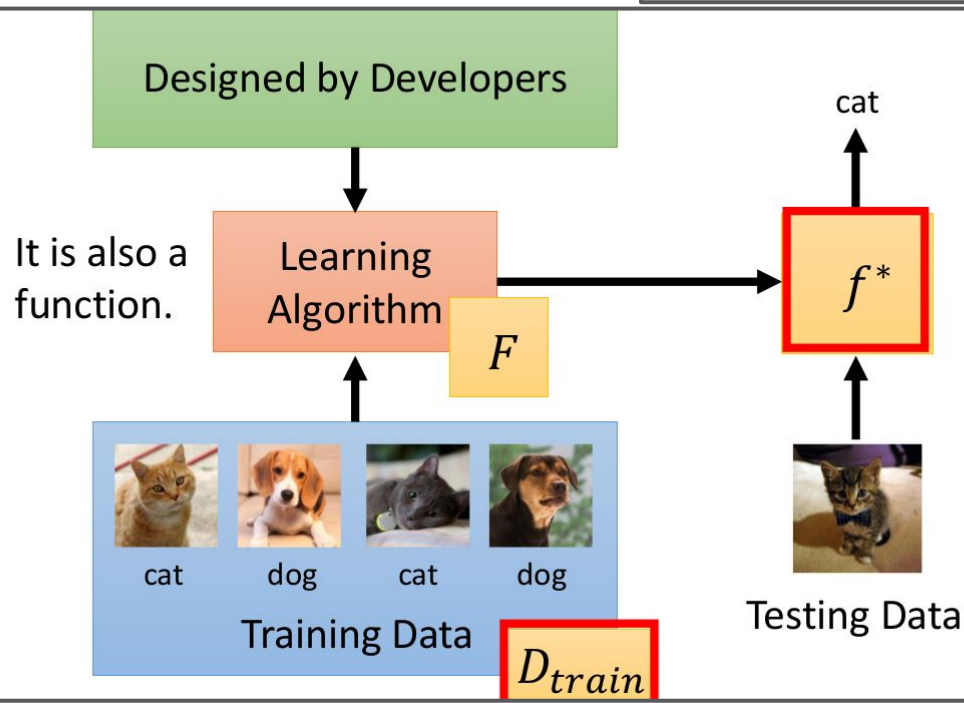


Meta Learning

Дилара Хамдеева,
172

“Learn to learn”

$$f^* = F(D_{train})$$

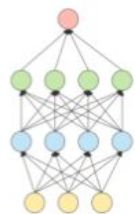


$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \dots, \mathcal{D}_n\}$$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \dots, (x_k^i, y_k^i)\}$$



Meta Learning vs Machine Learning



$f($



$) = \text{"Cat"}$

$F($



cat



dog



cat



dog

$) = f^*$



Training Data

1. Learning to learn by gradient descent
by gradient descent
2. MAML
3. Reptile

1. Learning to learn by gradient descent by gradient descent

ИДЕЯ: представить алгоритм оптимизации как задачу обучения

ИДЕЯ: представить алгоритм оптимизации как задачу обучения

Классическая задача оптимизации:

$$f(\theta) \rightarrow \min_{\theta \in \Theta}$$

Стандартное решение -- градиентный спуск:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

Проблема 1.

Игнорирование вторых производных.

Подход.

Масштабирование градиентного шага:
Гессиан, матр Фишера и тд.

Минус подхода.

Сложно считать.

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

Проблема 2.	Методы оптимизации подбираются для конкретных классов. А что насчет более общих методов?
-------------	--

Предлагается заменить hand-designed update rules на learned update rules.

