Na prethodnim predavanjima, radili smo sledeće algoritme: **“osnovni algoritam najstrmijeg pada”** (eng. *vanila steepest descent*, *vanila SD*), **algoritam najbržeg pada sa momentom** (eng. *SD with momentum*, *momentum SD*), **ubrzani algoritam Nesterov** (eng. *Nesterov SD*).

Svi ovi algoritmi imaju isti tok (zapravo svi “gradijenti” algoritmi imaju isti ovakav tok):

1. Inicijalizacija: svodi na izbor jedne početne tačke (početnog pogađanja, početnog rešenja)
2. Ažuriranje tekućeg rešenja: uvek ide kao $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{v}_k$
3. Ispitivanje uslova (kriterijuma) zaustavljanja.

Tačke 1 i 3 su manje-više zajedničke (iste su za sve algoritme ovog tipa). Razlike se kriju (isključivo) u tački 2.

$$
\begin{aligned}
\boldsymbol{v}_k &= \gamma \nabla f(\boldsymbol{x}_k) && \text{običan SD} \\
\boldsymbol{v}_k &= \omega \boldsymbol{v}_{k-1} + \gamma \nabla f(\boldsymbol{x}_k) && \text{SD sa momentom} \\
\boldsymbol{x}_{k,\text{pomoćno}} = \boldsymbol{x}_k - \omega \boldsymbol{v}_{k-1}, \quad \boldsymbol{v}_k &= \omega \boldsymbol{v}_{k-1} + \gamma \nabla f(\boldsymbol{x}_{k,\text{pomoćno}}) && \text{ubrani SD Nesterova}
\end{aligned}
$$

Oba osnovna adaptivna algoritma najbržeg pada imaju istu baznu strukturu, prema kojoj se svaka komponenta rešenja menja po sledećem pravilu

$$\boldsymbol{x}_{i,k+1} = \boldsymbol{x}_{i,k} + \underbrace{\frac{\gamma}{\sqrt{G_{i,k} + \varepsilon_1}}}_{\text{efektivna “brzina učenja” (“dužina koraka”)}} \underbrace{[\nabla f(\boldsymbol{x}_k)]_i}_{g_i}$$

$$G_{i,k} = \sum_{\ell=0}^{k} g_{i,\ell}^2 \qquad \text{ADAGRAD}$$

$$G_{i,k} = \omega G_{i,k-1} + (1-\omega) g_{i,k}^2 \qquad \text{RMSprop}$$

$$G_{i,0} = 0, \quad G_{i,1} = (1-\omega) g_{i,1}^2, \quad G_{i,2} = \omega(1-\omega) g_{i,1}^2 + (1-\omega) g_{i,2}^2, \ldots.$$

$$G_{i,k} = (1-\omega) \sum_{\ell=0}^{k} \omega^{\ell} g_{i,k-\ell}^2 = (1-\omega) \sum_{\ell=0}^{k} \omega^{k-\ell} g_{i,\ell}^2$$

Smisao ovog rekurentnog (rekurzivnog) načina sračunavanja G leži u tome da se veći značaj (relativno) da vrednostima gradijenta koje smo sračunali u ovoj i neposredno prethodnim iteracijama, a da se relativno mali značaj da onim vrednostima gradijenta koje smo "sreli" u "dalekoj prošlosti" (pre većeg broja iteracija).

Ova promena čini RMSprop algoritam daleko "responsivnijim" u odnosu na ADAGRAD: ADAGRAD se prilagođava "srednjem ponašanju kriterijuma optimalnosti", dok RMSprop čini to isto, ali on usrednjavanje efektivno vrši samo nad "skorijom istorijom".

INTERMEZZO. $G(s) = \frac{1}{\mathrm{sT}+1} \Rightarrow \dot{y}T + y = u \Rightarrow \frac{y_{k+1} - y_k}{\Delta T} T + y_k = u_k$

$$y_{k+1}T = y_k T - \Delta T y_k + \Delta T u_k \Rightarrow y_{k+1} = \underbrace{\frac{T - \Delta T}{T}}_{\omega} y_k + \underbrace{\frac{\Delta T}{T}}_{1-\omega} u_k$$

$$y_{k+1} = \omega y_k + (1 - \omega) u_k$$

ADAM algoritam jednovremeno koristi ideje vezane za uvođenje momenta (odnosno da je trenutni pravac pretrage uslovljen ne samo tekućom vrednošću gradijenta, već i njegovim prethodnim vrednostima) i RMSprop algoritma (odnosno adaptacije efektivne dužine koraka na osnovu kvadrata gradijenta po osi)

Vrlo grubo posmatrano, možemo pisati:

ADAM = MOMENT + RMSprop

Kod ADAM-a, ono što smo ranije zvali G_i sada obeležavamo sa v_i (a inače je ista stvar), a ono što smo ranije obeležavali sa v sada obeležavamo sa m (manje-više ...)

$$\begin{aligned} \boldsymbol{v}_k &= \omega \boldsymbol{v}_{k-1} + \gamma \nabla f(\boldsymbol{x}_k) && \text{SD sa momentom (osnovni)} \\ m_{i,k} &= \omega m_{i,k-1} + (1-\omega) \nabla f(\boldsymbol{x}_k), \ldots \boldsymbol{v}_k = \gamma \boldsymbol{m}_k && \text{ADAM} \end{aligned}$$