GRADIENT DESCENT: NOW IN 2D!

① Usually loss functions involve more than one variable. For instance, we might want to relaunch our "guess your age or win a prize" booth by predicting age (y) as a function of number of CS courses taken (x_1) and cholesterol level (x_2).

x_1	x_2	y
5	180	20
12	210	41

Assuming that $y \approx \theta_1 x_1 + \theta_2 x_2$ for some constants θ_1, θ_2, we get a loss function like:

$$L(\theta_1, \theta_2) = (20 - (5\theta_1 + 180\theta_2))^2 + (41 - (12\theta_1 + 210\theta_2))^2$$

Thus, we want to find the values of θ_1 and θ_2 that minimize our loss function $L(\theta_1, \theta_2)$:

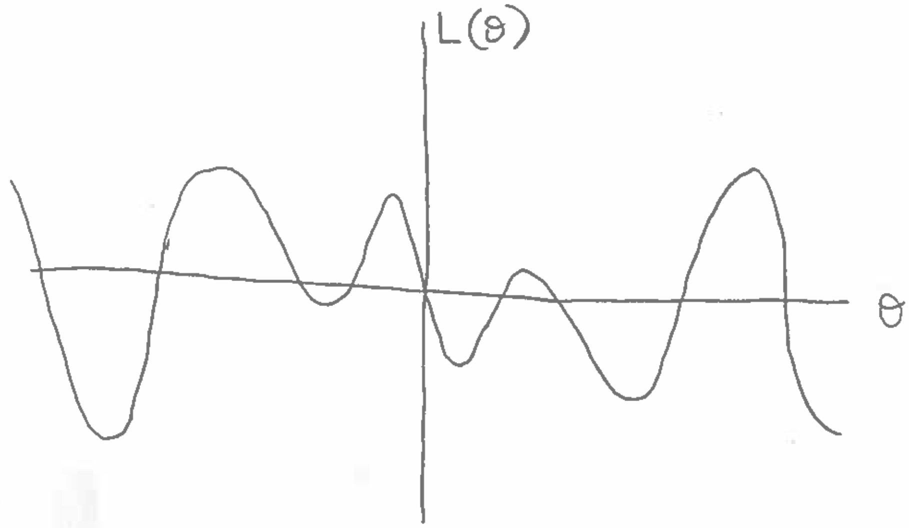
$$\underset{a}{\operatorname{argmin}}\ L(\theta_1, \theta_2)$$

② So now we have a loss function over 2 variables, not just 1.

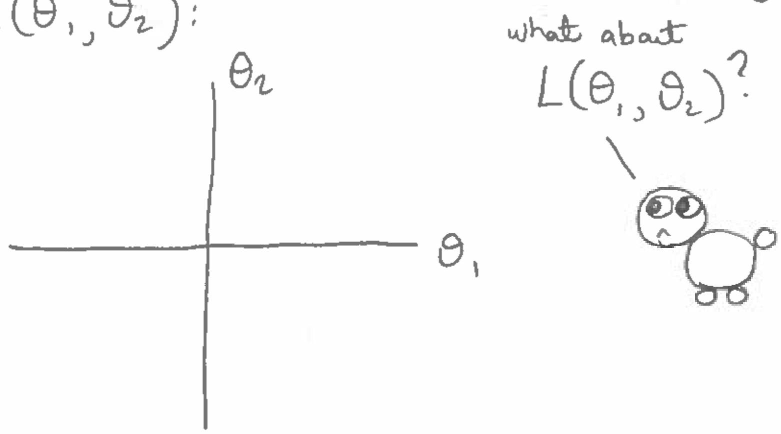
How do we do this minimization?

GRADIENT DESCENT: Now IN 2D!

