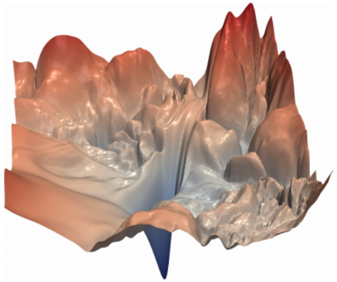
Deep Learning

Challenges in Optimization



Learning goals

-
-
-

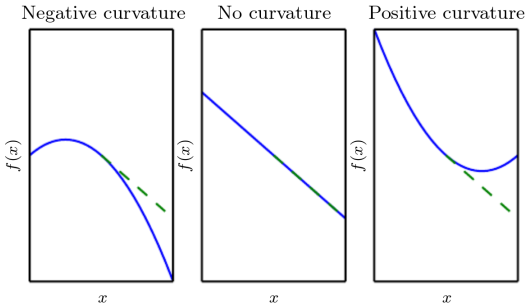
CHALLENGES IN OPTIMIZATION

- In this section, we summarize several of the most prominent challenges regarding training of deep neural networks.
- Traditionally, machine learning ensures that the optimization problem is convex by carefully designing the objective function and constraints. But for neural networks we are confronted with the general nonconvex case.
- Furthermore, we will see in this section that even convex optimization is not without its complications.

Ill-Conditioning

EFFECTS OF CURVATURE

Intuitively, the curvature of a function determines the outcome of a GD step…



Source: Goodfellow *et al.*, (2016), ch. 4

Figure: Quadratic objective function $f(\mathbf{x})$ with various curvatures. The dashed line indicates the first order taylor approximation based on the gradient information alone. Left: With negative curvature, the cost function decreases faster than the gradient predicts; Middle: With no curvature, the gradient predicts the decrease correctly; Right: With positive curvature, the function decreases more slowly than expected and begins to increase.

SECOND DERIVATIVE AND CURVATURE

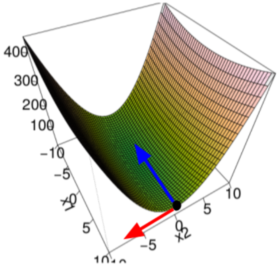
To understand better how the curvature of a function influences the outcome of a gradient descent step, let us recall how curvature is described mathematically:

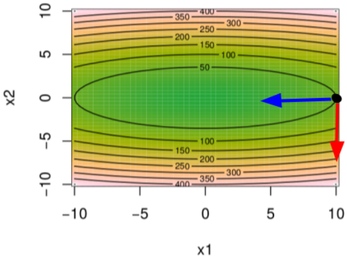
- The second derivative corresponds to the curvature of the graph of a function.
- The **Hessian** matrix of a function $\mathcal{R}(\boldsymbol{\theta}) : \mathbb{R}^m \to \mathbb{R}$ is the matrix of second-order partial derivatives

$$H_{ij} = \frac{\partial^2}{\partial\theta_i \partial\theta_j}\mathcal{R}(\boldsymbol{\theta}).$$

SECOND DERIVATIVE AND CURVATURE

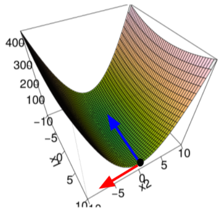
- The second derivative in a direction **d**, with $\|\mathbf{d}\| = 1$, is given by $\mathbf{d}^\top \mathbf{H}\, \mathbf{d}$.
- What is the direction of the highest curvature (red direction), and what is the direction of the lowest curvature (blue)?

