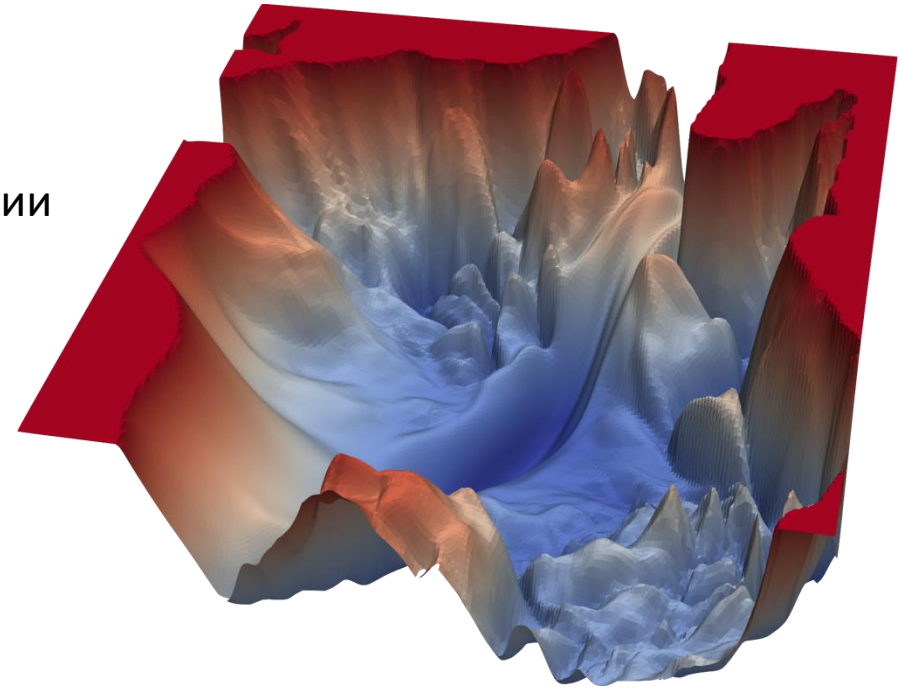
Основы практического использования нейронных сетей.

Лекция 3. Алгоритмы обучения для глубоких НС.

Дмитрий Буряк.
к.ф.-м.н.
dyb04@yandex.ru

Трудности обучения глубоких НС

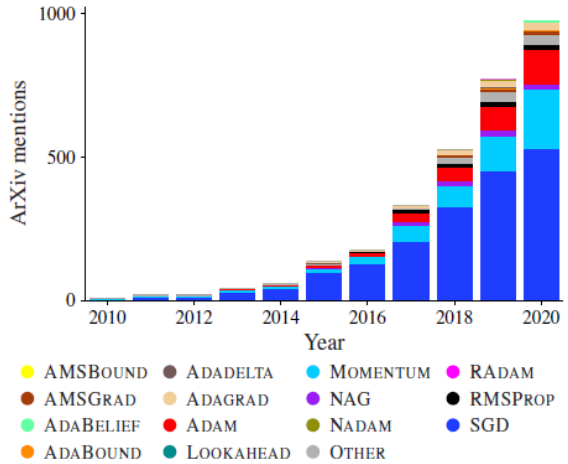
- ❑ Локальные минимумы функции ошибки
- ❑ Сложный ландшафт целевой функции
- ❑ Неравномерность обновления параметров сети
- ❑ Выбор закона изменения скорости обучения



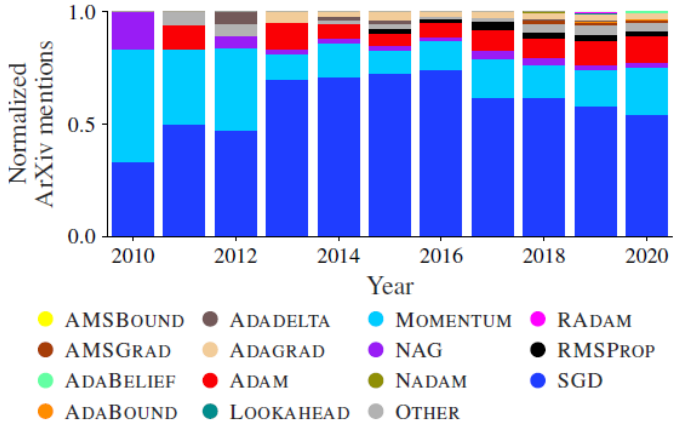
https://www.cs.umd.edu/~tomg/projects/landscapes/

Выбор алгоритма обучения

- ❑ Существует более 100 алгоритмов оптимизации для НС*.
- ❑ Эмпирическое сравнение эффективности различных алгоритмов.
- ❑ Не существует исследований, которые сравнивают все (почти все) алгоритмы.



Количество упоминаний оптимизаторов в статьях на сайте arxiv.org*



Относительное число упоминаний оптимизаторов в статьях на сайте arxiv.org*

** https://arxiv.org/abs/2007.01547*

Стохастический градиентный спуск

- ❑ Основной алгоритм для глубокого обучения.
- ❑ Базируется на принципах градиентного спуска.
- ❑ Вычисление градиента по каждому примеру вычислительно затратно → выбор случайных подмножест из обучающей выборки.

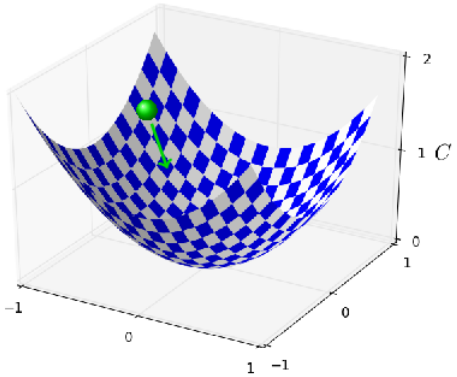
$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \to b'_l = b_l - \eta \frac{\partial C}{\partial b_l}.$$

$$\nabla C \approx \frac{1}{m} \sum_{j=1}^{m} \nabla C_{X_j}$$

$$w_k \to w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \to b'_l = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l},$$

