

Основы практического использования нейронных сетей.

Лекция 3. Алгоритмы обучения для глубоких НС.

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

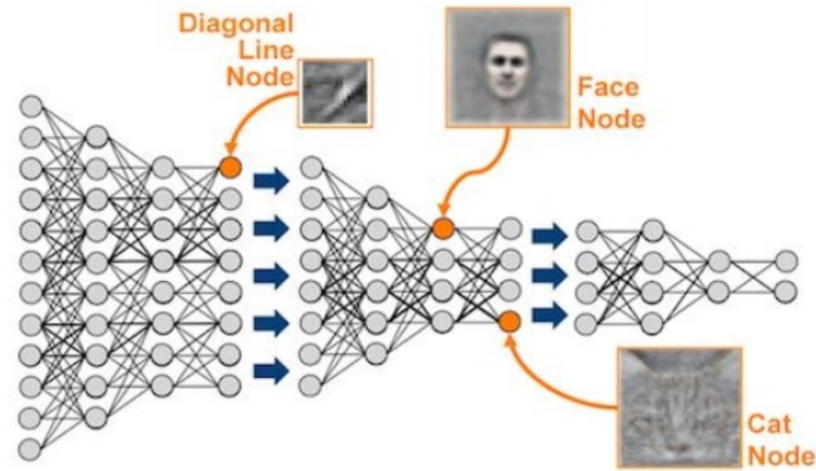
Глубокие НС vs традиционные НС

❑ Традиционные (shallow) НС:

- 1-2 внутренних слоев;
- необходимость предварительного выделения признаков, предобработка данных;
- «простота» обучения.

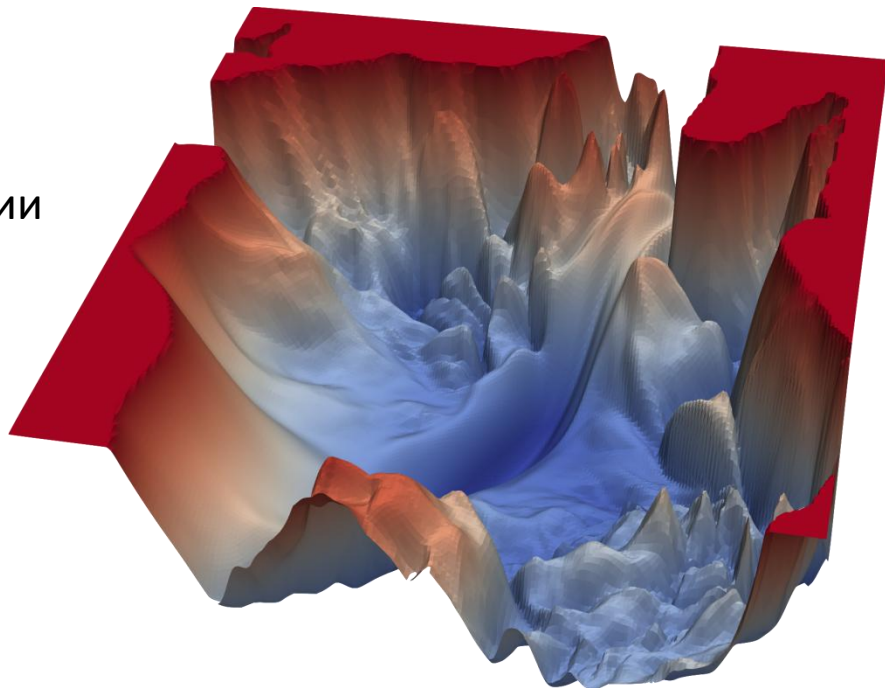
❑ Глубокие НС:

- >2 слоев;
- моделирование абстракций в данных;
- большое число слоев и нейронов -> большое число весов.
- адаптирование алгоритмов обучения.
- высокие требования к вычислительным ресурсам.



Трудности обучения глубоких НС

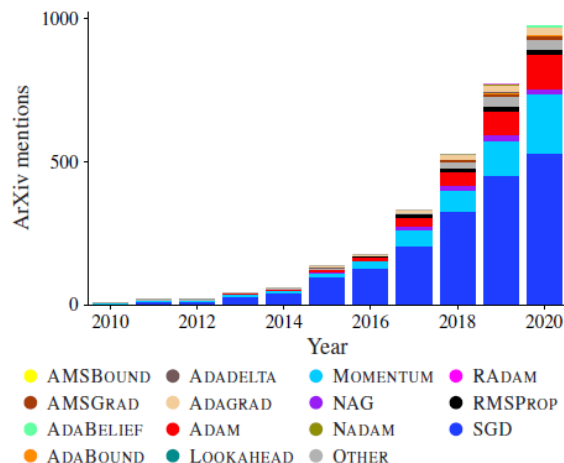
- ❑ Локальные минимумы функции ошибки
- ❑ Сложный ландшафт целевой функции
- ❑ Неравномерность обновления параметров сети
- ❑ Выбор закона изменения скорости обучения



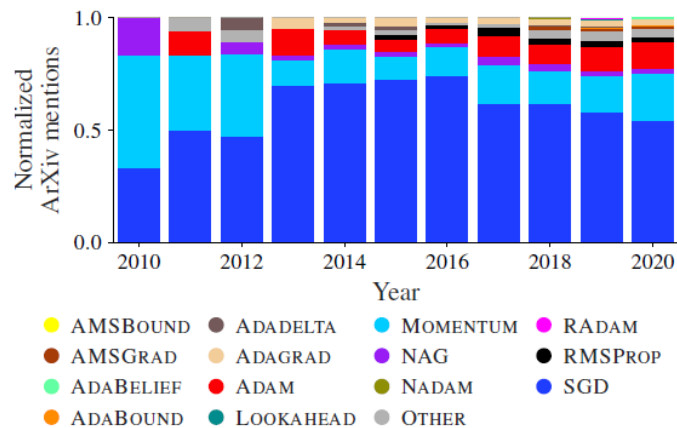
<https://www.cs.umd.edu/~tomg/projects/landscapes/>

