

Meta-learning

Марьин Никита, 171

Introduction

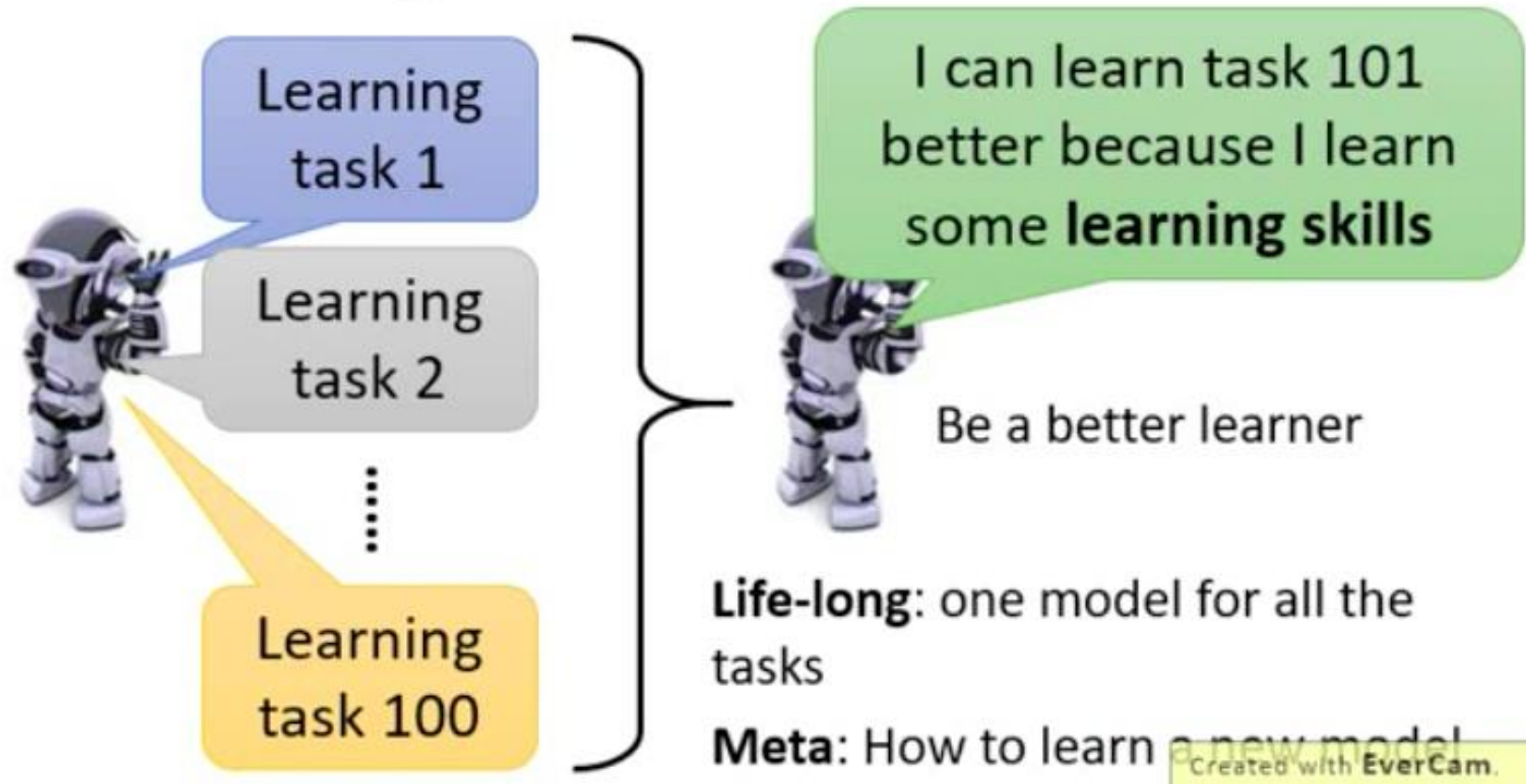
Task 1: speech recognition

Task 2: image recognition

⋮

Task 100: text classification

- Meta learning = Learn to learn



$$f^* = F(D_{train}) \quad D_{train} = (D_i, \theta_i)^{N_1}$$



Примеры задач мета-обучения

- ▶ Классификатор обучали на изображениях собак и велосипедов, давайте дообучим его определять еще и кошек
- ▶ Бот для игр, способный быстро обучиться новой игре
- ▶ Робот, выполняющий задачу на пригорке во время теста даже если он обучался на ровной поверхности