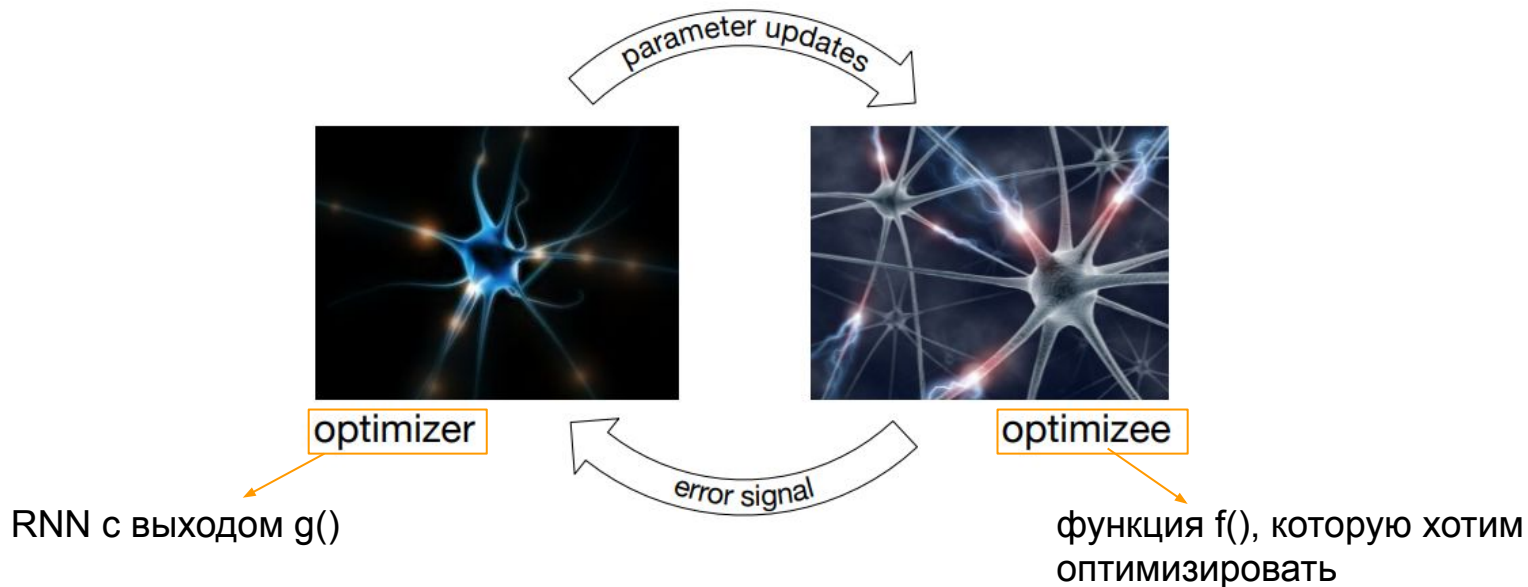
Заменим

hand-designed update rules

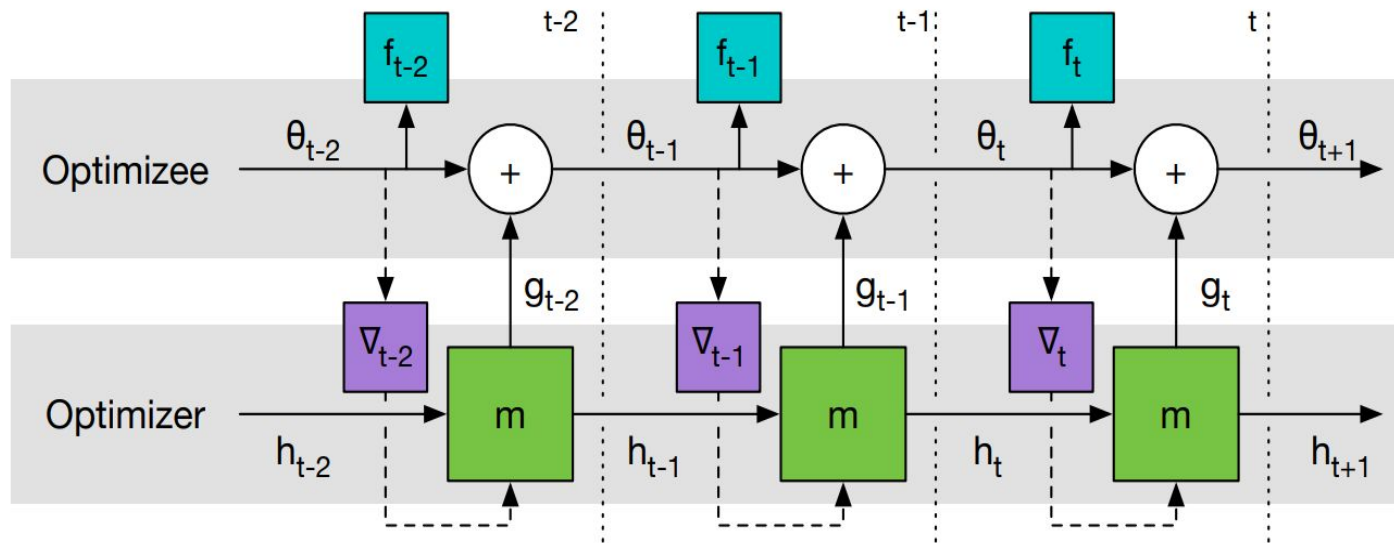
на learned update rules:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

$$\theta_{t+1} = \theta_t + \underline{g_t(\nabla f(\theta_t), \phi)}$$



RNN



Метод

- $\theta^*(f, \phi)$ -- финальное значение параметра
- $L(\phi) = E_f[f(\theta^*(f, \phi))]$
- Функция f зависит только от финального значения параметра $\theta^*(f, \phi)$
- \Rightarrow не можем использовать BPTT, чтобы обучить optimizer.