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*

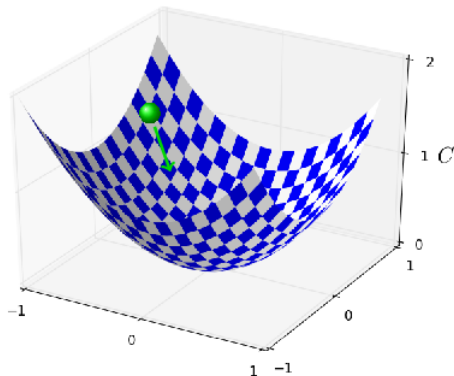


Относительное число упоминаний оптимизаторов в статьях на сайте arxiv.org*

* <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 подмножеств из обучающей выборки.



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

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

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

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

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

Методы оптимизации 2-го порядка

$$C(w + \Delta w) \approx C(w) + \nabla C \cdot \Delta w + \frac{1}{2} \Delta w^T H \Delta w.$$

$$\Delta w = -H^{-1} \nabla C.$$

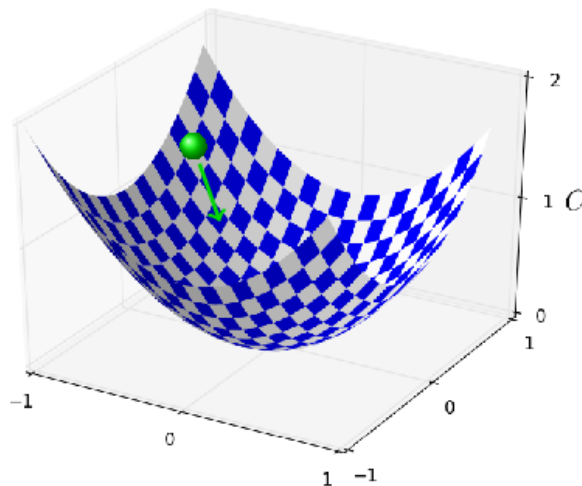
- ❑ Оценка Гессиана $\sim$ скорость изменения градиента.
- ❑ Требуется много памяти: 1000000 весов $\rightarrow 10^{12}$ элементов Гессиана $\rightarrow >3\text{Tb}$ RAM .
- ❑ Методы частичной аппроксимации Гессиана (L-BFGS).
- ❑ Аппроксимация требует использования всех примеров из обучающей выборки.

Использование момента

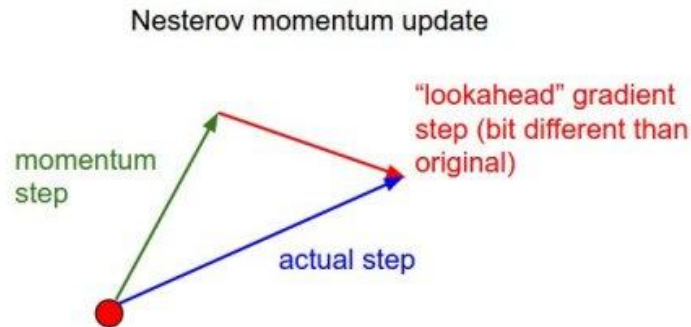
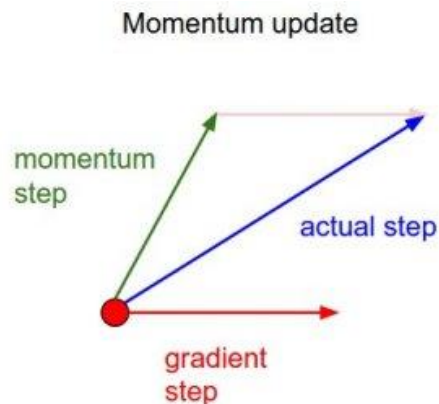
- ❑ Идея накопления импульса.
- ❑ Оценка «скорости» градиента.
- ❑ Применение коэффициента «трения» ~ коэффициент момента.