**Learning to learn by gradient
descent by gradient descent**

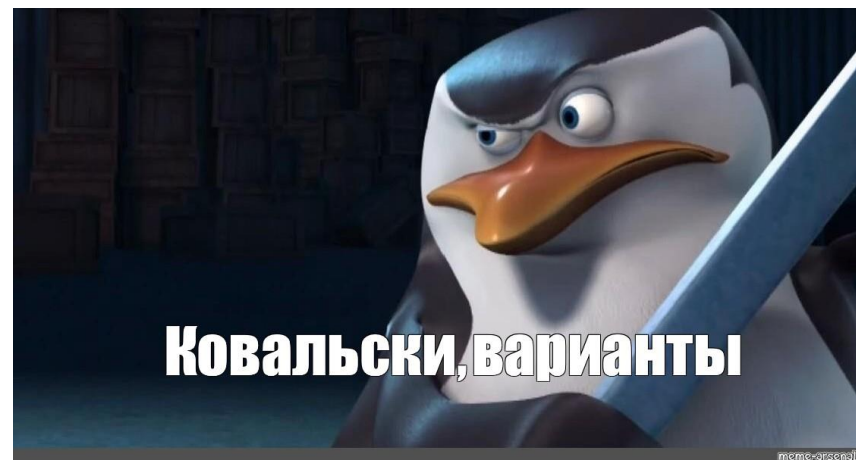
Есть обычная задача оптимизации : $\theta^* = \arg \min_{\theta \in \Theta} f(\theta)$

Простейший вариант решения :
градиентный спуск $\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$.

Плюсы : прост

Минусы:

- ▶ Такая оптимизация не использует информацию о вторых производных для корректировки шага, например.
- ▶ Вручную разработанные правила обновления

