# optimisation in neural networks

Patrick van der Smagt

# What is optimisation?

We have a model $p_{\mathbf{w}}(z \mid x)$. This can, e.g., be a neural network.

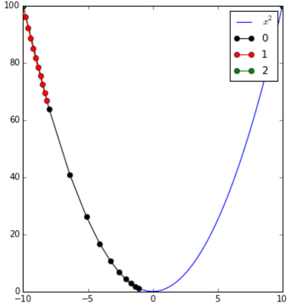
We want to minimise the loss

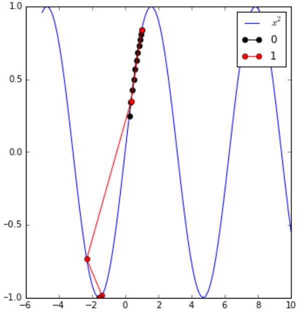
$$\mathcal{L}(\mathbf{w}) = -\log \prod_i p_{\mathbf{w}}(z_i \mid x_i)$$

by finding better values of $\mathbf{w}$.

# How do we do optimisation?

- if finding the best **w** is a convex problem, good methods exist (remember SVD from linear algebra).
- In general, finding the best **w** is not a convex problem. Only incremental methods are known.



# Convex optimisation problems

Practical example: $\{(x_i, z_i)\} = \{(0,1); (1,2.1); (2,2.9)\}$.

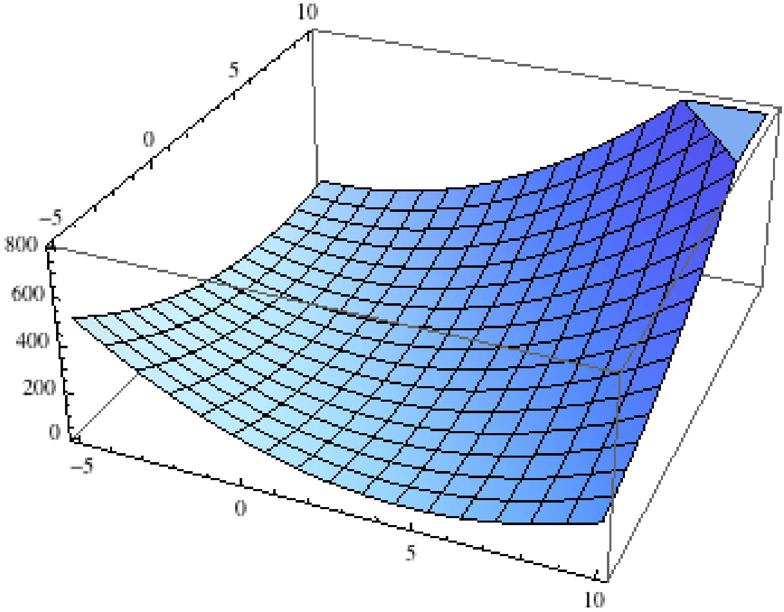
Our model: $y_{\mathbf{w}}(x) = ax + b$ with $\mathbf{w} = (a, b)$; the MLE loss is

$$\mathcal{L}(\mathbf{w}) = \sum_i \big(y_{\mathbf{w}}(x_i) - z_i\big)^2$$

How do we find the minimum of $\mathcal{L}$? It is there where $\partial\mathcal{L}/\partial w_i = 0$.

$$\frac{\partial \mathcal{L}_i(\mathbf{w})}{\partial w_1 \equiv a} = 2x_i(b + ax_i - z_i) = 0$$
$$\frac{\partial \mathcal{L}_i(\mathbf{w})}{\partial w_2 \equiv b} = 2(b + ax_i - z_i) = 0$$

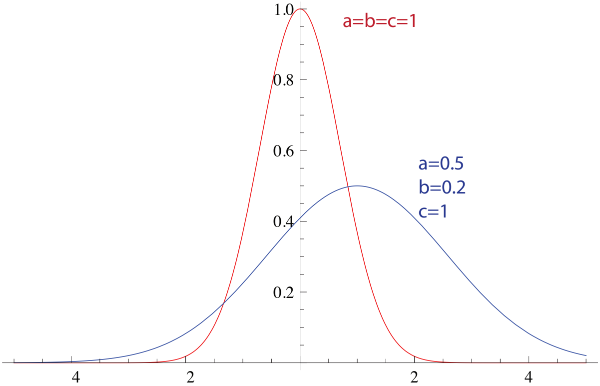
We can solve that!