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

- ❑ Изменение коэффициента момента в процессе обучения $0.5 \rightarrow 0.9$



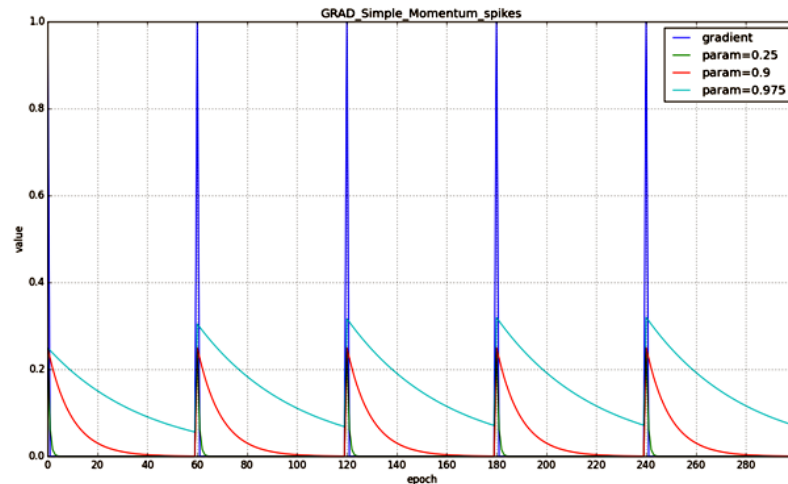
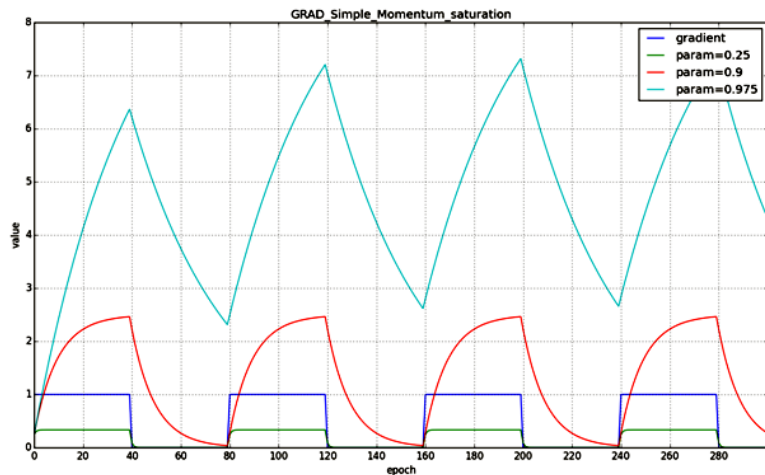
Момент Нестерова (NAG)



- ❑ Оценка градиента в предсказанной позиции функции ошибки

Обзор алгоритмов обучения*. Момент.

$$v_t = \gamma v_{t-1} + \eta g_t \quad g_t = \nabla_{\theta} C(\theta)$$
$$\theta_{t+1} = \theta_t - v_t \quad \gamma = 0.9$$



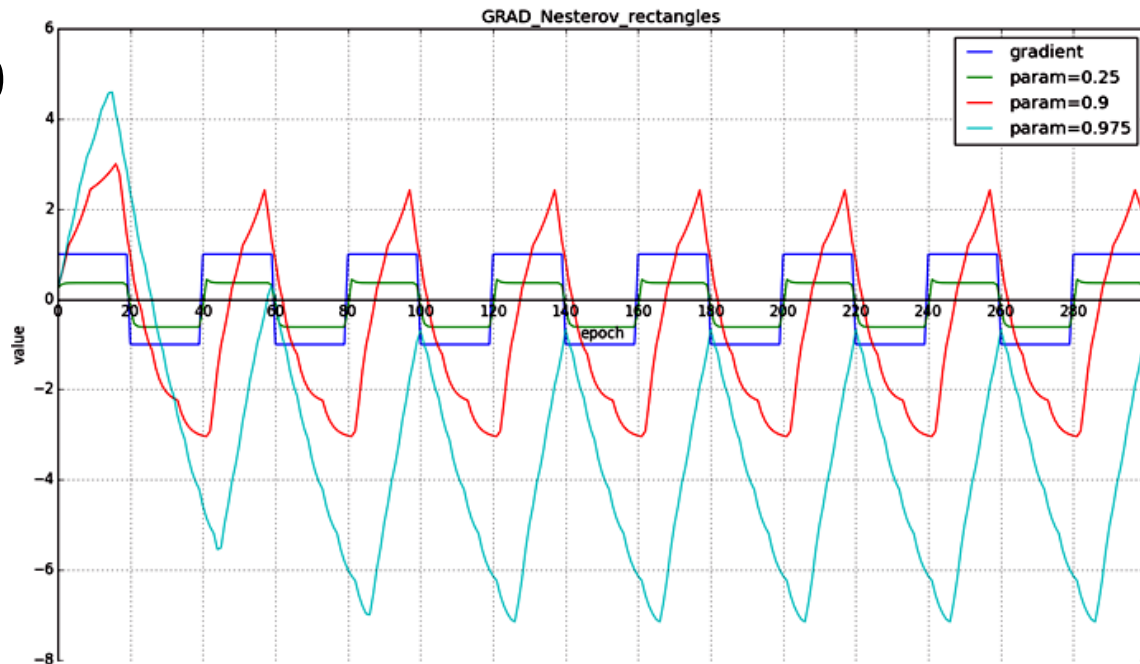
* <https://habrahabr.ru/post/318970/>

Обзор алгоритмов обучения.

Момент Нестерова (NAG).

$$v_t = \mathcal{W}_{t-1} + \eta g_t (\theta - \mathcal{W}_{t-1})$$

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



Обзор алгоритмов обучения.

Adagrad.