(3) Multivariable functions are immediately annoying because they're hard to visualize. For 1-variable functions, we can use one axis for θ and the second for the value of $L(\theta)$:



For a 2-variable function $L(\theta_1, \theta_2)$, if we use one axis for θ_1 and the second for θ_2, we run out of axes for the value of $L(\theta_1, \theta_2)$:



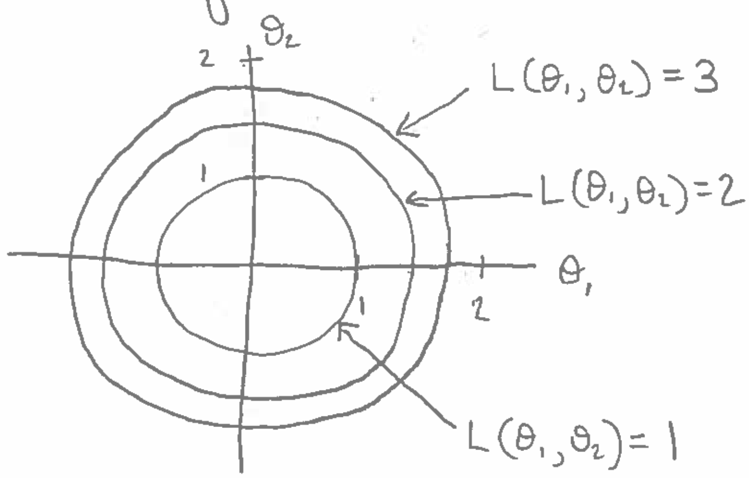
Gradient Descent: Now in 2D!

4) A common solution are contour plots. Say

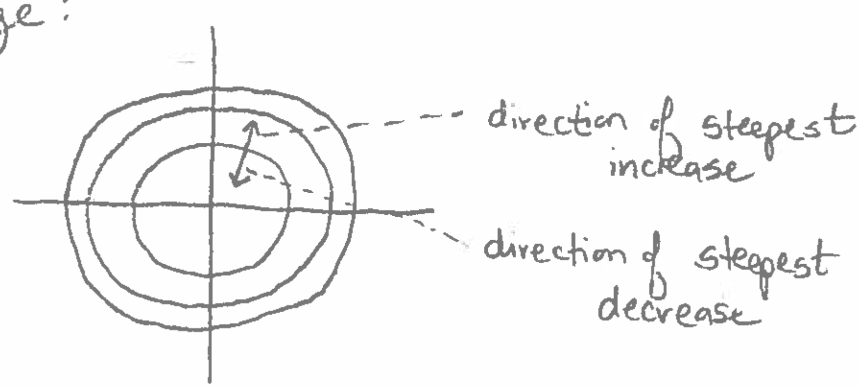
$$L(\theta_1, \theta_2) = \theta_1^2 + \theta_2^2$$

To "show" the value of $L(\theta_1, \theta_2)$, let's select a couple of values we're interested in (say, $L(\theta_1, \theta_2) = 1$, $L(\theta_1, \theta_2) = 2$, $L(\theta_1, \theta_2) = 3$)

Now, we draw all points on the (θ_1, θ_2)-plane where $L(\theta_1, \theta_2)$ equals one of the chosen values:

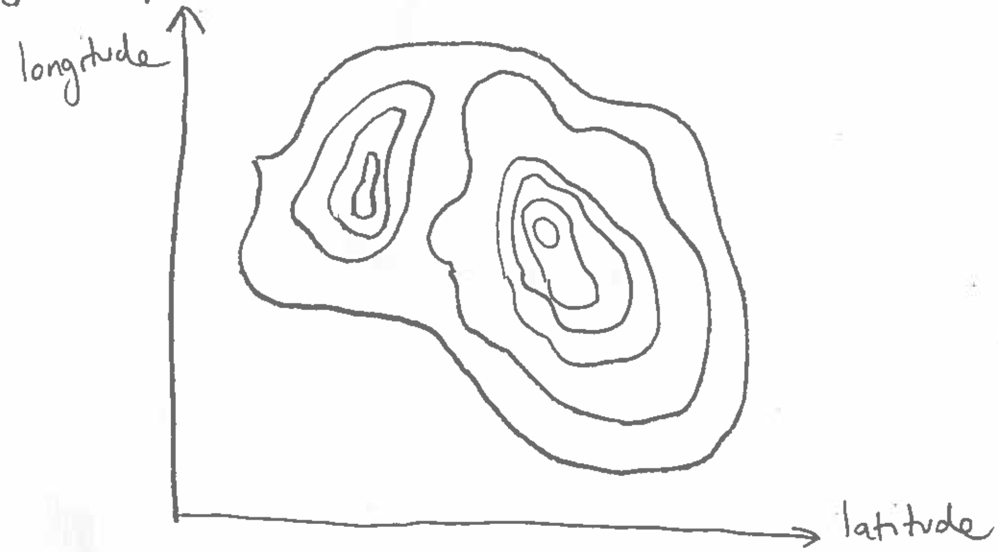


5) Geometrically, we can see that as we travel along one of these "contours" (e.g. the circle $L(\theta_1, \theta_2) = 1$), then the value of $L(\theta_1, \theta_2)$ doesn't change. It turns out that travelling perpendicularly to a contour gives the most drastic change:



GRADIENT DESCENT: Now in 2D!

