MAML (Model-Agnostic Meta-Learning)

MAML (Model-Agnostic Meta-Learning) [1]は、近年最もよく使われているメタ学習アルゴリズムの1つで、メタ学習の研究分野に大きなブレークスルーをもたらしたとされています。MAMLは、次の2つの特徴を持ちます。

- Model-Agnostic
  微分可能である以外、モデルや損失関数の具体的な形式を仮定しない

- Task-Agnostic
  回帰、分類、強化学習など、様々なタスクに適用できる

このように適用条件に制限が少なく、汎用性の高いメタ学習アルゴリズムとなっています。本解説プリントでは、MAMLの提案論文 (Chelsea Finn+, Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, arXiv:1703.03400) を元に、MAMLの基本的なアイディア、アルゴリズムについて説明します。

♣ 背景

近年の深層学習に関する技術の発展は目覚ましく、画像認識や囲碁ゲームなどのタスクにおいて計算機は人間を超える性能を示すようになっています。しかしながら、このような深層学習モデルが高い性能を発揮するためには、大量のラベル付き訓練データが必要となります。具体例として、いくつかの画像認識のプロジェクトで必要となったデータ数 (画像数) をまとめてみると、以下の表のようになります。

プロジェクト名	タスク	データ数
MNIST	手書き文字認識	70,000
CIFAR-10	物体認識	60,000
MegaFace	顔認識	5,700,000
FaceNet	顔認識	450,000
MITCSAIL	画像アノテーション	897,000

学習する画像数が少ないと、「過学習」と呼ばれる問題が起こることが知られています。過学習とは、訓練データ (教師データ) のみを学習し過ぎてしまった状態で、少しでも教師データと異なると正しい判定ができなくなってしまうことです。この過学習を防ぐために、多くの深層学習モデルでは大量のラベル付き訓練データが必要となるのです。

しかしながら、一般に大量のラベル付きデータを用意するのはコストがかかり非常に大変です。そのため、少数のデータからどのように大量のパラメータを有する深層学習モデルを効率的

[1]そのままアルファベット読みする他にも、「マムル」や「マーマル」のように発音されることもあります。

に学習させるかという課題が、近年重要となってきています。このような問題に対し、有用な事前知識をモデルに組み込むことが効果的であることが知られています。特に、複数の物事に対する共通の概念を自ら学習し、得られた知識を未知の問題に適用する **メタ学習 (meta-learning)** という手法が大きな注目を集めています。メタ学習では類似したタスクの集合から、機械学習モデルが自らタスク間に **メタ知識 (meta-knowledge)** と呼ばれる共通の概念を学習することを目的としています。これは、人間の学習方法とよく似ています。人間は様々な経験から学習し、その経験を未知の状況に生かすことができます。例えば、英語とイタリア語が得意な人がいれば、その人は比較的素早くスペイン語を習得できるでしょう。また将棋が得意な人は、チェスの上達も早いはずです。メタ学習では、このような能力を機械学習モデルに獲得させることを目指すのです。

深層学習モデルに対してメタ学習を適用した代表的な手法が、この解説プリントのテーマである **MAML (Model-Agnostic Meta-Learning)** と呼ばれる手法です。この MAML のアイディアはとてもシンプルです。詳しくは後ほど説明しますが、大まかにはメタ知識としてタスク間に共通の最適な初期パラメータを学習することにより、少数データしか存在しない新たなタスクに対しても、少ないパラメータ更新で効率的に適応させることを目指します。提案論文ではこの手法の有効性も検証されており、画像分類、回帰、強化学習などのタスクにおいてパラメータ調整が効率化されることが実証されています。

MAML は、メタ学習を大きく変えたと言われています。従来、メタ学習は非常に計算コストが高くなってしまうため、実装には高いハードルがありました。そもそも深層学習で学習を行うためには前述したように大量のデータが必要となりますが、メタ学習ではタスク間に共通の概念を学習するために従来の手法ではより多くのデータを取り扱う必要があるため、計算コストが非常に高くなってしまうのです。しかしながら、MAML ではモデルパラメータの初期値を最適化することでこのメタ学習の高コストの計算を大幅に減らすことを可能とし、現在では多くのメタ学習で使われる有名なアルゴリズムになっています。