Добавим информацию о траектории

- $L(\phi) = E_f[f(\theta^*(f, \phi))]$

- Добавим информацию о траектории.
- Новый лосс выглядит так

$$L(\phi) = E_f \left[\sum_{t=1}^T w_t f(\theta_t) \right] \quad \begin{aligned} \theta_{t+1} &= \theta_t + g_t, \\ \begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} &= m(\nabla_t, h_t, \phi). \end{aligned}$$

↓
RNN-модель с параметрами ϕ и состоянием h_t

-
- При $w_t = 1[t = T]$ старая и новая функции потерь эквивалентны
- Берем $w_t > 0$, чтобы BPTT считался

RNN

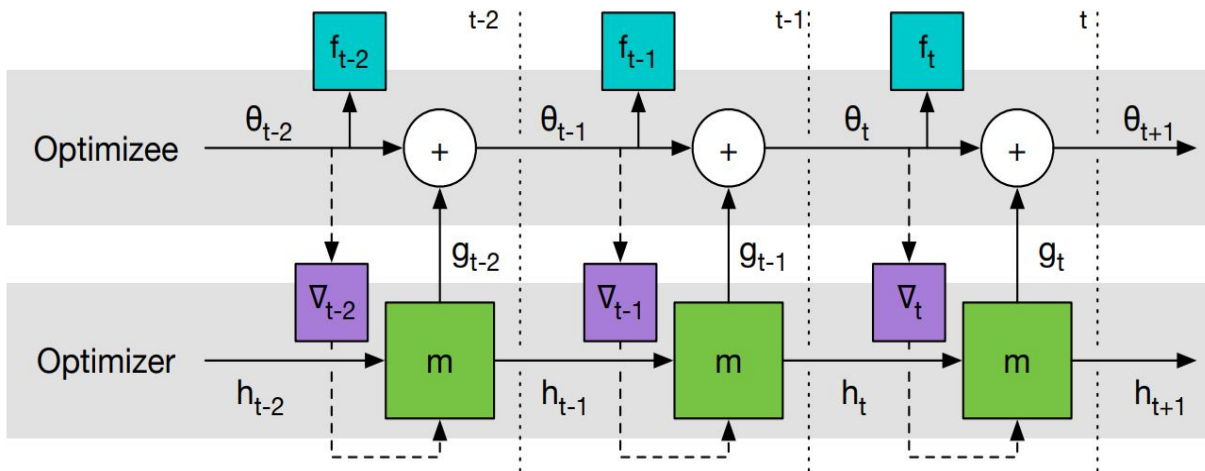
- $L(\phi) \rightarrow \min_{\phi}$ с помощью градиентного спуска по ϕ

- Оценка градиента $\partial L(\phi)/\partial \phi$ считается с помощью BPTT

НО

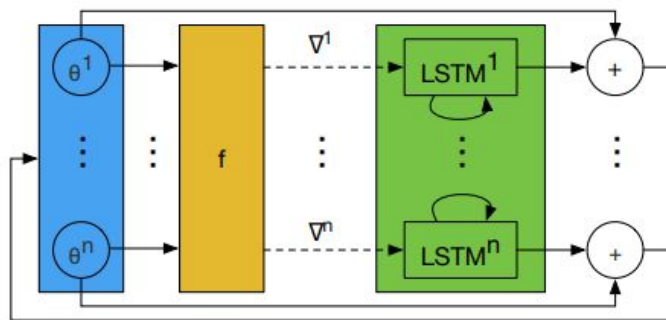
- Градиенты по пунктирным линиям *игнорируются*
 $\partial \nabla_t / \partial \phi = 0$

- Можно не считать вторые производные



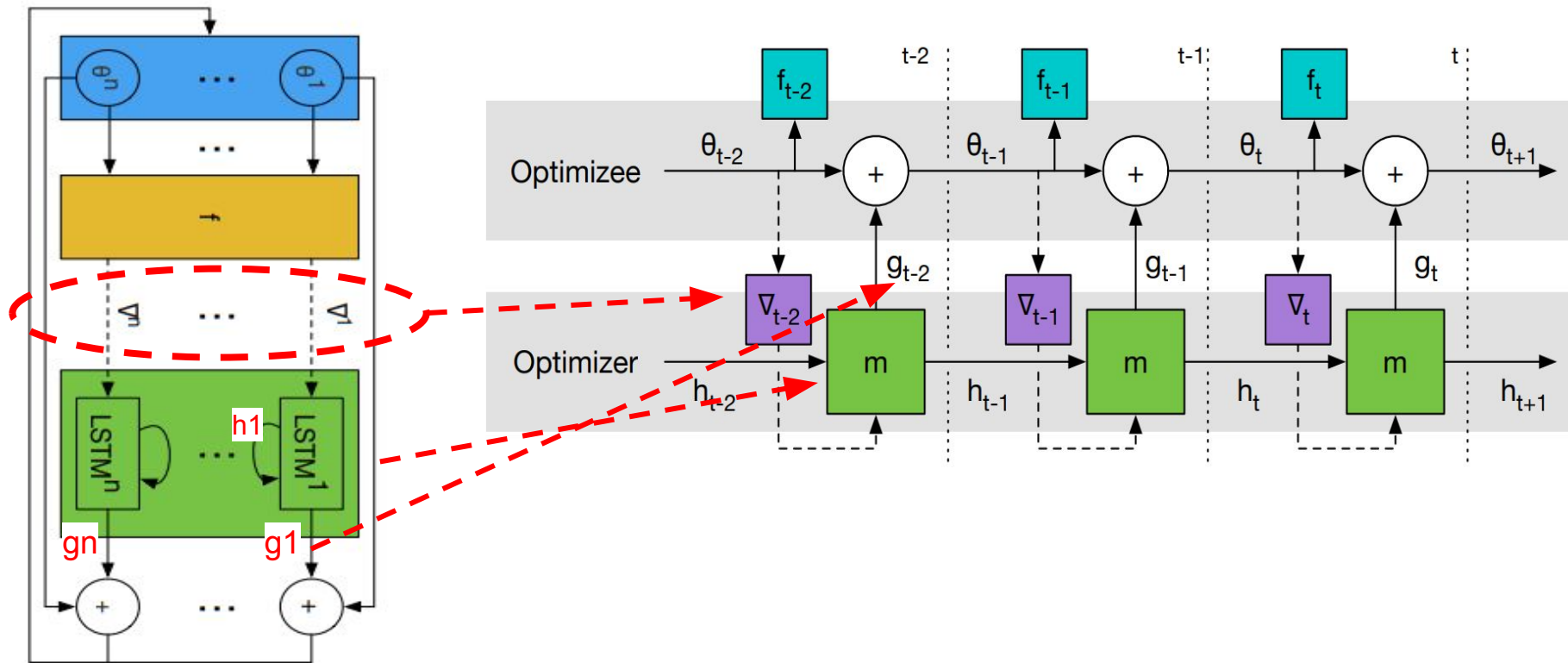
Покоординатный LSTM optimizer.