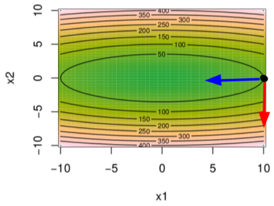




SECOND DERIVATIVE AND CURVATURE

- Since **H** is real and symmetric (why?), eigendecomposition yields $\mathbf{H} = \mathbf{V}\mathrm{diag}(\boldsymbol{-})\mathbf{V}^{-1}$ with **V** and $\boldsymbol{\lambda}$ collecting eigenvectors and eigenvalues, respectively.
- It can be shown, that the eigenvector $\boldsymbol{v}_{\text{max}}$ with the max. eigenvalue λ_{max} points into the direction of highest curvature ($\boldsymbol{v}_{\text{max}}^\top \boldsymbol{H}\boldsymbol{v}_{\text{max}} = \lambda_{\text{max}}$), while the eigenvector $\boldsymbol{v}_{\text{min}}$ with the min. eigenvalue λ_{min} points into the direction of least curvature.

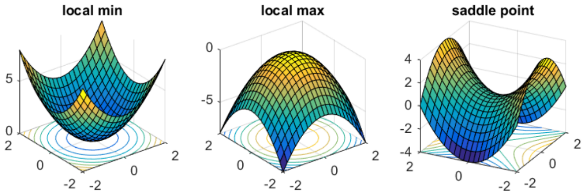




SECOND DERIVATIVE AND CURVATURE

- At a stationary point $\boldsymbol{\theta}$, where the gradient is 0, we can examine the eigenvalues of the Hessian to determine whether the $\boldsymbol{\theta}$ is a local maximum, minimum or saddle point:

$$\begin{aligned}
\forall i : \lambda_i > 0 \, (\mathbf{H} \text{ positive definite at } \boldsymbol{\theta}) \quad &\Rightarrow \quad \text{minimum at } \boldsymbol{\theta} \\
\forall i : \lambda_i < 0 \, (\mathbf{H} \text{ negative definite at } \boldsymbol{\theta}) \quad &\Rightarrow \quad \text{maximum at } \boldsymbol{\theta} \\
\exists\, i : \lambda_i < 0 \wedge \exists i : \lambda_i > 0 \, (\mathbf{H} \text{ indefinit at } \boldsymbol{\theta}) \quad &\Rightarrow \quad \text{saddle point at } \boldsymbol{\theta}
\end{aligned}$$

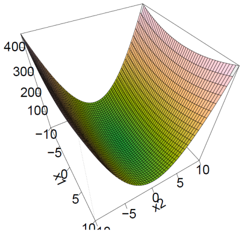


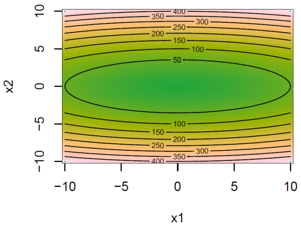
Credit: Rong Ge (2016)

ILL-CONDITIONED HESSIAN MATRIX

The condition number of a symmetric matrix $\boldsymbol{A}$ is given by the ratio of its min/max eigenvalues $\kappa(\boldsymbol{A}) = \frac{|\lambda_{\text{max}}|}{|\lambda_{\text{min}}|}$. A matrix is called ill-conditioned, if the condition number $\kappa(\boldsymbol{A})$ is very high.

An **ill-conditioned** Hessian matrix means that the ratio of max. / min. curvature is high, as in the example below:





CURVATURE AND STEP-SIZE IN GD

What does it mean for gradient descent if the Hessian is ill-conditioned?

- Let us consider the second-order Taylor approximation as a local approximation of the of $\mathcal{R}$ around a current point $\boldsymbol{\theta}^0$ (with gradient $\mathbf{g}$)

$$\mathcal{R}(\boldsymbol{\theta}) \approx T_2 f(\boldsymbol{\theta}, \boldsymbol{\theta}^0) := \mathcal{R}(\boldsymbol{\theta}^0) + (\boldsymbol{\theta} - \boldsymbol{\theta}^0)^\top \mathbf{g} + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^0)^\top \mathbf{H}\, (\boldsymbol{\theta} - \boldsymbol{\theta}^0)$$

- Furthermore, Taylor's theorem states (proof in Koenigsberger (1997), p. 68)

$$\lim_{\boldsymbol{\theta} \to \boldsymbol{\theta}^0} \frac{\mathcal{R}(\boldsymbol{\theta}) - T_2 f(\boldsymbol{\theta}, \boldsymbol{\theta}^0)}{||\boldsymbol{\theta} - \boldsymbol{\theta}^0||^2} = 0$$

