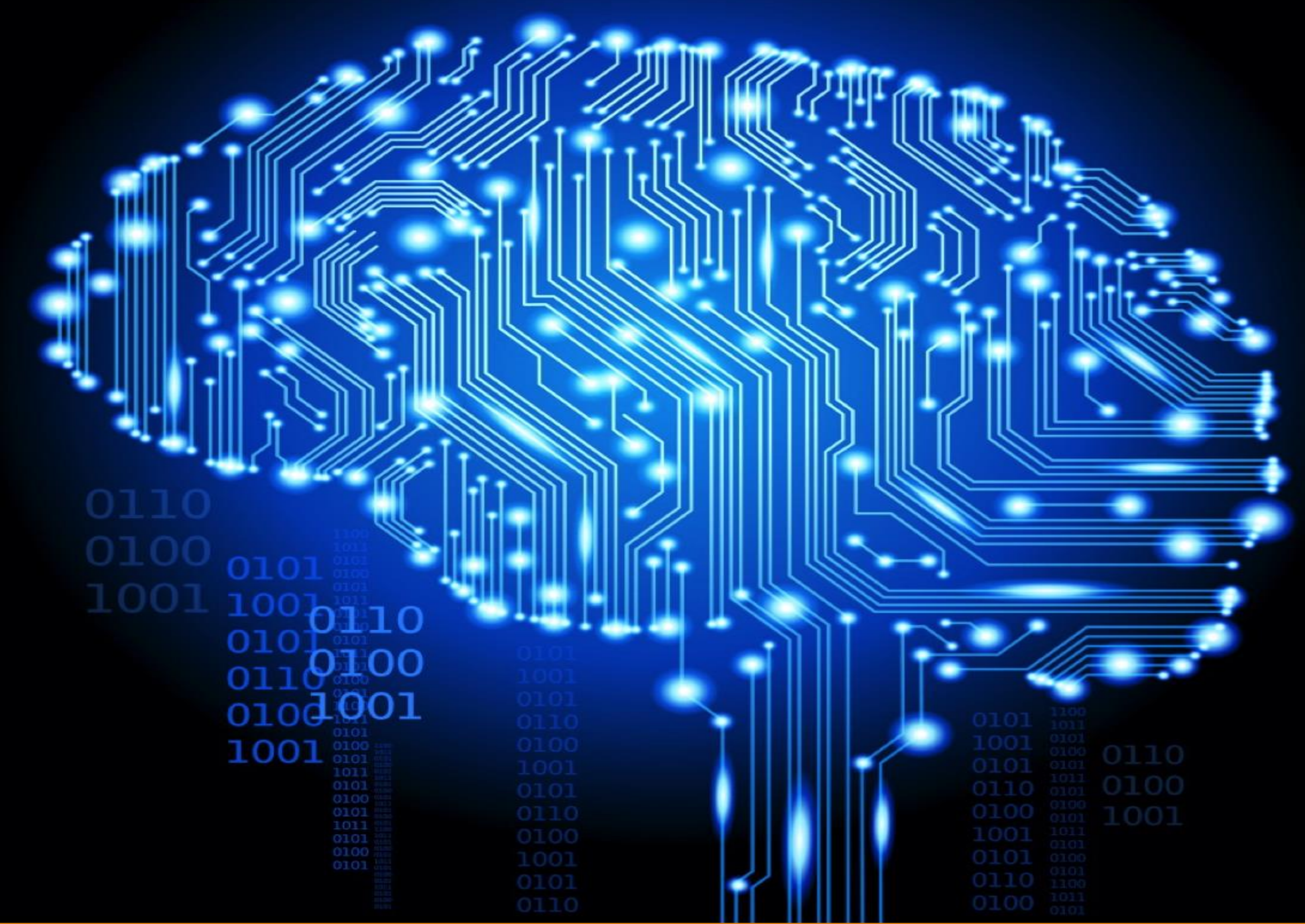


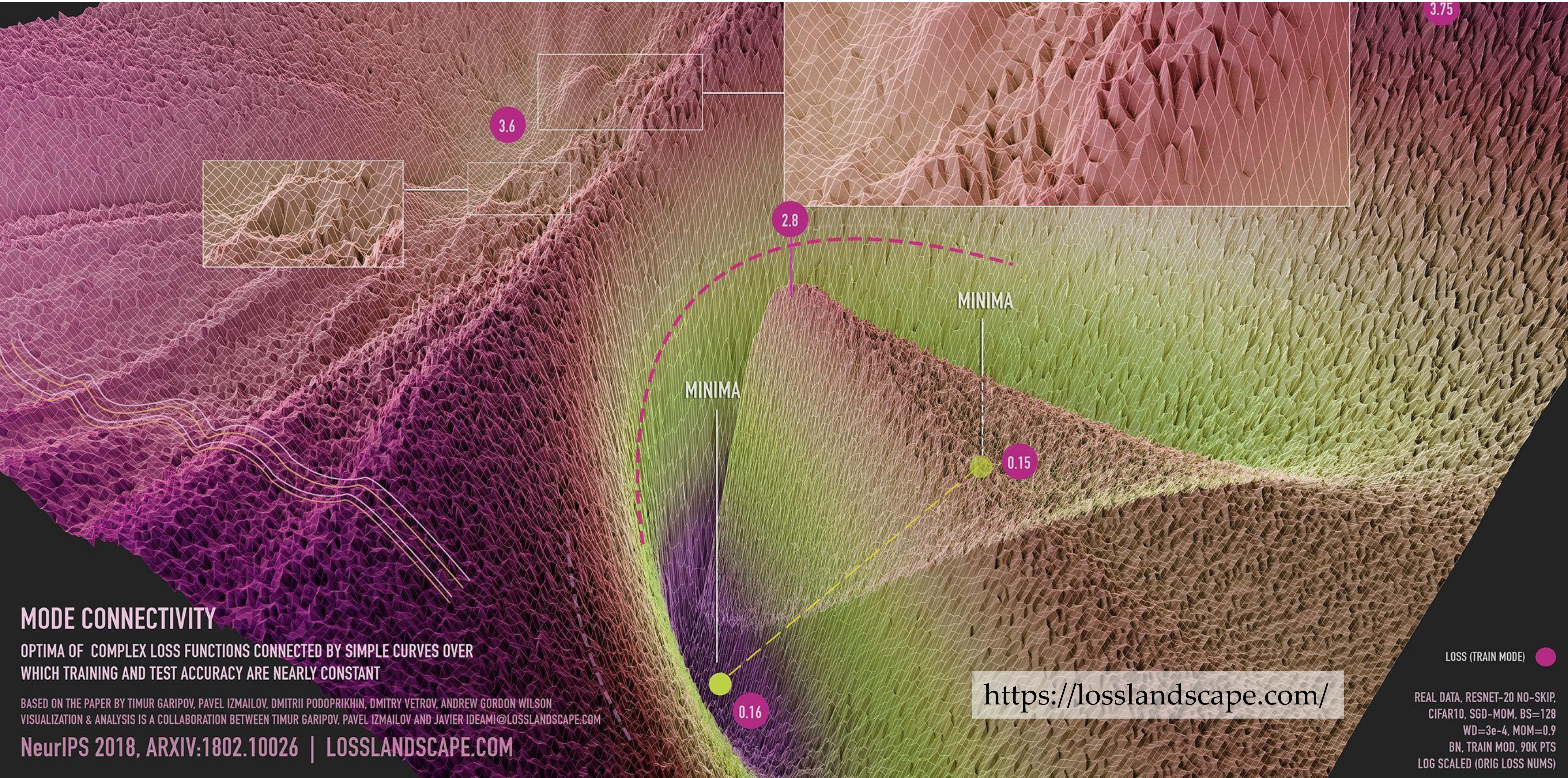
Lecture 3: Deep Learning Optimizations

Deep Learning @ UvA

Lecture overview

- Advanced optimizers
- Initialization
- Normalization
- Regularization
- Hyperparameters

Stochastic gradient descent: $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \frac{d\mathcal{L}}{d\boldsymbol{w}}$



Challenges in optimization

- Ill conditioning → a strong gradient might not even be good enough
- Local optimization is susceptive to local minima
- Plateaus, cliffs and pathological curvatures
- Vanishing and exploding gradients