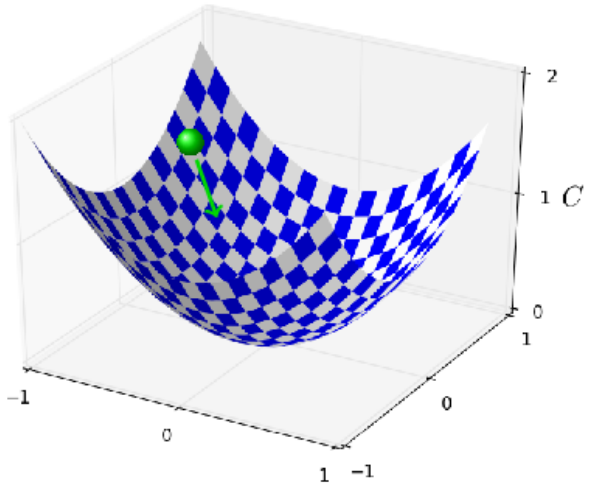


Использование момента

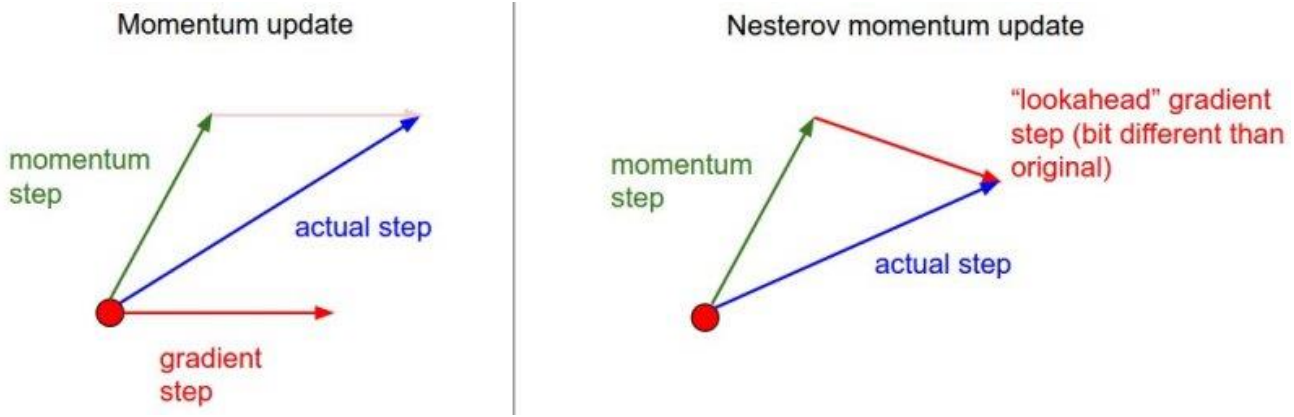
- Идея накопления импульса.
- Оценка «скорости» градиента.
- Применение коэффициента «трения» ~ коэффициент момента.

$$v \rightarrow v' = \mu v - \eta \nabla C$$
$$w \rightarrow w' = w + v'.$$

- Изменение коэффициента момента в процессе обучения 0.5→0.9



Момент Нестерова (NAG)

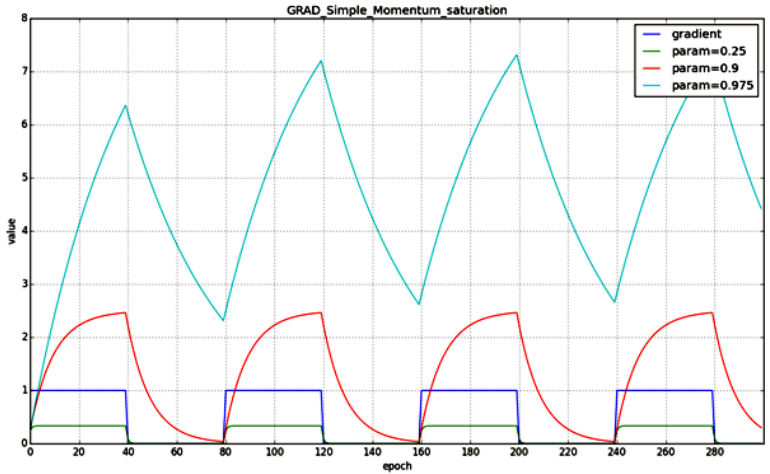


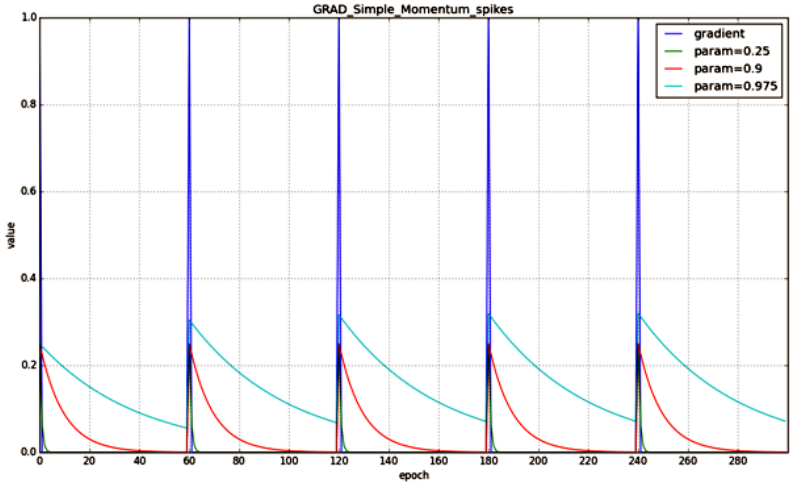
- Оценка градиента в предсказанной позиции функции ошибки

Обзор алгоритмов обучения*. Момент.

