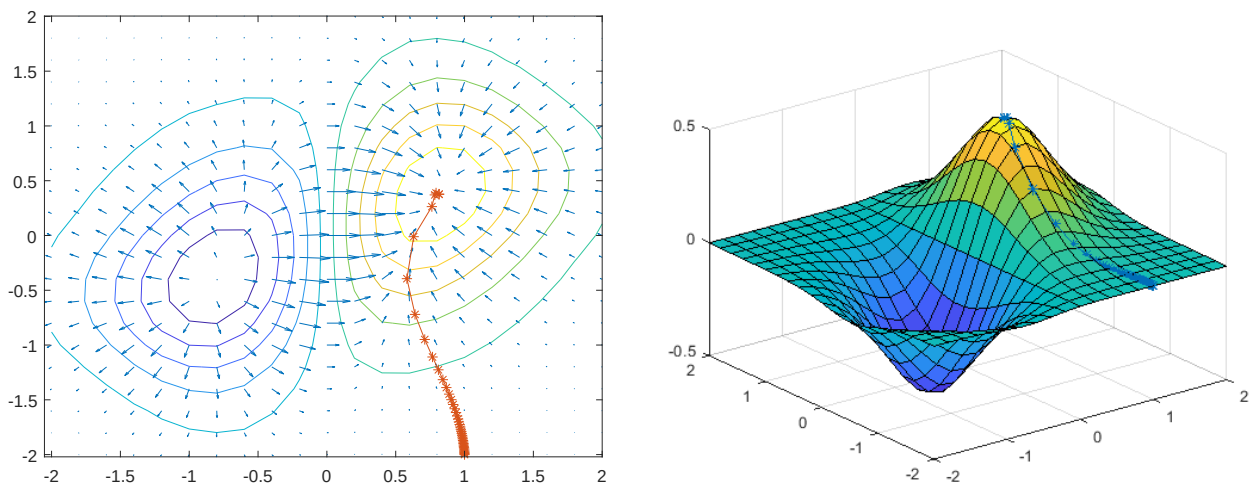

Figure 24.1: Gradient ascent on $f(x_1, x_2) = x_1 e^{-x_1^2 - x_2^2 + \sin(x_1 x_2)}$.

Optimization in multiple variables

If a differentiable function of multiple variables, f, has a maximum or minimum at point $\mathbf{x}^*$, then we will have:

$$\nabla f(\mathbf{x}^*) = \mathbf{0} \quad \text{the zero vector.}$$

Thus one way to locate max or min points for a specific function is to solve $\nabla f(\mathbf{x}^*) = \mathbf{0}$. This will often lead to systems of non-linear equations as we encountered in Lecture 13. For example, for $f(x_1, x_2) = x_1 e^{-x_1^2 - x_2^2 + \sin(x_1 x_2)}$, to attempt to find critical point we would need to try to solve:

$$(1 + x_1(-2x_1 + \cos(x_1 x_2) x_2)) e^{-x_1^2 - x_2^2 + \sin(x_1 x_2)} = 0 \quad \text{and} \quad x_1(-2x_2 + \cos(x_1 x_2) x_1) e^{-x_1^2 - x_2^2 + \sin(x_1 x_2)} = 0.$$

That, obviously, is going to be very hard and in fact is impossible using algebra. Thus we will have to use numerical methods to find any maximum or minimum points. The multiple variable Newton's method that we learned in Lecture 13 would be one option. In the next section we introduce another common algorithm.

The gradient ascent or descent method

We will take advantage of the fact that the gradient vector points in the direction of greatest increase in the function f. If f is a function of x and y, then we can think of the graph of $z = f(x, y)$ as a landscape and the gradient points "uphill". That gives us a very handy way to find maximum points, just follow the gradient vectors as they will take you toward higher values of z. See Figure 24.1. Since the gradient vector field may change from point to point, we can only "follow" it numerically in finite steps. That is, starting from an initial guess $\mathbf{x}_0$, we generate a sequence of points by the formula

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \epsilon \nabla f(\mathbf{x}_n).$$

The hope then is that

$$\mathbf{x}_n \to \mathbf{x}^*.$$

The number ϵ is often called the "learning rate". It should not be too small or too large. If it is too small the algorithm will be slow. If it is too large, the steps might skip over a critical point. There is ongoing research about how to choose it optimally.

Notice the similarities between this method and the Newton's method for nonlinear systems in Lecture 13. In both cases we have a function of a vector and generate a sequence of vectors $\mathbf{x}_0, \mathbf{x}_1, \ldots$.

Exercises

24.1 Find by hand the gradient of the function $f(x_1, x_2) = \sin(x_1)\ e^{-x_1^2 - x_2^2}$. Plot it on the rectangle $-3 \le x_1 \le 3$, $-2 \le x_2 \le 2$ using `meshgrid` and `quiver`. On the same figure, add the contour plot of f.

24.2 Write a well-commented **script** program that uses the gradient descent method to find a minimum point for the function $f(x_1, x_2) = \sin(x_1)\ e^{-x_1^2 - x_2^2}$, starting from $\mathbf{x}_0 = [-1; -2]$. Compare with the plot in exercise 24.1 to check that it worked. (Hint: Review the program `mymultnewton` in Lecture 13 for examples of making functions of vectors.)