Numerical solution: $a = 0.95, b = 1.05$.

# What if. . .

$$y_{(a,b,c)}(x) = a \exp(-b(\mathbf{x} - c)^2) + d$$



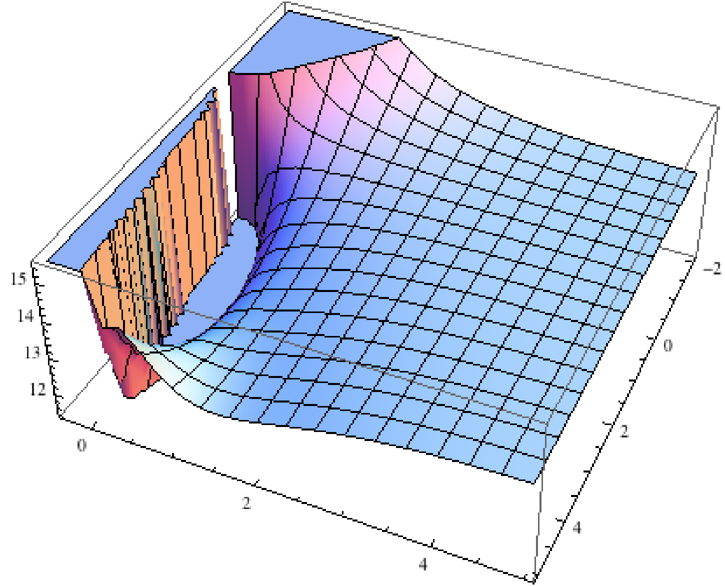

We can't find a closed-form solution for that!

$$2e^{-bc^2}\left(ae^{-bc^2}-1\right)+2e^{-b(1-c)^2}\left(ae^{-b(1-c)^2}-2.1\right)+ 2e^{-b(2-c)^2}\left(ae^{-b(2-c)^2}-2.9\right)=0$$

$$-2ac^2e^{-bc^2}\left(ae^{-bc^2}-1\right)-2a(1-c)^2e^{-b(1-c)^2}\left(ae^{-b(1-c)^2}-2.1\right)- 2a(2-c)^2e^{-b(2-c)^2}\left(ae^{-b(2-c)^2}-2.9\right)=0$$

$$-4abce^{-bc^2}\left(ae^{-bc^2}-1\right)+4ab(1-c)e^{-b(1-c)^2}\left(ae^{-b(1-c)^2}-2.1\right)+ 4ab(2-c)e^{-b(2-c)^2}\left(ae^{-b(2-c)^2}-2.9\right)=0$$