⑥ You've likely seen contour plots before in the form of topological maps, where the "loss function" is the altitude of a particular point.

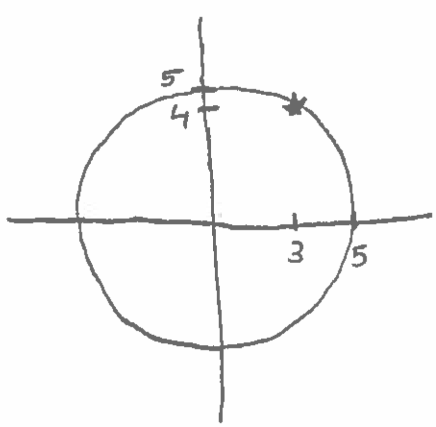


Our goal is to find the lowest point of the map (the deepest valley).

---

⑦ How do we extend gradient descent from 1 to 2 variables? Just pretend the other variable doesn't exist.

Suppose the loss function is $L(\theta_1, \theta_2) = \theta_1^2 + \theta_2^2$. The first step of gradient descent is to guess the solution, so let's guess $\hat{\theta}_1 = 3$, $\hat{\theta}_2 = 4$.

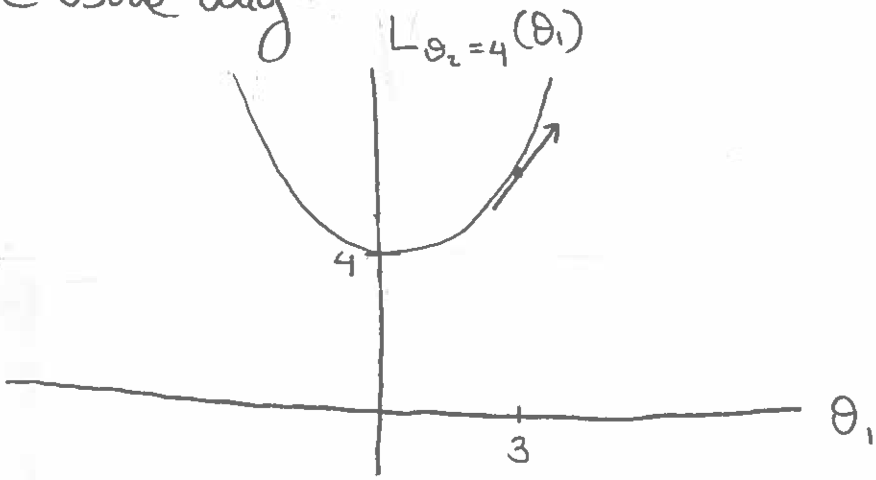


GRADIENT DESCENT: NOW IN 2D!

8) If we pretend $\hat{\theta}_2$ is already correct, then we can focus on just minimizing L as a function of θ_1 (where θ_2 is treated like any old constant), i.e.

$$L_{\theta_2=\hat{\theta}_2}(\theta_1) = \theta_1^2 + \hat{\theta}_2^2$$

Now that the function has only one variable, we can plot it in the usual way



and find its derivative in the usual way:

$$\frac{dL_{\theta_2=\hat{\theta}_2}(\theta_1)}{d\theta_1} = 2\theta_1$$

So our step size (using vanilla gradient descent) at $\hat{\theta}_1 = 3$ is:

$$-\alpha \cdot \frac{dL_{\theta_2=\hat{\theta}_2}(3)}{d\theta_1}$$

$$= -6\alpha$$

GRADIENT DESCENT: NOW IN 2D!