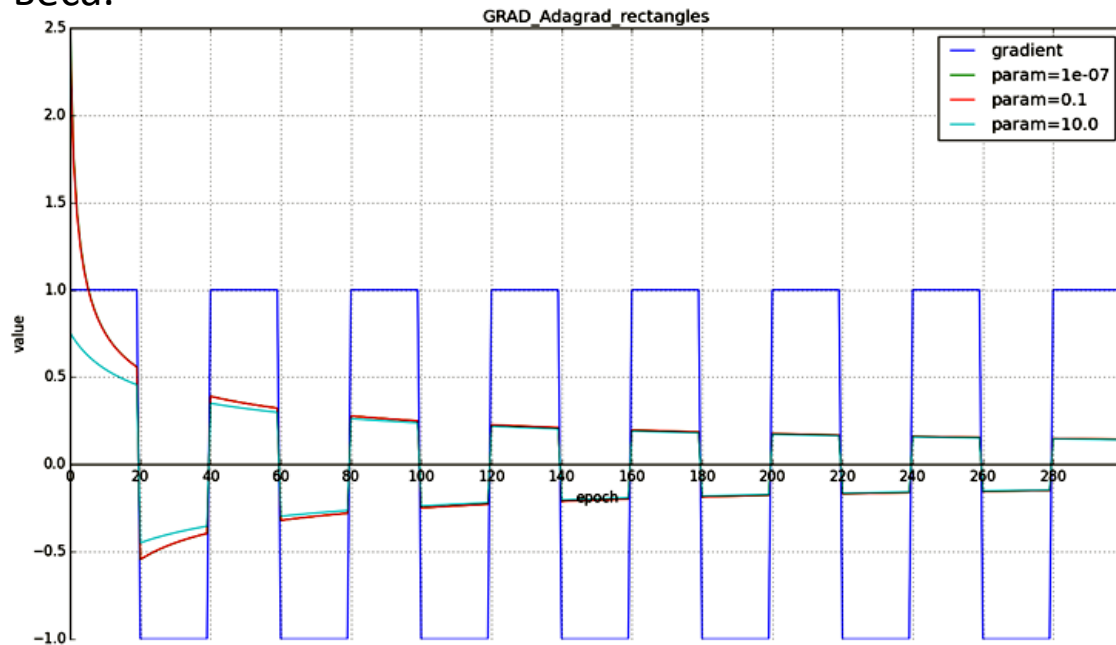
□ Учет частоты обновления веса.

□ 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;

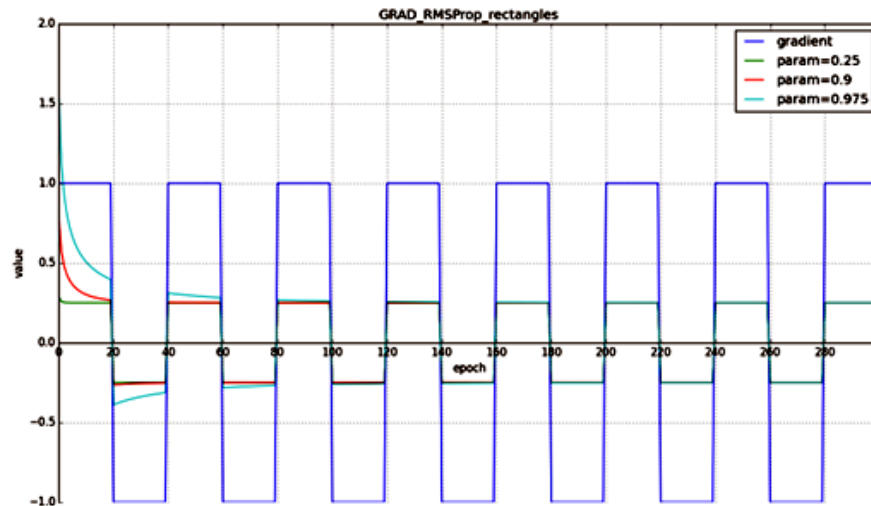
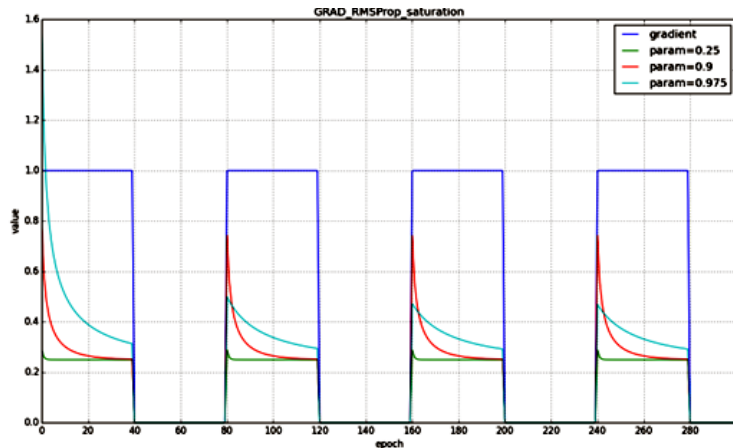


Обзор алгоритмов обучения. RMSProp.

- Учет частоты обновления веса за счет усреднения.

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

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



Обзор алгоритмов обучения.

