Basic optimization methods, part 1

Aapo Hyvärinen

for the NNDL course, 14 March 2025

- Problem statement: Optimization of $f(\mathbf{w})$ with $\mathbf{w} \in \mathbb{R}^n$
- Definition and meaning of gradient
- Basic gradient method
- Constrained optimization with $\mathbf{w} \in S$

Literature:

Prince's book sections 6.1 (only partly the same but with further illustrations)

Optimization problem in real space

- We are given a function $f(\mathbf{w})$ with $\mathbf{w} = (w_1, \ldots, w_n)$ in $\mathbb{R}^n$ to optimize
  - f is called the objective (function) or criterion in optimization; also loss function or cost function in ML
- Here, we consider finding maximum:

$$\max_{\mathbf{w}} f(\mathbf{w}) \tag{1}$$

- Most literature talks about minimum, which is equivalent

$$\max_{\mathbf{w}} f(\mathbf{w}) = -\min_{\mathbf{w}} -f(\mathbf{w}) \tag{2}$$

- In probabilistic inference, it may be more natural to maximize (likelihood, for example)

Gradient: definition

- The gradient of f is denoted by ∇f, or more precisely $\nabla_{\mathbf{w}} f$
- Defined as the vector of the partial derivatives:

$$\nabla f(\mathbf{w}) = \begin{pmatrix} \frac{\partial f(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial f(\mathbf{w})}{\partial w_n} \end{pmatrix} \tag{3}$$

- Partial derivatives are obtained by taking the ordinary derivative with respect to w_i so that all other variables are fixed

Meaning of gradient

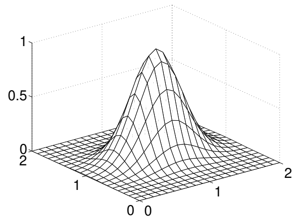
- Gradient points at the direction where the function *grows the fastest*.
- Suppose we want to find a vector $\boldsymbol{\epsilon}$ such that $f(\mathbf{w} + \boldsymbol{\epsilon})$ is as large as possible
  - while constraining $\|\boldsymbol{\epsilon}\|$ to be fixed and very small.
- Then the optimal $\boldsymbol{\epsilon}$ is given by the gradient multiplied by a small positive constant. "Steepest ascent"
- Likewise, the vector that reduces the value of f as much as possible is given by $-\nabla f(\mathbf{w})$, multiplied by a small constant. "Steepest descent"
- These properties come from the fundamental first-order approximation (Taylor series)

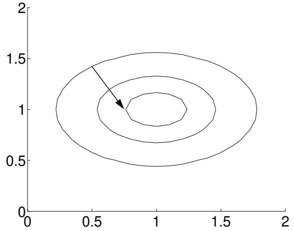
$$f(\mathbf{w} + \boldsymbol{\epsilon}) = f(\mathbf{w}) + (\nabla f(\mathbf{w}))^T \boldsymbol{\epsilon} + o(\boldsymbol{\epsilon}) \qquad (4)$$

Some further properties of gradient

- At the *maximizing points*, the gradient is zero
  - if you want to increase the function, "there's nowhere to go"
  - ... but gradient can be zero at other points as well (especially, the minima)
  - generalization of elementary calculus result: in 1D the maxima (and minima) of a function are obtained at those points where the derivative is zero
- The gradient is always *orthogonal* to the curves in a contour plot of the function pointing in the direction of *growing* f.

Illustration





Left: 3D plot
Right: Contour plots, gradient in one point given by arrow

- Consider the function

$$f(\mathbf{w}) = \exp(-5(w_1 - 1)^2 - 10(w_2 - 1)^2)$$

- The gradient is equal to

$$\nabla f(\mathbf{w}) = \begin{pmatrix} -10(w_1 - 1)\exp(-5(w_1 - 1)^2 - 10(w_1 - 1)^2) \\ -20(w_2 - 1)\exp(-5(w_1 - 1)^2 - 10(w_2 - 1)^2) \end{pmatrix}$$

- Taking a small step in the direction of the gradient, one gets closer to the maximizing point (1,1).
- However, too big a step will miss the maximizing point

Gradient method (for finding maximum)

- Start at some point $\mathbf{w} := \mathbf{w}_0$ (perhaps randomly generated)
- Repeatedly take small steps in the direction of $\nabla f(\mathbf{w})$
    - Recompute the gradient at the current point after each step.
- In other words:

$$\mathbf{w} \leftarrow \mathbf{w} + \mu \nabla f(\mathbf{w}) \tag{5}$$

  where the scalar parameter $\mu > 0$ is a small step size, typically much smaller than 1
- We have to take small steps because this leads to an increase in the value of f only locally
- This iteration is repeated over and over again until the algorithm converges to a point (or you run out of time)
- In case of minimization, the sign of the increment is negative:

$$\mathbf{w} \leftarrow \mathbf{w} - \mu \nabla f(\mathbf{w}) \tag{6}$$

Intuitive thinking in 1D

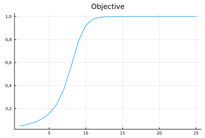
- To gain intuition, we can also consider 1D space
- Gradient is just the derivative

$$w \leftarrow w + \mu f'(w) \tag{7}$$

- At points where the function is increasing, derivative is positive, so the method will take a steps to the right
- At points where the function is decreasing, the derivative is negative, so it will tell us to move to the left.
- When the derivative is zero, you're done!