- Хотим оптимизировать хотя бы десятки из тысяч параметров.
- Но с RNN это не совсем осуществимо: огромный hidden state, много параметров. => используем покоординатный LSTM.

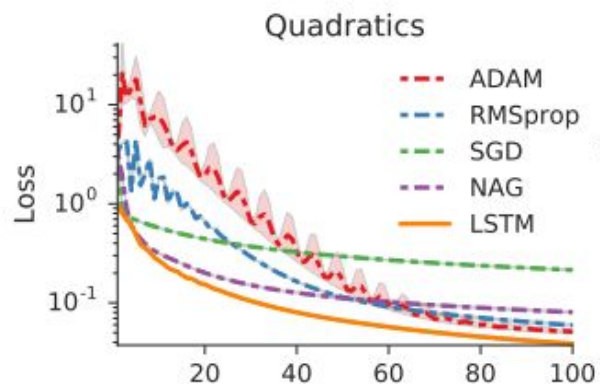


Один шаг LSTM оптимайзера

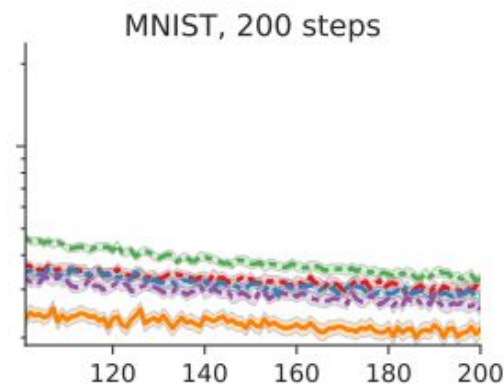
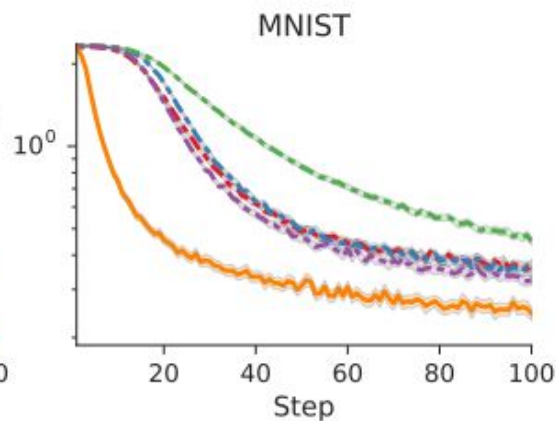
Покоординатный LSTM optimizer.



Experiments



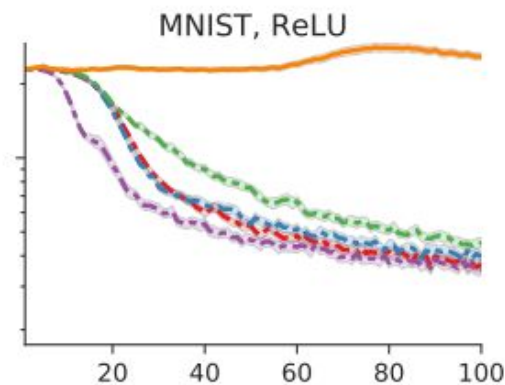
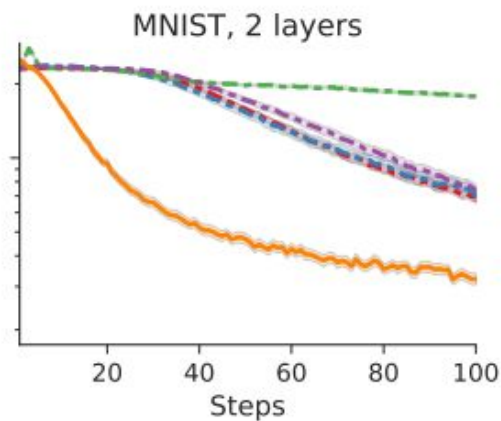
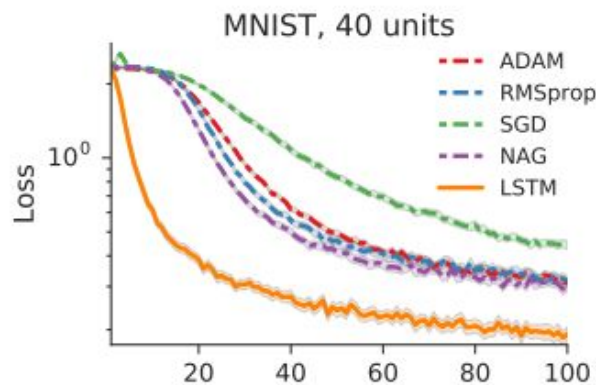
$$f(\theta) = \|W\theta - y\|_2^2$$