$$v_t = \gamma v_{t-1} + \eta g_t \qquad g_t = \nabla_\theta C(\theta)$$

$$\theta_{t+1} = \theta_t - v_t \qquad \gamma = 0.9$$



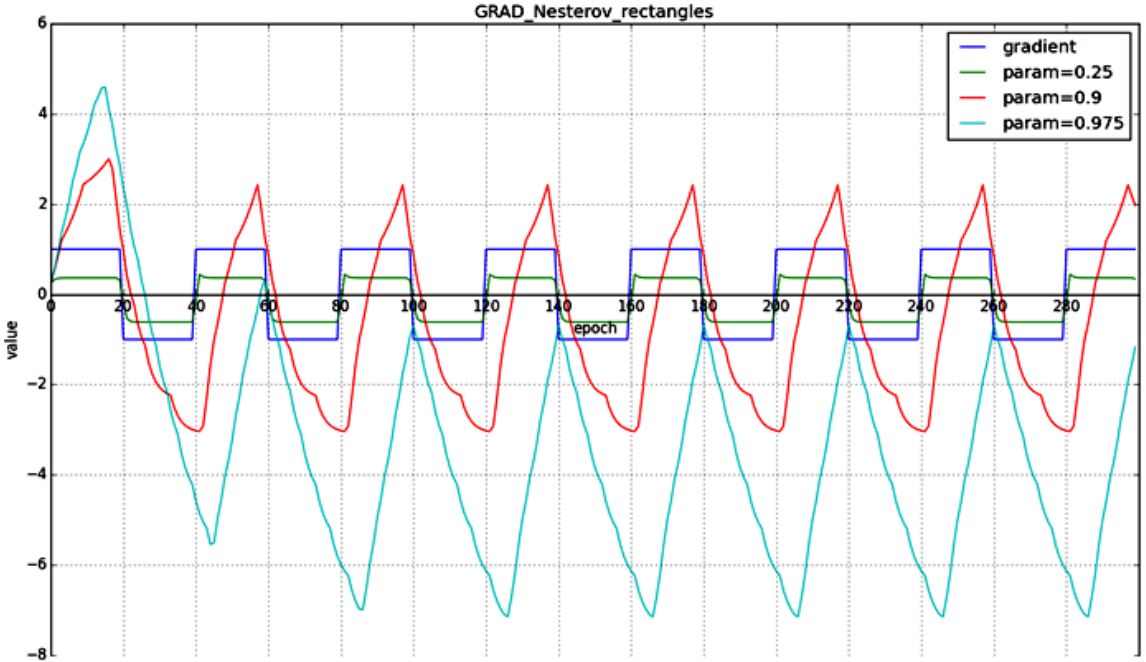


** https://habrahabr.ru/post/318970/*

Обзор алгоритмов обучения. Момент Нестерова (NAG).

$$v_t = \gamma v_{t-1} + \eta g_t(\theta - \gamma v_{t-1})$$

$$\theta_{t+1} = \theta_t - v_t$$



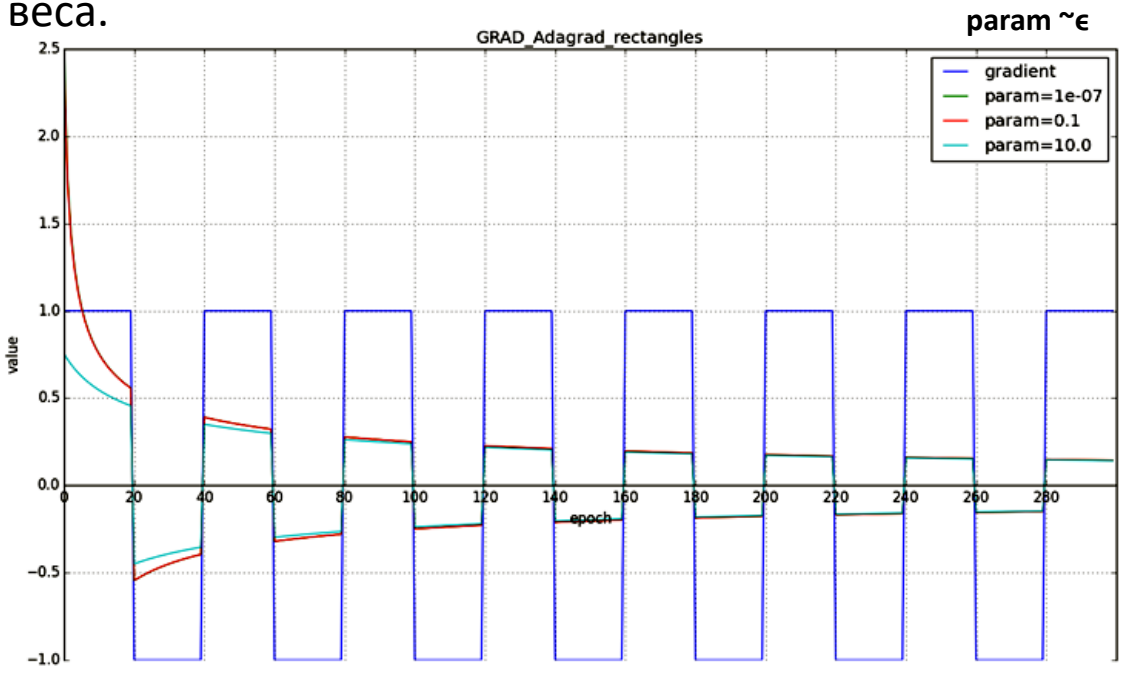
Обзор алгоритмов обучения. Adagrad.

- Учет частоты обновления веса.
- Устойчивость к выбору скорости обучения.

$$G_t = G_t + g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t$$

ϵ — сглаживающий параметр



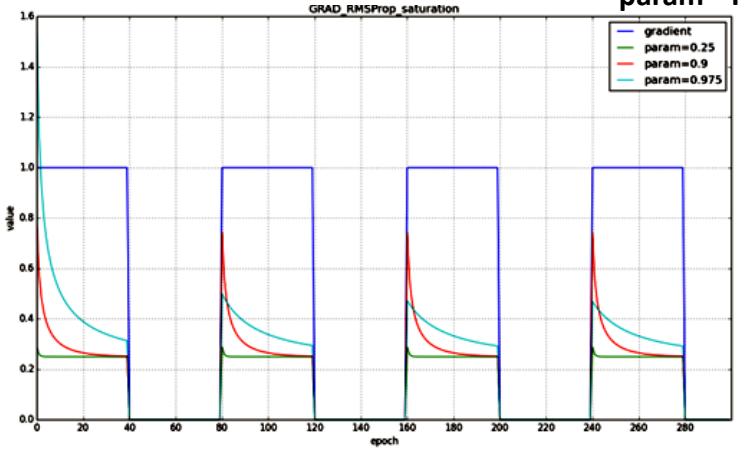
Обзор алгоритмов обучения. RMSProp.

- Учет частоты обновления веса за счет усреднения.

