TOOLS FOR LEARNING: FUNDAMENTALS OF NUMERICAL OPTIMIZATION – PART 1

Michele Rossi

rossi@dei.unipd.it

Dept. of Information Engineering

University of Padova, IT

Outline

- Basic tools
  - From the *numerical optimization* field
- Used by nearly all learning models
  - To determine optimal parameters
  - Method: *minimization* of a differentiable function
  - Used to train nearly all supervised neural architectures
- Methods
  - Gradient descent
  - Newton method

Taylor expansions (1/3)

- We are given a many times differentiable function $g(w)$
- Consider a point v
- We form a linear approximation of $g(w)$ near this point
  - It is a tangent line passing through the point $(v, g(v))$
  - It is given by the *Taylor series expansion* of $g(w)$
  - Written as:

$$h(w) = g(v) + g'(v)(w - v)$$

  - This linear approximation is:
    - 1) linear in w
    - 2) tangent to $g(w)$ at v since:

$$\left.\frac{\partial h(w)}{\partial w}\right|_{w=v} = h'(v) = g'(v)$$

Taylor expansions (2/3)

- We can go a step further and build a quadratic approximation
  - Contains both *first* and *second* order information
  - It is provided by the second-order Taylor series expansion:

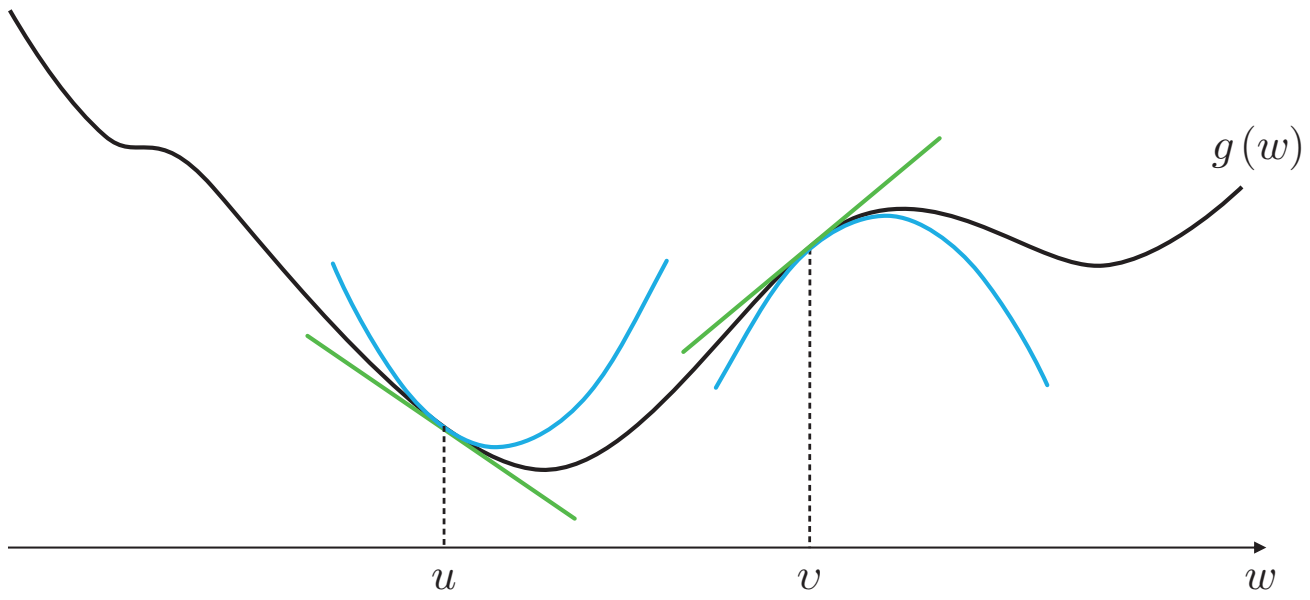
$$h(w) = g(v) + g'(v)(w - v) + \frac{1}{2}g''(v)(w - v)^2$$

  - This quadratic has:

$$h(v) = g(v)$$
$$h'(v) = g'(v)$$
$$h''(v) = g''(v)$$

  - It more closely resembles the underlying function around v
  - The second derivative contains the so called *curvature information*

Taylor expansions (3/3)



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.

Vector form – linear approx (1/2)

- Often we deal with (*column*) vectors (of *weights*)

$$\boldsymbol{w} \in \mathbb{R}^N \, , \boldsymbol{w} = [w_1, w_2, \ldots, w_N]^T$$

- Linear approx. becomes

$$h(\boldsymbol{w}) = g(\boldsymbol{v}) + \nabla g(\boldsymbol{v})^T (\boldsymbol{w} - \boldsymbol{v})$$

  - where:

$$\nabla g(\boldsymbol{v}) = \left[\frac{\partial}{\partial w_1} g(\boldsymbol{v}) \quad \frac{\partial}{\partial w_2} g(\boldsymbol{v}) \quad \ldots \quad \frac{\partial}{\partial w_N} g(\boldsymbol{v}) \right]^T$$

  - is the Nx1 gradient of partial derivatives

Vector form – quadratic approx (2/2)

- We can also write a quadratic approximation as:

$$h(\boldsymbol{w}) = g(\boldsymbol{v}) + \nabla g(\boldsymbol{v})^T(\boldsymbol{w} - \boldsymbol{v}) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{v})^T \nabla^2 g(\boldsymbol{v})(\boldsymbol{w} - \boldsymbol{v})$$

- where:

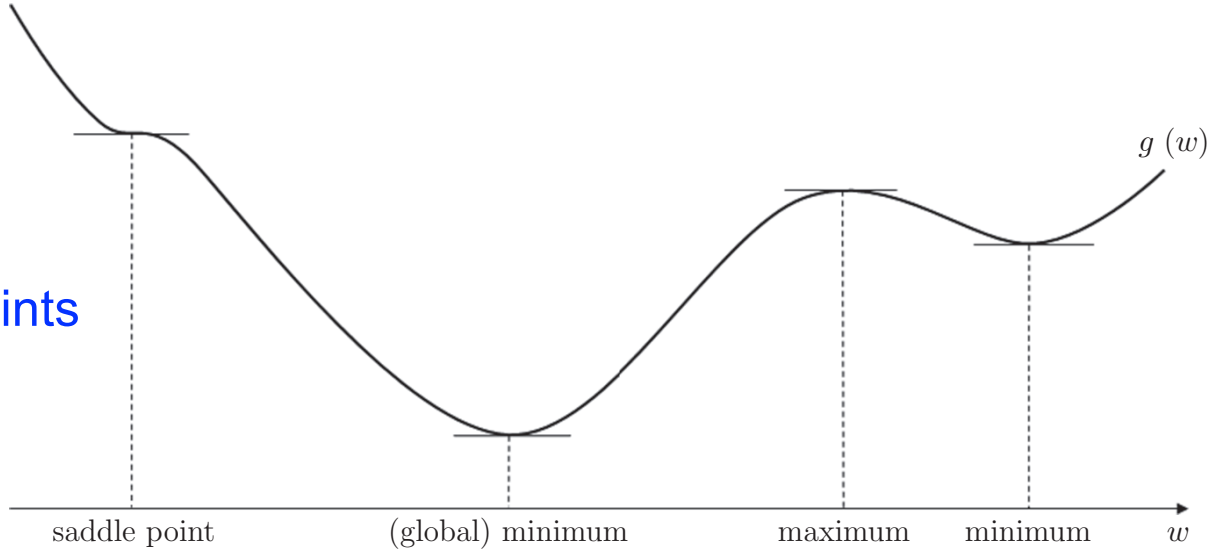
$$\nabla^2 g(\boldsymbol{v}) = \begin{bmatrix} \frac{\partial^2}{\partial w_1 \partial w_1} g(\boldsymbol{v}) & \frac{\partial^2}{\partial w_1 \partial w_2} g(\boldsymbol{v}) & \cdots & \frac{\partial^2}{\partial w_1 \partial w_N} g(\boldsymbol{v}) \\ \frac{\partial^2}{\partial w_2 \partial w_1} g(\boldsymbol{v}) & \frac{\partial^2}{\partial w_2 \partial w_2} g(\boldsymbol{v}) & \cdots & \frac{\partial^2}{\partial w_2 \partial w_N} g(\boldsymbol{v}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial w_N \partial w_1} g(\boldsymbol{v}) & \frac{\partial^2}{\partial w_N \partial w_2} g(\boldsymbol{v}) & \cdots & \frac{\partial^2}{\partial w_N \partial w_N} g(\boldsymbol{v}) \end{bmatrix}$$