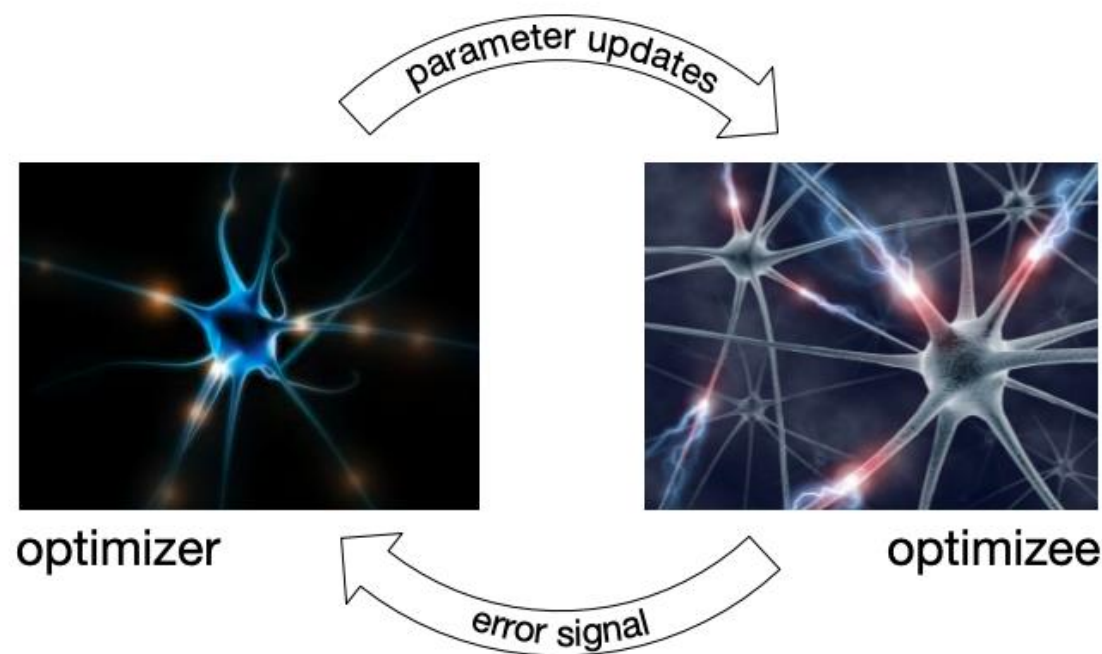


Параметры будут
изменяться по динамическому
правилу, их будет менять
RNN

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

Optimizer - RNN,
 $g()$, ее параметры -
 ϕ

Optimizee -
Оптимизируемая
функция f



Как понять, что optimizer хороший?

No Free Lunch теорема

Возникает естественный вопрос - существует ли некоторый лучший эволюционный *алгоритм*, который дает всегда лучшие результаты при решении всевозможных проблем?

NFL теорема. Для любой пары алгоритмов a_1 и a_2 имеет место равенство

$$\sum_f P(d_m^y | f, m, a_1) = \sum_f P(d_m^y | f, m, a_2).$$

Сумма условных вероятностей получения данного частного решения при выбранной функции потерь после m -итерации алгоритма для двух алгоритмов равны.

Оценка качества optimizer

$$\mathcal{L}(\phi) = \mathbb{E}_f \left[f(\theta^*(f, \phi)) \right]$$

Заметим, что зависит только от искомым параметров θ^*

Изменим немного функцию качества, добавив положительные веса (для простоты можно взять их единичными) и используя результат всех предыдущих итераций.

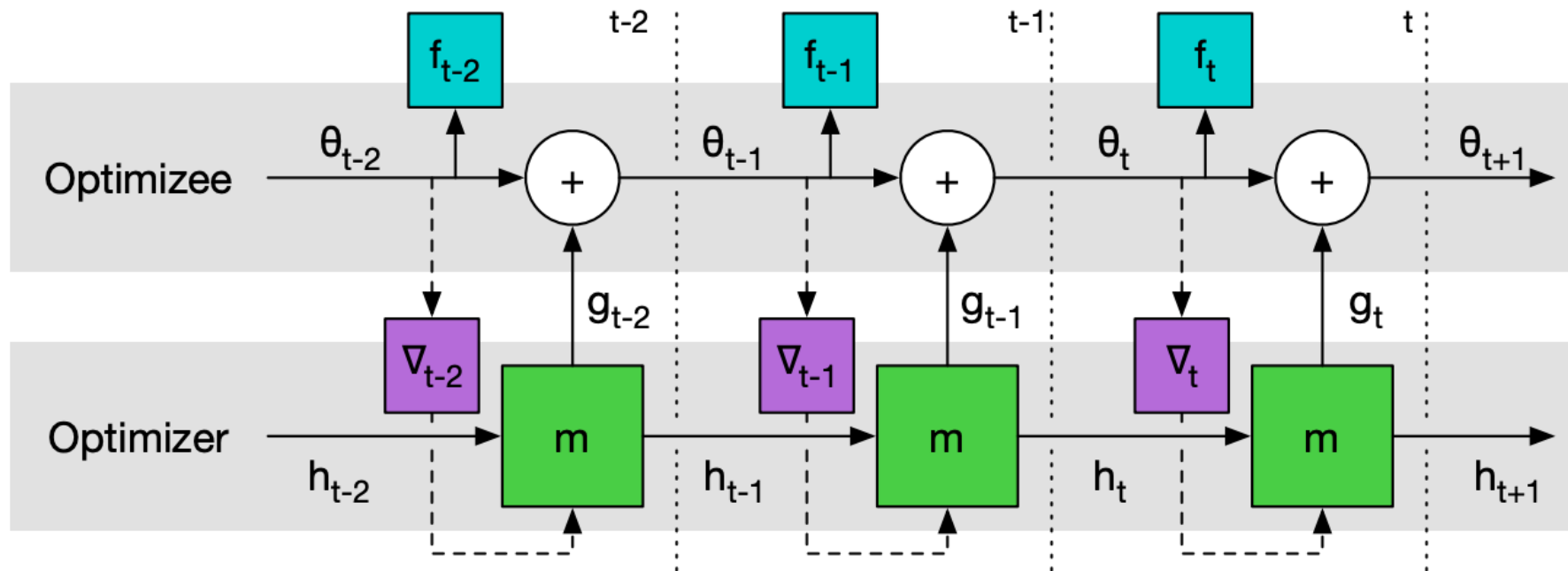
$$\mathcal{L}(\phi) = \mathbb{E}_f \left[\sum_{t=1}^T w_t f(\theta_t) \right] \quad \text{where}$$