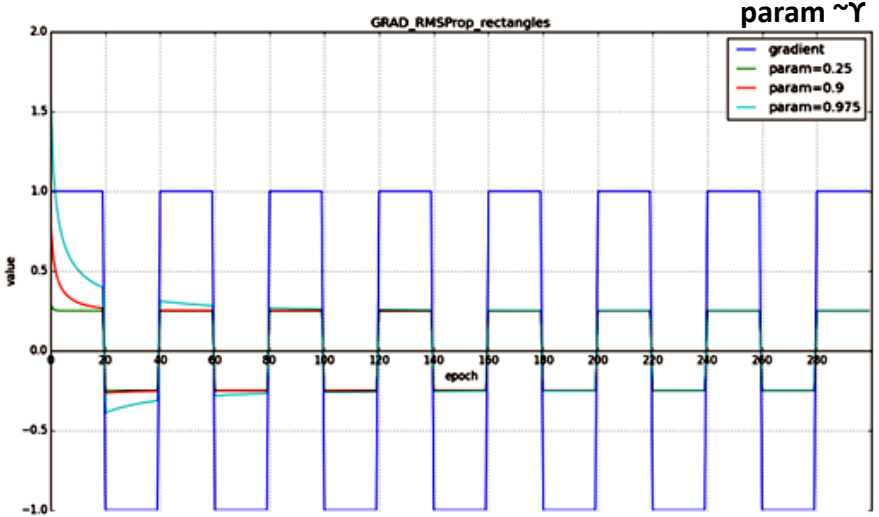
$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

param ~ϒ



param ~ϒ



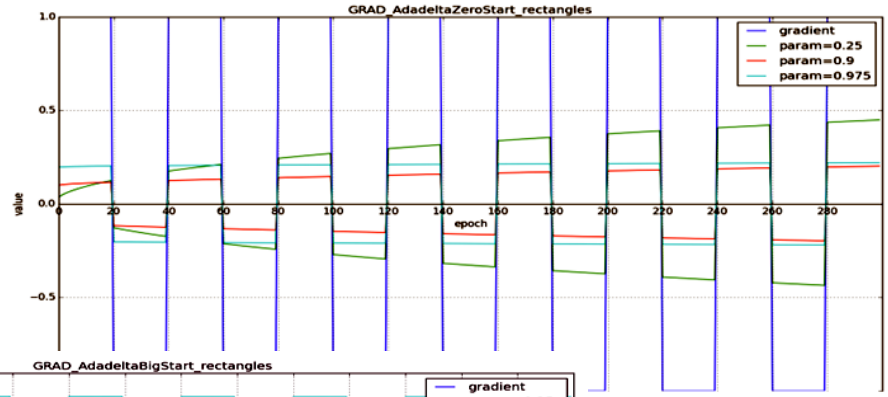
Обзор алгоритмов обучения. Adadelta.

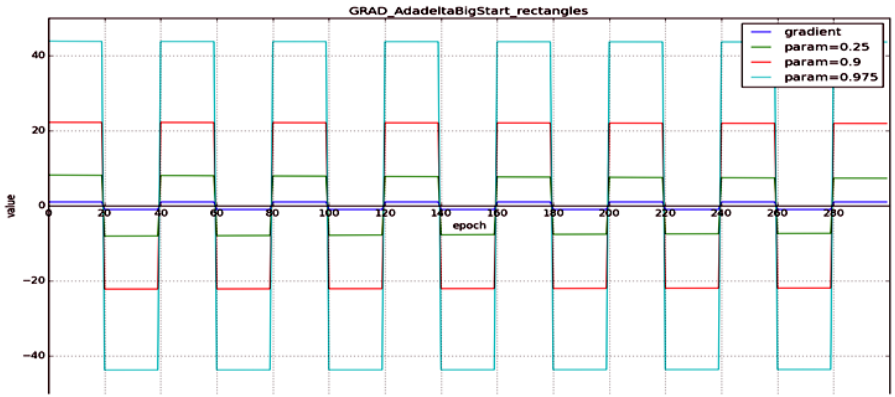
$$\Delta\theta = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

$$\theta_{t+1} = \theta_t - \frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2$$

$$RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon}$$





param ~ϒ

Обзор алгоритмов обучения. Adam.

- Момент + учет частоты обновления веса.

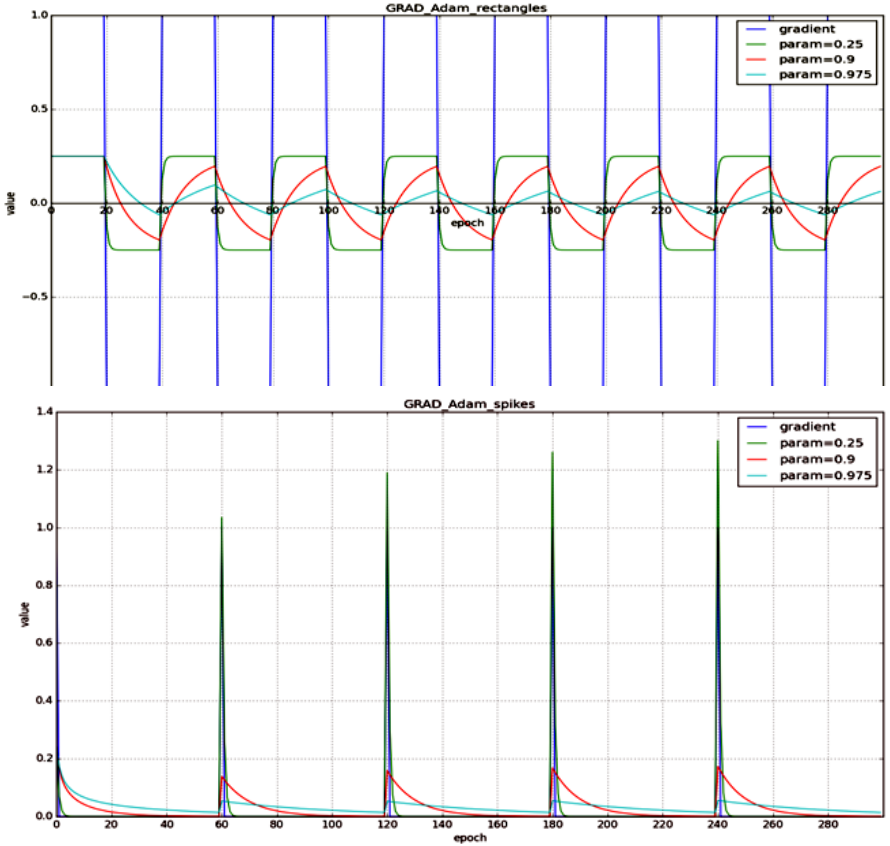
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \; \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