Out[56]=



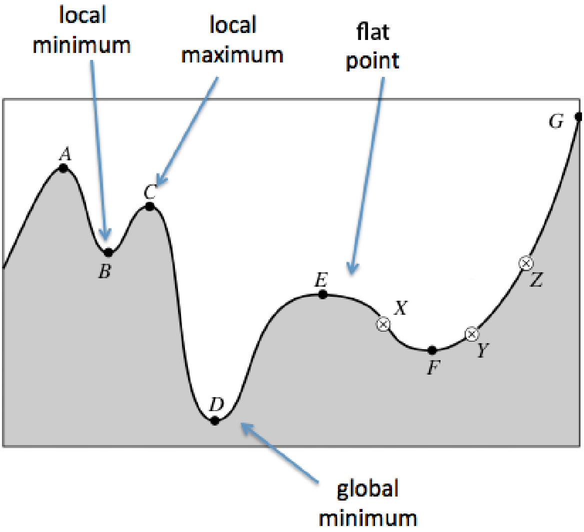

$c = -1$

# But

We can compute $\partial \mathcal{L}(\mathbf{w}) / \partial w_i$

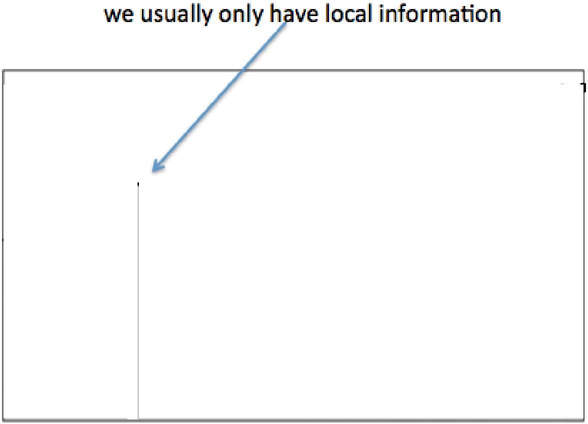
# the value of $\mathcal{L}$

we are interested in finding $\arg\min_{\mathbf{w}} \mathcal{L}$



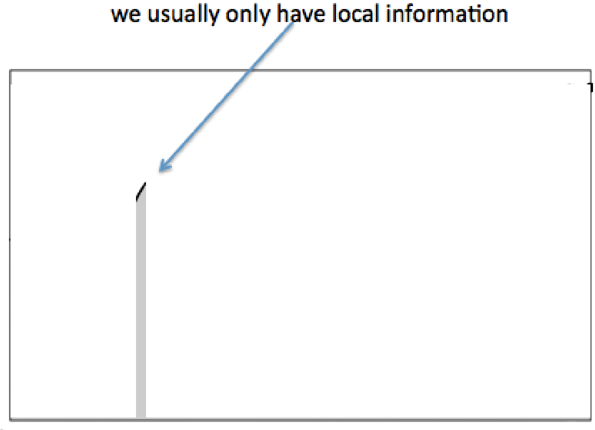
# using local information $\mathcal{L}(\mathbf{w})$

we are interested in finding $\arg\min_{\mathbf{w}} \mathcal{L}$



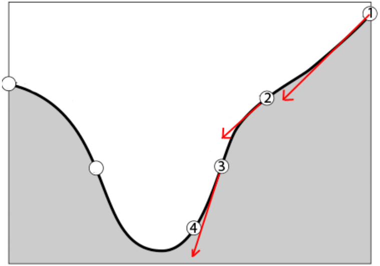
# using local information $\mathcal{L}(\mathbf{w})$ as well as $\partial\mathcal{L}(\mathbf{w})/\partial w$

we are interested in finding $\arg\min_{\mathbf{w}} \mathcal{L}$



# using the gradient $\mathbf{g}$ of $\mathcal{L}$

The direction $\mathbf{u}$ in which to optimise is given by the gradient: $\mathbf{u} = -\mathbf{g}$




Searching the minimum by repeated evaluation of $\mathcal{L}$ and $\mathbf{g} \equiv \nabla \mathcal{L}$.
$-\mathbf{g}$ gives us a direction $\mathbf{u}$ in which we want to optimise.
We change the parameter vector as follows:

$$\mathbf{u}_i = -\mathbf{g}_i \tag{1}$$

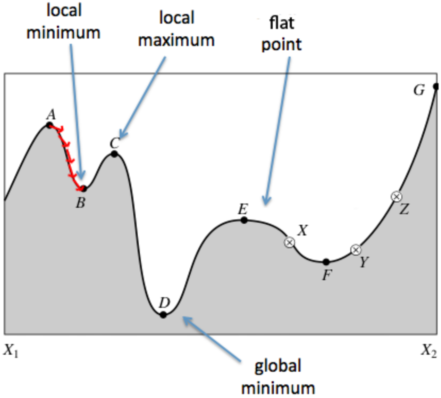
$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha \mathbf{u}_i \tag{2}$$

we call $\mathbf{u}$ the search direction
we call $\alpha$ the learning parameter or step size
we call this method steepest descent or gradient descent
it belongs to the class of greedy algorithms

# the value of $\alpha$

a *too small* value for $\alpha$ has two drawbacks:

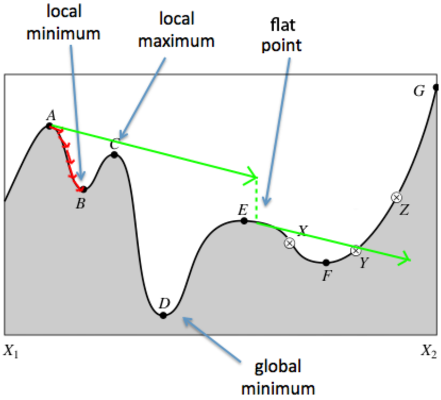
- we find the minimum more slowly
- we end up in local minima or saddle/flat points



# the value of $\alpha$

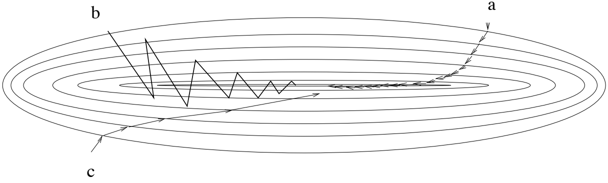
a *too large* value for $\alpha$ has one drawback:

- you may never find a minimum; oscillations usually occur




we only need 2 steps to overshoot!