CURVATURE AND STEP-SIZE IN GD

- One GD step with a learning rate α yields new parameters $\boldsymbol{\theta}^0 - \alpha\mathbf{g}$ and a new approximated loss value

$$\mathcal{R}(\boldsymbol{\theta}^0 - \alpha\mathbf{g}) \approx \mathcal{R}(\boldsymbol{\theta}^0) - \alpha\mathbf{g}^\top\mathbf{g} + \frac{1}{2}\alpha^2\mathbf{g}^\top\mathbf{H}\,\mathbf{g}\,.$$

- Theoretically, if $\mathbf{g}^\top\mathbf{H}\mathbf{g}$ is positive, we can solve the equation above for the optimal step size which corresponds to

$$\alpha^* = \frac{\mathbf{g}^\top\mathbf{g}}{\mathbf{g}^\top\mathbf{H}\,\mathbf{g}}\,.$$

CURVATURE AND STEP-SIZE IN GD

- Let us assume the gradient $\mathbf{g}$ points into the direction of $\boldsymbol{v}_{\text{max}}$ (i.e. the direction of highest curvature), the optimal step size is given by

$$\alpha^* = \frac{\mathbf{g}^\top \mathbf{g}}{\mathbf{g}^\top \mathbf{H}\, \mathbf{g}} = \frac{\mathbf{g}^\top \mathbf{g}}{\lambda_{\text{max}} \mathbf{g}^\top \mathbf{g}} = \frac{1}{\lambda_{\text{max}}},$$

  which is very small. Choosing a too large step-size is bad, as it will make us “overshoot” the stationary point.

- If, on the other hand, $\mathbf{g}$ points into the direction of the lowest curvature, the optimal step size is

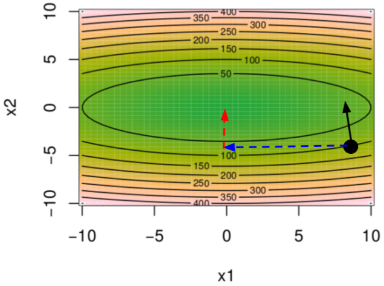
$$\alpha^* = \frac{1}{\lambda_{\text{min}}},$$

  which corresponds to the largest possible optimal step-size.

- We summarize: We want to perform big steps in directions of low curvature, but small steps in directions of high curvature.

CURVATURE AND STEP-SIZE IN GD

- But what if the gradient does not point into the direction of one of the eigenvectors?
- Let us consider the 2-dimensional case: We can decompose the direction of $\boldsymbol{g}$ (black) into the two eigenvectors $\boldsymbol{v}_{\text{max}}$ and $\boldsymbol{v}_{\text{min}}$
- It would be optimal to perform a **big** step into the direction of the smallest curvature $\boldsymbol{v}_{\text{min}}$, but a **small** step into the direction of $\boldsymbol{v}_{\text{max}}$, but the gradient points into a completely different direction.



ILL-CONDITIONING

- GD is unaware of large differences in curvature, and can only walk into the direction of the gradient.
- Choosing a too large step-size will then cause the descent direction change frequently (“jumping around”).
- α needs to be small enough, which results in a low progress.

ILL-CONDITIONING

- This effect is more severe, if a Hessian has a poor condition number, i.e. the ratio between lowest and highest curvature is large; gradient descent will perform poorly.

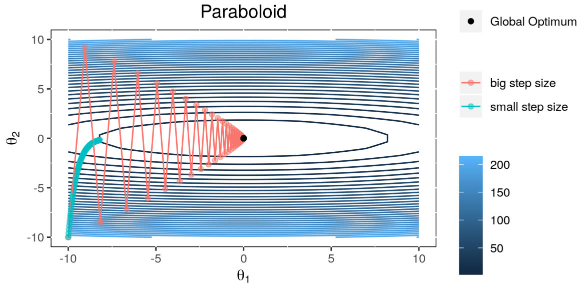


Figure: The contour lines show a quadratic risk function with a poorly conditioned Hessian matrix. The plot shows the progress of gradient descent with a small step-size vs. larger step-size. In both cases, convergence to the global optimum is rather slow.