⑨ Simultaneously, we can pretend $\hat{\theta}_1$ is already correct, and focus on minimizing L as a function of θ_2:

$$L_{\theta_1=\hat{\theta}_1}(\theta_2) = \hat{\theta}_1^2 + \theta_2^2$$

Its derivative is:

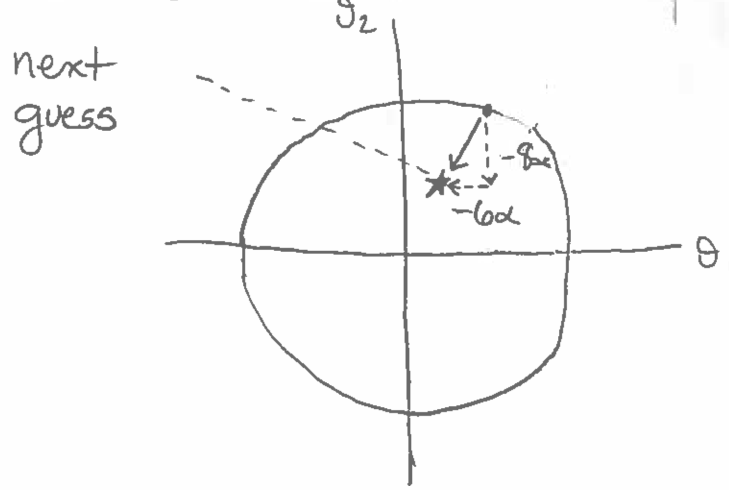
$$\frac{dL_{\theta_1=\hat{\theta}_1}(\theta_2)}{d\theta_2} = 2\theta_2$$

So our step size at our current guess $\hat{\theta}_2 = 4$ is:

$$-\alpha \cdot \frac{dL_{\theta_1=\hat{\theta}_1}}{d\theta_2}(4)$$

$$= -8\alpha$$

⑩ If we take both of these steps in quick sequence, then we get a new guess in the 2D variable space:



GRADIENT DESCENT: NOW IN 2D!

(11) The operation of "treating all variables except one as a constant" is called a <u>partial derivative</u>, and denoted:

$$\frac{\partial L}{\partial \theta_1}(\theta_1) = \frac{dL_{\theta_2=\hat{\theta}_2}(\theta_1)}{d\theta_1}$$

$$\frac{\partial L}{\partial \theta_2}(\theta_2) = \frac{dL_{\theta_1=\hat{\theta}_1}(\theta_2)}{d\theta_2}$$

This works for functions of any finite number of variables, e.g. for $L(\theta_1, \theta_2, \theta_3)$, we have:

$$\frac{\partial L}{\partial \theta_1}(\theta_1) = \frac{dL_{\theta_2=\hat{\theta}_2,\, \theta_3=\hat{\theta}_3}(\theta_1)}{d\theta_1}$$

$$\frac{\partial L}{\partial \theta_2}(\theta_2) = \frac{dL_{\theta_1=\hat{\theta}_1,\, \theta_3=\hat{\theta}_3}(\theta_2)}{d\theta_2}$$

$$\frac{\partial L}{\partial \theta_3}(\theta_3) = \frac{dL_{\theta_1=\hat{\theta}_1,\, \theta_2=\hat{\theta}_2}(\theta_3)}{d\theta_3}$$

---

(12) It will often be convenient to organize these partial derivatives into a vector, called the <u>gradient</u> and denoted ∇L, e.g.

$$\nabla L(\theta_1, \theta_2, \theta_3) = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \frac{\partial L}{\partial \theta_3} \end{bmatrix}$$

GRADIENT DESCENT: NOW IN 2D!

(13) Exercise: If $L(\theta_1, \theta_2, \theta_3) = 5\theta_1 + \theta_1\theta_2 + \theta_3^3$, what is the gradient of L?

Answer: $$\nabla L(\theta_1, \theta_2, \theta_3) = \begin{bmatrix} \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \frac{\partial L}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} 5 + \theta_2 \\ \theta_1 \\ 3\theta_3^2 \end{bmatrix}$$

(14) Generalizing single-variable gradient descent to multiple variables is straightforward:

GRADIENT DESCENT (loss function L, learning rate α):