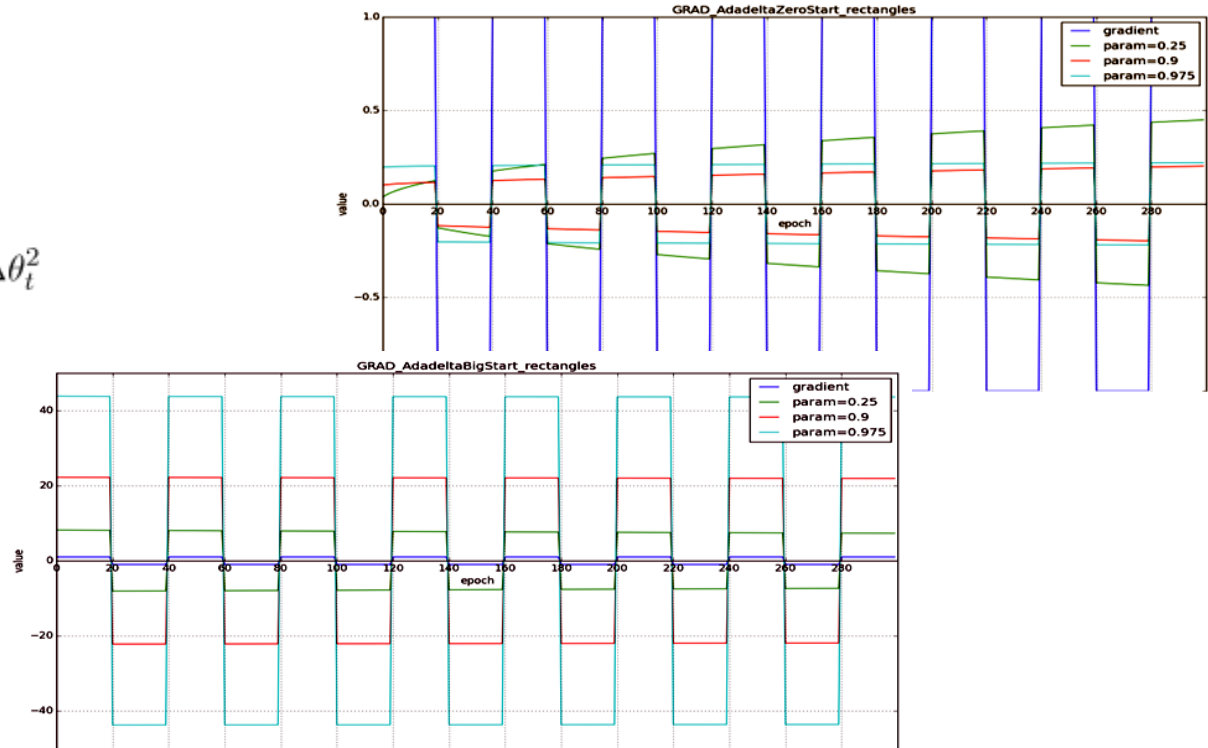
Adadelta.

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

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

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

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



Обзор алгоритмов обучения. Adam.

□ Момент + учет частоты обновления веса.

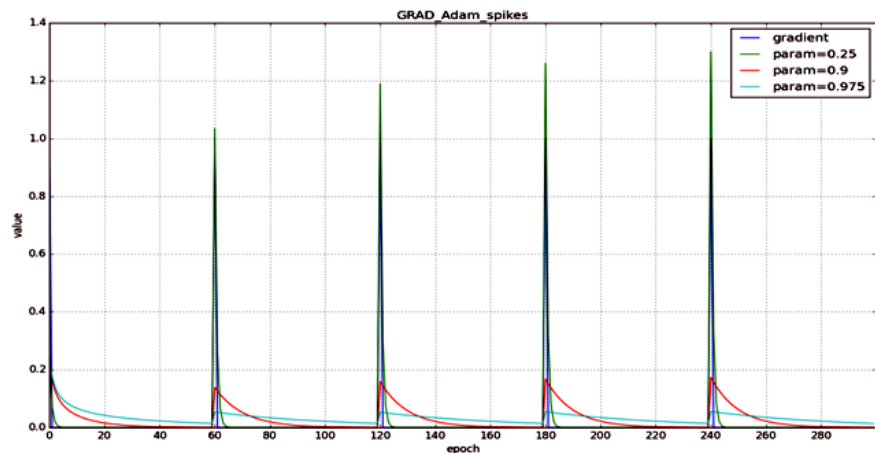
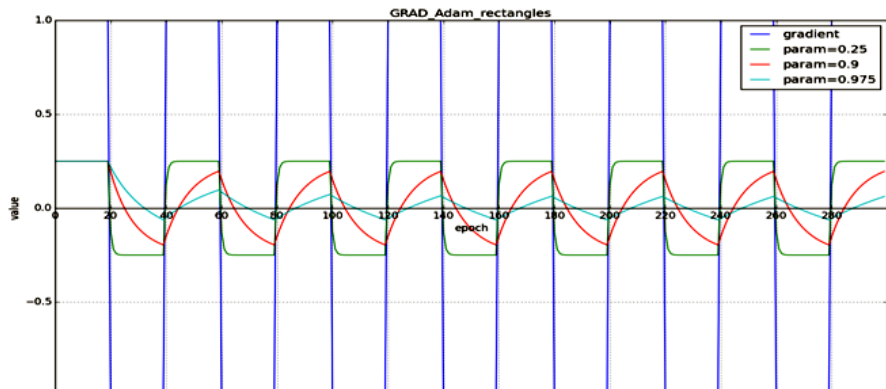
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

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

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

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

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



Обзор алгоритмов обучения.

Adamax.

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

$$\begin{aligned} u_t &= \lim_{p \rightarrow \infty} v_t^{\frac{1}{p}} = \lim_{p \rightarrow \infty} \left[\beta_2^p v_{t-1} + (1 - \beta_2^p) |g_t|^p \right]^{\frac{1}{p}} = \lim_{p \rightarrow \infty} \left(\sum_{i=1}^t \beta_2^{p(t-i)} |g_i|^p \right)^{\frac{1}{p}} \\ &= \max(\beta_2^{t-1} |g_1|, \beta_2^{t-2} |g_2|, \dots, \beta_2 |g_{t-1}|, |g_t|) \end{aligned}$$

