Aprendizaje en pocos pasos con Reptile

Elizabeth, Victor, Ximena

Mayo 2021

1 Introducción

En esta práctica final, investigamos acerca de los problemas de meta-aprendizaje. Es decir, cómo es que un algoritmo aprende a funcionar cuando se le presentan tareas distintas. La idea es la siguiente: en general, los algoritmos de aprendizaje de máquina necesitan de muchos más datos que los humanos para descifrar la regla. Es decir, si un humano que no sabe qué es un perro ve un pug, un chihuahua, un dálmata y un pastor alemán y en cada ocasión le dicen que eso es un perro, entonces es muy probable que el humano distinga que está viendo un perro cuando en el futuro vea a un labrador dorado. Para que una computadora sea capaz de hacer esto, la práctica común en los algoritmos es que reciba miles de fotos de perros antes de lograr descrifrar la regla.

El código que analizamos presenta un algoritmo que limita la cantidad de información que recibe el agente y le enseña a adaptarse frente a cada tarea. El algoritmo Reptile fue desarrollado por OpenAI para realizar un meta-learning agnóstico del modelo. Es decir, este algoritmo fue diseñado para aprender rápidamente a realizar nuevas tareas con un entrenamiento mínimo. Reptile trabaja con descenso en gradiente estocástico utilizando la diferencia entre los pesos entrenados en un mini lote de datos nunca vistos y los pesos del modelo antes del entrenamiento sobre un número fijo de meta-iteraciones.

2 Resúmen técnico del artículo

Artículo: Nichol, Achiam, Schulman, "On First-Order Meta-Learning Algorithms", 2018

El algoritmo Reptile esta basado en el artículo On First-Order Meta-Learning Algorithms de Alex Nichol, Joshua Achiam y John Schulman (OpenAI). En dicho artículo se menciona que el meta-learning ha surgido como un enfoque para aprender a partir de pequeñas cantidades de datos. En lugar de intentar emular la inferencia bayesiana (que puede ser computacionalmente intratable), el meta-learning busca optimizar directamente un algoritmo de aprendizaje rápido, utilizando un conjunto de datos de tareas. En concreto, se asume el acceso a una distribución sobre las tareas, donde cada tarea es, por ejemplo, un problema de clasificación. A partir de esta distribución, se muestrea un conjunto de tareas de entrenamiento y un conjunto de tareas de prueba. Por lo que, el algoritmo Reptile se alimenta del conjunto de entrenamiento y debe producir un agente que tenga un buen rendimiento medio en el conjunto de prueba. Dado que cada tarea corresponde a un problema de aprendizaje, un buen rendimiento en una tarea corresponde a un aprendizaje rápido.

Por otra parte, se han propuesto diversos enfoques para el meta-learning cada uno con sus propios pros y contras. Chelsea Finn y Sergey Levine en su artículo Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm propusieron una variante llamada MAML de primer orden (FOMAML), que se define ignorando los términos de la segunda derivada, evitando este problema pero a costa de perder algo de información del gradiente. Así que, los autores de este algoritmo exploraron el potencial de los algoritmos de meta-learning basados en información de gradiente de primer orden, motivados en la aplicación de problemas en los que es demasiado engorroso aplicar técnicas que dependen de gradientes de orden supe-

rior (como el MAML completo). Con ello hicieron las siguientes contribuciones:

- Presentaron Reptile un algoritmo estrechamente relacionado con FOMAML, que es sencillo de implementar. Reptile es tan similar al entrenamiento conjunto (es decir, el entrenamiento para minimizar la pérdida en la expectativa sobre tareas de entrenamiento) que funciona como un algoritmo de meta-learning. A diferencia de FOMAML, Reptile no necesita una división de entrenamiento-prueba para cada tarea, lo que puede hacer que sea una opción más natural en ciertos entornos. También está relacionado con la idea más antigua de pesos rápidos/pesos lentos.

- Proporcionaron un análisis teórico que se aplica tanto a MAML de primer orden como a Reptile, mostrando que ambos optimizan la generalización dentro de la tarea.

- Sobre la base de la evaluación empírica en los conjuntos de datos Mini-ImageNet y Omniglot, proporcionaron algunas ideas sobre las mejores prácticas de implementación.