♣ MAML のアイディア

それでは、MAML のアイディアを具体的に説明していきます。その前に、通常のニューラルネットワークの学習方法について確認しておきましょう。ニューラルネットワークは**パラメータ (parameter)** と呼ばれる数値の集合を持っており、ネットワーク上でどのように情報が伝わるかはそのパラメータに依存しています。パラメータを変えることで最終的な出力 (予測値) が変わるので、どのようにパラメータを設定するかがニューラルネットワークにとって重要な課題となります。

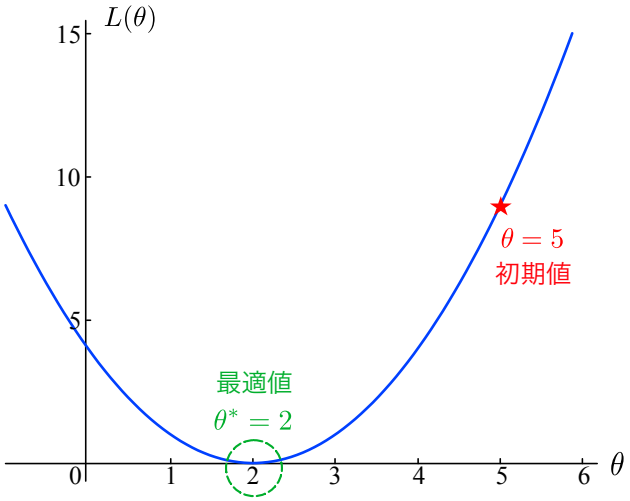
ニューラルネットワークの学習とは、予測値と予め分かっている正解値との「誤差」がなるべく小さくなるようにネットワークのパラメータを最適化することです。一般に、この「誤差」を測る指標のことを**損失関数 (loss function)** と呼び、タスクに応じて様々な関数が用いられます。例えば回帰問題に対しては平均二乗誤差、分類問題に対してはクロスエントロピーなど

が用いられます。いずれにしても、タスクに応じて適切な損失関数を設定し、その損失関数を最適化することでパラメータを学習することができます。

では、具体的にどのようして目的関数は最適化されるのでしょうか？最適化の代表的な方法の 1 つが **勾配降下法 (gradient descent)** と呼ばれる方法です。この方法のイメージを掴むために、パラメータを 1 変数 θ として損失関数が

$$L(\theta) = (\theta - 2)^2$$

のような 2 次関数で与えられる簡単な場合を考えてみましょう[2]。



勾配降下法で最適パラメータ θ^* を求める場合、まずパラメータの初期値 θ を乱数を用いてランダムに与えます。上図では、$\theta = 5$ という初期値が与えられたとしています。この初期パラメータを更新するために、$\theta = 5$ での勾配 (微分係数) を計算します。

$$\left.\frac{dL(\theta)}{d\theta}\right|_{\theta=5} = 2(\theta - 2)|_{\theta=5} = 6$$

この勾配は、θ を増加させた際に損失関数 $L(\theta)$ が増加する方向を意味しています。今は $L(\theta)$ の値を小さくしたいので、この勾配の逆方向に θ を変化させれば良いことになります。具体的には、勾配を今の θ の値 $(= 5)$ から引けば良くて

$$\theta \leftarrow \theta - \eta \left.\frac{dL(\theta)}{d\theta}\right|_{\theta=5}$$

となります[3]。ここで、パラメータ θ の一度の更新量の幅を調整するために、**学習率 (learning rate)** と呼ばれる係数 η を勾配に乗じています[4]。例えば、この学習率を $\eta = 0.8$ とすれば、初

[2]実際のニューラルネットワークの目的関数は、多次元でもっと複雑な関数形をしています
[3]上式の $\leftarrow$ は、右の値を θ に代入 (θ の値を更新) することを表します。
[4]学習率は、ステップ数に応じて変化させる場合もありますが、ここでは簡単のため定数としています。