- Long-term dependencies

Ill-conditioning

- We can analyze possible behaviors of the neural network loss function
  - Resort to the 2nd order Taylor dynamics around the current weight $\mathbf{w}'$

$$\mathcal{L}(\boldsymbol{w}) = \mathcal{L}(\boldsymbol{w}') + \boldsymbol{g}(\boldsymbol{w} - \boldsymbol{w}') + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}')^{\mathrm{T}}\mathbf{H}(\mathbf{w} - \mathbf{w}') \text{ where } \boldsymbol{g} = \frac{d\mathcal{L}}{d\boldsymbol{w}}$$

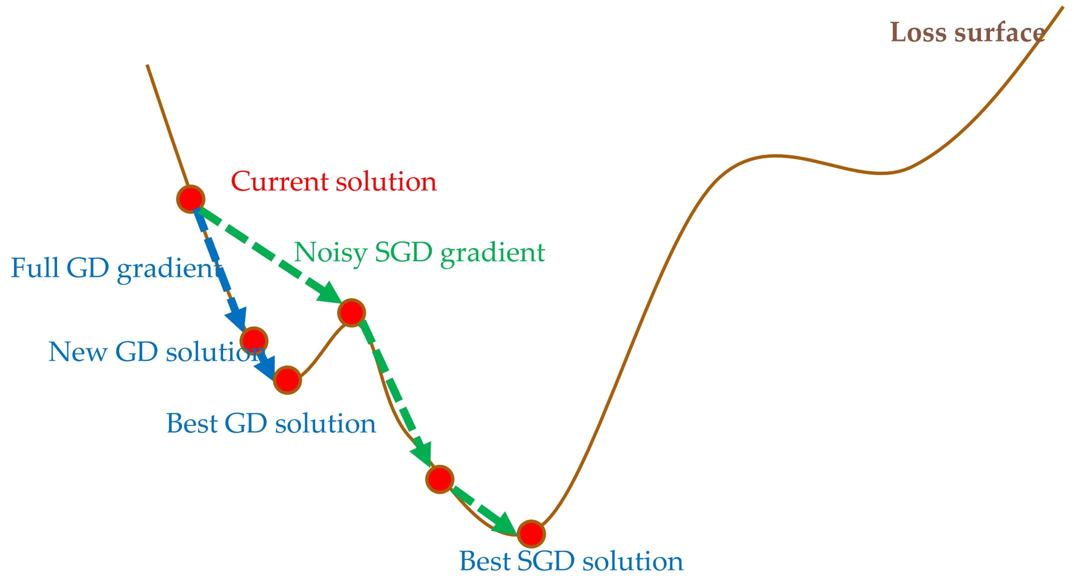
- If we analyze the loss around the current weight $\mathbf{w}'$ plus a small step