Для чего изменили функцию? - Для того чтобы Backpropagation Through Time (BPTT) имел смысл.

$$\begin{aligned} \theta_{t+1} &= \theta_t + g_t, \\ \begin{bmatrix} g_t \\ h_{t+1} \end{bmatrix} &= m(\nabla_t, h_t, \phi). \\ \nabla_t &= \nabla_{\theta} f(\theta_t) \end{aligned}$$

Оптимизация функции $\mathcal{L}(\phi)$

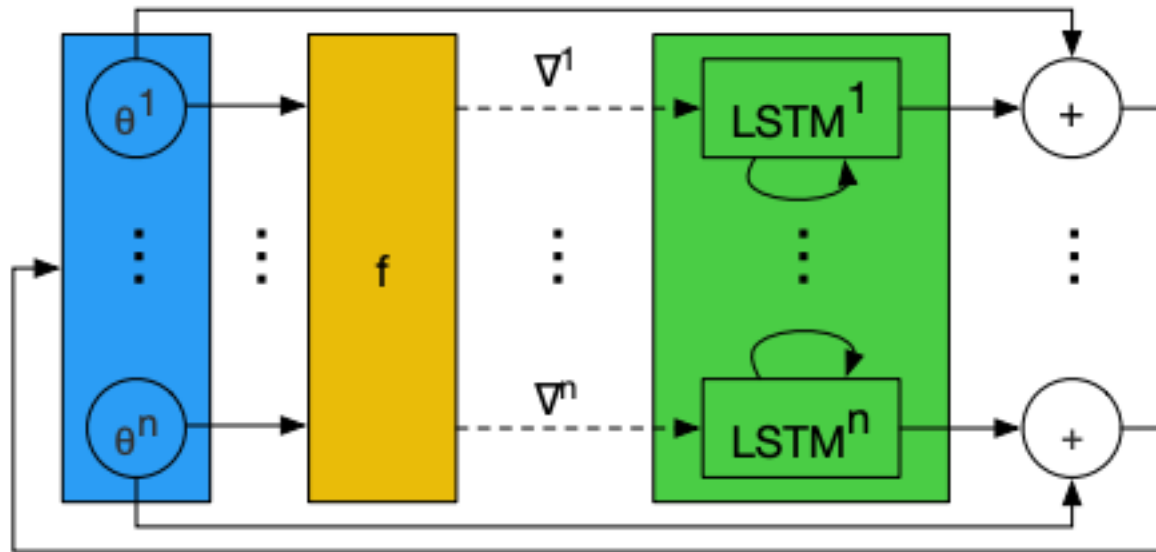
Будем использовать градиентный спуск по ϕ , в предположении, что $\partial \nabla_t / \partial \phi = 0$
 $\partial \mathcal{L}(\phi) / \partial \phi$ можно посчитать выбрав функцию f и запустить BPTT:



Покоординатный LSTM optimizer


Наша цель оптимизировать как можно больше параметров (десятки тысяч). Оптимизация в этом масштабе с RNN не представляется возможной, так как потребовалось бы огромное количество скрытых состояний и огромное количество параметров.

