Hipoteza "Lottery winning ticket"

1st Michał Sumiński
Politechnika Łódzka
PŁ
Łódź, Polska
249999

I. Streszczenie

Jednym z obecnych wyzwań trenowania sieci neuronowych jest tzw. overparametrization(pol. przeparametryzowanie), czyli problem z ogromna liczba parametrów modelu. Technika 'pruningu' sieci neuronowych może znaczaco zredukować liczbe tych parametrów (nawet powyżej 90%) oraz potrzebna pamieć, zwiekszajac przy tym wydajność oraz dokładność modelu. W niniejszej pracy zdecydowano sie przetestować 3 rózne warianty wykorzystania mechanizmu pruningu: pruning iteracyjny, pruning iteracyjny z przywracaniem początkowych wag(Lotery winning ticket"), pruning iteracyjny z losowa reinicjalizacja wag na sieciach CNN oraz GNN. Wyniki zaprezentowano w tabelach 1-5. Eksperymenty pokazuja, ze kazdy z wariantow daje dokladnosc bardzo bliska orginalnej, przy wykorzystaniu $< 5\%$.

II. Wprowadzenie

Artykuł ten poświecony jest przebadaniu na własna reke hipotezy Lotery winning ticket" zaproponowanej przez Jonathana Frankle'a oraz Michaela Carbin'a [1] rozszerzajaca podstawowe pojecie pruningu o dodatkowy mechanizm przywracania wag z poczatkowej inicjalizacji. Hipoteza ta brzmi nastepujaco: *Losowo zainicjowana, sieć neuronowa zawiera podsieć, która jest zainicjowana w taki sposób, że podczas uczenia w izolacji może odpowiadać testowej dokładności oryginalnej sieć, kiedy jest wytrenowana przez co najwyżej taka sama liczbe iteracji.* Możemy rozpatrzyć nastepujacy przykład:
- Tworzymy sieć X, która składa sie ze 100 wag
- Po wytrenowaniu tej sieci przez e epok osiagamy wartość accuracy $(X) = a$.
- Obcinamy wagi o połowe, zostaje ich 50, przywracamy wagi z poczatkowej inicjalizacji, otrzymujemy sieć X'.
- Trenujemy sieć X' przez e' epok ,gdzie $e'e$.
- Otrzymujemy miare accuracy $(X') = a'$, gdzie zgodnie z hipoteza $a'a$.

III. Powiazane prace

Problem przeparametryzowania jest rozwijany na wiele sposobów, Jonathan Frankle oraz Michael Carbin [1] w swojej pracy przytaczaja miedzy innymi takie prace jak:
- Restrykcja optymalizacji do małych, losowych zbiorów parametrów z całej przestrzeni parametrów. [6]
- Modele SqueezeNeti MobileNet, które sa o rzad wielkości niższe od standardowych architektur. [7]
- Destylacja (ang. Distillation) – przekazanie wiedzy z wiekszego modelu do mniejszego. [8]

IV. Metoda

A. Konwolucyjne sieci neuronowe (CNN)

Do eksperymentów zdecydowano sie użyć architektury sieci CNN z zadania nr.2, tzn. sieć CNN z dwoma warstwami konwolucyjnymi typu Conv2d oraz jedna warstwa w pełni połaczona, użyta funkcja aktywacji w warstwach konwolucyjnych to leaky_relu, a w warstwie w pełni połaczonej to sigmoid, użyto optymalizatora Adagrad oraz krzyżowej entropii jako funkcji celu. Całość została zaimplementowana przy użyciu biblioteki PyTorch[2] (a ścislej jej podzbiorowi torchvision[4]). Użyty został zbiór danych MNIST.

Zdecydowano sie przeprowadzić 3 różne eksperymenty, tj. pierwszy - pruning iteracyjny, tzn. uczenie modelu przez 'x' iteracji, nastepnie wykonanie operacji pruningu, i tak kilka razy, drugi - również pruning iteracyjny natomiast po każdej operacji pruningu pozostałe wagi zostawały losowo reinicjalizowane, trzeci - również pruning iteracyjny, a po każdej operacji pruningu zostały przywracane wagi z pierwszej inicjalizacji - jest to implementacja właśnie tytułowego "Winning ticketa". Warto przypomnieć, że oryginalnie model po 10 epokach osiagnał wartość miary accuracy równa **(0.988)**.

Do pruningu zdecydowano sie zastosować również biblioteke PyTorch. Pruning za każdym razem wykonywanie był globalnie (global pruning) na 2 warstwach konwolucyjnych, tzn. że gdy podawana była wartość parametru pruningu, np. 0.2 mówiaca o tym jaki procent (0.2 = 20%) wag ma zostać usuniety, to nie dotyczyła ona każdej warstwy pojedynczo, tzn. z każdej warstwy nie zostało usuwane 20% wag tylko. 20% wag zostało usuwanych z obu warstw, np. z pierwszej warstwy zostało usuniete 5% wag, a z drugiej 15%.