- is the NxN symmetric Hessian matrix of second derivatives

First order conditions 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
- The *gradient* of g provides this *slope* information
- In N dimensions $\boldsymbol{v}$ is a potential minimum if $\nabla g(\boldsymbol{v}) = \mathbf{0}_{N\times 1}$

stationary points



Stationary points of the general function g include minima, maxima, and saddle points. At all such points the gradient is zero.

The convenience of convexity

- Solving a machine learning problem
  - Eventually reduces to finding the minimum of a cost function
  - Of all possible minima of a function, we are interested in the smallest → global minimum

- Convex functions
  - All their stationary points are *global* minima
  - Are free of maxima, saddle points and local minima
  - To determine whether a function is convex at a point
    - We check its second derivative (curvature) information

$$g''(v) \geq 0 \leftrightarrow g \text{ is convex at } v$$

$$g''(v) \leq 0 \leftrightarrow g \text{ is concave at } v$$

Convexity for vector functions

- For general vectors (N points)
  - Convexity is checked looking at the Hessian matrix

- Function g is convex at **v** iff
  - The Hessian evaluated at **v** has all non-negative eigenvalues
  - Notation:

$$\nabla^2 g(\boldsymbol{v}) \succeq 0$$

- Function g is concave at **v** iff
  - The Hessian evaluated at **v** has all non-positive eigenvalues
  - Notation:

$$\nabla^2 g(\boldsymbol{v}) \preceq 0$$

Example: quadratic function

- A *quadratic function* in **w** takes the form

$$g(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{Q}\boldsymbol{w} + \boldsymbol{r}^T\boldsymbol{w} + d$$

- Note that:

$$\frac{\partial}{\partial \boldsymbol{w}}(\boldsymbol{r}^T\boldsymbol{w}) = \boldsymbol{r}^T \quad \text{(constant)}$$

$$\nabla^2 g(\boldsymbol{w}) = \nabla^2\left(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{Q}\boldsymbol{w}\right) = \frac{1}{2}(\boldsymbol{Q} + \boldsymbol{Q}^T)$$

(descends from Appendices 1 and 2)

Quadratic function examples (1/2)

- Note that if **Q** is symmetric: $\nabla^2 g(\boldsymbol{x}) = (\boldsymbol{Q} + \boldsymbol{Q}^T)/2 = \boldsymbol{Q}$

  - When $\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ the Hessian $\nabla^2 g(\mathbf{w}) = \mathbf{Q}$ has two eigenvalues equaling 2, so the corresponding quadratic, shown in the left panel of Fig. 2.4, is convex.
  - When $\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$ again the Hessian $\nabla^2 g(\mathbf{w}) = \mathbf{Q}$ has two eigenvalues (2 and 0), so the corresponding quadratic, shown in the middle panel of Fig. 2.4, is convex.
  - When $\mathbf{Q} = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$ again the Hessian is $\nabla^2 g(\mathbf{w}) = \mathbf{Q}$ and has eigenvalues 2 and -2, so the corresponding quadratic, shown in the right panel of Fig. 2.4, is non-convex.

- For the eigenvalues of diagonal matrix see Appendix 3
- Fig. 2.4 (from reference book) is shown in the next slide

Quadratic function examples (2/2)

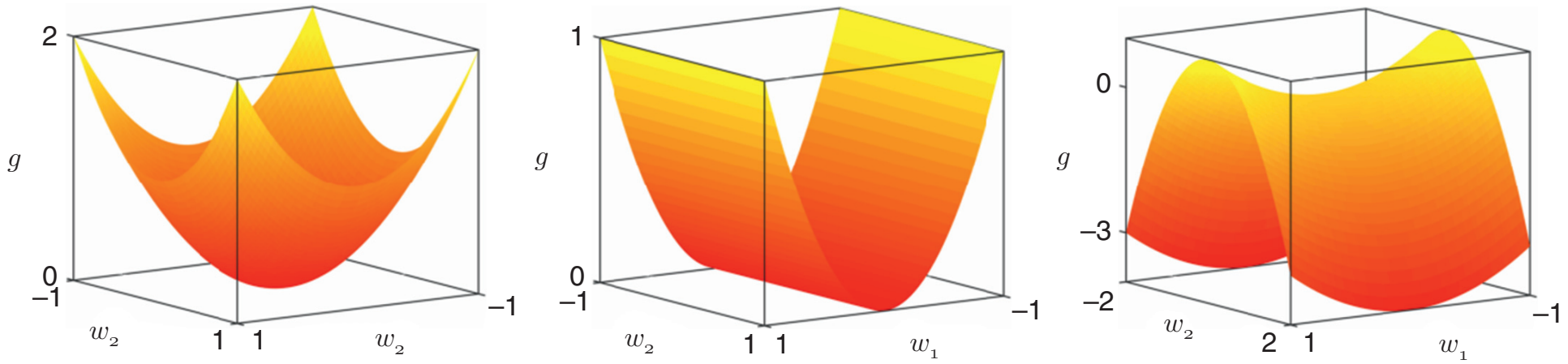


Fig. 2.4 Three quadratic functions of the form $g(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{r}^T\mathbf{w} + d$ generated by different instances of matrix $\mathbf{Q}$ in Example 2.3. In all three cases $\mathbf{r} = \mathbf{0}_{2\times 1}$ and $d = 0$. As can be visually verified, only the first two functions are convex. The last "saddle-looking" function on the right has a saddle point at zero!