Решение : обращаться к каждой координате вектора параметров θ поотдельности



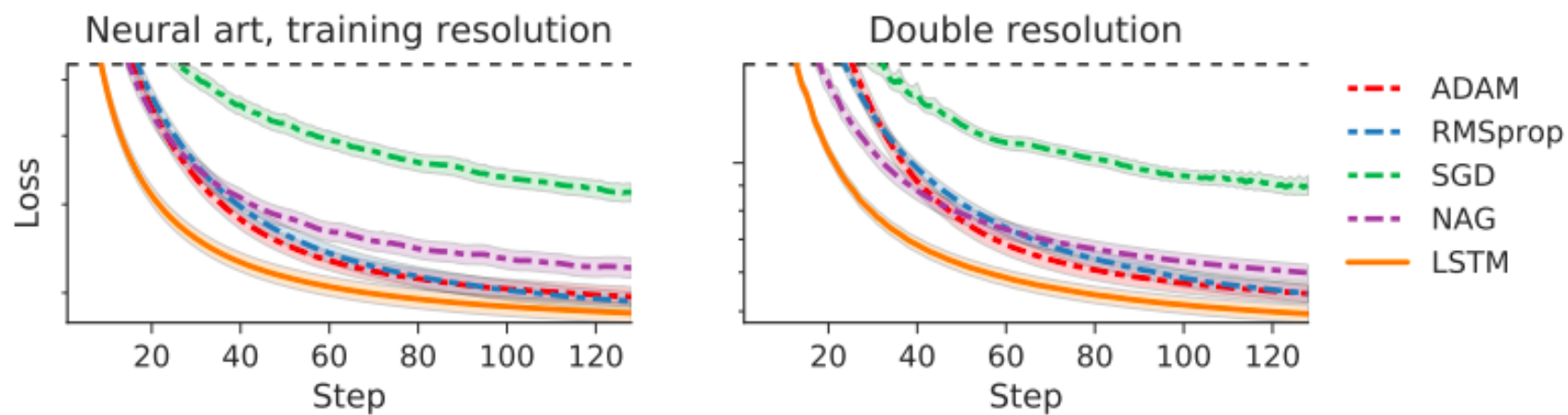
Препоцесссинг входных данных оптимайзера

Из-за разного масштаба входных параметров, сеть смотрит на большие по модулю значения и концентрируется на них. Поэтому для улучшения качества модели предлагается использовать такое преобразование:

$$\nabla^k \rightarrow \begin{cases} \left(\frac{\log(|\nabla|)}{p}, \text{sgn}(\nabla) \right) & \text{if } |\nabla| \geq e^{-p} \\ (-1, e^p \nabla) & \text{otherwise} \end{cases}$$


Эксперименты. Neural Art

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



Model-Agnostic Meta-Learning (MAML)