Por consiguiente, al igual que MAML, Reptile aprende una inicialización para los parámetros de un modelo de red neuronal, de manera que cuando se optimizan estos parámetros en el momento de la prueba, el aprendizaje es rápido, es decir, el modelo generaliza a partir de un pequeño número de ejemplos de la tarea de prueba. El algoritmo de Reptile es el siguiente:

* Inicializas ϕ, el vector de parámetros iniciales - para la iteración = 1, 2, . . . hacer - Tarea de muestreo τ , correspondiente a la pérdida $L\tau$ en los vectores de peso $\tilde{\phi}$ - Calcular $\tilde{\phi} = U\tau^k(\phi)$, denotando k pasos de SGD o Adam. - Actualización $\phi \leftarrow \phi + \epsilon(\tilde{\phi} - \phi)$

En el último paso, en lugar de simplemente actualizar ϕ en la dirección $\tilde{\phi} - \phi$ podemos tratar $(\tilde{\phi} - \phi)$ como un gradiente y conectarlo a un algoritmo adaptativo como Adam. También podemos definir una versión paralela o por lotes del algoritmo que evalúa sobre n tareas en cada iteración y actualiza la inicialización a

$$\phi \leftarrow \phi + \epsilon \frac{1}{n} \sum i = 1^n (\tilde{\phi} i - \phi)$$

donde $\tilde{\phi} i = U\tau i^k(\phi)$; los parámetros actualizados en la i^{th} tarea.

Este algoritmo se parece notablemente al entrenamiento conjunto sobre la pérdida esperada $\mathrm{E}\tau[L\tau]$. Por lo que, si se defe U como un solo paso de descenso de gradiente ($k = 1$), entonces este algoritmo corresponde a descenso de gradiente estocástico sobre la pérdida esperada:

$$g\text{Reptile},\ k = 1 = \mathrm{E}\tau[\phi - U\tau(\phi)]\ /\ \alpha = \mathrm{E}\tau[\bigtriangledown \phi L\tau(\phi)].$$

Sin embargo, si realizamos múltiples actualizaciones de gradiente en la minimización parcial ($k > 1$), entonces la actualización esperada $\mathrm{E}\tau[U\tau^k(\phi)]$ no corresponde a tomar un paso de gradiente sobre la pérdida esperada $\mathrm{E}\tau[L\tau]$. Por lo tanto, Reptile converge a una solución que es muy diferente del minimizador de la pérdida esperada $\mathrm{E}\tau[L\tau]$.

Aparte del parámetro del tamaño de los pasos ϵ y el muestreo de las tareas, la versión por lotes de Reptile es la misma que el algoritmo SimuParallelSGD. SimuParallelSGD es un método de optimización distribuida eficiente en cuanto a la comunicación, en el que los trabajadores realizan actualizaciones de gradientes localmente y promedian con poca frecuencia sus parámetros, en lugar del enfoque estándar de promediar los gradientes.

3 Código documentado

Ver código en el repositorio.

4 Exploración de limitantes y capacidades de los modelos

Los autores de Reptile señalan que hace falta investigar más por qué funciona tan bien el algortimo. Presentan dos ideas al respecto:

- Aproximando la actualización que hace el algoritmo con series de Taylor, los autores demostraron que SDG proporciona un término parecido al término de segundo orden que MAML utiliza. Este término ajusta los pesos para maximizar el producto punto entre los gradientes de distintas minibatches. Esto ayuda al meta-aprendizaje porque fomenta que se generalicen los gradientes entre las mismas tareas.
- Presentan un argumento informal en el que explican que Reptile converge a una solución que está muy cerca, en distancia euclideana, a la solución óptima.

5 Jugando con el código

El código que analizamos tardó 14.18 minutos en la fase de entrenamiento. En total, el código necesitó de 18 minutos para correr por completo. Esto fue en Google Colab utilizando GPU.

El algoritmo es rápido y preciso, con precisión muy cercana a 1 tanto en entrenamiento como en validación.

Para entender un poco mejor el algoritmo, trabajamos también aplicandolo a una función de seno. El algoritmo recibe solo 10 puntos (x,y) y puede predecir la función seno a la que pertenecen. El código fue desarrollado por los autores del paper y lo incluiremos en el repositorio con nuestros comentarios.

Como extensión, jugamos con este código probándolo con la función coseno y con menos puntos de entrenamiento. Vimos que aunque el modelo es menos preciso con el coseno, este sigue siendo rápido y atinado. Al disminuir la cantidad de puntos de entrenamiento, vimos que tanto en seno como en coseno, el modelo fue más tardado pero aumentó la precisión.

6 Conclusión

El algoritmo Reptile es un algoritmo de meta aprendizaje que utiliza SDG y gradiente de primer orden. Como vimos en los experimentos, Reptile es prometedor pues encuentra un punto cercano a la solución óptima con poca información. Mientras que Reptile funciona de manera parecida a MAML en el sentido de que busca la inicialización de los parámetros de una red neuronal y los ajusta dependiendo de la tarea, Reptile es más eficiente pues no utiliza las segundas derivadas sino que hace SDG estándar.