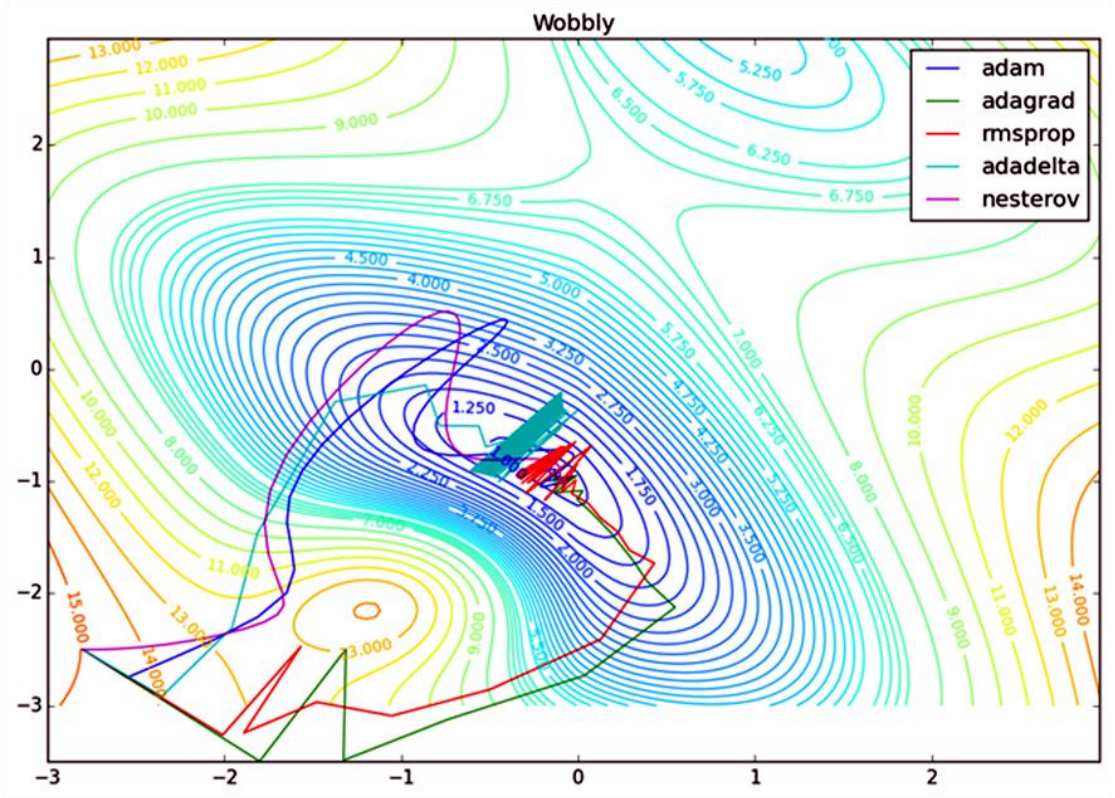
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t$$