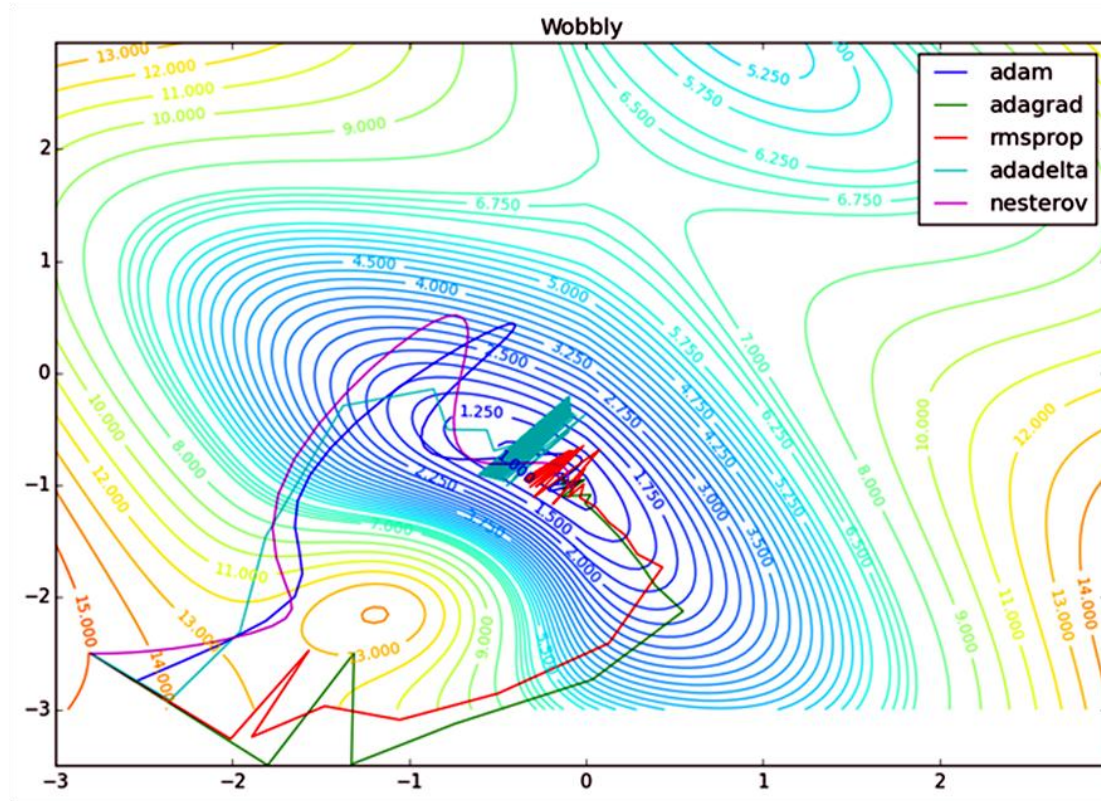
$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

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

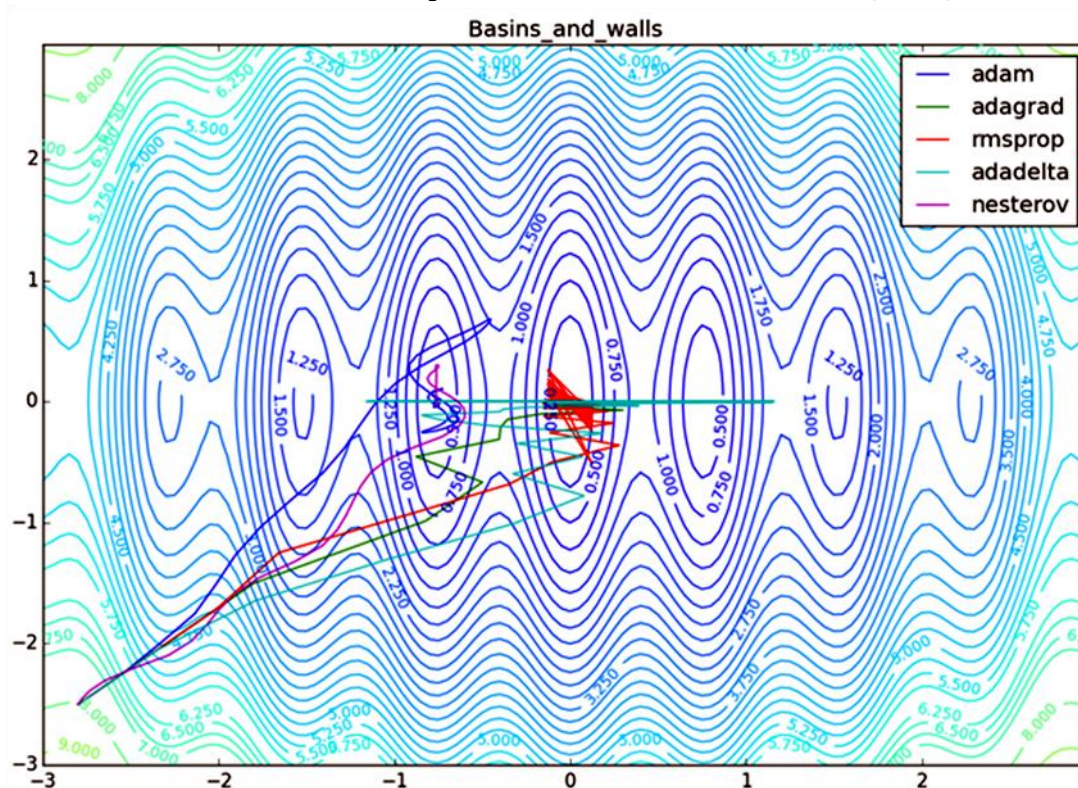
Обзор алгоритмов обучения.

Эксперименты (1).