$f(\theta)$ is the cross entropy

Experiments.

Разные архитектуры.



Experiments. Neural Art.



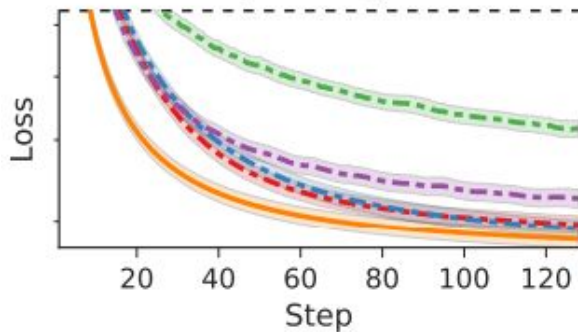
content image результат style image



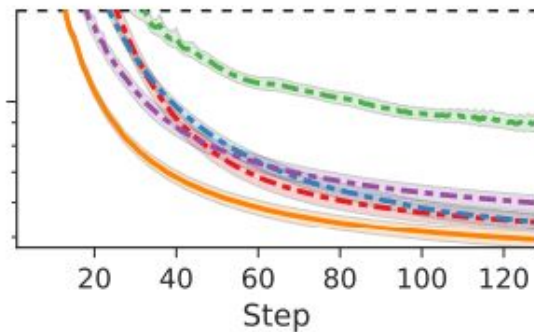
content image результат style image

$$f(\theta) = \alpha \mathcal{L}_{\text{content}}(c, \theta) + \beta \mathcal{L}_{\text{style}}(s, \theta) + \gamma \mathcal{L}_{\text{reg}}(\theta)$$

Neural art, training resolution



Double resolution



- ADAM
- RMSprop
- SGD
- NAG
- LSTM

1. Learning to learn by gradient descent
by gradient descent

2. **MAML**

3. Reptile

2 MAML: Model-Agnostic Meta-Learning

MAML - мета-лернинг алгоритм, который

1. не зависит от модели* и задачи
2. быстро обучается на новых задачах
(хорошие результаты уже за 1-2 градиентных спусков).

* (но модель должна уметь обучаться градиентным спуском)

2 MAML: Model-Agnostic Meta-Learning

ИДЕЯ: обучить начальные параметры модели так, чтобы модель быстро адаптировалась к новой задаче: достигала максимальной производительности после обновления параметров градиентным спуском.

MAML. Идея.

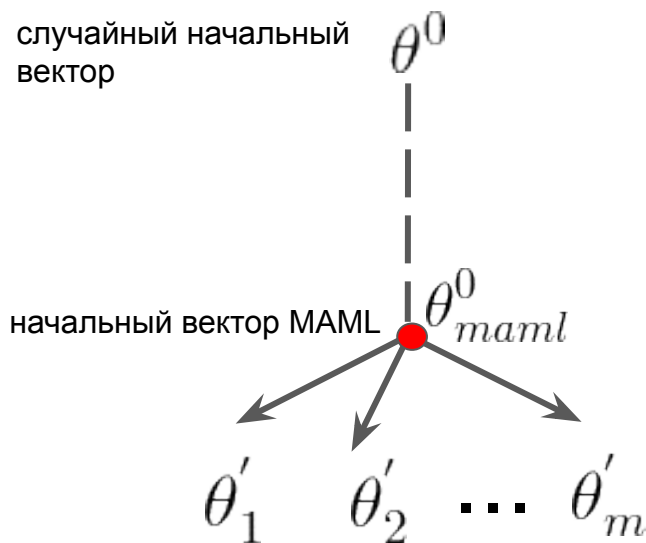
Обычно	MAML
<div data-bbox="150 394 784 836"><ol style="list-style-type: none">1. Есть задача T (классификация, регрессия и тд)2. Хотим найти вектор параметров θ.3. Берем случайно какой-то начальный вектор θ^04. И оптимизируем:</div> <div data-bbox="229 885 633 945">$\theta^{k+1} = \theta^k + \alpha \nabla L(\theta^k)$</div>	<div data-bbox="832 563 1818 803"><p>Идея: Для решения задачи T на шаге 3. возьмем не случайный вектор θ^0, а обученный с помощью MAML. Назовем его θ_{maml}^0</p></div>

MAML. Как найти начальный вектор параметров?