Алгоритм MAML

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**
-

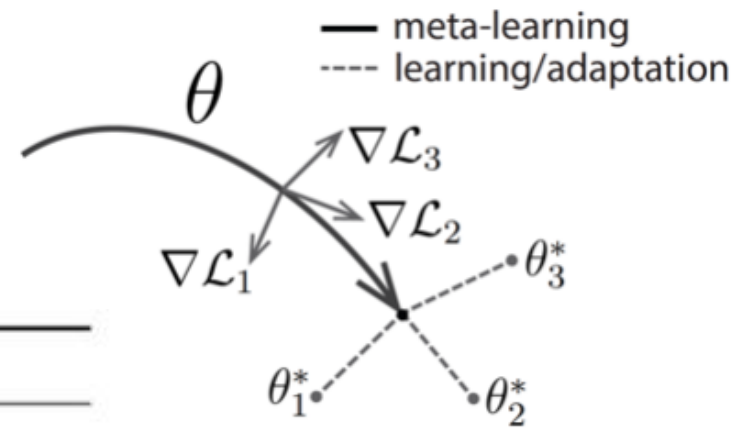
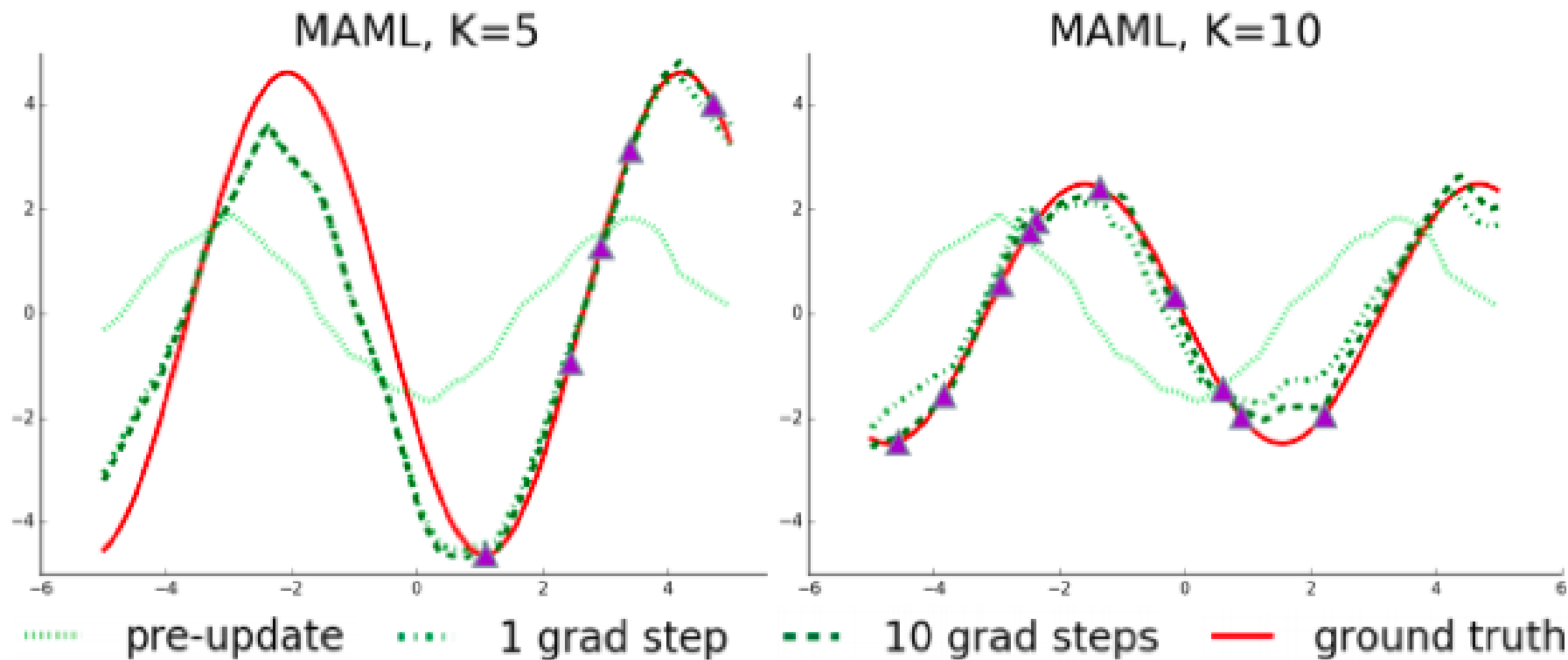


Diagram of the MAML approach.

Суть алгоритма заключается в поиске наилучшей начальной инициализации параметра, из которого за малое количество итераций градиентного спуска можно будет достигнуть оптимальных параметров для любых задач.