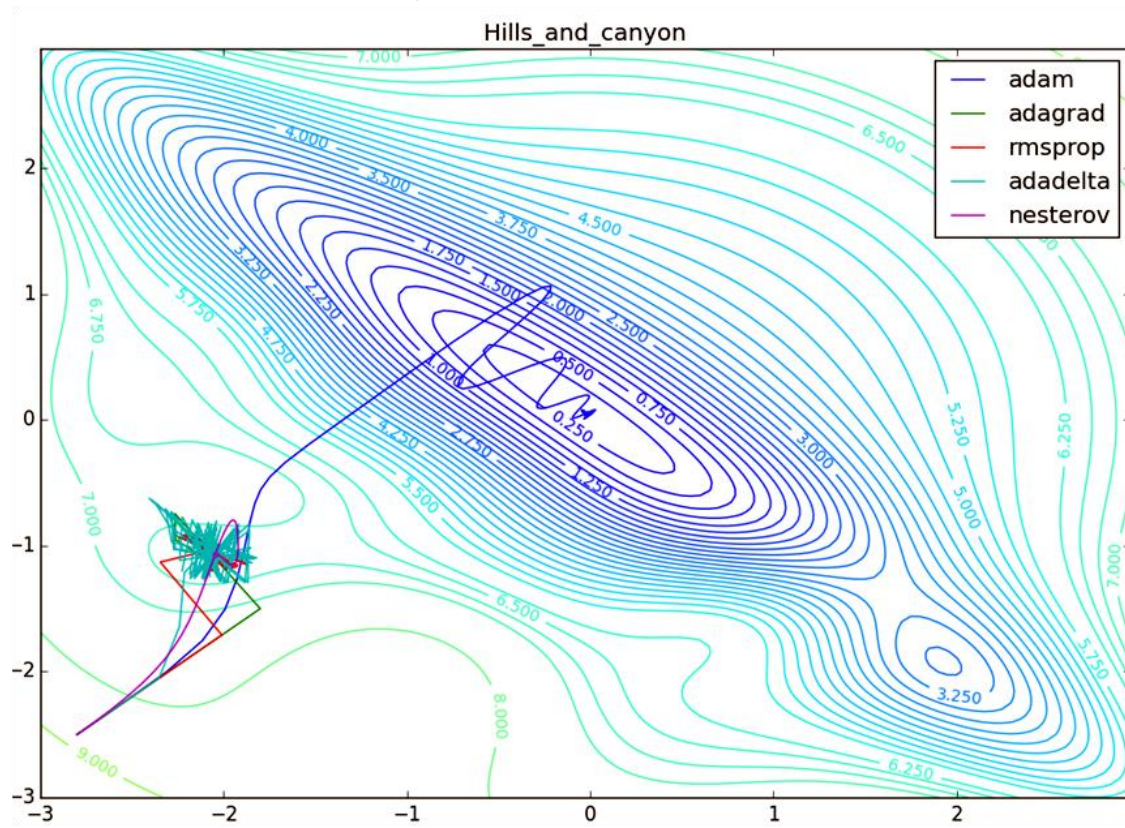
Обзор алгоритмов обучения.

Эксперименты (2).



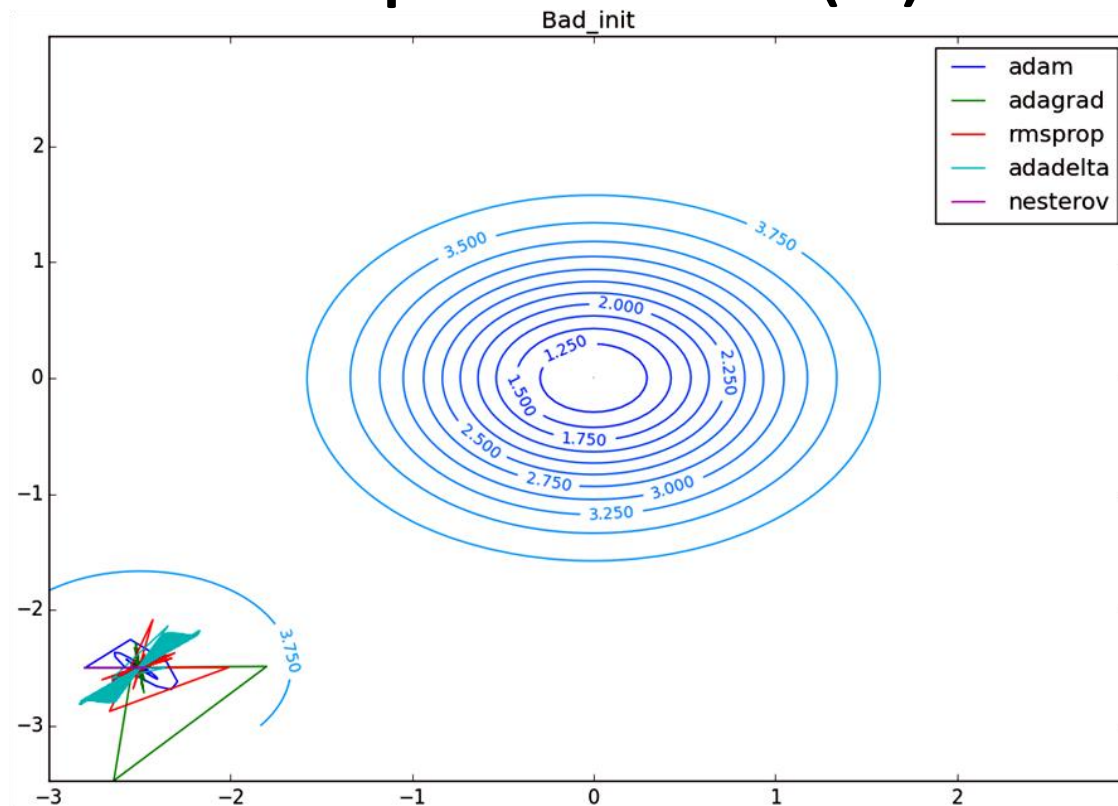
Обзор алгоритмов обучения.