ILL-CONDITIONING

- In the worst case, ill-conditioning of the Hessian matrix and a too big step-size will cause the risk to increase

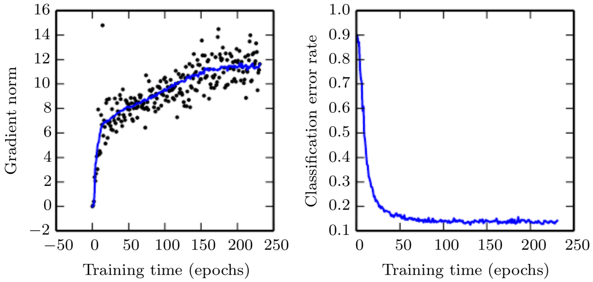
$$\mathcal{R}(\boldsymbol{\theta}^0 - \alpha \mathbf{g}) \approx \mathcal{R}(\boldsymbol{\theta}^0) - \alpha \mathbf{g}^\top \mathbf{g} + \frac{1}{2}\alpha^2 \mathbf{g}^\top \mathbf{H}\,\mathbf{g}\,,$$

  which happens if

$$\frac{1}{2}\alpha^2 \mathbf{g}^\top \mathbf{H}\,\mathbf{g} > \alpha \mathbf{g}^\top \mathbf{g}.$$

- To determine whether ill-conditioning is detrimental to the training, the squared gradient norm $\mathbf{g}^\top \mathbf{g}$ and the risk can be monitored.

ILL-CONDITIONING



Source: Goodfellow, ch. 6

- Gradient norms **increase** over time, showing that the training process is not converging to a stationary point $\mathbf{g} = 0$.
- At the same time, we observe that the risk is approx. constant, but the gradient norm increases

$$\underbrace{\mathcal{R}(\boldsymbol{\theta}^0 - \alpha \mathbf{g})}_{\text{approx. constant}} \approx \mathcal{R}(\boldsymbol{\theta}^0) - \underbrace{\alpha \mathbf{g}^\top \mathbf{g}}_{\text{increase}} + \frac{1}{2} \underbrace{\alpha^2 \mathbf{g}^\top \mathbf{H}\, \mathbf{g}}_{\rightarrow\text{increase}} .$$

Local Minima

UNIMODAL VS. MULTIMODAL LOSS SURFACES

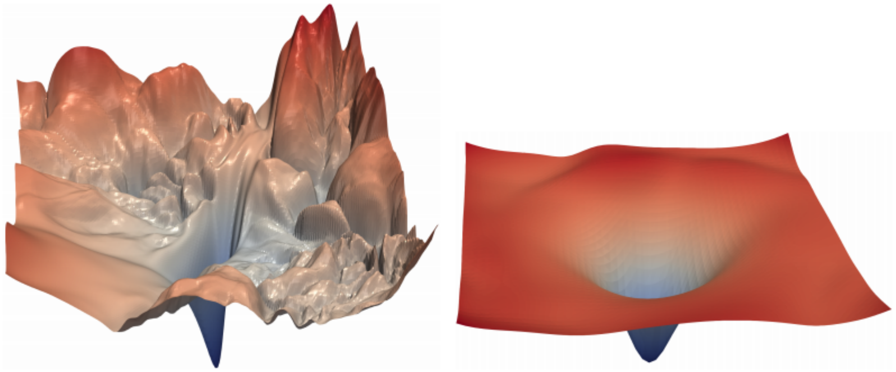
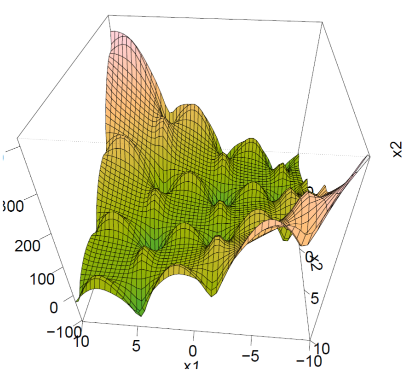
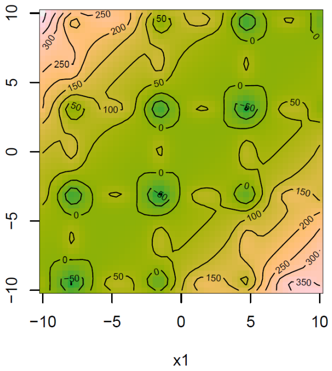