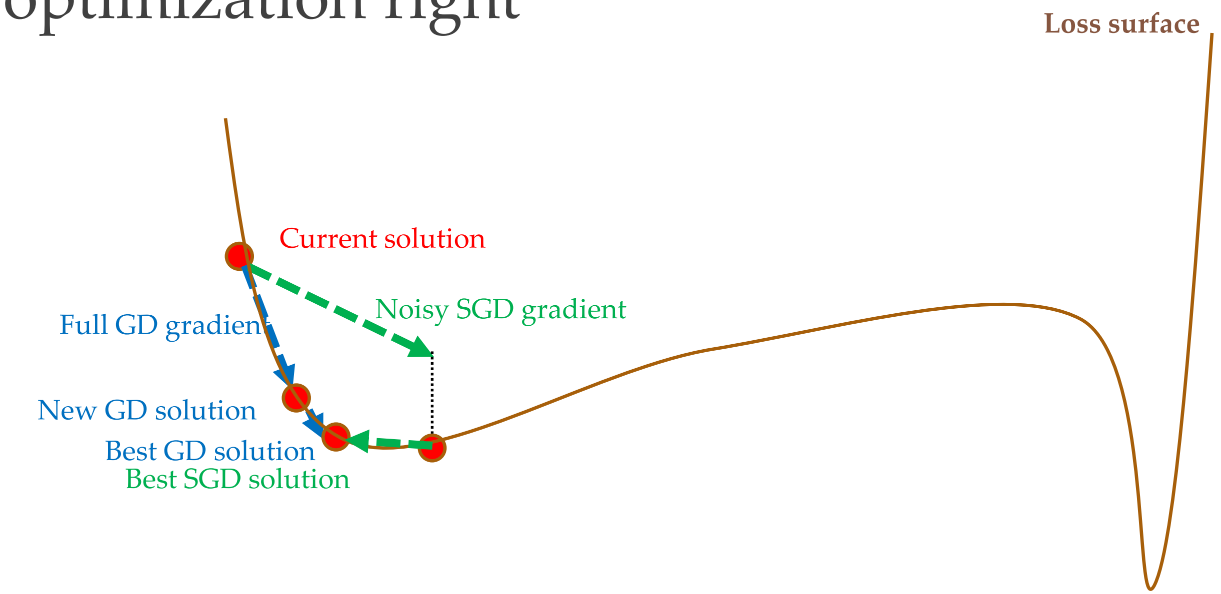
$$\boldsymbol{w} \leftarrow \boldsymbol{w}' - \varepsilon \boldsymbol{g} \Rightarrow \mathcal{L}(\boldsymbol{w}' - \varepsilon \boldsymbol{g}) \approx \mathcal{L}(\boldsymbol{w}') - \varepsilon \boldsymbol{g}^{\mathrm{T}}\boldsymbol{g} + \varepsilon^2 \frac{1}{2} \boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g}$$

- There are cases where $\boldsymbol{g}$ is “strong” but $\varepsilon \frac{1}{2} \boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g} > \boldsymbol{g}^{\mathrm{T}} \boldsymbol{g}$
  - In these cases, the loss would still go higher after we take a gradient step

Local minima

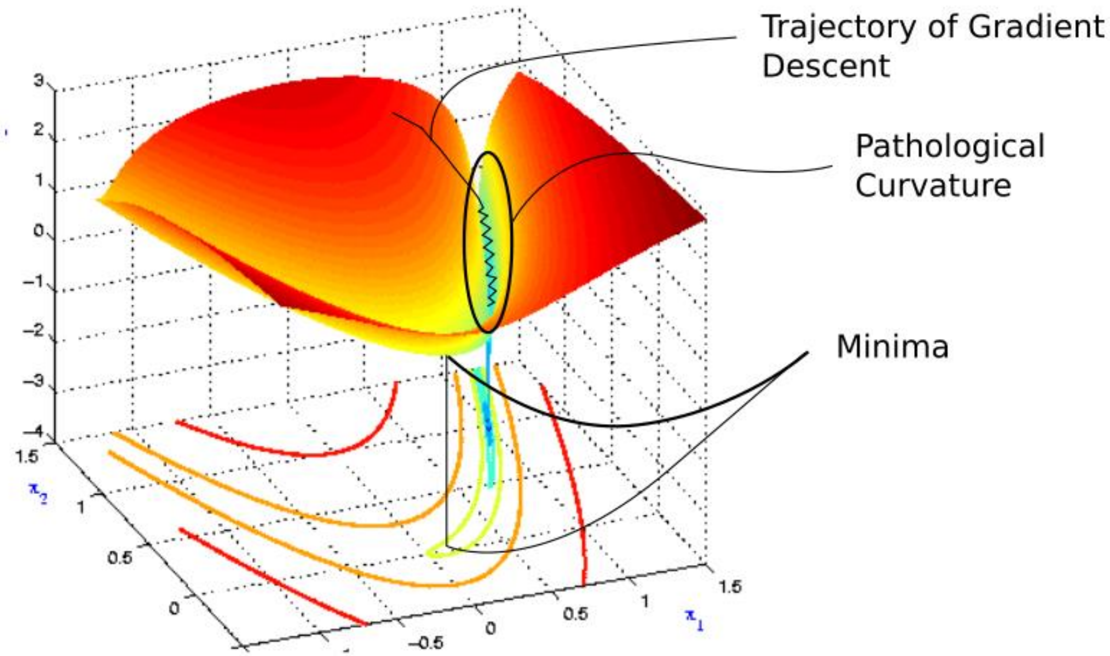
- Stochasticity alone is not always enough to escape local minima
- You must realize that these nice visualization are our own imagination
- In practice, we (and the NNs) are blind of what the landscape really looks like
  - Our best hope is to simply get the optimization right