- Есть ряд задач:
- Для каждой есть оптимальный вектор:

T_1, T_2, \dots, T_m

$\theta'_1, \theta'_2, \dots, \theta'_m$



MAML. Как найти начальный вектор параметров?

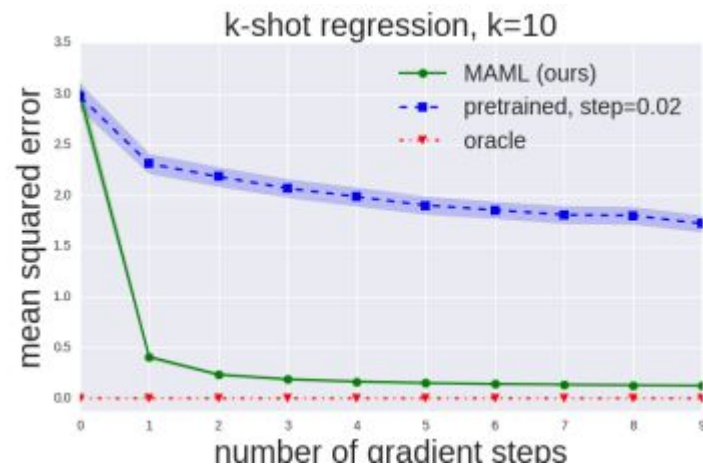
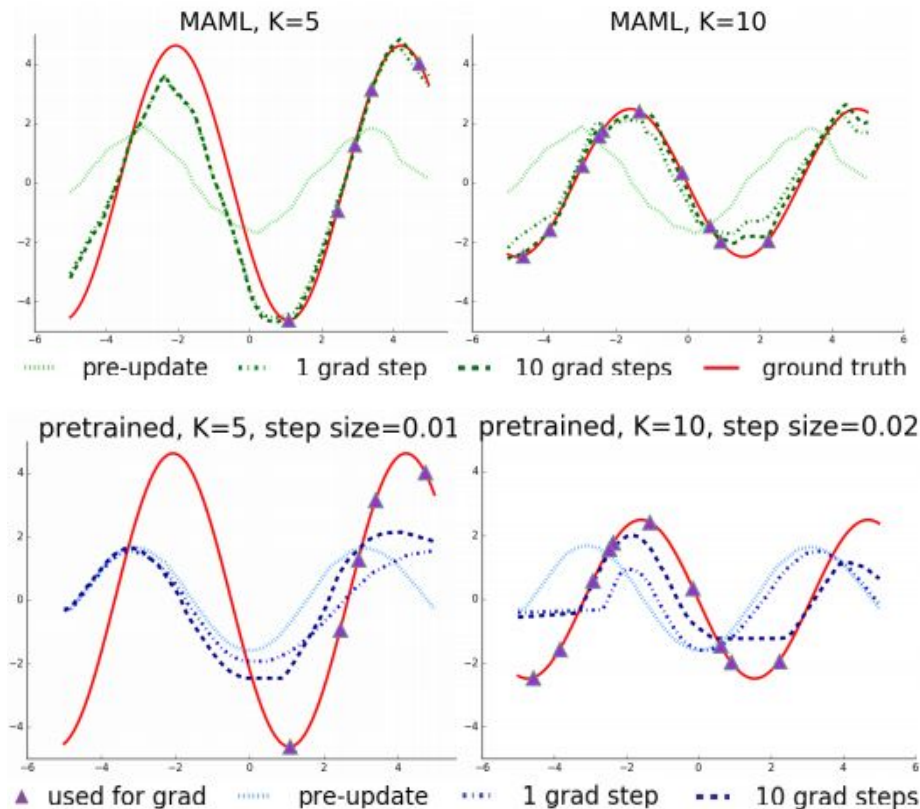
Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β : step size hyperparameters

- 1: randomly initialize θ
 - 2: **while** not done **do**
 - 3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
 - 4: **for all** \mathcal{T}_i **do**
 - 5: Evaluate $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$ with respect to K examples
 - 6: Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$
 - 7: **end for**
 - 8: Update $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$
 - 9: **end while**
-

MAML. Experiments.



MAML. Experiments.