$$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$$

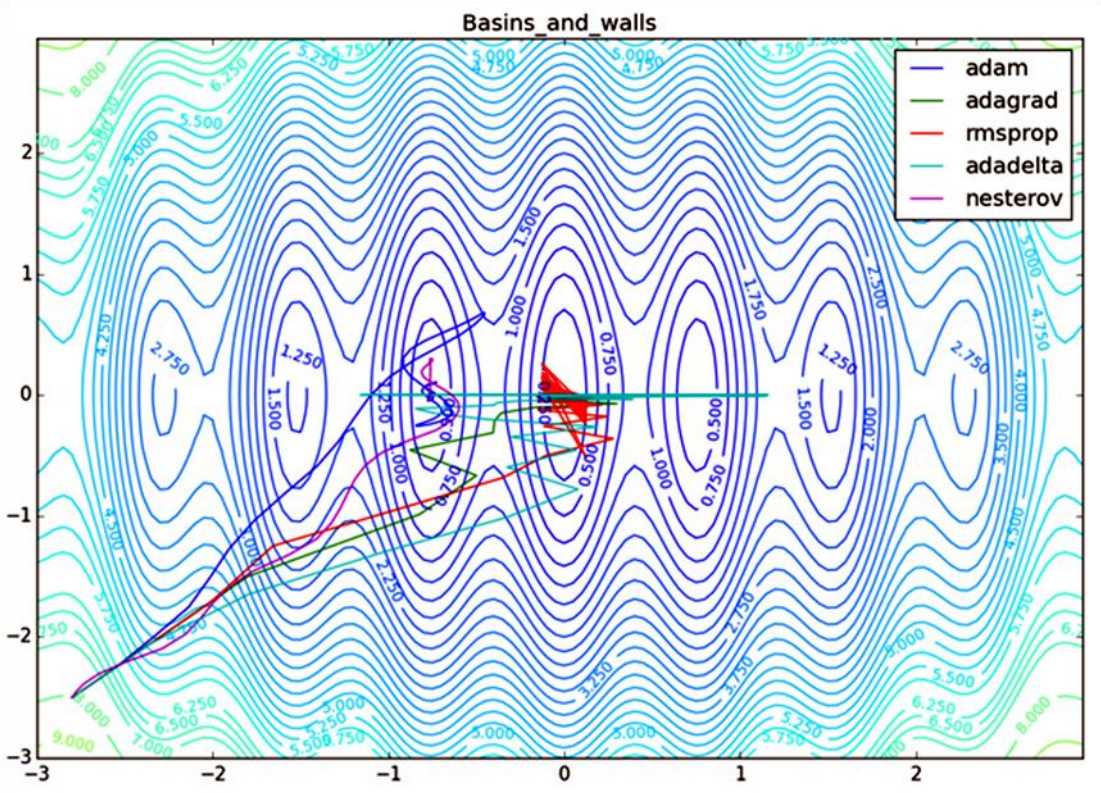


param ~β_2

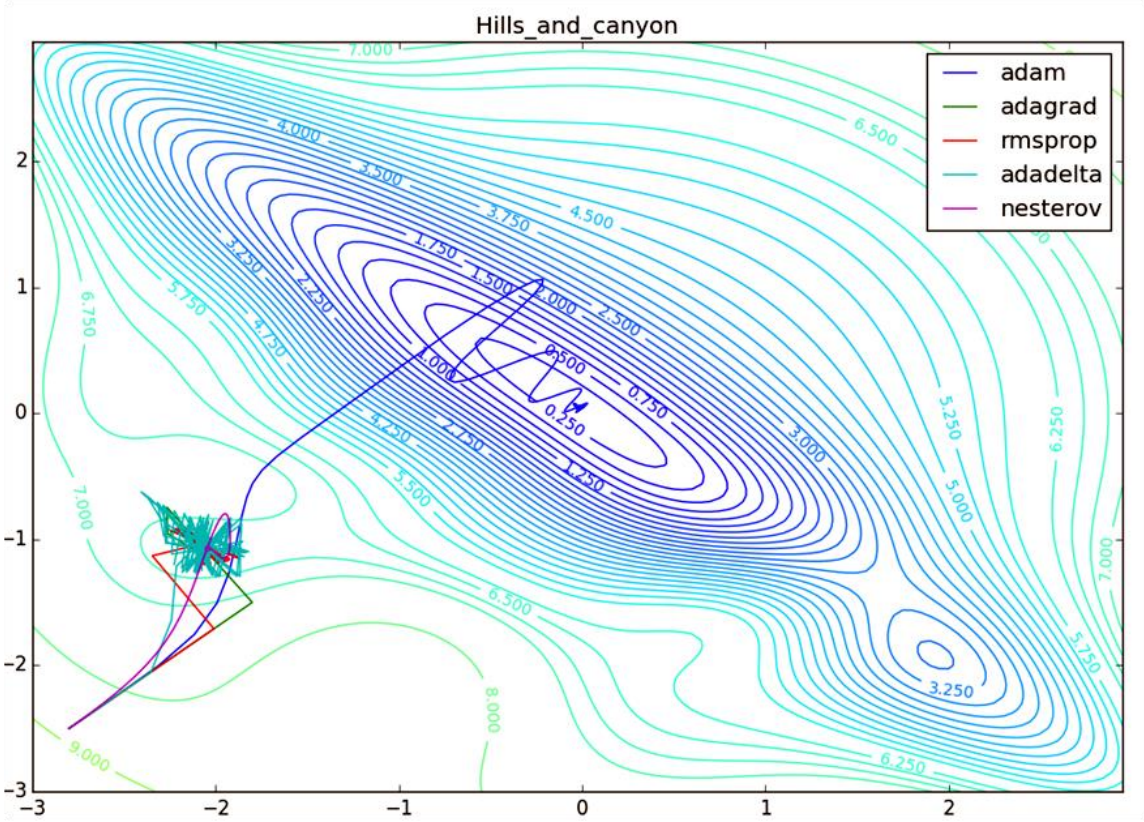
Обзор алгоритмов обучения. Эксперименты (1).



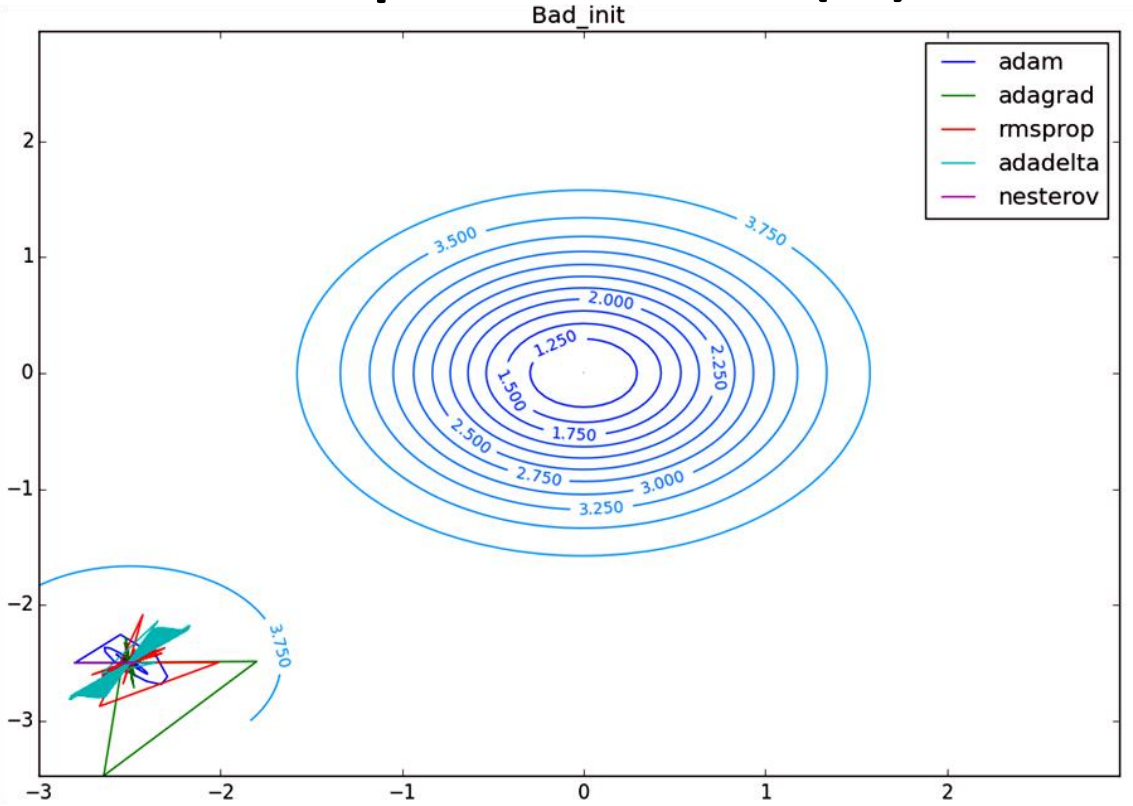
Обзор алгоритмов обучения. Эксперименты (2).



Обзор алгоритмов обучения. Эксперименты (3).



Обзор алгоритмов обучения. Эксперименты (4).



Алгоритм обучения AdaBelief.

