**The lecture**

**Fundamentals of numerical optimization**

**Calculus-defined optimality**

In this section we briefly review how calculus is used to describe the local geometry of a function, as well as its minima or lowest points. As we will see later in the chapter, powerful numerical algorithms can be built using these simple concepts.

**Taylor series approximations**

To glean some basic insight regarding the geometry of a many times differentiable function *g (w)* near a point *v* we may form a *linear* approximation to the function near this point. This is just a tangent line passing through the point *(v*, *g (v))*, as illustrated in Fig. 2.1, which contains the first derivative information *g*’*(v)*. Such a linear approximation (also known as a first order Taylor series) is written as 

Note that indeed this function is **a)** linear in *w*, **b)** tangent to *g*(*w*) at *v* since *h (v)* = *g (v)* and because it contains the first derivative information of *g* at *v* i.e., *h*’*(v)* = *g*’*(v)*. Thislinear approximation holds particularly well near *v* because the derivative contains slopeinformation.

To understand even more about *g* near *v* we may form a quadratic approximation (also illustrated in Fig. 2.1) that contains both first and second derivative information *g*’*(v)* and *g*’’*(v)*. This quadratic, referred to as the second order Taylor series approximation, is written as





Linear (in green) and quadratic (in blue) approximations to a differentiable function *g (w)* at two

points: *w* = *u* and *w* = *v*. Often these linear and quadratic approximations are equivalently

referred to as first and second order Taylor series approximations, respectively.

This quadratic contains the same tangency and first order information of the linear approximation (i.e., *h (v)* = *g (v)* and *h*’*(v)* = *g*’*(v)*) with additional second order derivative information as well at *g* near *v* since *h*’’*(v)* = *g*’’*(v)*. The second order Taylor series approximation more closely resembles the underlying function around *v* because the second derivative contains so-called curvature information. We may likewise define linear and quadratic approximations for a many time differentiable function *g (***w***)* of vector valued input $w=[w\_{1},w\_{2},w\_{3},…,w\_{n}]$. In general, we may formally write the linear approximation as  where 

We may also generally write the quadratic approximation as  where is the *N* × *N* symmetric Hessian matrix of second derivatives (which is just the second derivative  when *N* = 1) defined as



**The first order condition for optimality**

Minimum values of a function *g* are naturally located at “valley floors” where the line or hyperplane tangent to the function has zero slope. Because the derivative/gradient contains this slope information, calculus thereby provides a convenient way of finding minimum values of *g*. In *N* = 1 dimension any point *v* where *g*’*(v)* = 0 is a potential minimum. Analogously with general *N*-dimensional input any point **v** where ∇*g (***v***)* = **0***N*×1 is a potential minimum as well. Note that the condition ∇*g (***v***)* = **0***N*×1 can be equivalently written as a system of *N* equations:





However, for a general function *g* minima are not the only points that satisfy this condition.

As illustrated in Fig. 2.2, a function’s maxima as well as saddle points (i.e., points at which the curvature of the function changes from negative to positive or vice-versa) are also points at which the function has a vanishing gradient. Together minima, maxima, and saddle points are referred to as *stationary points* of a function.

In short, while calculus provides us with a useful method for determining minima of a general function *g*, this method unfortunately determines other undesirable points (maxima and saddle points) as well.1 Regardless, as we will see later in this chapter the condition ∇*g (***w***)* = **0***N*×1 is a hugely important tool for determining minima, generally referred to as the *first order condition for optimality*, or in short the first order condition.

**The convenience of convexity**

Solving a machine learning problem eventually reduces to finding the minimum of an associated cost function. Of all (potentially many) minima of a cost function, we are especially interested in the one that provides the lowest possible value of the function, known as the *global minimum*. For a special family of functions, referred to as convex functions, the first order condition is particularly useful because *all stationary points of a convex function are global minima*. In other words, convex

functions are free of maxima and saddle points as well as non-global minima. To determine if a function *g* is *convex* (facing upward) or *concave* (facing downward) at a point *v* we check its curvature or second derivative information there



Here, if a statement on one side of the symbol ⇐⇒ (which reads “if and only if”) is true then the statement on the other side is true as well (likewise if one is false then the other is false as well). Similarly, for general *N* an analogous statement can be made regarding the eigenvalues of ∇2*g (***v***)*, i.e., *g* is convex (or concave) at **v** if and only if the Hessian matrix evaluated at this point has all non-negative (or non-positive) *eigenvalues*, in which case the Hessian is called positive semi-definite (or negative semi-definite). Based on this rule, *g (w)* is convex everywhere, a convex function, if its second derivative *g*’’*(w)* is always non-negative. Likewise, *g (***w***)* is convex if ∇2*g (***w***)* always has non-negative eigenvalues.