	5-way Accuracy		20-way Accuracy	
	1-shot	5-shot	1-shot	5-shot
Omniglot (Lake et al., 2011)				
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MAML, no conv (ours)	$89.7 \pm 1.1\%$	$97.5 \pm 0.6\%$	–	–
Siamese nets (Koch, 2015)	97.3%	98.4%	88.2%	97.0%
matching nets (Vinyals et al., 2016)	98.1%	98.9%	93.8%	98.5%
neural statistician (Edwards & Storkey, 2017)	98.1%	99.5%	93.2%	98.1%
memory mod. (Kaiser et al., 2017)	98.4%	99.6%	95.0%	98.6%
MAML (ours)	$98.7 \pm 0.4\%$	$99.9 \pm 0.1\%$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$

	5-way Accuracy	
	1-shot	5-shot
MiniImagenet (Ravi & Larochelle, 2017)		
fine-tuning baseline	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
nearest neighbor baseline	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
matching nets (Vinyals et al., 2016)	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
meta-learner LSTM (Ravi & Larochelle, 2017)	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
MAML, first order approx. (ours)	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
MAML (ours)	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$

1. Learning to learn by gradient descent
by gradient descent
2. MAML
3. Reptile

3 Reptile

ИДЕЯ: обучить начальные параметры модели так,
(как у MAML) чтобы модель быстро адаптировалась к новой задаче.

+

использовать только производные 1-го порядка.

Reptile. Начало.

- Очевидный минус MAML:
 - В силу наличия вторых производных вычисления становятся ресурсоемкими
- Инсайт, пришедший авторам:
 - FOMAML (first-order MAML) -- тот же MAML, но с игнорированием вторых производных.

Reptile. Начало.

- Очевидный минус MAML:
 - В силу наличия вторых производных вычисления становятся ресурсоемкими
- Инсайт, пришедший авторам:
 - FOMAML (first-order MAML) -- тот же MAML, но с игнорированием вторых производных.

FOMAML довольно
хорошо себя показал.



Это сподвигло к исследованию мета-лернинг
алгоритмов, основанных на
градиентах 1-го порядка.

так появился Reptile.
(first-order gradient-based meta-learning algorithm)

Reptile. Алгоритм.


Algorithm 2 Reptile, batched version

Initialize θ


for iteration = 1, 2, ... **do**

 Sample tasks $\tau_1, \tau_2, \dots, \tau_n$

for $i = 1, 2, \dots, n$ **do**

 Compute $W_i = \text{SGD}(L_{\tau_i}, \theta, k)$  к шагов SGD, начиная с θ

end for

 Update $\theta \leftarrow \theta + \beta \frac{1}{n} \sum_{i=1}^n (W_i - \theta)$  градиент Reptile = $(\theta - W)/\alpha$

end for

Reptile vs FOMAML.

Пусть

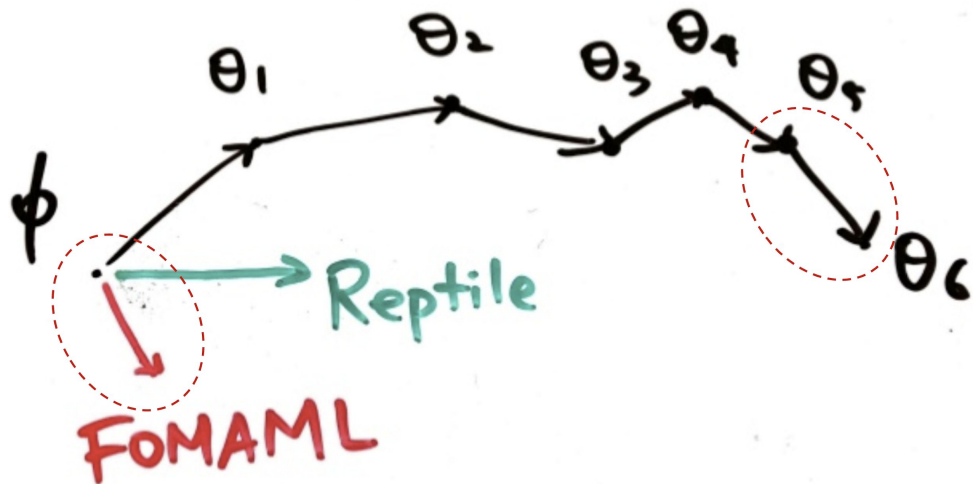
$$\theta_0 = \theta$$

$$\theta_1 = \theta_0 - \alpha \nabla_0^{(0)}$$

$$\theta_2 = \theta_1 - \alpha \nabla_1^{(1)} = \theta_0 - \alpha \nabla_0^{(0)} - \alpha \nabla_1^{(1)}$$

$$(*) \nabla_{FOMAML} = \nabla_1^{(1)}$$

$$\nabla_{Reptile} = (\theta_0 - \theta_2) / \alpha = \nabla_0^{(0)} + \nabla_1^{(1)}$$



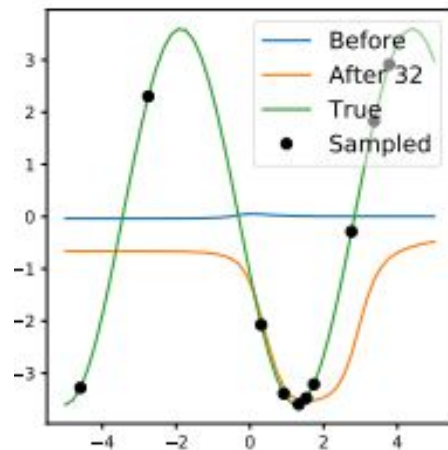
(*) градиент FOMAML -- последнее обновление градиента
(док-во на следующем слайде)

Reptile vs FOMAML.