B. Grafowe sieci neuronowe (GNN)

Ten sam mechanizm co w powyższym rozdziale o sieciach konwolucyjnych zastosowano do sieci grafowej. Eksperymenty zostały przeprowadzone na architekturze sieci z zadania nr.2, tzn. sieć GNN z dwoma warstwami konwolycujnografowymi typu GCNConv oraz jedna warstwa w pełni połaczona, użyta fukncja aktywacji w warstwach konwolucyjnych to leaky_relu, a w warstwie w pełni połaczonej to sigmoid, użyto optymalizatora Adam oraz krzyżowej entropii jako funkcji celu. Całość została zaimplementowana przy

użyciu biblioteki PyTorch[2] (a ścislej jej podzbiorowi pytorch_geometric[3]). Użyty został zbiór danych CORA. Warto przypomnieć, ze oryginalnie model po 10 epokach osiagnał wartość miary accuracy równa **(0.792)**.

C. Trening

Treningi modeli wygladały nastepujaco:
1) zainicjalizowanie modelu (w przypadku "Winning ticket'ów" po tym kroku należało skopiować macierz wag modelu w celu późniejszego ich przywrócenia)
2) pierwszy trening modelu (dla sieci CNN 10 iteracji, dla sieci GNN 1000)
3) wykonanie operacji pruningu 20%
4) w przypadku eksperymentów z losowareinicjalizacja badź "Winning ticket'ami" w tym kroku zostały przywracane badź reinicjalizowane wagi
5) ponowny trening modelu
6) wykonanie operacji pruningu 52%
7) Powtórzenie kroków 4,5
8) wykonanie operacji pruningu 80%
9) Powtórzenie kroków 4,5
10) wykonanie operacji pruningu 96%
11) Powtórzenie kroków 4,5
12) wykonanie operacji pruningu 99.6%(GNN) badź 99.96%(CNN)
13) Powtórzenie kroków 4,5

V. Wyniki

Poniżej przedstawiono tabele z wynikami eksperymentów dla sieci CNN, liczba epok w każdej kolejnej iteracji pruningu wynosiła 10:

TABLE I
PRUNING ITERACYJNY

%pruningu	accuracy
20	0.9884
52	0.9897
80	0.9904
96	0.9888
99.96	0.1135

TABLE II
PRUNING ITERACYJNY Z ZASTOSOWANIEM "WINNING TICKET'A"

%pruningu	accuracy
20	0.9889
52	0.9896
80	0.9905
96	0.9883
99.96	0.5114

TABLE III
PRUNING ITERACYJNY Z LOSOWA REINICJALIZACJA WAG

%pruningu	accuracy
20	0.9895
52	0.9898
80	0.9903
96	0.9888
99.96	0.4238

Podobnie jak dla sieci konwolucyjnych przeprowadzono dokładnie te same 3 eksperymenty dla sieci GNN. Poniżej przedstawiono tabele z wynikami eksperymentów, liczba epok w ka zdej kolejnej iteracji pruningu wynosiła 1000:

TABLE IV
PRUNING ITERACYJNY

%pruningu	accuracy
20	0.9884
52	0.9897
80	0.9904
96	0.9888
99.6	0.361

TABLE V
PRUNING ITERACYJNY Z ZASTOSOWANIEM "WINNING TICKET'A"

%pruningu	accuracy
20	0.9889
52	0.9896
80	0.9905
96	0.9883
99.6	0.684

TABLE VI
PRUNING ITERACYJNY Z LOSOWA REINICJALIZACJA WAG

%pruningu	accuracy
20	0.9895
52	0.9898
80	0.9903
96	0.9888
99.96	0.319

VI. Podsumowanie

W niniejszych eksperymentach hipoteza "Wining ticket'a" nie rysuje sie tak klarownie jak to miało miejsce u J. Frankle'a, M. Carbin'a [1]. Wyniki z każdego z 3 eksperymentów daja praktycznie taka sama wartość miary accuracy - różnia sie minimalnie. Natomiast można spostrzec jedna zależność, zarówno w CNN jak i w GNN sieć z "Winning ticket'em" osiagneła najwieksza wartośćmiary accuracy przy najwiekszym procencie usunietych wag, co można zaobserwować w ostatnich wierszach każdej z tabel. Widać zatem, że gdy pozostały procent parametrów wynosił $< 1\%$, to sieć z "Winning ticket'em" radziła sobie najlepiej. Może to być pewien punkt zaczepienia. Być może przebadane sieci były zbyt małe,

posiadały zbyt mało parametrów, aby mechanizm ”Winning ticket'a” zadziałał. Co jednak jest pewne to to, że w każdym z eksperymentów wystarczył $< 5\%$ parametrów, aby osiagnać wyniki porównywalne do pełnej, gestej sieci.

VII. Spis literatury

References

[1] J. Frankle, M. Carbin, “The lottery ticket hypothesis: finding sparse, trainable neural networks”.

[2] https://pytorch.org/

[3] https://pytorch-geometric.readthedocs.io/en/latest/

[4] https://pytorch.org/vision/stable/index.html

[5] https://pytorch.org/tutorials/intermediate/pruning_tutorial.html

[6] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. Proceedings of ICLR, 2018.

[7] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. arXiv preprint arXiv:1602.07360, 2016.

[8] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In Advances in neural information processing systems, pp. 2654–2662, 2014