- initialize $\theta^{(0)} = \begin{bmatrix} \theta_1^{(0)} \\ \vdots \\ \theta_d^{(0)} \end{bmatrix}$; $t \leftarrow 0$
- repeat until happy:
  - let step $\sigma^{(t)} = -\alpha \cdot \nabla L(\theta^{(t)})$
  - let next guess $\theta^{(t+1)} = \theta^{(t)} + \sigma^{(t)}$
  - let $t = t + 1$

e.g. $\begin{bmatrix} \sigma_1^{(t)} \\ \sigma_2^{(t)} \end{bmatrix} = -\alpha \cdot \begin{bmatrix} \frac{\partial L(\theta^{(t)})}{\partial \theta_1} \\ \frac{\partial L(\theta^{(t)})}{\partial \theta_2} \end{bmatrix}$ for 2-dimensions

e.g. $\begin{bmatrix} \theta_1^{(t+1)} \\ \theta_2^{(t+1)} \end{bmatrix} = \begin{bmatrix} \theta_1^{(t)} \\ \theta_2^{(t)} \end{bmatrix} + \begin{bmatrix} \sigma_1^{(t)} \\ \sigma_2^{(t)} \end{bmatrix}$ for 2-dimensions

Gradient Descent: Now in 2D!

15) A key to understanding multivariable gradient descent is that it's simply running several single variable gradient descents in parallel, one for each variable.

Say $L(\theta_1, \theta_2) = \theta_1^2 + \theta_2^2$ and the first guess $\theta^{(0)} = \begin{bmatrix} \theta_1^{(0)} \\ \theta_2^{(0)} \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, and learning rate $\alpha = 0.25$

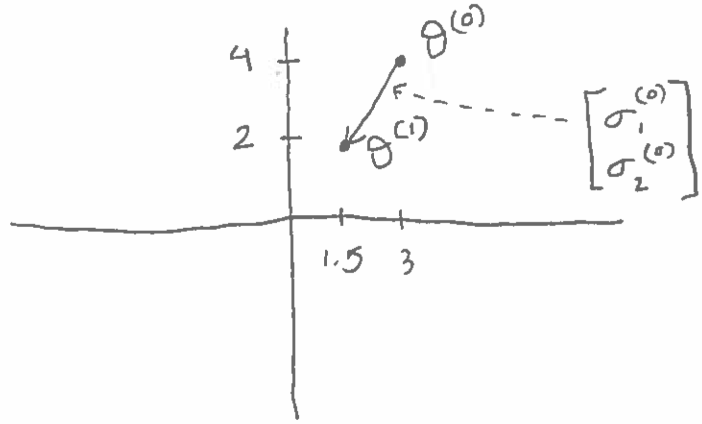
gradient descent for θ_1

- let step $\sigma_1^{(0)} = -\alpha \cdot \frac{\partial L}{\partial \theta_1}(\theta_1^{(0)}, \theta_2^{(0)})$
$= -6\alpha$
$= -1.5$
- let next guess $\theta_1^{(1)} = \theta_1^{(0)} + \sigma_1^{(0)}$
$= 3 - 1.5$
$= 1.5$

gradient descent for θ_2

- let step $\sigma_2^{(0)} = -\alpha \cdot \frac{\partial L}{\partial \theta_2}(\theta_1^{(0)}, \theta_2^{(0)})$
$= -8\alpha$
$= -2$
- let next guess $\theta_2^{(1)} = \theta_2^{(0)} + \sigma_2^{(0)}$
$= 4 - 2$
$= 2$

next guess $\theta^{(1)} = \begin{bmatrix} 1.5 \\ 2 \end{bmatrix}$



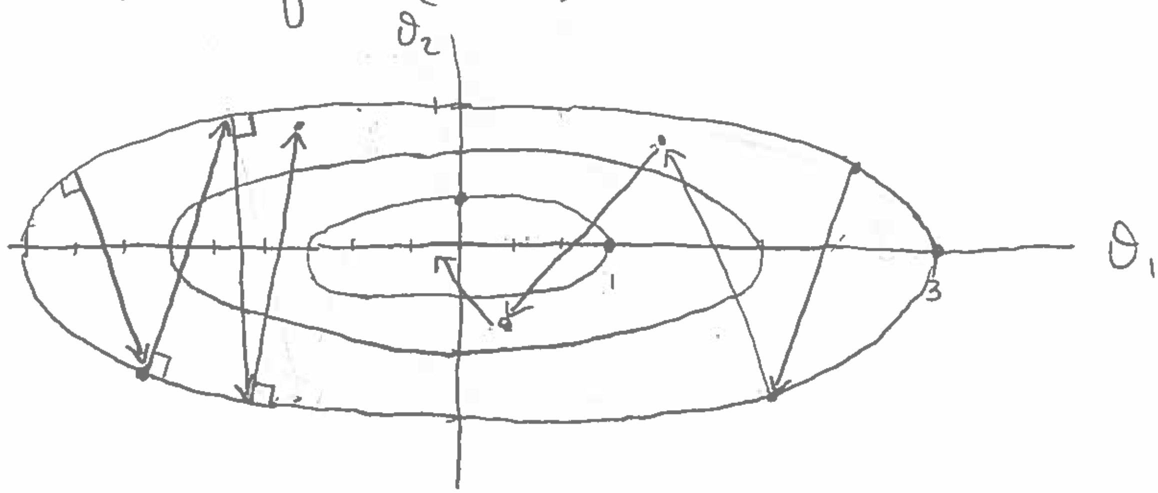
GRADIENT DESCENT: NOW IN 2D!