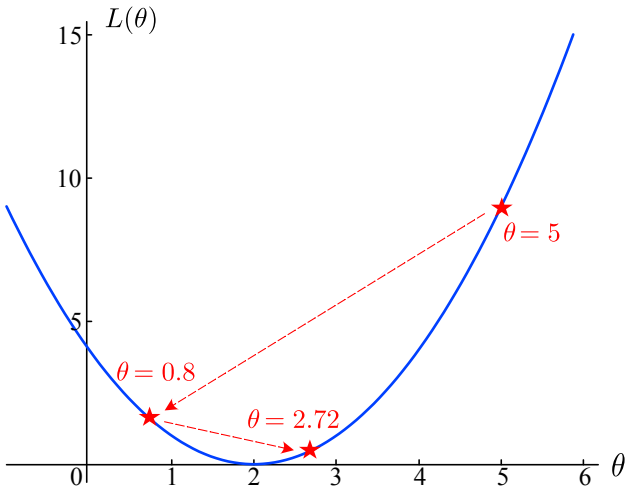
期値 $\theta = 5$ は

$$\theta \to 5 - 0.8 \times 6 = 0.8$$

と更新されます。そして更新された点で再び勾配を計算し、パラメータの更新を繰り返していきます。$\theta = 0.8$ での勾配は、$2(\theta - 2)|_{\theta=0.8} = -2.4$ となるため、

$$\theta \to \theta - \eta \left.\frac{dL(\theta)}{d\theta}\right|_{\theta=0.8} = 0.8 - 0.8 \times (-2.4) = 2.72$$

と更新されます。これを図にすると次のようになります。



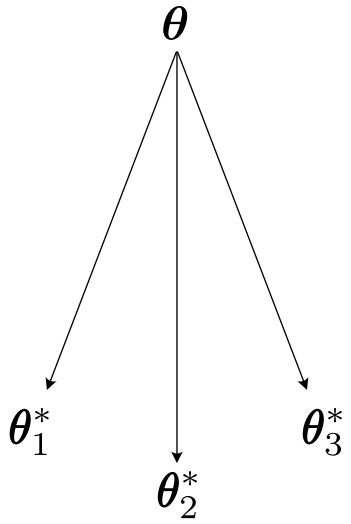
このように各点での勾配を用いて、$5 \to 0.8 \to 2.72 \to \cdots$ とパラメータ更新していくことで、徐々に損失関数 $L(\theta)$ の最適値 $(\theta^* = 2)$ に近づいていくことがわかります。これが勾配降下法の基本的なアイディアです。一般に、パラメータ θ は 1 次元の変数ではなく、多次元の変数 (ベクトル) $\boldsymbol{\theta}$ となります。その場合には、今の 1 次元の場合を拡張して、

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}) = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

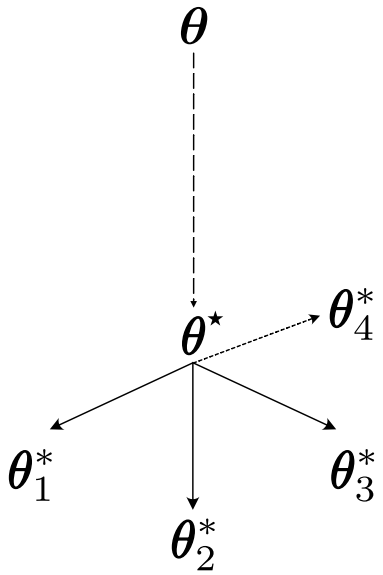
とパラメータ更新されることになります (この式のことを更新式と言います)。ここで $\nabla_{\boldsymbol{\theta}}$ は、1 次元の場合の微分を多次元に拡張したもので、**勾配 (gradient)** と呼ばれます[5]。