Numerical methods for optimization

- We now introduce two
  - Basic, widely-used, *hugely important* techniques
  - 1) Gradient descent
  - 2) Newton's method

- Objective:

$$\underset{\boldsymbol{w}}{\text{minimize}}\ g(\boldsymbol{w})$$

- Both methods operate sequentially, by finding points at which g(w) gets smaller and smaller
- They are both guaranteed to find stationary points of g(w)

The general algorithm

- All the numerical optimization algorithms for the minimization of a generic function work as follows:

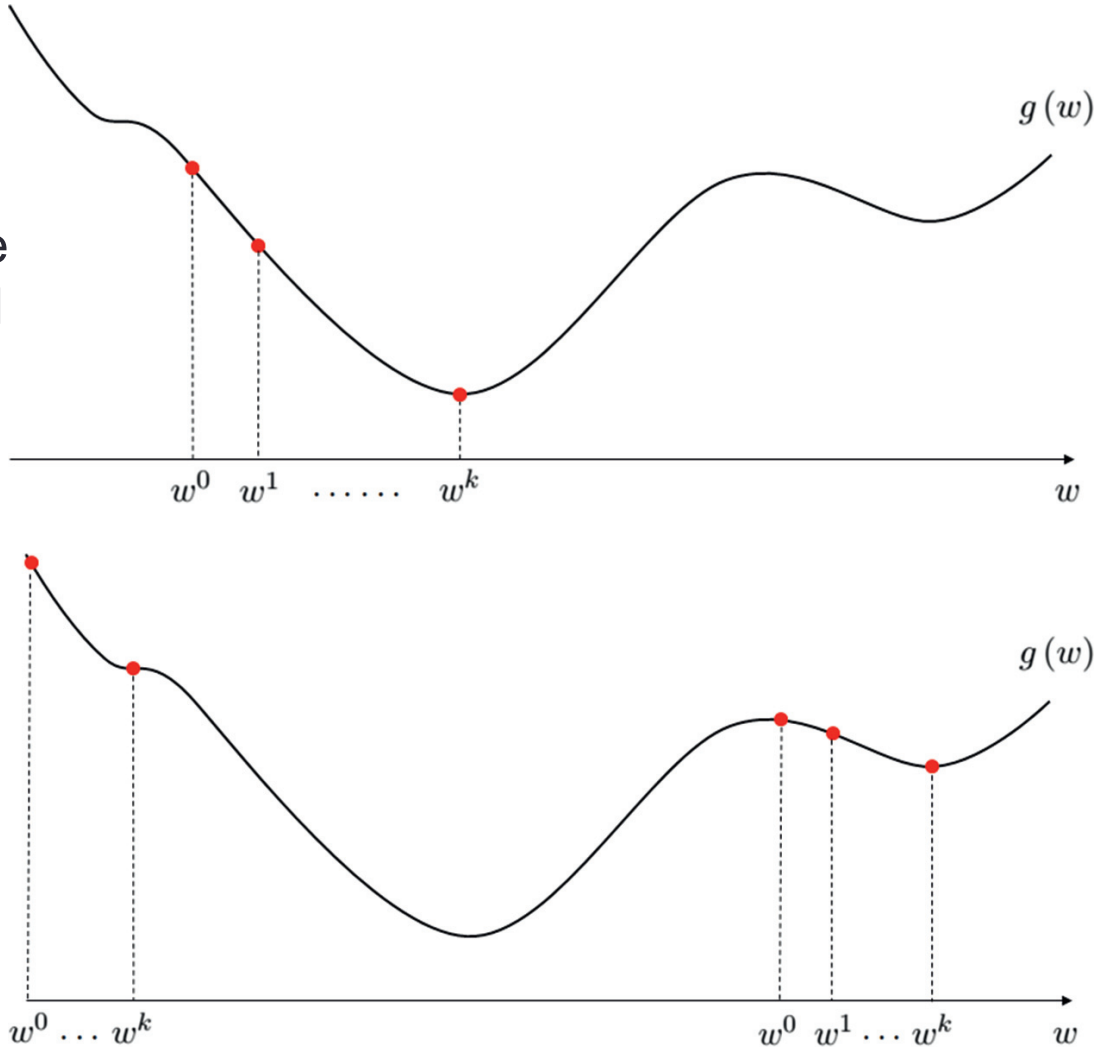
(1) Start the minimization process from some *initial point* $\mathbf{w}^0$.

(2) Take *iterative* steps denoted by $\mathbf{w}^1, \mathbf{w}^2, \ldots$, going "downhill" towards a stationary point of g.

(3) Repeat step (2) until the sequence of points converges to a stationary point of g.

Graphically

- For convex functions the minimum is always reached
- Regardless of initial point
- If function is non-convex
  - It depends on w^0
  - See also later…



The stationary point of a non-convex function found via numerical optimization is dependent on the choice of initial point w^0. In the top panel our initialization leads us to find the global minimum, while in the bottom panel the two different initializations lead to a saddle point on the left, and a non-global minimum on the right.

Stopping conditions

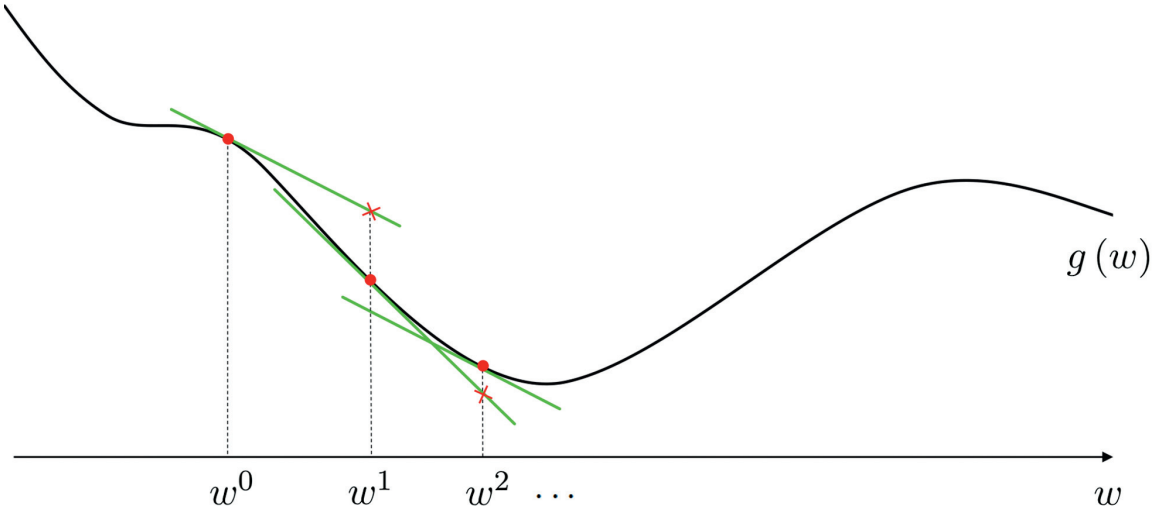
- The two most common stopping criteria are:

(1) When a pre-specified number of iterations are complete.

(2) When the gradient is small enough, i.e., $\left\|\nabla g\left(\mathbf{w}^k\right)\right\|_2 < \epsilon$ for some small $\epsilon > 0$.