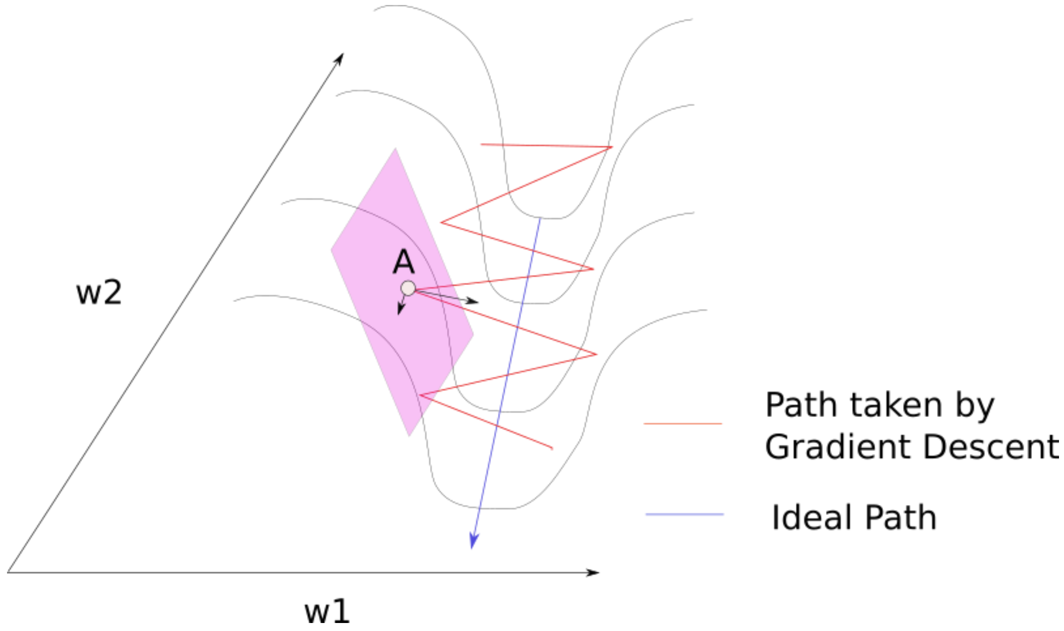




Ravines

- Locations where the gradient is large in one direction and small in another

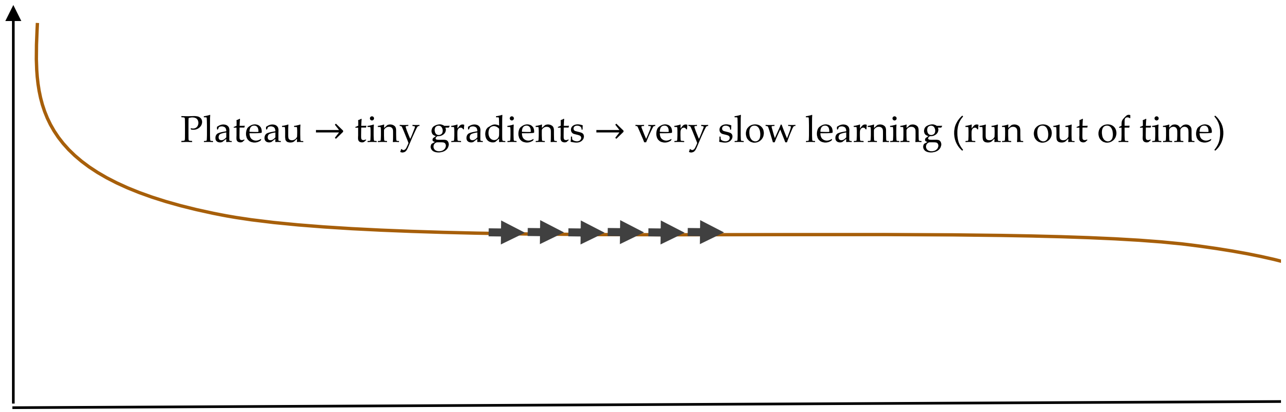




Picture credit: [Team Paperspace]()

Plateuaus/Flat areas

- In flat areas, there is almost zero-gradients → no updates → no learning
- That said, flat areas that are minima generalize well



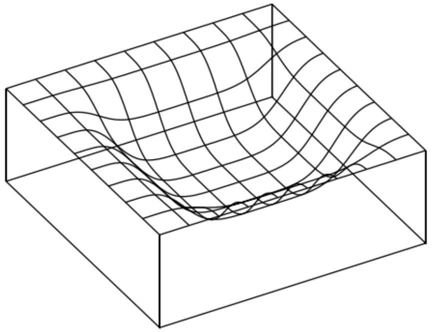


Figure 1: *Example of a "flat" minimum.*

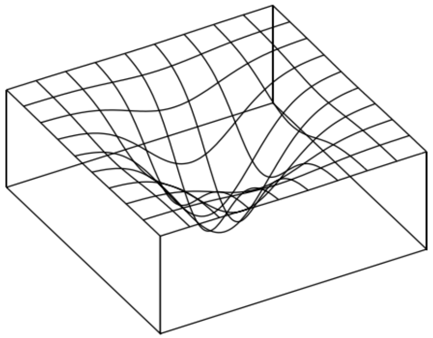
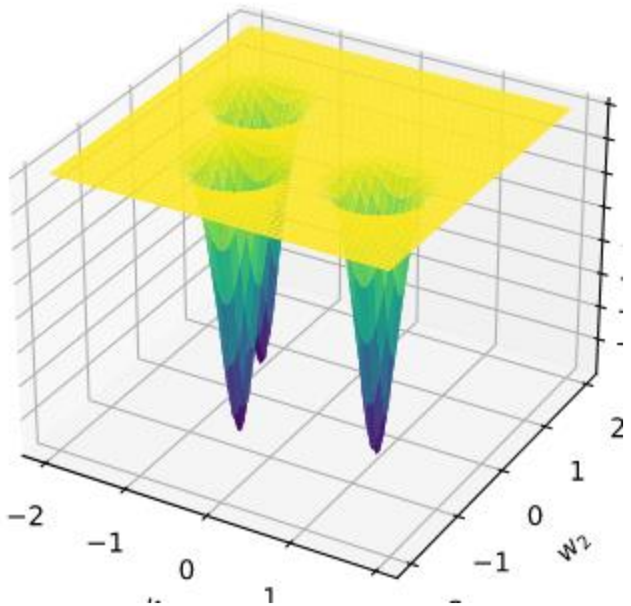


Figure 2: *Example of a "sharp" minimum.*

[Link]

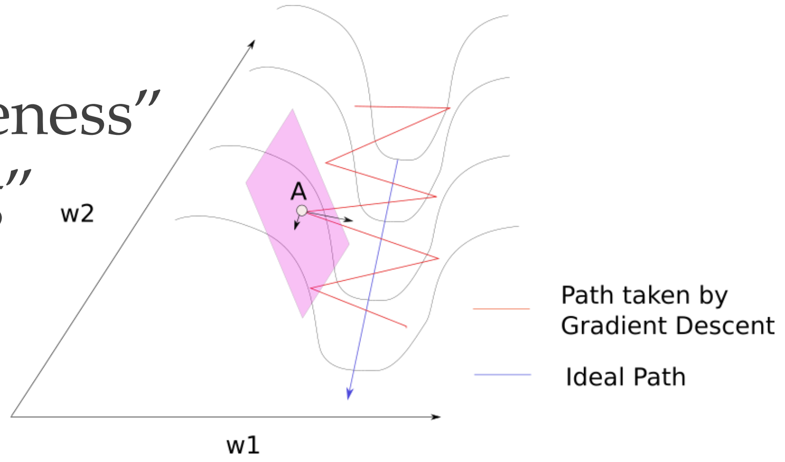
Flat areas, steep minima

- When combining flat areas with very steep minima → very challenging
- How do we even get to the area where the steep minima starts?