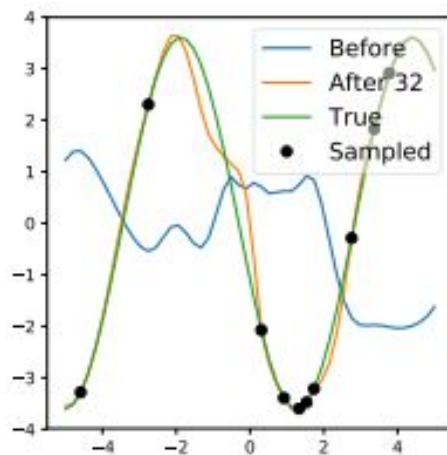
(*)

$$\begin{aligned}\theta' &= \theta - \alpha \nabla_{\theta} L(\theta) \\ \nabla_{FOMAML} &= \nabla_{\theta} L(\theta') = (\nabla_{\theta'} L(\theta')) \cdot (\nabla_{\theta} \theta') \\ &= (\nabla_{\theta'} L(\theta')) \cdot (\nabla_{\theta} (\theta - \alpha \nabla_{\theta} L(\theta))) \\ &\approx (\nabla_{\theta'} L(\theta')) \cdot (\nabla_{\theta} \theta) \\ &= \nabla_{\theta'} L(\theta')\end{aligned}$$

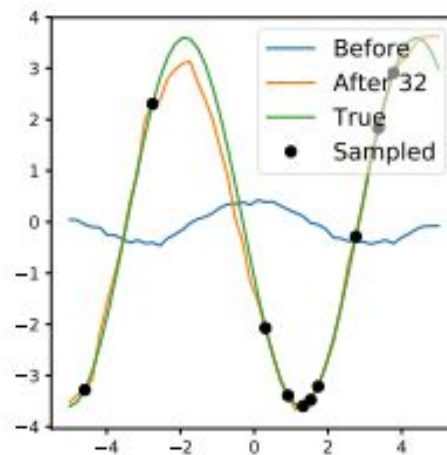
Reptile. Experiments.



(a) Before training



(b) After MAML training



(c) After Reptile training

Reptile. Выводы.

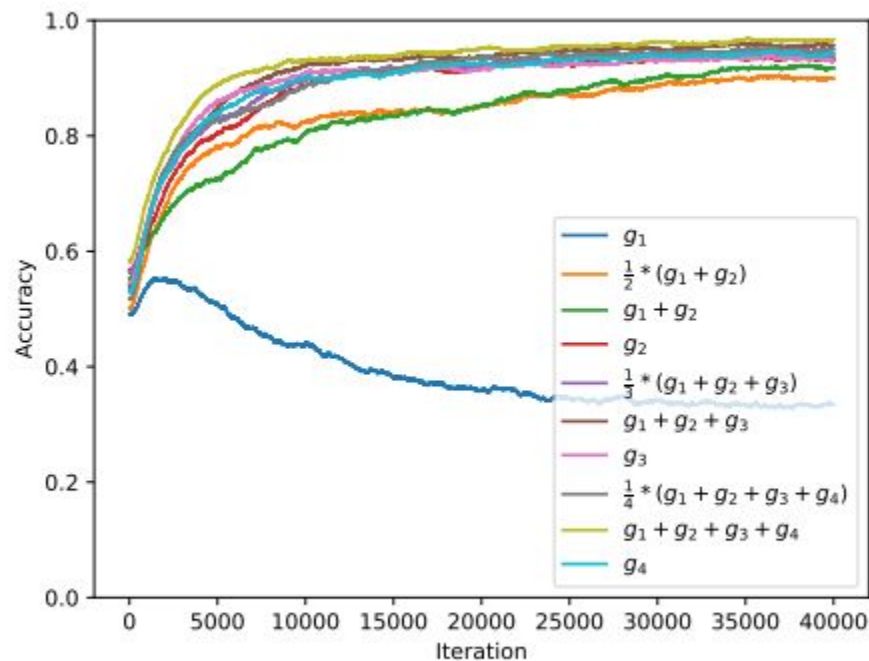


Figure 3: Different inner-loop gradient combinations on 5-shot 5-way Omniglot.

Вопросы

1. Что такое optimizer и optimizee? Как они между собой взаимодействуют. Нарисовать схему сети и обозначить их там.
2. В чем заключаются особенности(свойства, основная идея) MAML? Описать алгоритм.
3. Что такое FOMAML? Сходства и различия Reptile и FOMAML.

Источники

- <https://arxiv.org/pdf/1606.04474.pdf>
- <https://arxiv.org/pdf/1803.02999.pdf>
- <https://arxiv.org/pdf/1703.03400.pdf>
- <https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html#maml>
- [https://en.wikipedia.org/wiki/Meta_learning_\(computer_science\)](https://en.wikipedia.org/wiki/Meta_learning_(computer_science))
- <http://neerc.ifmo.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%B0-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5>