Gradient descent (1/4)

- What differentiates different numerical optimization methods
  - Is the way iterative steps are taken to reduce the value of g(w)
- Gradient and Newton's methods
  - Both use (build) a local model of the function at each step
- Gradient descent – basic idea
  - **1)** build a local linear model for g(w) (acting as a *"surrogate"*, or *"proxy"* of the function itself), **2)** determine the "downward" direction on this hyperplane, **3)** travel a short distance along this direction



Gradient descent (2/4)

- Formally, we start at an initial point $\boldsymbol{w}^0$
- The linear model of g(w) at this point is precisely given by the first order Taylor series approximation:

$$h(\boldsymbol{w}) = g(\boldsymbol{w}^0) + \nabla g(\boldsymbol{w}^0)^T(\boldsymbol{w} - \boldsymbol{w}^0)$$

- We take our first step in the direction towards which
  - The tangent hyperplane most sharply angles downwards
  - This is the steepest descent direction

Intermission: steepest descent (1/2)

$$h(\boldsymbol{w}) = g(\boldsymbol{w}^0) + \nabla g(\boldsymbol{w}^0)^T(\boldsymbol{w} - \boldsymbol{w}^0)$$

- Taking a unit length direction **d**, this equation is rewritten as

$$h(\boldsymbol{d}) = g(\boldsymbol{w}^0) - \nabla g(\boldsymbol{w}^0)^T\boldsymbol{w}^0 + \nabla g(\boldsymbol{w}^0)^T\boldsymbol{d}$$

- The first two terms on the RHS are *constant* wrt **d**
- So minimizing h(**d**) corresponds to minimizing the last term
- But this last term is an inner product, i.e.,

$$\nabla g(\boldsymbol{w}^0)^T\boldsymbol{d} \stackrel{\text{def}}{=} < \nabla g(\boldsymbol{w}^0), \boldsymbol{d} > = \|\nabla g(\boldsymbol{w}^0)\|_2 \|\boldsymbol{d}\|_2 \cos(\theta)$$

- Where θ is the angle between the two vectors

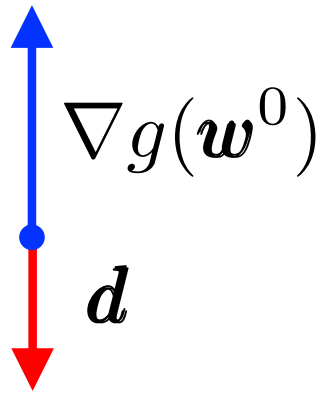
Intermission: steepest descent (2/2)

$$\nabla g(\boldsymbol{w}^0)^T \boldsymbol{d} \overset{\text{def}}{=} < \nabla g(\boldsymbol{w}^0), \boldsymbol{d} >= \|\nabla g(\boldsymbol{w}^0)\|_2 \|\boldsymbol{d}\|_2 \cos(\theta)$$

- This inner product is minimized if we pick direction:

$$\boldsymbol{d} = -\frac{\nabla g(\boldsymbol{w}^0)}{\|\nabla g(\boldsymbol{w}^0)\|_2}$$

- In fact,



$$< \nabla g(\boldsymbol{w}^0), -\nabla g(\boldsymbol{w}^0)/\|\nabla g(\boldsymbol{w}^0)\|_2 >= \|\nabla g(\boldsymbol{w}^0)\|_2 \times 1 \times -1$$

- The cosine between the two vectors is at a minimum (-1) !!!

$$\cos(\theta) \in [-1, 1]$$

Gradient descent (3/4)

- So, we have found that the steepest descent direction
- It is given precisely as $$\boldsymbol{d} = -\frac{\nabla g(\boldsymbol{w}^0)}{\|\nabla g(\boldsymbol{w}^0)\|_2}$$
- Now, this **d** provides a *unit step length*, in the *best direction*
- We may want to adjust it and travel *a bit more* or *a bit less*
- For this reason, we often take: $$\boldsymbol{d} = -\alpha \nabla g(\boldsymbol{w}^0)$$
- Where α is a positive constant, called the step length
- Armed with this knowledge, the next point is simply computed as:

$$\boldsymbol{w}^1 = \boldsymbol{w}^0 - \alpha \nabla g(\boldsymbol{w}^0)$$

Gradient descent (4/4)

- The gradient descent algorithm is then summarized as:

Input: differentiable function g, fixed step length α, and initial point $\mathbf{w}^0$

$k = 1$

Repeat until stopping condition is met:

$\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha \nabla g\left(\mathbf{w}^{k-1}\right)$

$k \leftarrow k + 1$

Example (1/2)

- Take $g(\boldsymbol{w}) = -\cos(2\pi \boldsymbol{w}^T \boldsymbol{w}) + 2\boldsymbol{w}^T \boldsymbol{w}$

  (non-convex function)

- With

$$\boldsymbol{w} = [w_1 \ w_2]^T$$

$$\nabla g(\boldsymbol{w}) = 4\pi \sin(2\pi \boldsymbol{w}^T \boldsymbol{w})\boldsymbol{w} + 4\boldsymbol{w}$$

(see appendix 4)

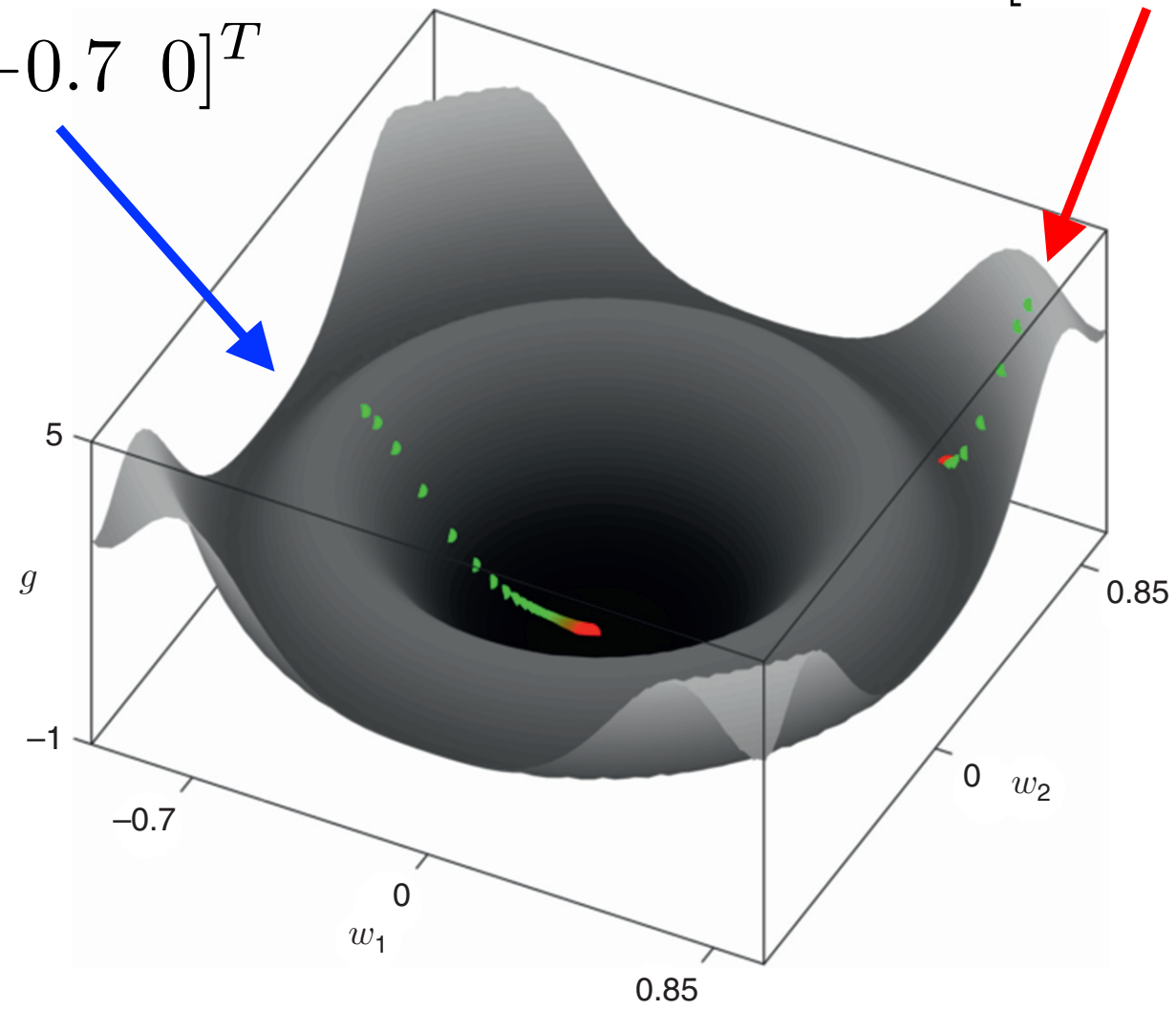
$$\alpha = 10^{-3}$$

Example (2/2)