Second order optimization

- Normally all weights updated with same "aggressiveness"
  - Often some parameters could enjoy more "teaching"
  - While others are already about there

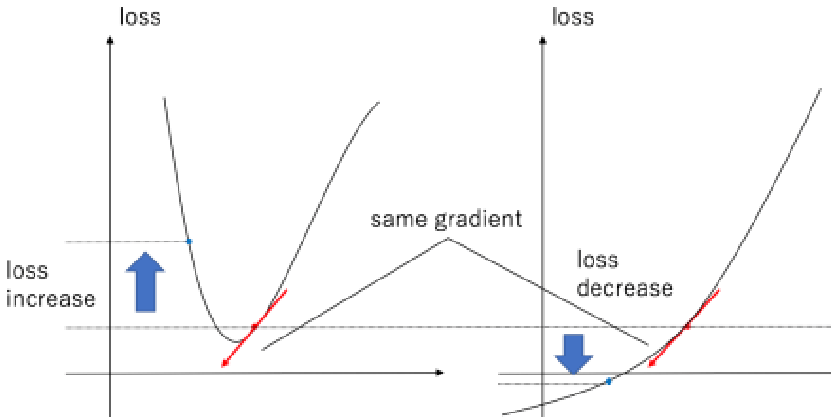


- Adapt learning per parameter

$$w_{t+1} = w_t - H_{\mathcal{L}}^{-1} \eta_t g_t$$

- $H_{\mathcal{L}}$ is the Hessian matrix of $\mathcal{L}$: second-order derivatives

$$H_{\mathcal{L}}^{ij} = \frac{\partial \mathcal{L}}{\partial w_i \partial w_j}$$



Second order optimization methods in practice

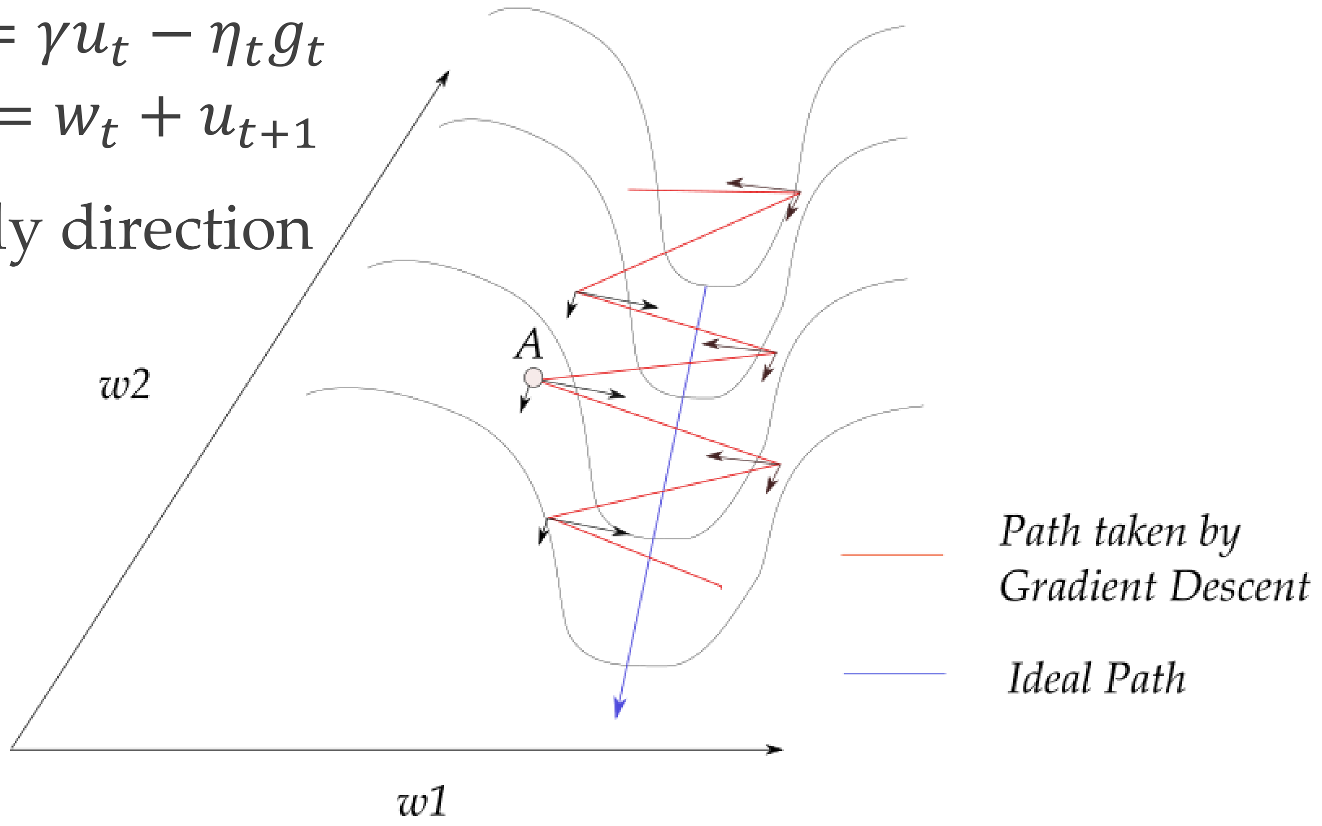
- Inverse of Hessian usually very expensive
  - Too many parameters
- Approximating the Hessian, e.g. with the L-BFGS algorithm
  - Keeps memory of gradients to approximate the inverse Hessian
  - L-BFGS works alright with Gradient Descent. What about SGD?
- In practice, SGD with momentum works just fine quite often

SGD with momentum

- Don't switch update direction all the time
- Maintain "momentum" from previous updates → dampens oscillations