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

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

Update

$\theta_t \leftarrow \Pi_{\mathcal{F}, \sqrt{\hat{v}_t}} \left(\theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \right)$

Algorithm 2: AdaBelief Optimizer

Initialize $\theta_0, m_0 \leftarrow 0, s_0 \leftarrow 0, t \leftarrow 0$

While θ_t not converged

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$

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

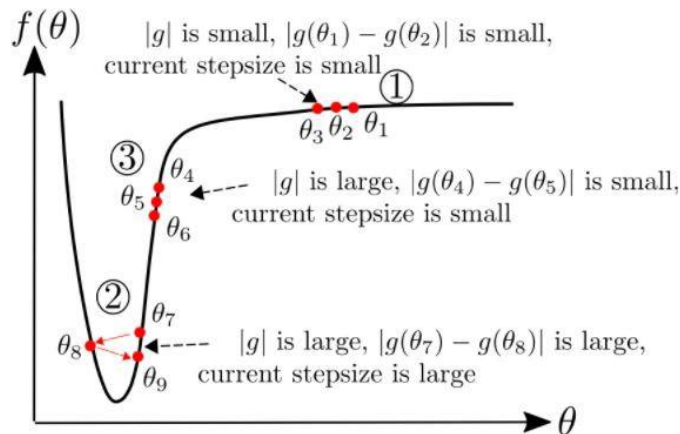
$s_t \leftarrow \beta_2 s_{t-1} + (1 - \beta_2) (g_t - m_t)^2$

Bias Correction

$\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}, \hat{s}_t \leftarrow \frac{s_t + \epsilon}{1 - \beta_2^t}$

Update

$\theta_t \leftarrow \Pi_{\mathcal{F}, \sqrt{\hat{s}_t}} \left(\theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{s}_t} + \epsilon} \right)$

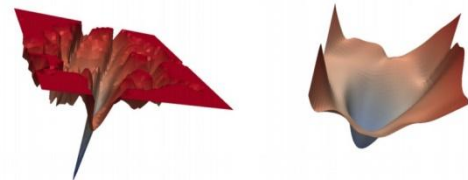


	Область 1	Область 2	Область 3
SGD/Момент	-	-	+
Adam	+	+	-
AdaBelief	+	+	+

Алгоритм обучения SAM.

□ SAM: Sharpness-aware Minimization (2020)

- Поиск минимума в окрестности с близкими значениями → повышение обобщающей способности



$$L_S(w) \triangleq \frac{1}{n} \sum_{i=1}^n l(w, x_i, y_i)$$

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

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

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

$$\nabla_w L_S^{SAM}(w) \approx \nabla_w L_S(w)|_{w+\hat{\epsilon}(w)}$$

$$\hat{\epsilon}(w) = \rho \operatorname{sign}(\nabla_w L_S(w)) \frac{|\nabla_w L_S(w)|^{q-1}}{\left(\|\nabla_w L_S(w)\|_q^q\right)^{1/p}}$$

<https://arxiv.org/abs/2010.01412>

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^n \{(x_i, y_i)\}$, Loss function $l: \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, Batch size b , Step size $\eta > 0$, Neighborhood size $\rho > 0$.

Output: Model trained with SAM

Initialize weights w_0 , $t = 0$;

while *not converged* **do**

 Sample batch $\mathcal{B} = \{(x_1, y_1), \dots, (x_b, y_b)\}$;

 Compute gradient $\nabla_w L_{\mathcal{B}}(w)$ of the batch's training loss;

 Compute $\hat{\epsilon}(w)$ per equation 2;

 Compute gradient approximation for the SAM objective (equation 3): $g = \nabla_w L_{\mathcal{B}}(w)|_{w+\hat{\epsilon}(w)}$;

 Update weights: $w_{t+1} = w_t - \eta g$;

$t = t + 1$;

end

return w_t

Algorithm 1: SAM algorithm

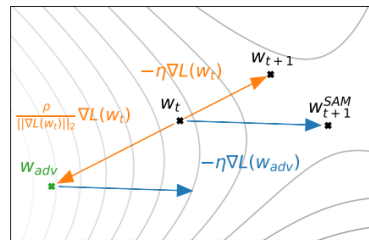


Figure 2: Schematic of the SAM parameter update.

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

Изменение скорости обучения

- ❑ Изменение скорости обучения повышает эффективность обучения НС.
- ❑ Существует множество схем изменения скорости обучения.
- ❑ Простейшие схемы:

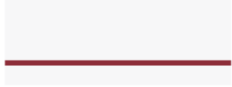
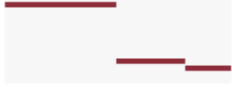
- Ступенчатое уменьшение

- в k раз каждые n итераций обучения;
- уменьшение коэффициента обучения, если ошибка на подтверждающей выборке перестала уменьшаться.

- Экспоненциальное $\eta = \alpha_0 e^{-\beta t}$

- Гиперболическое $\eta = \frac{\alpha_0}{1 + \beta t}$

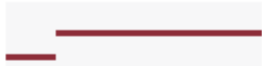
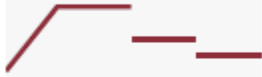
Схемы изменения скорости обучения (1)

Name	Ref.	Illustration
Constant		
Step Decay	constant factor	
	multi-step	
Smooth Decay	linear decay	
	polynomial decay	
	exponential decay	
	inverse time decay	
	cosine decay	
	linear cosine decay	

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)	

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	Icycle	(Smith & Topin, 2017)	

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) $\approx$ 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
- ❑ Почему в алгоритмах обучения не применяют методы оптимизации 2-го порядка?
- ❑ Отличие обучения с моментом от момента Нестерова?