- We take two initial values for $\mathbf{w}^0$

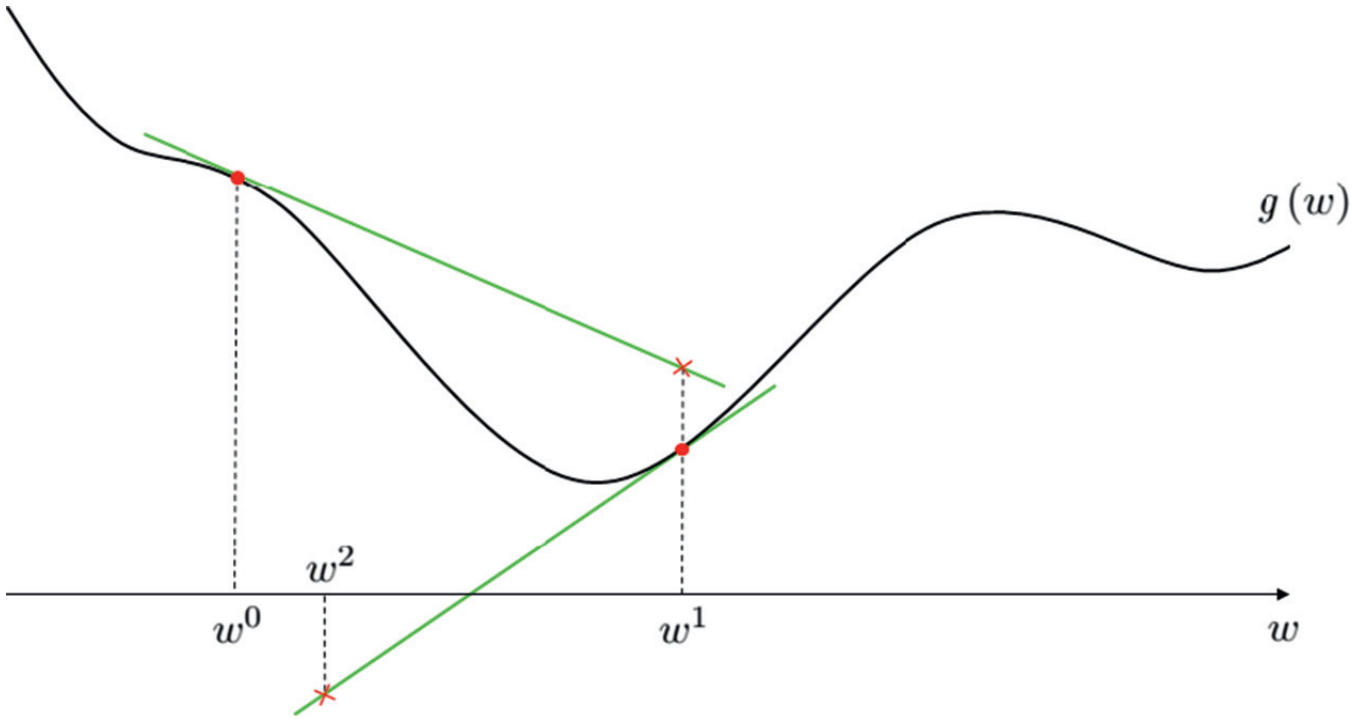
$$\boldsymbol{w}^0 = [0.85 \;\; 0.85]^T$$

$$\boldsymbol{w}^0 = [-0.7 \;\; 0]^T$$



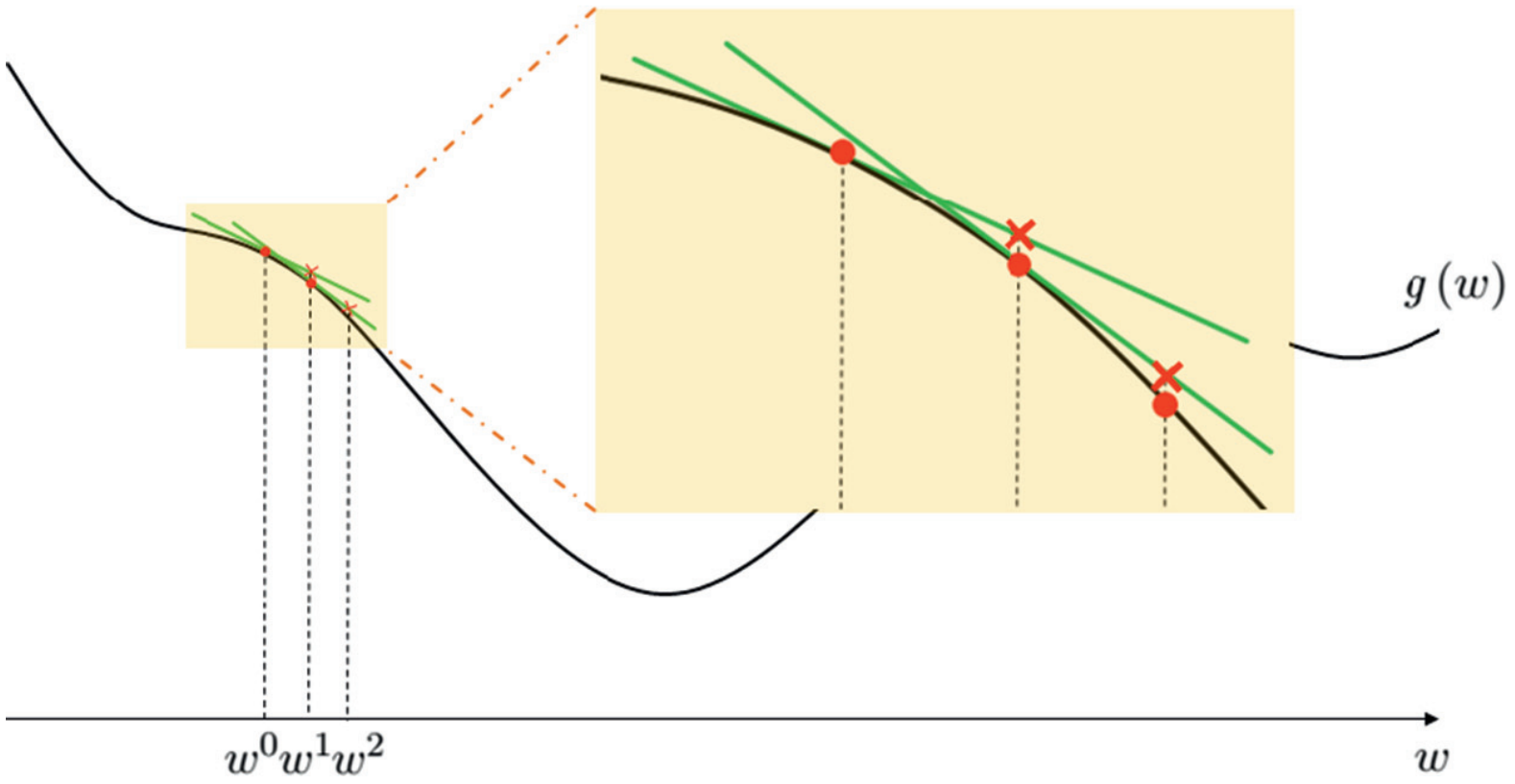
Problems ahead (1/2)