Figure: Left: Multimodal loss surface with saddle points; Right: (Nearly) unimodal loss surface (Hao Li et al. (2017))

MULTIMODAL FUNCTION

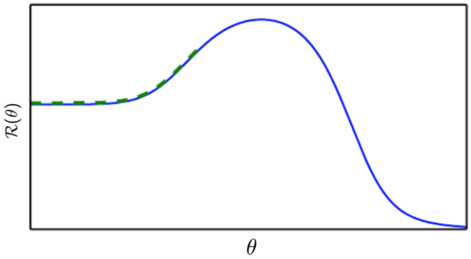
Potential snippet from a loss surface of a deep neural network with many local minima:





ONLY LOCALLY OPTIMAL MOVES

- If the training algorithm makes only *locally* optimal moves (as in gradient descent), it may move away from regions of *much* lower cost.



Source: Goodfellow, Ch. 8

- In the figure above, initializing the parameter on the "wrong" side of the hill will result in suboptimal performance.
- In higher dimensions, however, it may be possible for gradient descent to go around the hill but such a trajectory might be very long and result in excessive training time.

LOCAL MINIMA

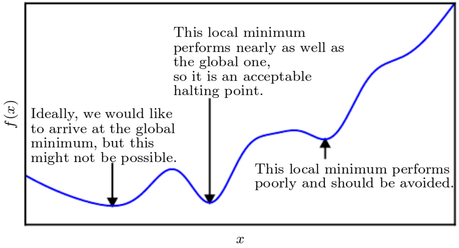
Weight space symmetry:

- If we swap incoming weight vectors for neuron *i* and *j* and do the same for the outcoming weights, modelled function stays unchanged.
  - $\Rightarrow$ with *n* hidden units and one hidden layer there are *n*! networks with the same empirical risk
- If we multiply incoming weights of a ReLU neuron with β and outcoming with $1/\beta$ the modelled function stays unchanged.

$\Rightarrow$ The empirical risk of a NN can have very many minima with equivalent empirical risk.

LOCAL MINIMA

- In practice only local minima with a high value compared to the global minimium are problematic.



Source: Goodfellow, Ch. 4

- Current literature suspects that most local minima have low empirical risk.
- Simple test: Norm of gradient should get close to zero.

Saddle Points

SADDLE POINTS

- In optimization we look for areas with zero gradient.
- A variant of zero gradient areas are saddle points.
- For the empirical risk $\mathcal{R}$ of a neural network, the expected ratio of the number of saddle points to local minima typically grows exponentially with m

$$\mathcal{R} : \mathbb{R}^m \to \mathbb{R}$$

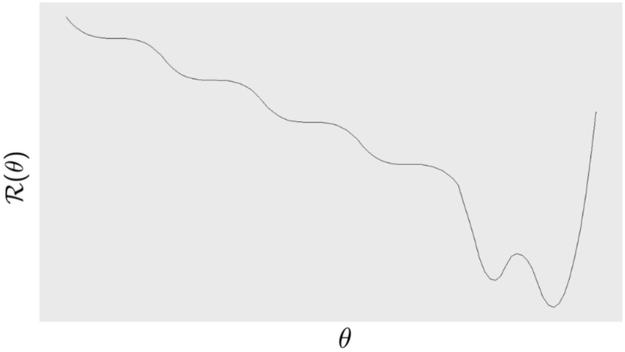
  In other words: Networks with more parameters (deeper networks or larger layers) exhibit a lot more saddle points than local minima.
- Why is that?
- The Hessian at a local minimum has only positive eigenvalues. At a saddle point it is a mixture of positive and negative eigenvalues.