(16) This "single variable gradient descent in parallel" interpretation is true even for the fancy variants. For instance, multivariate gradient descent with momentum becomes:

> GDWithMomentum (loss L, learning rate α, momentum rate μ):
> initialize $\theta^{(0)} = \begin{bmatrix} \theta_1^{(0)} \\ \vdots \\ \theta_d^{(0)} \end{bmatrix}$; $t = 0$; $\sigma^{(-1)} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$
> repeat until happy:
> - let next step $\sigma^{(t)} = \mu \sigma^{(t-1)} - \alpha \cdot \nabla L(\theta^{(t)})$
> - let next guess $\theta^{(t+1)} = \theta^{(t)} + \sigma^{(t)}$
> - let $t \leftarrow t+1$

---

(17) This gives interesting behavior when visualized. Consider the contour plot of $L(\theta_1, \theta_2) = \theta_1^2 + 9\theta_2^2$



If we start vanilla GD from the left side, it tends to make non-optimal moves, where momentum (started on the right side) "speeds up" as it continues to step in the western direction along the θ_1-axis.

Gradient Descent: Now in 2D!

(18) For reference, here are the multivariable versions of AdaGrad and RMSprop:

AdaGrad(loss L, init learning rate α, tiny delta $\delta > 0$):

initialize $\theta^{(0)} = \begin{bmatrix} \theta_1^{(0)} \\ \vdots \\ \theta_d^{(0)} \end{bmatrix}$; $t \leftarrow 0$

here, division and square root are applied elementwise

repeat until happy:

- let learning rate $\alpha^{(t)} \leftarrow \dfrac{\alpha}{\delta + \sqrt{\sum_{t'=0}^{t} \nabla L(\theta^{(t')}) \odot \nabla L(\theta^{(t')})}}$

- let update $\sigma^{(t)} \leftarrow -\alpha^{(t)} \odot \nabla L(\theta^{(t)})$
- let next guess $\theta^{(t+1)} \leftarrow \theta^{(t)} + \sigma^{(t)}$
- let $t \leftarrow t+1$

this is called Hadamard product:

$$\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \odot \begin{bmatrix} y_1 \\ \vdots \\ y_d \end{bmatrix} = \begin{bmatrix} x_1 y_1 \\ \vdots \\ x_d y_d \end{bmatrix}$$

Gradient Descent: Now in 2D!

(19) RmsProp (loss L, init learning rate α, decay rate β, tiny delta δ):

initialize $\theta^{(0)} \leftarrow \begin{bmatrix} \theta_1^{(0)} \\ \vdots \\ \theta_d^{(0)} \end{bmatrix}$; $t \leftarrow 0$; $m^{(-1)} \leftarrow 0$

repeat until happy:

- let $m^{(t)} \leftarrow \beta m^{(t-1)} + (1-\beta) \cdot \left[\nabla L(\theta^{(t)}) \odot \nabla L(\theta^{(t)}) \right]$
- let learning rate $\alpha^{(t)} \leftarrow \dfrac{\alpha}{\sqrt{\delta + m^{(t)}}}$
- let update $\sigma^{(t)} \leftarrow -\alpha^{(t)} \odot \nabla L(\theta^{(t)})$
- let next guess $\theta^{(t+1)} \leftarrow \theta^{(t)} + \sigma^{(t)}$
- let $t \leftarrow t+1$