# putting a trace on $\mathbf{u}$




a: $\mathbf{u} = -\mathbf{g}$; small $\alpha$;
b: $\mathbf{u} = -\mathbf{g}$; large $\alpha$;
c:

$$\begin{aligned} \mathbf{u}_0 &= -\mathbf{g}_0 && (3) \\ \mathbf{u}_i &= -\mathbf{g}_i + \beta \mathbf{u}_{i-1} && (4) \\ &= -\mathbf{g}_i - \beta \mathbf{g}_{i-1} - \beta^2 \mathbf{g}_{i-2} - \beta^3 \mathbf{g}_{i-3} \ldots && (5) \\ \mathbf{w}_{i+1} &= \mathbf{w}_i + \alpha \mathbf{u}_i && (6) \end{aligned}$$

we call $\alpha$ the learning rate
we call $\beta$ the momentum
we usually take $\beta \gg \alpha$

# trick: momentum

How do we choose $\alpha$ and $\beta$? if, for the sake of the argument, assume that $\mathbf{g} \equiv \nabla E$ does not change:

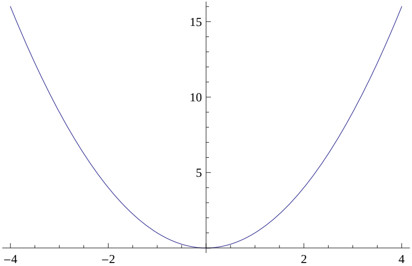
$$\begin{aligned} \Delta \mathbf{w} &= -\alpha \, \mathbf{g} \, (1 + \beta + \beta^2 + \ldots) \\ &= -\frac{\alpha}{1-\beta} \, \mathbf{g} \end{aligned}$$

Assuming a perfect $\nabla E$, the best values for $\alpha$ and $\beta$ are when

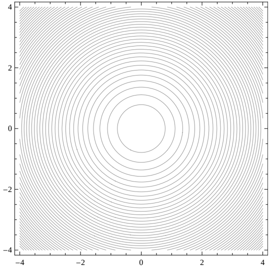
$$\frac{\alpha}{1-\beta} = 1 \qquad \Rightarrow \qquad \alpha + \beta = 1$$

Typically we choose $\alpha$ small and $\beta$ large (of course, $\alpha, \beta > 0$).

# bird's eye view



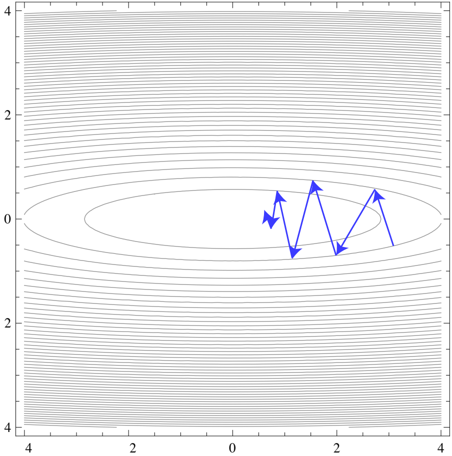

=




$$\mathcal{L}(x) = \mathcal{L}(0) + x \underbrace{\frac{\partial \mathcal{L}}{\partial w}}_{g} + x^2 \underbrace{\frac{\partial^2 \mathcal{L}}{\partial w^2}}_{\text{Hessian} H}$$

# optimising

following the gradient is not always the best choice

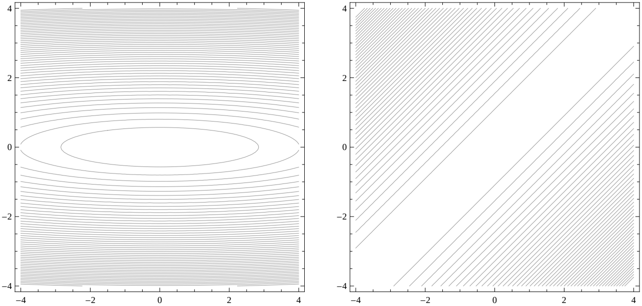



Close to minima, it appears that Loss functions are close to quadratic

# condition of the Hessian

Condition = largest EV / smallest EV

Condition 5 Condition 100




What does $H$ look like?

A large condition number means that some directions of $H$ are very steep compared to others. In neural networks, a condition of $10^{10}$ is not uncommon.

A class of optimisers (CG, Adam, rprop, adadelta, ...) deal with such $H$.