SADDLE POINTS

- Imagine the sign of each eigenvalue is generated by coin flipping:
  - In a single dimension, it is easy to obtain a local minimum (e.g. “head” means positive eigenvalue).
  - In an *m*-dimensional space, it is exponentially unlikely that all *m* coin tosses will be head.
- A property of many random functions is that eigenvalues of the Hessian become more likely to be positive in regions of lower cost.
- For the coin flipping example, this means we are more likely to have heads *m* times if we are at a critical point with low cost.
- That means in particular that local minima are much more likely to have low cost than high cost and critical points with high cost are far more likely to be saddle points.
- See Dauphin et al. (2014) for a more detailed investigation.

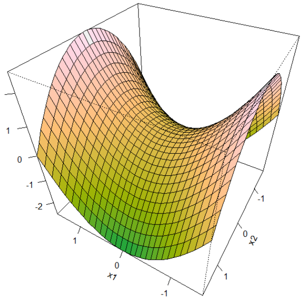
SADDLE POINTS

- “Saddle points are surrounded by high error plateaus that can dramatically slow down learning, and give the illusory impression of the existence of a local minimum” (Dauphin et al. (2014)).



SADDLE POINTS: EXAMPLE

$$f(x_1, x_2) = x_1^2 - x_2^2$$

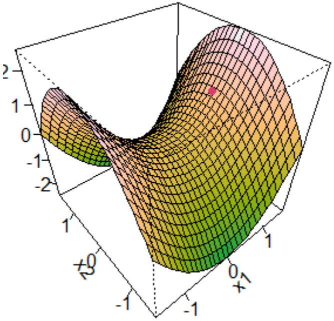


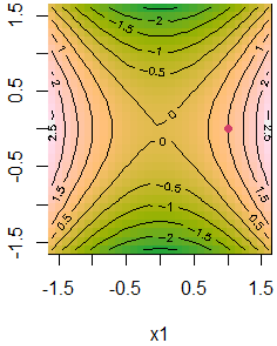
- Along x_1, the function curves upwards (eigenvector of the Hessian with positive eigenvalue). Along x_2, the function curves downwards (eigenvector of the Hessian with negative eigenvalue).

SADDLE POINTS

- So how do saddle points impair optimization?
- First-order algorithms that use only gradient information **might** get stuck in saddle points.
- Second-order algorithms experience even greater problems when dealing with saddle points. Newtons method for example actively searches for a region with zero gradient. That might be another reason why second-order methods have not succeeded in replacing gradient descent for neural network training.

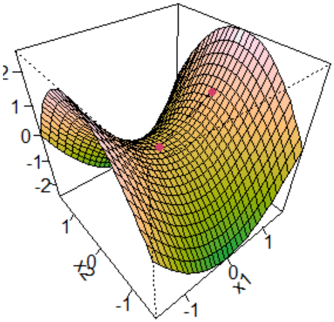
EXAMPLE: SADDLE POINT WITH GD

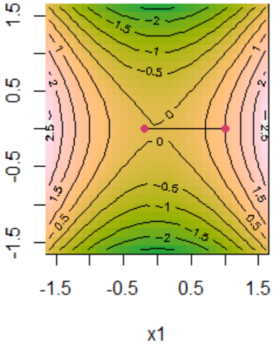




Red dot: Starting location

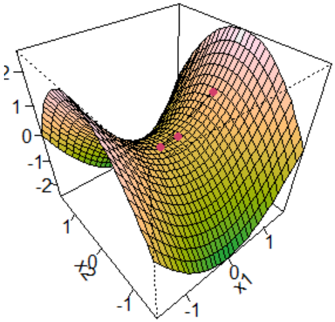
EXAMPLE: SADDLE POINT WITH GD

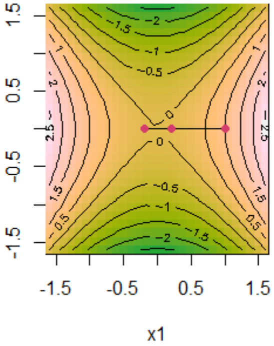




First step...