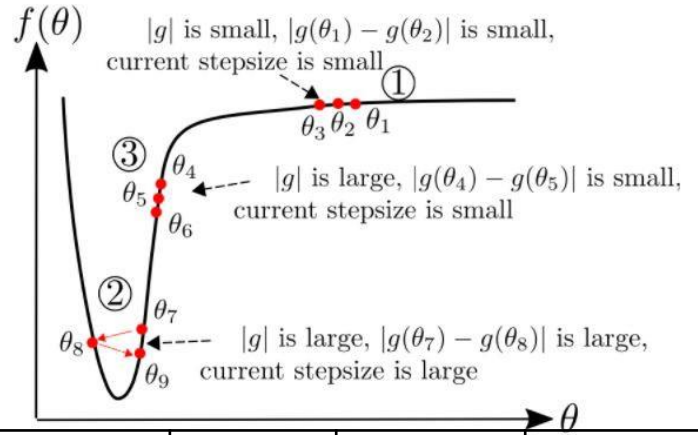
- AdaBelief : Adapting Stepsizes by the Belief in Observed Gradients (2020)
  - Высокая скорость сходимости;
  - Хорошая обобщающая способность

Algorithm 1: Adam Optimizer

Initialize $\theta_0, m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0$
While θ_t not converged
$t \leftarrow t+1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
Bias Correction
$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}, \widehat{v_t} \leftarrow \frac{v_t}{1-\beta_2^t}$
Update
$\theta_t \leftarrow \prod_{\mathcal{F}, \sqrt{\widehat{v_t}}} \left(\theta_{t-1} - \frac{\alpha \widehat{m_t}}{\sqrt{\widehat{v_t}+\epsilon}}\right)$

Algorithm 2: AdaBelief Optimizer

Initialize $\theta_0, m_0 \leftarrow 0, s_0 \leftarrow 0, t \leftarrow 0$
While θ_t not converged
$t \leftarrow t+1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
$s_t \leftarrow \beta_2 s_{t-1} + (1-\beta_2)(g_t - m_t)^2$
Bias Correction
$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}, \widehat{s_t} \leftarrow \frac{s_t+\epsilon}{1-\beta_2^t}$
Update
$\theta_t \leftarrow \prod_{\mathcal{F}, \sqrt{\widehat{s_t}}} \left(\theta_{t-1} - \frac{\alpha \widehat{m_t}}{\sqrt{\widehat{s_t}+\epsilon}}\right)$



	Область 1	Область 2	Область 3
SGD/Момент	**S**	**L**	**L**
Adam	**L**	**S**	**S**
AdaBelief	**L**	**S**	**L**
Идеальный	**L**	**S**	**L**

L – большой шаг; S – маленький шаг

https://arxiv.org/abs/2010.07468

Алгоритм обучения SAM.

- SAM: Sharpness-aware Minimization (2020)
  - Поиск минимума в окрестности с близкими значениями → повышение обобщающей способности

$$L_S(\boldsymbol{w}) \triangleq \frac{1}{n}\sum_{i=1}^{n} l(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i)$$

$$L_{\mathscr{D}}(\boldsymbol{w}) \triangleq \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[l(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})]$$

$$L_{\mathscr{D}}(\boldsymbol{w}) \leq \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\epsilon}) + h\left(\frac{\|\boldsymbol{w}\|_2^2}{\rho^2}\right)$$

$$L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) \triangleq \max_{||\boldsymbol{\epsilon}||_p \leq \rho} L_S(\boldsymbol{w}+\boldsymbol{\epsilon})$$

$$\nabla_{\boldsymbol{w}} L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) \approx \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(w)|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$$

$$\hat{\boldsymbol{\epsilon}}(\boldsymbol{w}) = \rho \ \text{sign}\left(\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})\right) \frac{|\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})|^{q-1}}{\left(\|\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})\|_q^q\right)^{1/p}}$$

https://arxiv.org/abs/2010.01412

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_w L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

Algorithm 1: SAM algorithm

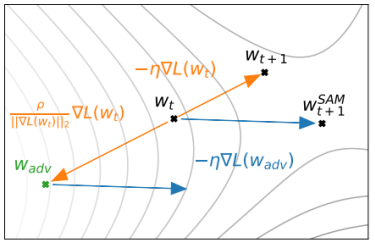
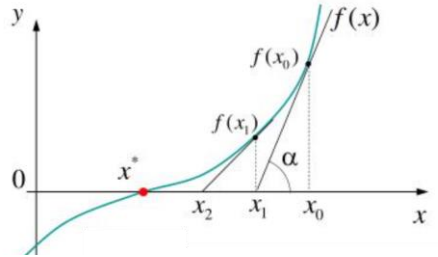


Figure 2: Schematic of the SAM parameter update.

- Применение стандартных оптимизаторов
- Специальная функция потерь
- Удвоенные вычислительные затраты

LBFGS.

- L-BFGS (**B**royden, **F**letcher, **G**oldfarb, **S**hanno) – оптимизатор 2-го порядка
  - Квазиньютоновский метод оптимизации



$$\operatorname{tg}\alpha = \frac{f(x_0)}{x_0 - x_1}$$

$$\operatorname{tg}\alpha = f'(x_0)$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

- Алгоритм BFGS

$$f(x) \to \min \Rightarrow \nabla f(x) = 0$$

Given starting point x_0, convergence tolerance $\epsilon > 0$,
inverse Hessian approximation H_0;
$k \leftarrow 0$;
while $\|\nabla f_k\| > \epsilon$;
Compute search direction

$$p_k = -H_k \nabla f_k;$$

x_k	the input of the function at step k	$x_{k+1} = x_k + \alpha_k p_k$
∇f_k	Gradient at x_k(at step k)	$\nabla f_k = \nabla f(x_k)$
B_k	the approximation of Hessian matrix at step k	...
α_k	step length at step k	search by wolfe condition
p_k	minimizer, search direction, at step k	$p_k = -B_k^{-1}\nabla f_k$
s_k	$s_k = x_{k+1} - x_k$	$s_k = \alpha_k p_k$
y_k	$y_k = \nabla f_{k+1} - \nabla f_k$	$B_{k+1}s_k = y_k$
ρ_k	$\rho_k = \frac{1}{y_k^T s_k}$	...
H_k	$H_k = B_k^{-1}$	$p_k = -H_k\nabla f_k$, $H_{k+1}y_k = s_k$

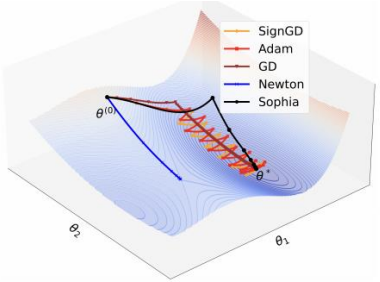
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