- We need to carefully select the step length (trial and error?)
- Too large a step size → we may overshoot the solution



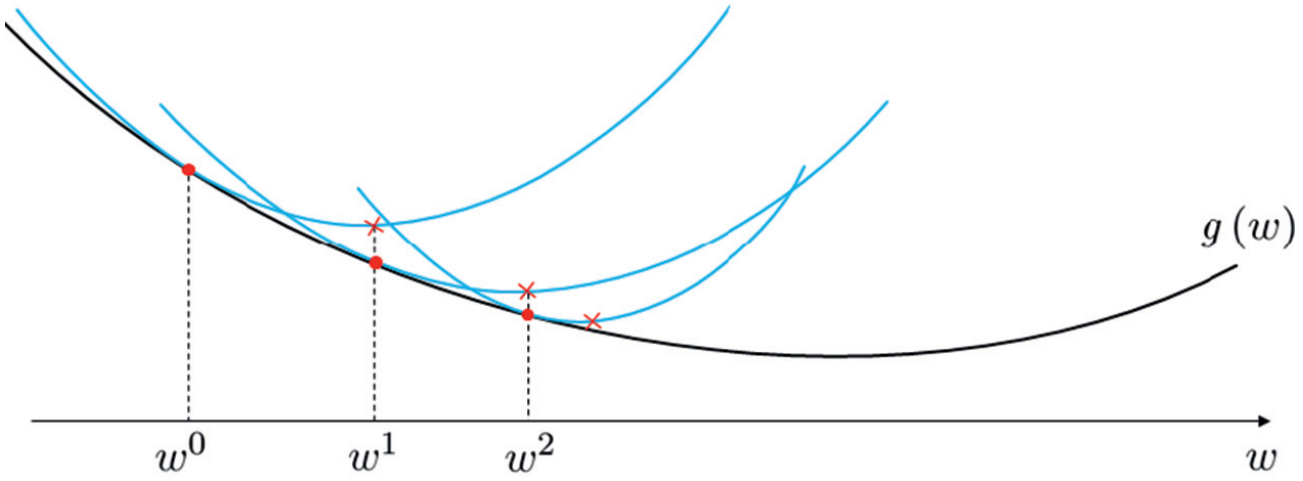
Problems ahead (2/2)

- We need to carefully select the step length
- Too small a step size → too many iterations to converge



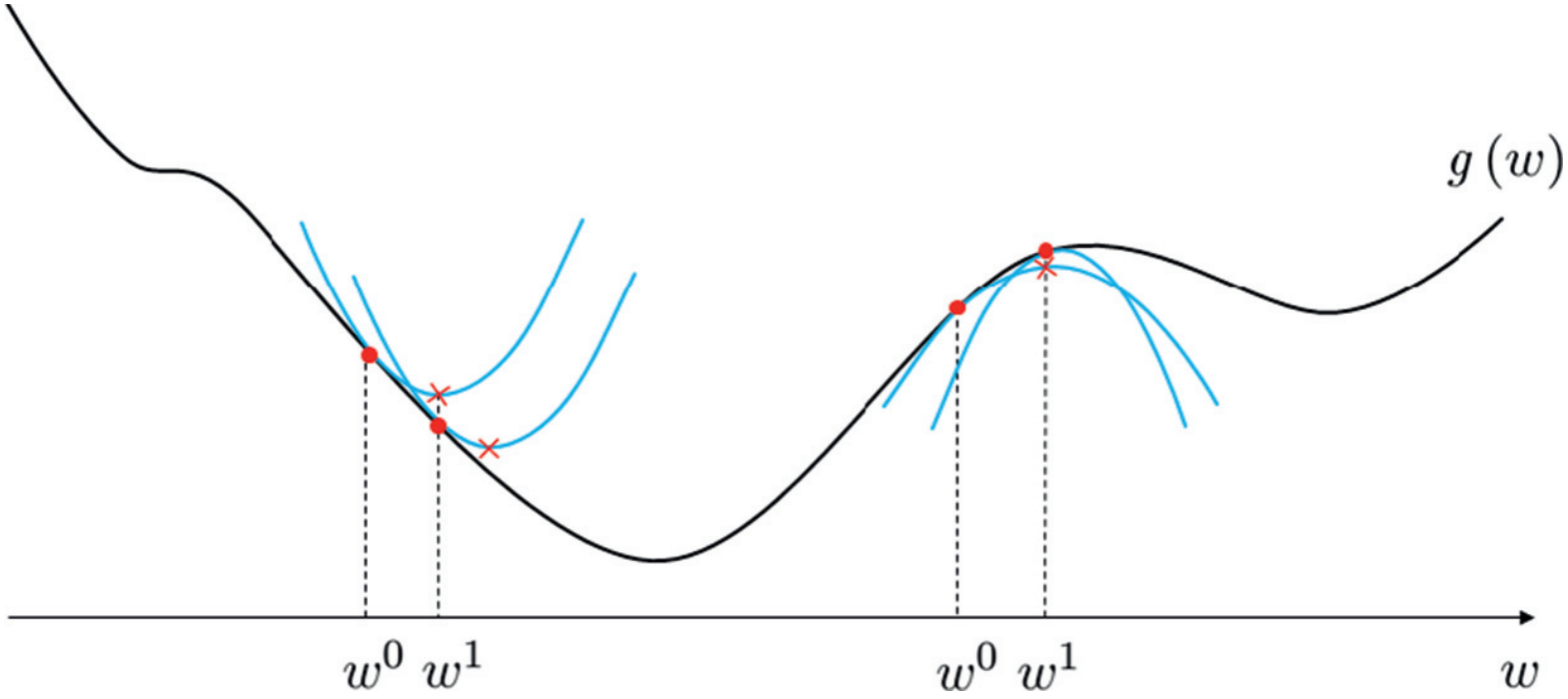
Newton's method

- Like gradient descent it also uses a local approximation of a function at each step to lower its value
- However, a quadratic approximation (generated through Taylor series expansion) is used
- Rationale: **1)** repeatedly creating a quadratic approx. to the function, **2)** traveling to a stationary point of this quadratic and **3)** hopping back onto the function



Newton's method

- It is often *much more effective* than gradient descent
  - As the quadratic approx. more closely mimics the function
- But for non-convex functions
  - At concave portions it may climb to maximum (see example below)



Newton's method – the math (1/3)