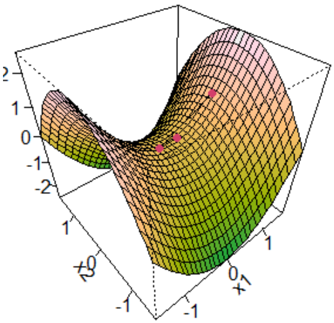
EXAMPLE: SADDLE POINT WITH GD

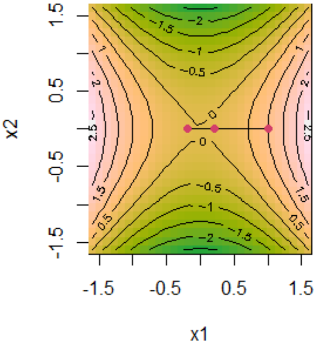




...second step...

EXAMPLE: SADDLE POINT WITH GD





...tenth step got stuck and cannot escape the saddle point!

Cliffs and Exploding Gradients

CLIFFS AND EXPLODING GRADIENTS

- As a result from the multiplication of several parameters, the emprirical risk for highly nonlinear deep neural networks often contain sharp nonlinearities.
  - That may result in very high derivatives in some places.
  - As the parameters get close to such cliff regions, a gradient descent update can catapult the parameters very far.
  - Such an occurrence can lead to losing most of the optimization work that had been done.
- However, serious consequences can be easily avoided using a technique called **gradient clipping**.
- The gradient does not specify the optimal step size, but only the optimal direction within an infinitesimal region.

CLIFFS AND EXPLODING GRADIENTS

- Gradient clipping simply caps the step size to be small enough that it is less likely to go outside the region where the gradient indicates the direction of steepest descent.
- We simply "prune" the norm of the gradient at some threshold *h*:

$$\text{if } ||\nabla\boldsymbol{\theta}|| > \text{h} : \nabla\boldsymbol{\theta} \leftarrow \frac{h}{||\nabla\boldsymbol{\theta}||}\nabla\boldsymbol{\theta}$$

EXAMPLE: CLIFFS AND EXPLODING GRADIENTS

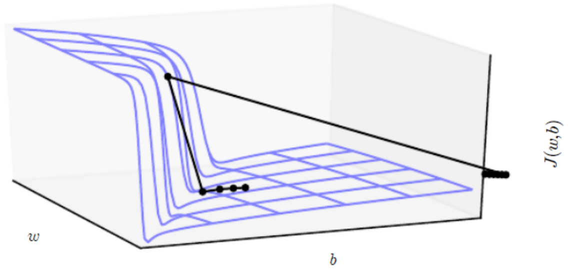


Figure: “The objective function for highly nonlinear deep neural networks or for recurrent neural networks often contains sharp nonlinearities in parameter space resulting from the multiplication of several parameters. These nonlinearities give rise to very high derivatives in some places. When the parameters get close to such a cliff region, a gradient descent update can catapult the parameters very far, possibly losing most of the optimization work that had been done” (Goodfellow et al. (2016)).

REFERENCES

Ian Goodfellow, Yoshua Bengio and Aaron Courville (2016)

Deep Learning

http://www.deeplearningbook.org/

Yann Dauphin, Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, Surya Ganguli, Yoshua Bengio (2014)

Identifying and attacking the saddle point problem in high-dimensional non-convex optimization

https://arxiv.org/abs/1406.2572

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, Tom Goldstein (2017)

Visualizing the Loss Landscape of Neural Nets

https://arxiv.org/abs/1712.09913

Konrad Koenigsberger (1997)

Analysis 2, Springer

Rong Ge (2016)

Escaping from Saddle Points

http://www.offconvex.org/2016/03/22/saddlepoints/