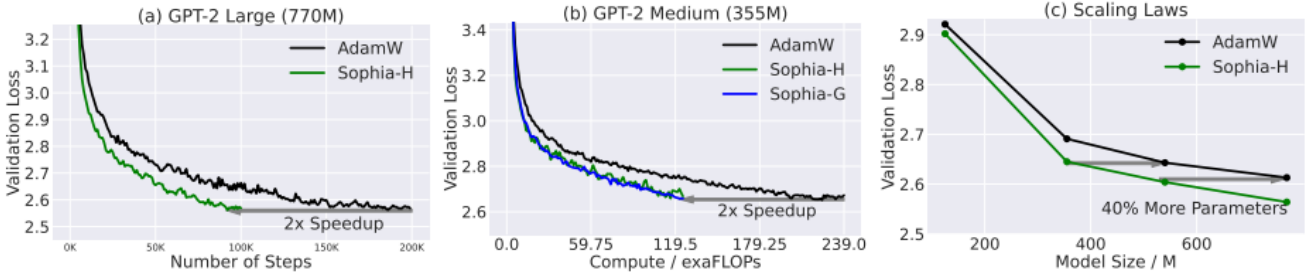
Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is computed from a line search procedure to satisfy the Wolfe conditions
Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$;
Compute H_{k+1}
$k \leftarrow k + 1$;
end (while)

x_{k+1}	$x_k + \alpha_k p_k$	...
α_{k+1}	using line search under Wolfe condition $y_k^T s_k \geq (c_2 - 1)\nabla f_k^T p_k$	$c_2 < 1$. ....
B_{k+1}	$(I - \rho_k y_k s_k^T)B_k(I - \rho_k s_k y_k^T) + \rho_k y_k y_k^T$	...
H_{k+1}	$(I - \rho_k s_k y_k^T)H_k(I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$	...

Sophia – оптимизатор 2-го порядка

- ❑ Усреднение градиента (скользящее среднее)
- ❑ Аппроксимация диагональных элементов Гессиана (2 варианта).
- ❑ Аппроксимация выполняется каждые n эпох.
- ❑ Применение клипирования для избегания слишком резких изменений весов.





Algorithm 3 Sophia

Input: θ_1, learning rate $\{\eta_t\}_{t=1}^{T}$, hyperparameters $\lambda, \beta_1, \beta_2, \epsilon$, and estimator choice Estimator $\in$ {Hutchinson,Gauss-Newton-Bartlett}

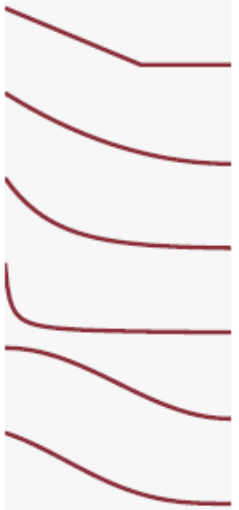
Set $m_0 = 0$, $v_0 = 0$, $h_{1-k} = 0$
for $t = 1$ **to** T **do**
  Compute minibach loss $L_t(\theta_t)$.
  Compute $g_t = \nabla L_t(\theta_t)$.
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  if $t \bmod k = 1$ **then**
    Compute $\hat{h}_t = \text{Estimator}(\theta_t)$.
    $h_t = \beta_2 h_{t-k} + (1 - \beta_2)\hat{h}_t$
  else
    $h_t = h_{t-1}$
  $\theta_t = \theta_t - \eta_t \lambda \theta_t$ (weight decay)
  $\theta_{t+1} = \theta_t - \eta_t \cdot \text{clip}(m_t / \max\{h_t, \epsilon\}, \rho)$

- ❑ Представлен, как эффективный алгоритм обучения LLM: 2x меньше вычислений для того же минимума чем Adam, достижения такой же эффективности на моделях меньшего размера.

Изменение скорости обучения

- Изменение скорости обучения повышает эффективность обучения НС.
- Существует множество схем изменения скорости обучения.
- Простейшие схемы:
  - Ступенчатое уменьшение
    - в ***к*** раз каждые ***n*** итераций обучения;
    - уменьшение коэффициента обучения, если ошибка на подтверждающей выборке перестала уменьшаться.
  - Экспоненциальное $\eta = \alpha_0 e^{-\beta t}$
  - Гиперболическое $\eta = \frac{\alpha_0}{1+\beta t}$

Схемы изменения скорости обучения (1)

Name		Ref.	Illustration
Constant			
Step Decay	constant factor		
	multi-step		
Smooth Decay	linear decay	e.g. (Goodfellow et al., 2016)	
	polynomial decay		
	exponential decay		
	inverse time decay	e.g. (Bottou, 2012)	
	cosine decay	(Loshchilov & Hutter, 2017)	
	linear cosine decay	(Bello et al., 2017)	

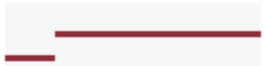
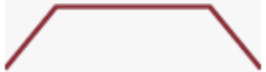
** https://arxiv.org/abs/2007.01547*

Схемы изменения скорости обучения (2)

Name		Ref.	Illustration
Cyclical	triangular	(Smith, 2017)	
	triangular + decay	(Smith, 2017)	
	triangular + exponential decay	(Smith, 2017)	
	cosine + warm restarts	(Loshchilov & Hutter, 2017)	
	cosine + warm restarts + decay	(Loshchilov & Hutter, 2017)	

** https://arxiv.org/abs/2007.01547*

Схемы изменения скорости обучения (3)

Name		Ref.	Illustration
Warmup	constant warmup	e.g. (He et al., 2016)	
	gradual warmup	(Goyal et al., 2017)	
	gradual warmup + multi-step decay	(Goyal et al., 2017)	
	gradual warmup + step number decay	(Vaswani et al., 2017)	
	slanted triangular	(Howard & Ruder, 2018)	
	long trapezoid	(Xing et al., 2018)	
Super-Convergence	1cycle	(Smith & Topin, 2017)	

** https://arxiv.org/abs/2007.01547*

Сравнение оптимизаторов

- R.M. Schmidt, et al., 2021 (*https://arxiv.org/abs/2007.01547)*
- <15 оптимизаторов> x <8 задач> x <4 схемы изменения скорости обучения> x <4 вида настройки гиперпараметров>
- **Выводы:**
  - **Не существует «лучшего» оптимизатора, эффективность зависит от задачи.**
  - **Есть пул наиболее стабильных оптимизаторов: Adam, NAG, RMSProp, … .**
  - **Проверка разных оптимизаторов (с настройками по-умолчанию) ≈ настройка гиперпараметров для одного оптимизатора.**
  - **Эффективность схемы изменения скорости обучения зависит от задачи.**
  - **Направление будущих исследований: разработка методов эффективных для конкретных типов задач, автоматическая оптимизация внутренних параметров, новые типы алгоритмов.**

Вопросы

- ❑ Основные проблемы обучения НС
- ❑ Отличие обучения с моментом от момента Нестерова?
- ❑ Основные компоненты алгоритма Adam