さて、通常の深層学習における学習方法を確認したところで、本題の MAML に戻りましょう。ここでは、MAML のイメージを掴むために次の簡単な例を考えてみましょう：類似した 3 つのタスク T_1、T_2、T_3 が与えられたとします。そしてモデルパラメータ $\boldsymbol{\theta}$ にランダムな初期値を与え、タスク T_1 に関して勾配降下法によりネットワークの学習を行なったとします。その結果、タスク T_1 に対する最適パラメータ $\boldsymbol{\theta}_1^*$ を得ることができます。同様のことを他のタスクに対しても行い、それぞれのタスクに対する最適パラメータ $\boldsymbol{\theta}_2^*$、$\boldsymbol{\theta}_3^*$ を得たとします。この作業を図で表すと次のようになります。

[5]「勾配　ベクトル解析」などのキーワードで検索すれば、この勾配に関する解説が見つかります。



このように、初期値 $\boldsymbol{\theta}$ がランダムであっても、そこからそれぞれのタスクに関して勾配降下法で学習を行うことで (上図での矢印が学習に対応)、最適解 $\boldsymbol{\theta}^*_{1,2,3}$ が得られるでしょう。しかしながら、上図のような初期値 $\boldsymbol{\theta}$ から学習を開始するのはあまり賢い方法とは言えません。なぜなら、初期値をいい加減なところに取るのではなく 3 つのタスクの近傍に取るようにすれば ($\boldsymbol{\theta}^\star$)、より少ない学習ステップで効率的に最適解を得ることができるからです (下図参照)。



さらに、3 つのタスクから最適な初期値 $\boldsymbol{\theta}^\star$ を見つけておけば、新たな類似タスク T_4 に対しても、$\boldsymbol{\theta}^\star$ から学習を開始することで素早くパラメータを更新し、モデルをタスクに適応させることが可能になるはずです。これが MAML の基本的なアイディアとなります。つまり MAML では、新たなタスクに対して数ステップの学習で最適化できる、タスク普遍な最適な初期モデルパラメータ $\boldsymbol{\theta}^\star$ を見つけることが目的となります。この最適な初期パラメータ $\boldsymbol{\theta}^\star$ こそが複数のタスク間での「共通の概念」、すなわちメタ知識にあたると解釈できます。

♣ MAML のアルゴリズム

では、MAML は具体的にどのようにしてタスク普遍な最適な初期値 $\boldsymbol{\theta}^\star$ を見つけるのでしょうか？ このセクションでは、MAML の具体的なアルゴリズムについて説明していきます。以下ではパラメータ $\boldsymbol{\theta}$ を持つ機械学習モデル $f_{\boldsymbol{\theta}}$ を考えます。このパラメータ $\boldsymbol{\theta}$ がメタ学習のメタパラメータとなっています。そしてタスクの集合を $\mathcal{T}$ として、集合内のタスクに関する確率分布を $p(\mathcal{T})$ とします。この時、MAML のアルゴリズムは次のようになります：

1. パラメータ $\boldsymbol{\theta}$ をランダムに初期化します。

2. タスク集合 $\mathcal{T}$ からいくつかのタスクを確率分布 $p(\mathcal{T})$ に従ってサンプリングします[6]。数式では、$T_i \sim p(\mathcal{T})$ $(i = 1, 2, \cdots)$ と表現できます。

3. モデル $f_{\boldsymbol{\theta}}$ をタスク T_1 に適応させます。具体的には、勾配降下法を用いて、パラメータ $\boldsymbol{\theta}$ をタスク T_1 の損失関数 L_{T_1} を最小化するように更新します。

$$\boldsymbol{\theta}_1' = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L_{T_1}(f_{\boldsymbol{\theta}})$$

   α は学習率 (ハイパーパラメータ) となります。また、以下では表記を簡単にするために 1 回の勾配ステップでパラメータを更新していますが、複数回の勾配ステップの場合でも同様の議論が成り立ちます[7]。

4. 3. と同様の作業をバッチ内の残りのタスクに対しても行い、$\boldsymbol{\theta}_i'$ $(i = 1, 2, \cdots)$ を得る。