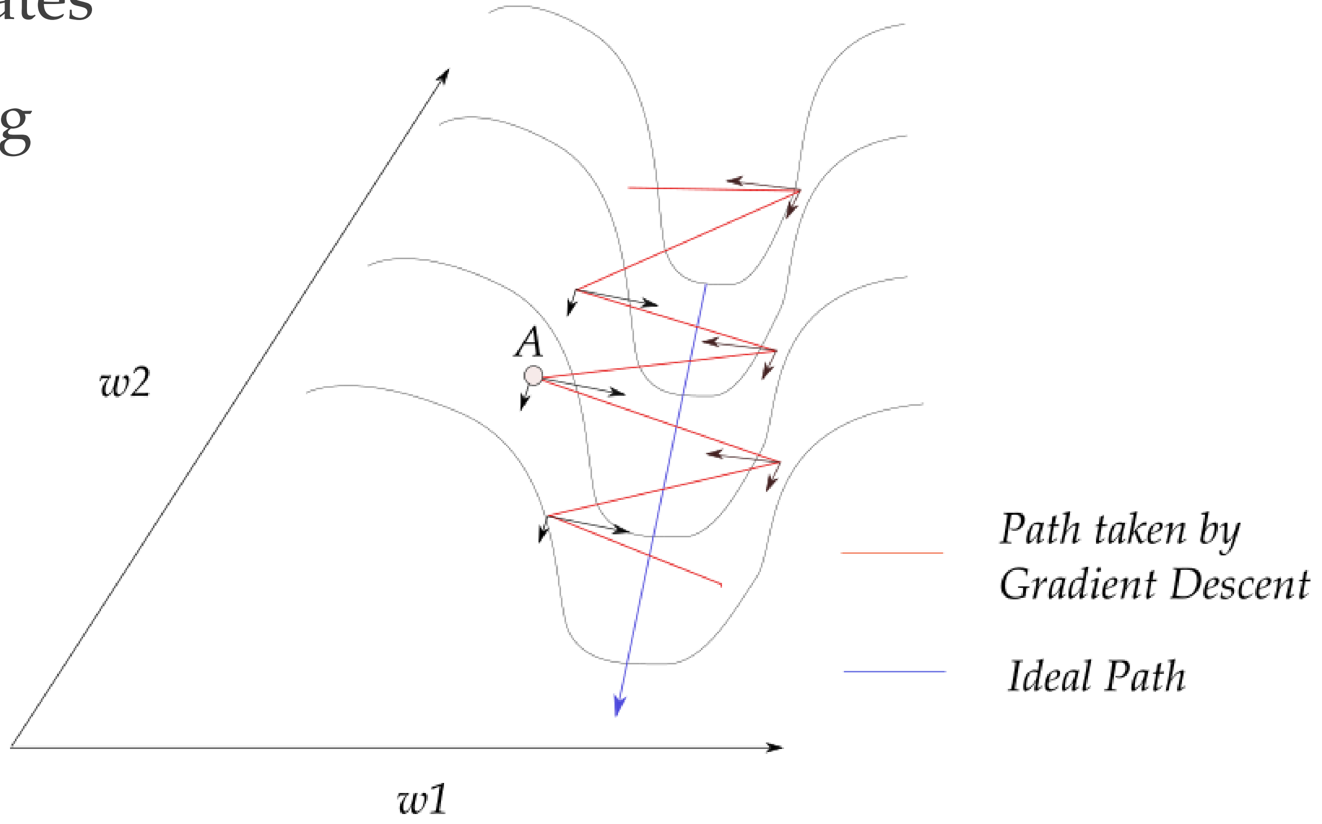
$$u_{t+1} = \gamma u_t - \eta_t g_t$$
$$w_{t+1} = w_t + u_{t+1}$$

- Exponential averaging keeps steady direction
- Example: $\gamma = 0.9$ and $u_0 = 0$
  - $u_1 \propto -g_1$
  - $u_2 \propto -0.9 g_1 - g_2$
  - $u_3 \propto -0.81 g_1 - 0.9 g_2 - g_3$



SGD with momentum

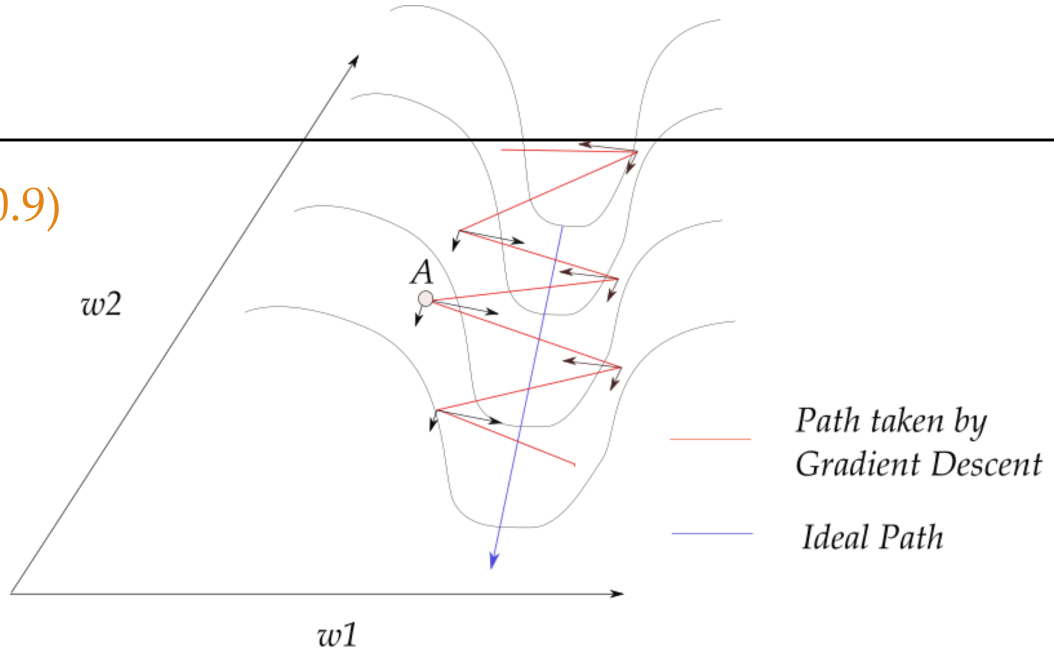
- The exponential averaging
  - cancels out the oscillating gradients
  - gives more weight to recent updates
- More robust gradients and learning
  → faster convergence
- In practice
  - $\gamma = \gamma_0 = 0.5$
  - Anneal to $\gamma_\infty = 0.9$