- Given an initial point $\mathbf{w}^0$
- The quadratic model is given precisely by the second-order Taylor expansion centered at $\mathbf{w}^0$:

$$h(\boldsymbol{w}) = g(\boldsymbol{w}^0) + \nabla g(\boldsymbol{w}^0)^T(\boldsymbol{w} - \boldsymbol{w}^0) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^0)^T \nabla^2 g(\boldsymbol{w}^0)(\boldsymbol{w} - \boldsymbol{w}^0)$$

- In the case where g is convex
  - We want to travel to a stationary point of h(**w**)
  - Such stationary point is a global minimum of h(**w**)
  - We need to compute the gradient of h(**w**) and set it to zero

$$\nabla h(\boldsymbol{w}) = \nabla g(\boldsymbol{w}^0) + \nabla^2 g(\boldsymbol{w}^0)(\boldsymbol{w} - \boldsymbol{w}^0) = \mathbf{0}_{N \times 1}$$

Newton's method – the math (2/3)

$$\nabla h(\boldsymbol{w}) = \nabla g(\boldsymbol{w}^0) + \nabla^2 g(\boldsymbol{w}^0)(\boldsymbol{w} - \boldsymbol{w}^0) = \mathbf{0}_{N\times 1}$$

- Solving for the Hessian, leads to a system of NxN equations

$$\nabla^2 g(\boldsymbol{w}^0)\boldsymbol{w} = \nabla^2 g(\boldsymbol{w}^0)\boldsymbol{w}^0 - \nabla g(\boldsymbol{w}^0)$$

- A solution to this system of equations gives the first point $\mathbf{w}^1$ that is traveled by Newton's method
- We then repeat the same step by starting at point $\mathbf{w}^1$
- This leads to the same system of equation, this time centered at $\mathbf{w}^1$, solving it leads to $\mathbf{w}^2$, …

$$\nabla^2 g(\boldsymbol{w}^1)\boldsymbol{w} = \nabla^2 g(\boldsymbol{w}^1)\boldsymbol{w}^1 - \nabla g(\boldsymbol{w}^1)$$

Newton's method – the math (3/3)

- We can summarize Newton's method as:

Input: twice differentiable function g, and initial point $\mathbf{w}^0$

$k = 1$

Repeat until stopping condition is met:

Solve the system $\nabla^2 g\left(\mathbf{w}^{k-1}\right)\mathbf{w}^k = \nabla^2 g\left(\mathbf{w}^{k-1}\right)\mathbf{w}^{k-1} - \nabla g\left(\mathbf{w}^{k-1}\right)$ for $\mathbf{w}^k$.

$k \leftarrow k + 1$

- At step k we solve a system of N equations
- If the Hessian matrix is invertible, the solution of the system of equations can be promptly obtained as:

$$\boldsymbol{w}^k = \boldsymbol{w}^{k-1} - \left[\nabla^2 g(\boldsymbol{w}^{k-1})\right]^{-1} \nabla g(\boldsymbol{w}^{k-1})$$

Newton's method – wrap up

- Newton method is *much faster* than gradient descent
- Works well for convex functions
  - At the cost of extra-computation
- The following solutions are equivalent
  - 1) Finding the inverse of the Hessian

$$\boldsymbol{w}^k = \boldsymbol{w}^{k-1} - \left[\nabla^2 g(\boldsymbol{w}^{k-1})\right]^{-1} \nabla g(\boldsymbol{w}^{k-1})$$

  - 2) Solving this linear system of N equations

$$\nabla^2 g(\boldsymbol{w}^{k-1})\boldsymbol{w}^k = \nabla^2 g(\boldsymbol{w}^{k-1})\boldsymbol{w}^{k-1} - \nabla g(\boldsymbol{w}^{k-1})$$

  - 2) is usually much faster using state-of-the-art linear algebra numerical solvers. Up to several thousands of input variables can be dealt with

Newton's method – example (1/2)

- Quadratic function

$$g(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{Q}\boldsymbol{w} + \boldsymbol{r}^T\boldsymbol{w} + d$$

- With:

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix}$$

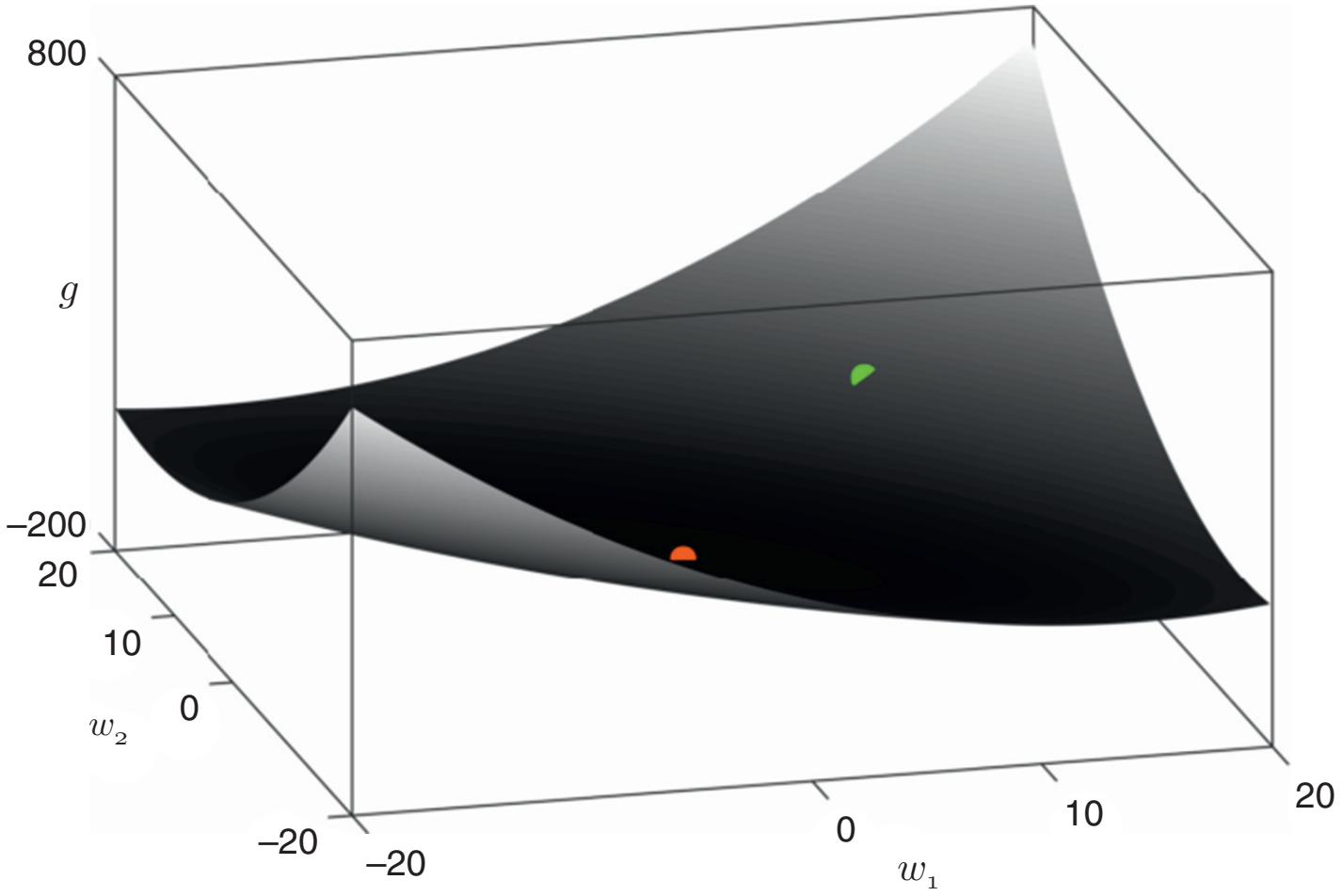
$$\boldsymbol{r} = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$$

$$d = 0$$

$$\boldsymbol{w}^0 = \begin{bmatrix} 10 & 10 \end{bmatrix}^T$$

Newton's method – example (2/2)