MAML для линейной регрессии

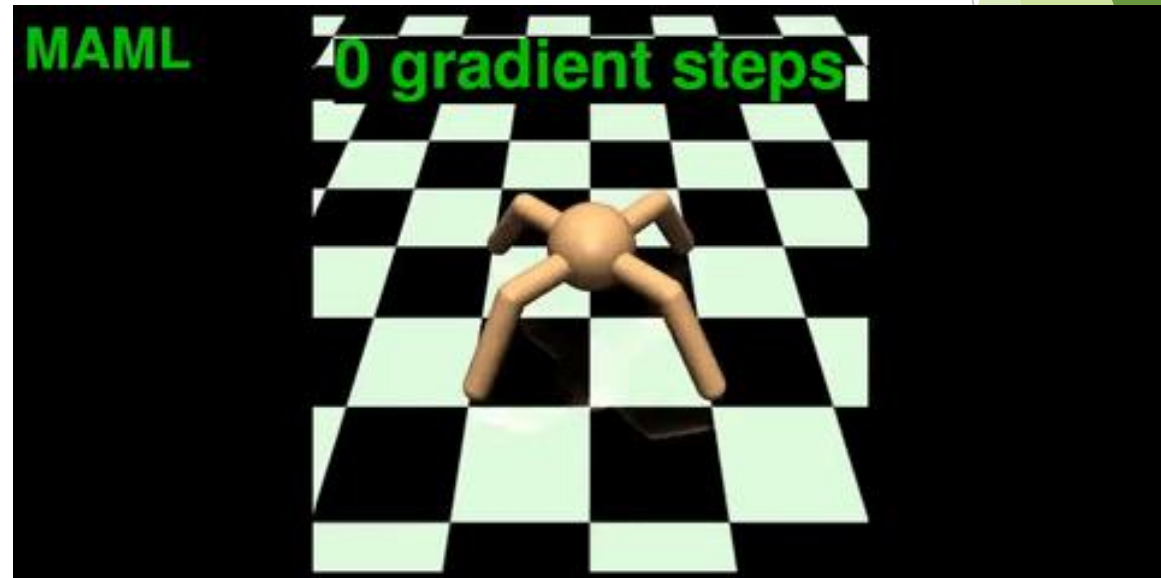
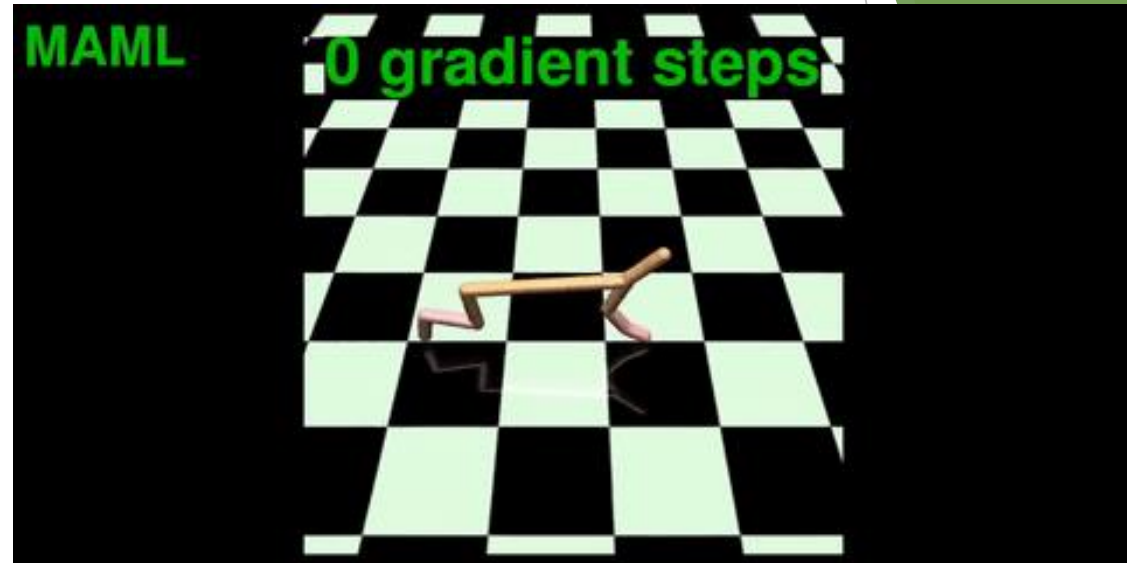


Примеры применения MAML

Целью было адаптировать поведение роботов для различных целей. Для этого использовали комбинацию из метода MAML и обучения с подкреплением.