5. パラメータの初期値 $\boldsymbol{\theta}$ を更新します (メタ学習)。具体的には、各タスクでパラメータを勾配降下により $\boldsymbol{\theta}_i'$ と更新した時、更新後の損失関数の値 $L_{T_i}(f_{\boldsymbol{\theta}_i'})$ の和が小さくなるように初期値 $\boldsymbol{\theta}$ を更新します。損失関数の値はモデルの予測値と正解値のズレを表しているため、各々のタスクでのズレの合計を減らすように初期値 $\boldsymbol{\theta}$ を学習することで、より良い位置に初期値を更新することができます。この最適化の作業を数式で表現すると、

$$\min_{\boldsymbol{\theta}} \sum_i L_{T_i}(f_{\boldsymbol{\theta}_i'}) = \min_{\boldsymbol{\theta}} \sum_i L_{T_i}(f_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L_{T_i}(f_{\boldsymbol{\theta}})})$$

   となります。ここで注意すべきなのは、最小化すべき損失関数は $\boldsymbol{\theta}_i'$ に基づいて計算されますが、この最適化問題は元のモデルパラメータ $\boldsymbol{\theta}$ に関する最適化問題であるという点です。先ほど MAML のアイディアを説明しましたが、このステップで新たなタスクに対しても少しの学習で適応できるような、最適な初期パラメータ $\boldsymbol{\theta}$ を学習しているのです。そして、この $\boldsymbol{\theta}$ に関する最適化は、ここまでの勾配降下法と同様にして、

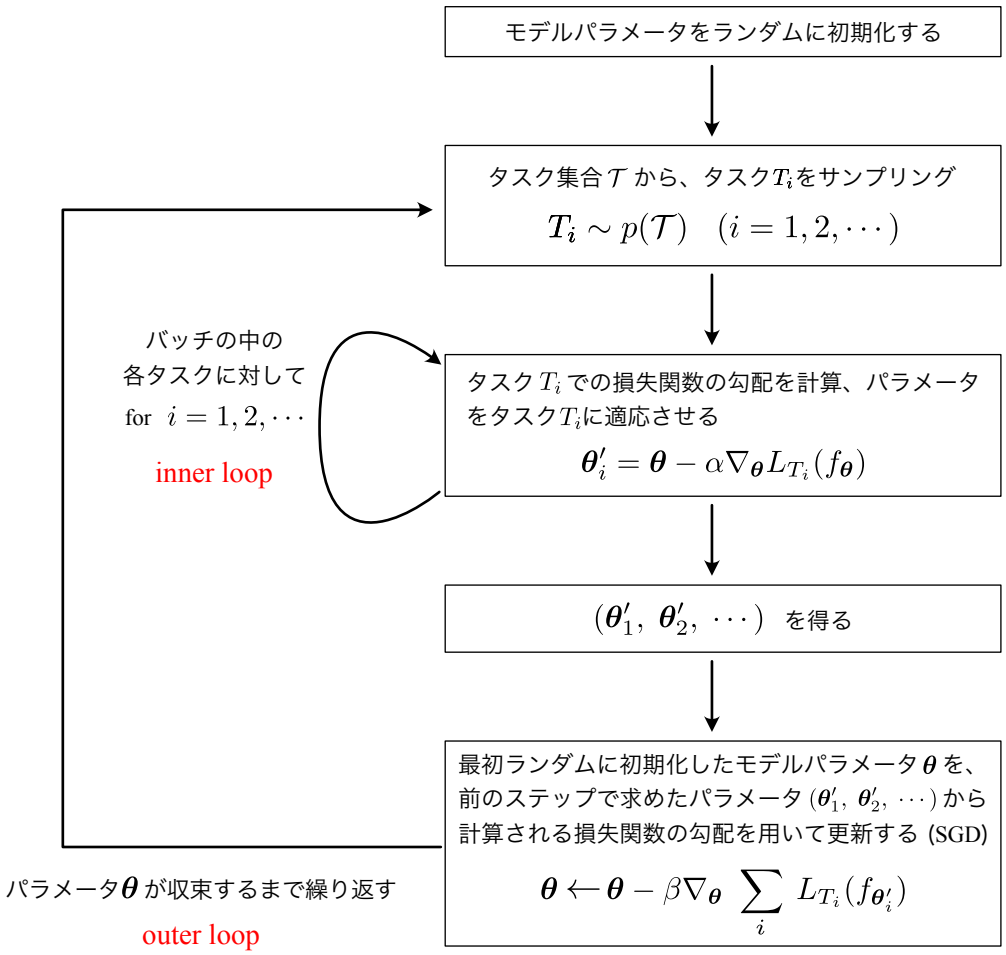
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_i L_{T_i}(f_{\boldsymbol{\theta}_i'})$$