- The minimum is reached in a single iteration (expected as the function is quadratic and Newton minimizes a quadratic "surrogate")



APPENDICES

Appendix 1 – gradient of a quadratic form (1/5)

- Let α be the quadratic form: $\alpha \stackrel{\Delta}{=} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}$
  - where: $\boldsymbol{x}$ is nx1, $\boldsymbol{A} = [a_{ij}]$ is nxn and does not depend on $\boldsymbol{x}$

- We define the gradient of α as the column vector:

$$\nabla\alpha(\boldsymbol{x}) = \left(\frac{\partial\alpha}{\partial x_1} \;\; \frac{\partial\alpha}{\partial x_2} \;\; \cdots \;\; \frac{\partial\alpha}{\partial x_n} \right)^T$$

- Then, we have that (column vector form):

$$\nabla\alpha(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{\partial\alpha}{\partial\boldsymbol{x}} = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{x}$$

Appendix 1 – gradient of a quadratic form (2/5)

Proof. by definition,

$$\alpha = \sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}x_i x_j$$

Differentiating with respect to the k-th element of **x** we get:

$$\frac{\partial \alpha}{\partial x_k} = \sum_{j=1}^{n} a_{kj}x_j + \sum_{i=1}^{n} a_{ik}x_i$$

for all k=1, 2,…, n. In compact form, this can be written as:

$$\nabla\alpha(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{\partial \alpha}{\partial \boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^T\boldsymbol{x} = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{x}$$

Appendix 1 – gradient of a quadratic form (3/5)

Let H be symmetric, e.g., a Hessian matrix

$$\alpha(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^0)^T \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^0)$$

Following the approach in the previous slide, we have:

$$2\alpha(\boldsymbol{w}) = \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(w_i - w_i^0)(w_j - w_j^0) = \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(w_i w_j) + \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(w_i^0 w_j^0)$$

$$- \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(w_i w_j^0) - \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(w_i^0 w_j) =$$

$$\stackrel{\text{def}}{=} \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$$

Appendix 1 – gradient of a quadratic form (4/5)

We have:

$$\frac{\partial \alpha_1}{\partial \boldsymbol{w}} = (\boldsymbol{H} + \boldsymbol{H}^T)\boldsymbol{w} \qquad \frac{\partial \alpha_2}{\partial \boldsymbol{w}} = \mathbf{0}$$

$$\frac{\partial \alpha_3}{\partial w_k} = -\sum_{j=1}^{n} h_{kj} w_j^0 \qquad \frac{\partial \alpha_4}{\partial w_k} = -\sum_{i=1}^{n} h_{ik} w_i^0$$

Compactly:

$$\frac{\partial \alpha_3}{\partial w_k} + \frac{\partial \alpha_4}{\partial w_k} = -\left(\sum_{j=1}^{n} h_{kj} w_j^0 + \sum_{i=1}^{n} h_{ik} w_i^0 \right)$$

In matrix form:

$$\frac{\partial \alpha_3}{\partial \boldsymbol{w}} + \frac{\partial \alpha_4}{\partial \boldsymbol{w}} = -(\boldsymbol{H} + \boldsymbol{H}^T)\boldsymbol{w}^0$$

Appendix 1 – gradient of a quadratic form (5/5)

- Putting it all together:

$$\frac{\partial \alpha}{\partial \boldsymbol{w}} = \frac{1}{2}\left(\frac{\partial \alpha_1}{\partial \boldsymbol{w}} + \frac{\partial \alpha_3}{\partial \boldsymbol{w}} + \frac{\partial \alpha_4}{\partial \boldsymbol{w}}\right) = \frac{1}{2}(\boldsymbol{H} + \boldsymbol{H}^T)(\boldsymbol{w} - \boldsymbol{w}^0)$$

- The gradient is finally obtained as:

$$\nabla(\alpha(\boldsymbol{w})) = \frac{\partial \alpha}{\partial \boldsymbol{w}} = \frac{1}{2}(\boldsymbol{H} + \boldsymbol{H}^T)(\boldsymbol{w} - \boldsymbol{w}^0) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^0)$$

since H is symmetric

Appendix 2 – Hessian of a quadratic form

- From Appendix 1, we have that:

$$\frac{\partial \alpha}{\partial x_k} = \sum_{j=1}^{n} a_{kj} x_j + \sum_{i=1}^{n} a_{ik} x_i$$

- Taking another derivative, leads to (*element-wise*):

$$\frac{\partial^2 \alpha}{\partial x_k \partial x_h} = a_{kh} + a_{hk}$$

- Putting this in matrix form, leads to:

$$\nabla^2 \alpha(\mathbf{x}) = \boldsymbol{A} + \boldsymbol{A}^T$$

q.e.d.

Appendix 3

- Eigenvectors and eigenvalues of a diagonal matrix

$$\boldsymbol{Dx} = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 x_1 \\ d_2 x_2 \\ \vdots \\ d_n x_n \end{bmatrix} = \boldsymbol{\lambda x}$$

- Is satisfied for the following eigenvectors / eigenvalues

$$\lambda_i = d_i \text{ and } \boldsymbol{x} = \boldsymbol{e}^i = [e_1\, e_2\, \dots\, e_n]^T \text{ with } e_i = 1 \text{ and } e_j = 0\, j \neq i$$

- Hence the eigenvalues of **D** are the elements of its diagonal and the eigenvectors are the canonical basis of $\mathbb{R}^n$

Appendix 4 – other useful things

- Square 2-norm: $$\alpha = \mathbf{x}^T\mathbf{x} = \sum_{i=1}^{n} x_i^2$$
- Which leads to: $$\frac{\partial \alpha}{\partial x_j} = 2x_j$$
- In vector form: $$\nabla\alpha(\mathbf{x}) = 2\mathbf{x}$$
- So we have (using *chain rule of derivatives*):

$$g(\mathbf{x}) = -\cos(2\pi\mathbf{x}^T\mathbf{x}) + 2\mathbf{x}^T\mathbf{x}$$

$$\nabla g(\mathbf{x}) = 4\pi\sin(2\pi\mathbf{x}^T\mathbf{x})\mathbf{x} + 4\mathbf{x}$$

References

Reference book:

[1] J. Watt, R. Borhani, A. K. Katsaggelos, “Machine Learning Refined: Foundations, Algorithms and Applications,” Cambridge University Press 2016.

Chapter 2 “Fundamentals of numerical optimization”

TOOLS FOR LEARNING: FUNDAMENTALS OF NUMERICAL OPTIMIZATION – PART 1

Michele Rossi

rossi@dei.unipd.it

Dept. of Information Engineering

University of Padova, IT