Плюсами такого подхода являются: с его помощью алгоритм становится более адаптивным, так же он может использоваться для большого спектра моделей, ведь единственное требование наличие производных 1-го и 2-го порядков.

Минусом является то, что используются 2 производные.



Reptile: A Scalable Meta-Learning Algorithm

Алгоритм Reptile

Algorithm 2 Reptile, batched version

Initialize ϕ

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

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

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

 Compute $W_i = \text{SGD}(L_{\tau_i}, \phi, k)$

end for

 Update $\phi \leftarrow \phi + \epsilon \frac{1}{k} \sum_{i=1}^n (W_i - \phi)$

end for

Плюсом является то, что данный алгоритм не требует вычисления вторых производных, в отличие от MAML

Здесь выполняется SGD, k итераций, начало в θ

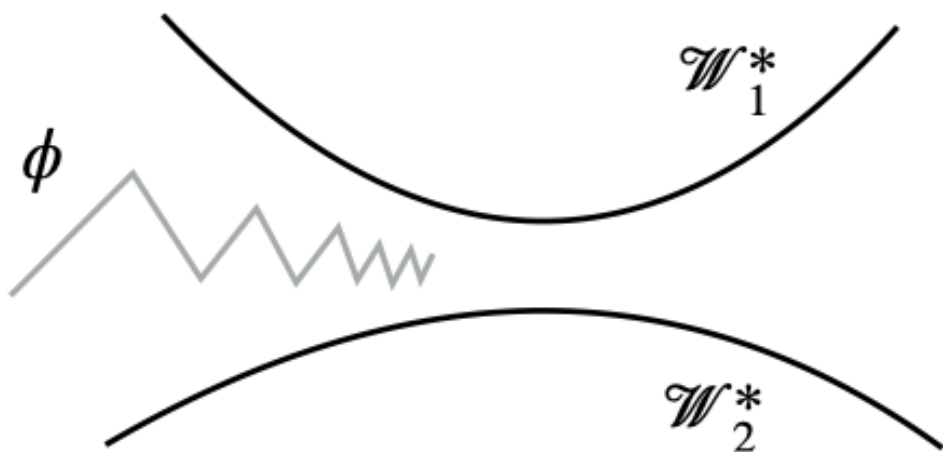
Градиент первого порядка:
 $(\phi - W)/\alpha$

Почему Reptile вообще работает?

$$\mathbb{E}_\tau [\text{SGD}(L_\tau, \phi, k)] \stackrel{?}{=} \text{SGD}(\mathbb{E}_\tau [L_\tau], \phi, k)$$

На практике было доказано, что Reptile так же минимизирует Loss, как и MAML.

Доказано это было с помощью разложения Loss в ряд Тейлора для обоих алгоритмов, было показано что Reptile приближает решение довольно точно.



На рисунке слева изображены два оптимальных многообразия, и последовательность сохраняет расстояние между ними.

Вопросы

- ▶ Что такое мета-обучение, чем оно отличается от обычного обучения, приведите примеры задач мета-обучения.
- ▶ Что такое optimizer, приведите формулу подсчета Loss оптимизатора.
- ▶ Какой препроцессинг входных данных оптимизатора был рассказан, для чего его нужно использовать?
- ▶ Приведите алгоритм работы и описание MAML или REPTILE на выбор.

СПИСОК ИСТОЧНИКОВ

- ▶ https://d4mucfpksywv.cloudfront.net/research-covers/reptile/reptile_update.pdf
- ▶ <https://openai.com/blog/reptile/#jump>
- ▶ <https://arxiv.org/pdf/1606.04474v2.pdf>
- ▶ <https://towardsdatascience.com/model-agnostic-meta-learning-maml-8a245d9bc4ac>
- ▶ <http://runopti.github.io/blog/2016/10/17/learningtolearn-1/>