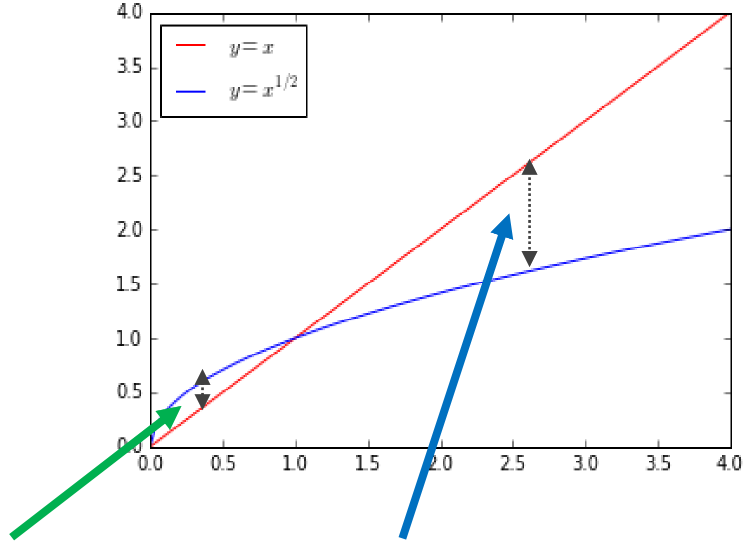


RMSprop

Decay hyper-parameter (usually 0.9)

- Schedule
  - $r_t = \alpha r_{t-1} + (1 - \alpha) g_t^2$
  - $u_t = -\frac{\eta}{\sqrt{r_t} + \varepsilon} g_t$
  - $w_{t+1} = w_t + u_t$
- **Large gradients**, e.g. too "noisy" loss surface
  - Updates are tamed
- **Small gradients**, e.g. stuck in plateau of loss surface
  - Updates become more aggressive
- Sort of performs simulated annealing





Adam [Kingma2014]

- One of the most popular learning algorithms

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$u_t = -\frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t$$
$$w_{t+1} = w_t + u_t$$

  - Recommended values: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$

- Adaptive learning rate as RMSprop, but with momentum & correction bias

Adagrad [Duchi2011]

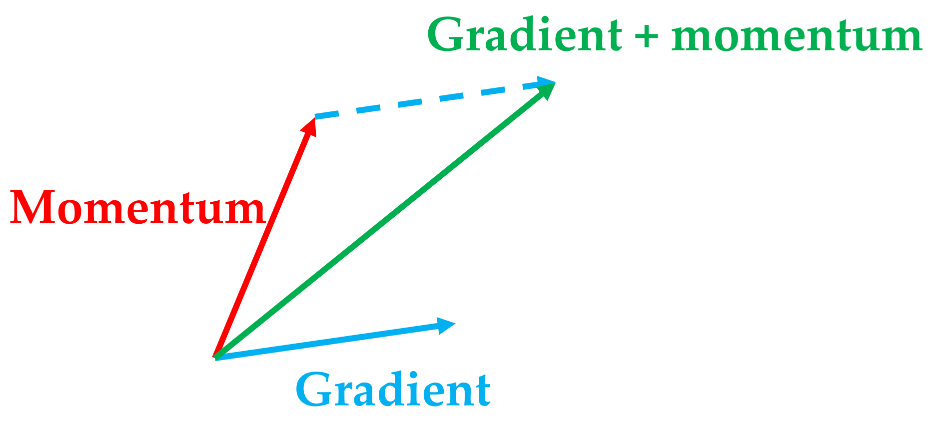
- Schedule
  - $r = \sum_t (\nabla_w \mathcal{L})^2 \implies w_{t+1} = w_t - \eta \frac{g_t}{\sqrt{r} + \varepsilon}$
  - Gradients become gradually smaller and smaller

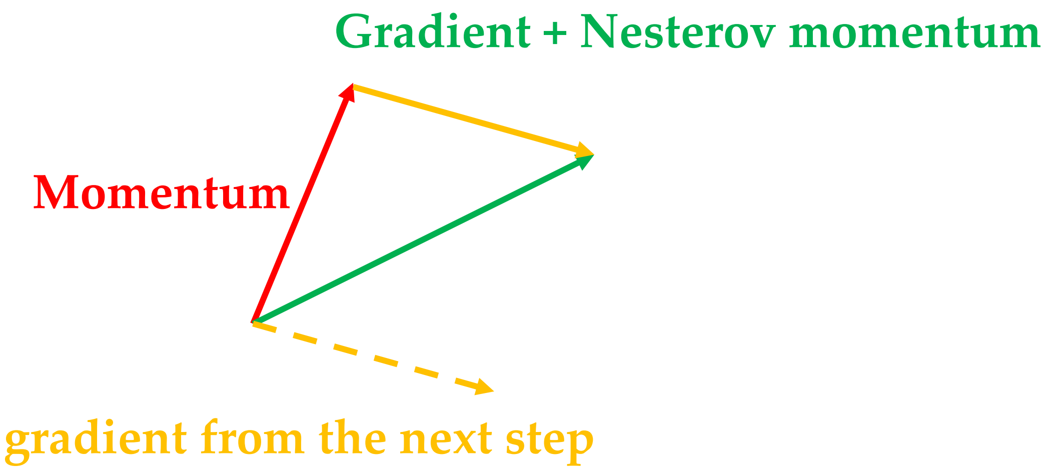
Nesterov Momentum [Sutskever2013]