[6] このようにサンプリングされた集合のことを、一般に**バッチ (batch)** と呼びます。
[7] 一般的には、5~10 回程度の更新が行われます。

とパラメータ更新することで実現できます[8]。単に 3. のステップで $L_{T_1}(f_{\boldsymbol{\theta}})$ としていた部分を、ここでは $\sum_i L_{T_i}(f_{\boldsymbol{\theta}'_i})$ を最小化したいので、それに置き換えただけです。このようなパラメータ更新の方法を、勾配降下法の中でも特に **確率的勾配降下法 (SGD: stochastic gradient descent)** と呼びます。

6. 手順 2. から手順 5. を、$\boldsymbol{\theta}$ がタスク普遍な最適な初期値 $\boldsymbol{\theta}^*$ に収束するまで繰り返す。

上記の流れを図にまとめてみると、次のようになります。



このように全体の構造を見直してみると、MAML のアルゴリズムは、大きく 2 つのループから構成されていることが分かります。それぞれ inner loop、outer loop と呼ばれ、

inner loop サンプリングしたタスク T_i に対する損失関数の勾配から、$\boldsymbol{\theta}'_i$ を求めるループ

outer loop inner loop で求めた $\boldsymbol{\theta}'_i$ を用いて、初期パラメータ $\boldsymbol{\theta}$ を更新するループ

となっています。

[8] β は、メタ学習における (メタ) 学習率 (ハイパーパラメータ) です。

このセクションの最後に、いくつか MAML のアルゴリズムに関してコメントしておきます。

- パラメータ $\boldsymbol{\theta}$ は、確率的勾配降下法 (SGD) を用いて

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_i L_{T_i}(f_{\boldsymbol{\theta}_i'}) = \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_i L_{T_i}(f_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L_{T_i}(f_{\boldsymbol{\theta}})})$$

  と更新されました。この時、等号の右側の式を見れば分かりますが、勾配の勾配として 2 次勾配を計算します[9]。そのため、勾配の更新回数に比例してメモリと計算が増大することになり、この部分の計算コストが非常に高くなるという問題があります。この問題に対する解決方法については、最後のセクションで簡単に説明します。

- ここまでアルゴリズムを見てきて分かるかと思いますが、MAML では具体的なモデルやタスクを仮定しません。仮定しているのは、モデルがパラメータ $\boldsymbol{\theta}$ で記述できること、そして損失関数が $\boldsymbol{\theta}$ に関して滑らかな関数になっている、すなわち微分可能であることのみです。この意味で、MAML は冒頭で述べたように Model-Agnostic、Task-Agnostic なメタ学習アルゴリズムと言えます。

♣ MAML の性能

このセクションでは、MAML の性能を確かめるために、原論文で行われている検証結果を見てみることにします。メタ学習では、数個の訓練データのみ学習して新しいタスクを解く Few-shot learning により汎化性能を測ることが一般的です。下の表は、Few-shot learning によるクラス分類タスクの精度を、MAML と他の手法 (Siamese nets や matching nets など) で比較した結果となります。

Table 1. Few-shot classification on held-out Omniglot characters (top) and the MiniImagenet test set (bottom). MAML achieves results that are comparable to or outperform state-of-the-art convolutional and recurrent models. Siamese nets, matching nets, and the memory module approaches are all specific to classification, and are not directly applicable to regression or RL scenarios. The ± shows 95% confidence intervals over tasks. Note that the Omniglot results may not be strictly comparable since the train/test splits used in the prior work were not available. The MiniImagenet evaluation of baseline methods and matching networks is from Ravi & Larochelle (2017).