Convergence of the gradient method

- If the algorithm arrives at a point where $\nabla f(\mathbf{w}) = 0$, it will not move away from it.
  - This is good: at the maximizing points, the gradient is zero.
- However, there is absolutely no guarantee that such a point will be found, especially for any finite computational resources
- In practice, convergence is tested e.g. by looking at
  - change in $\mathbf{w}$ between two subsequent iterations
  - norm of gradient (almost equal to the above)
  - change in objective function
- If one of the above is small enough, assume the algorithm has converged.
- IMHO, should always plot the evolution of objective function: It *must* change monotonically, and reach a "plateau"; otherwise something went wrong



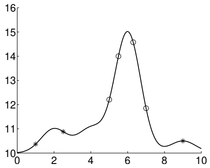
Choosing step size

- Choosing a good step size parameter μ is crucial
  - If μ is too large, the algorithm will not work at all
  - if μ is too small, the algorithm will be too slow
- No general method for choosing it *a priori* :(
- It is possible to adapt the step size during the iterations :)
  - At each step, consider the step size used in the previous iteration (say μ_0), and a larger one (say $2\mu_0$) and a smaller one ($\mu_0/2$).
  - Compute the value of the objective function that results from using any of these three step sizes in the current step
  - Choose the step size which gives the largest value for the objective function
- Ultimately, you can do a 1D optimization of the function

$$\max_{\mu} f(\mathbf{w} + \mu \nabla f(\mathbf{w})) \tag{8}$$

i.e. *line search*; but that is another story (very rare in DL)

Global and local maxima



Global max: $w = 6$

Further local max: $w = 2$ and $w = 9$

- Local maximum: a point in which the function obtains a value greater than in *all neighbouring points*.
- Global maximum: a point in which the function obtains a value great than *anywhere else*
- Gradient algorithm depends on the *initial point* $\mathbf{w}_0$
- The algorithm will find the local maximum "closest" to $\mathbf{w}_0$
- If started at a point marked by a circle $\rightarrow$ the global max
- If started at a point marked by a cross $\rightarrow$ a local maximum.
- Major problem in deep learning !!!
  "Getting stuck in a local minimum"
- Improvement: Run from many initial points, compare f in resulting final points, choose the one with the highest value

Optimization in matrix space

- Many functions we want to maximize are actually functions of a matrix (e.g. weight matrix in NN)
- No problem at all: matrices are treated just like vectors.
- In theory, vectorize $n \times m$ matrix to an nm-dim vector
- In practice, explicit vectorization not usually needed, can be even cumbersome
- Example:

$$f(\mathbf{W}) = \sum_i g(\mathbf{w}_i^T \mathbf{z}) \tag{9}$$

  where the $\mathbf{w}_i$ are the rows of matrix $\mathbf{W}$; $\mathbf{z}$ is some vector
- We can calculate (optional exercise)

$$\nabla f(\mathbf{W}) = g'(\mathbf{W}\mathbf{z})\mathbf{z}^T \tag{10}$$

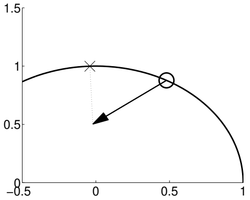
  (short-cut notation: g' is applied on each entry separately)
- Gradient method for maximizing f:

$$\mathbf{W} \leftarrow \mathbf{W} + \mu g'(\mathbf{W}\mathbf{z})\mathbf{z}^T \tag{11}$$

Constrained optimization

- Often, $\mathbf{w}$ is not allowed to take any arbitrary value in $\mathbb{R}^n$
  - Although this does not happen in our course, admittedly
- Example: $\|\mathbf{w}\| = 1$ (unit sphere)
- The set of allowed values is called the *constraint set*
- Two principal modifications to the gradient method:
  - Projection of $\mathbf{w}$ back to constraint set
  - Projection of the gradient to tangent of constraint set (omitted here)

Projecting $\mathbf{w}$ back to constraint set



- A function (not shown explicitly) is to be minimized on the unit sphere (constraint set)
- Starting at the point marked with "o", a small gradient step is taken, as shown by the arrow
- The point is projected to the closest point on the unit sphere, which is marked by "x". This is one iteration of the method.

Some examples of projections to constraint set

- Projecting means going to the point in the constraint set which is closest in Euclidean distance
- In general, computing the projection can be very difficult
- In some special cases, it is a simple operation
- For example, if the constraint set is the unit sphere: the projection is performed by

$$\mathbf{w} \leftarrow \mathbf{w}/\|\mathbf{w}\| \tag{12}$$

- Another example: orthogonality of a matrix. In that case, the projection onto the constraint set is given by

$$\mathbf{W} \leftarrow (\mathbf{W}\mathbf{W}^T)^{-1/2}\mathbf{W} \tag{13}$$

  Here, we see a rather involved operation: the inverse of the square root of the matrix (computable with standard libraries)