- Use the future gradient instead of the current gradient

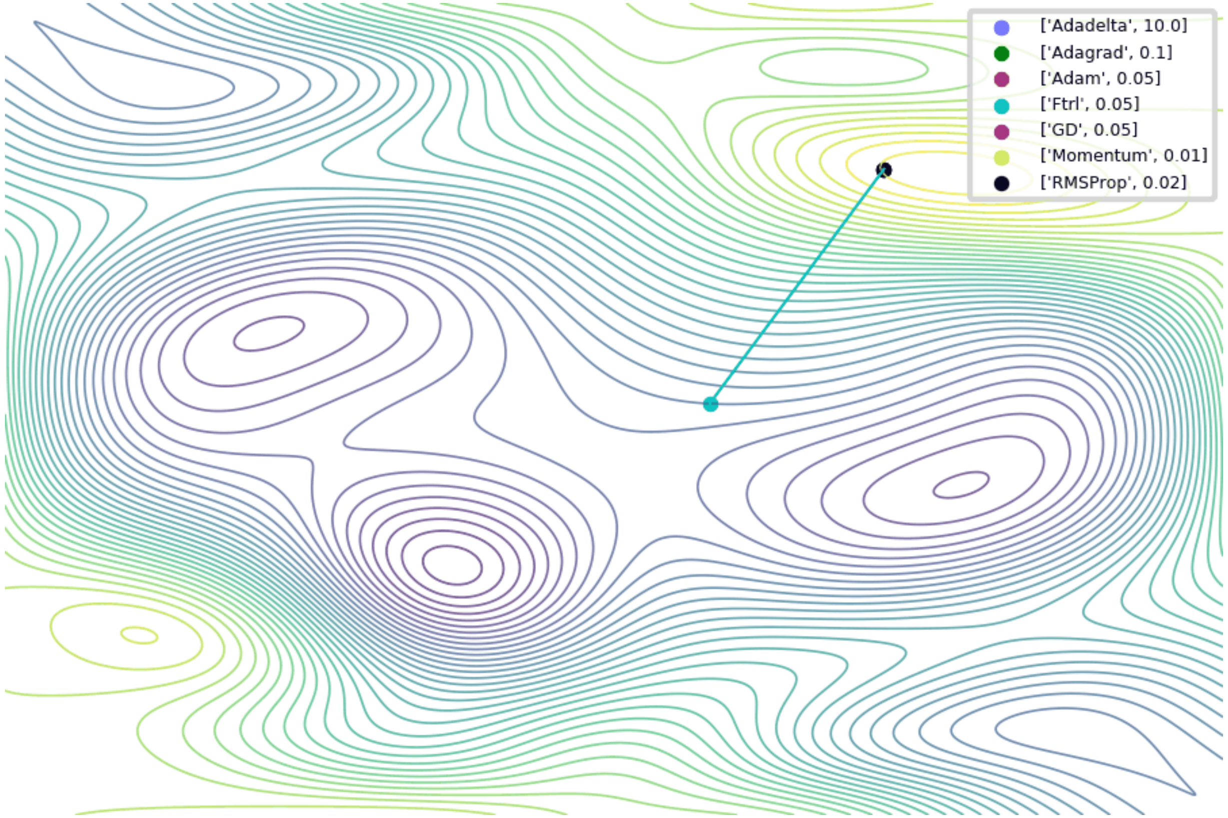
$$w_{t+0.5} = w_t + \gamma u_t$$
$$u_{t+1} = \gamma u_t - \eta_t \nabla_{w_{t+0.5}} \mathcal{L}$$
$$w_{t+1} = w_t + u_{t+1}$$

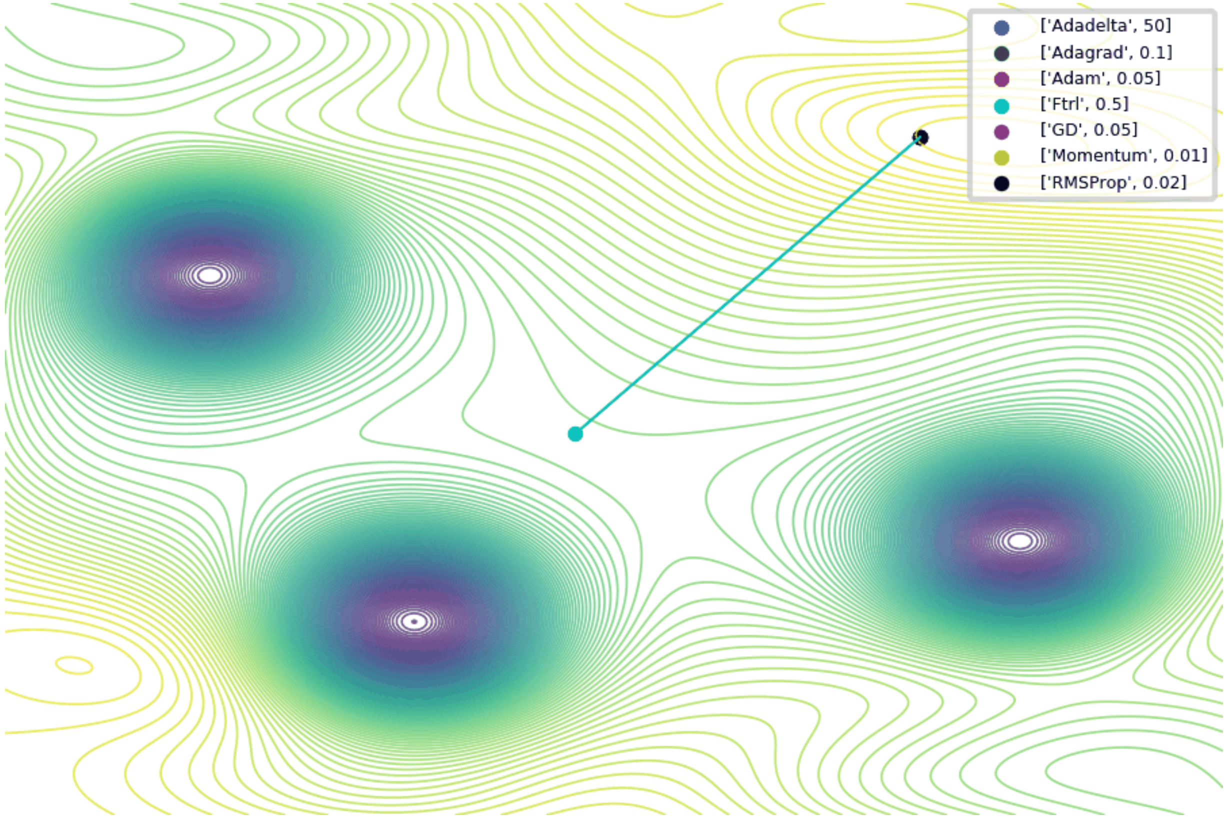
- Better theoretical convergence
- Generally works well with Convolutional Neural Networks





Visual overview





Picture credit: Jaewan Yun

In practice

- SGD works quite well for many cases
- For more complex models Adam is often the preferred choice
- However, Adam tends to "over-optimize"
- If you expect your data to be noisy, Adam might converge to suboptimal
  - Then, SGD with some momentum might work better