	5-way Accuracy		20-way Accuracy	
Omniglot (Lake et al., 2011)	1-shot	5-shot	1-shot	5-shot
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MAML, no conv (ours)	**89.7 ± 1.1%**	**97.5 ± 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 ± 0.4%**	**99.9 ± 0.1%**	**95.8 ± 0.3%**	**98.9 ± 0.2%**

提案論文 (C. Finn+, arXiv:1703.03400) より転載

ここで、5-way や 20-way はそれぞれ 5 クラス、20 クラスからサンプルしていることを表し、1-shot や 5-shot はそれぞれ 1 枚、5 枚の画像をサンプルして学習していることを意味します。そして、データセットには Omniglot (平仮名、片仮名、ラテン文字、ギリシャ文字など 50 種類の

[9] $\nabla_{\boldsymbol{\theta}}$ で微分される部分に、$\nabla_{\boldsymbol{\theta}}$ の勾配が含まれています。

アルファベット、1623 種類の文字からなるデータセット) が用いられています。上の表で太字になっている部分が、MAML の結果です。他の既存の手法に比べて高い精度を出していることが読み取れます。

尚、ここでは分類問題に関する性能の結果のみを見ましたが、提案論文では他にも回帰問題や強化学習といったタスクでも検証を行なっていますので、是非参照してみて下さい。

♣ MAML に対する近似

ここまでは、MAML が優れたメタ学習の手法であることを説明してきましたが、実は大きな問題を抱えています。それは、2 階の勾配計算が必要なため計算コストが非常に高くなり、実用的には inner loop のステップ数を大きくできないという問題です。この問題により、実はうまく学習するのは結構難しい手法となっています。この問題を解決するために、計算コストを減らす改良案 (近似方法) がいくつか提案されています。代表的なものが次の 2 つです。

- First-order MAML (FOMAML) [MAML の提案論文で提案されている手法]
  2 次以上の勾配を無視する MAML、高次の微分が不要となるため計算コストを大幅に抑えることができる。

- Reptile [A. Nichol+, On First-order meta-learning Algorithm, arXiv:1803.02999]
  inner loop を真面目に逆伝播するのをあきらめ、学習前後のパラメータの差のみをみる。

実は、2 次以上の勾配を無視した FOMAML でも、(何故だかはっきりとは分かりませんが[10]) かなり高い精度が出せることが知られています。下の表は、Few-shot learning によるクラス分類タスクの精度を、既存の手法、MAML、FOMAML で比較したものです。

	5-way Accuracy	
MiniImagenet (Ravi & Larochelle, 2017)	1-shot	5-shot
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)	$\mathbf{48.07 \pm 1.75\%}$	$\mathbf{63.15 \pm 0.91\%}$
MAML (ours)	$\mathbf{48.70 \pm 1.84\%}$	$\mathbf{63.11 \pm 0.92\%}$

提案論文 (C. Finn+, arXiv:1703.03400) より転載

太字になっている部分が、FOMAML(下から 2 つ目) と MAML(1 番下) の結果で、他の既存の手法と比べて精度が高くなっています。そして、FOMAML と MAML を比べると、ほとんど精度が同じであることが分かります。当然、FOMAML の方が高速で、MAML と比べて約 30% 高速と報告されています。このように、高速で比較的良い精度が出せると考えられていることから、FOMAML は MAML の中でも幅広く用いられるメタ学習アルゴリズムとなっています。

[10] 一応、ReLu 関数を用いたネットワークは基本的に線形であり、2 階の勾配がほとんど 0 になるケースが多いため、2 階の勾配を落とす近似が良くなっていると考えられています。