モデルのパラメータ数を減少させる手法、ProdSumNet を読む

MokkeMeguru

May 14, 2019<*2019-04-24 Wed*>

Contents

1 導入

ProdSumNet、というのは深層学習で **線形演算子** をより単純な形に書き換え、そのモデルの ***パラメータ数を減らす*** * フレームワークを提案した論文です。この線形演算子は CNN や KFC、Dilated CNN といったアーキテクチャ全てにこの手法は適用可能であることが特徴として挙げられ、それらのパラメータ数を減少することに成功しました。実験としては、MNIST と Fashion MNIST を用いました。

また別の内容としては、 **Convolutional Layer の代替となる手法** を提案しました。こいつは訓練することが出来るパラメータ数を動的に変更することができること、ややもすればパラメータ数と誤り率のトレードオフを調べることが出来ると考えられています。実験としては MNIST を用い、 3×10^6 以上の訓練パラメータを持つ CNN ネットワークを提案手法で置き換え、3554 個にまで訓練パラメータを減少させた上で 98.44% の Accuracy を達成しました。

2　モデルのパラメータ数を減少させると何が良いの？

昨今の深層学習モデルは、とてもでっかいサイズになりがちな傾向が見られます。例えば昨年度から本年度頭にかけて、自然言語処理の分野では BERT や GPT-2 なんていうクソデカ言語モデルが台頭しています。こいつは学習時や推論時に、目が飛び出るような値段の GPU や一昔前のストレージサイズばりのメモリなんかのリソースをもりもり消費します。

しかしそういった **強いけど重い深層学習モデル** というのは実用性があるのかと言われると、結構難しいところがあります。例えば (リアルタイム) 音声合成を行う機械学習モデルなんかはでかい計算機でビッグに計算してつよつよな精度が欲しいわけではなくて、寧ろモバイル機器でもとっととまともな出力が欲しい、という需要があります。(実際以前読んだ RawNet なんかや FaceBook の研究している LPCNet なんかの研究成果は、精度について競う他に、音声合成の速度やリソースの使用量に視点をおいて研究が行われています。)

3　関連研究はどんなものがあるの？

さて論文中ではモデルのパラメータ数を減少させる利点を２つ挙げ、その利点を得るための関連研究を挙げています。

１つ目は、 **訓練後のモデルの軽量化** です。これは訓練後のモデルのパラメータを主成分分析なり SVD なりしてパラメータ数を圧縮する方法が考えられています。

２つ目は、 **訓練中のモデル** の複雑さを軽減することです。そしてこれが本論文でフォーカスしている分野です。

２つ目についてもう少し詳しく述べると、以下の式について考えることが出来ます。

$$
\begin{aligned}
y_{i+1} &= f_i(W_i y_i + b_i)\ where\ i = 1, \ldots, N \qquad (1)\\
where\ f_i &\quad is \quad nonlinear\ function\ ex.\ ReLu\ or\ other\ identity\ function(R^{m_i} \to R^{n_{i+i}})\\
W_i &\quad is \quad a\ matrix\ which\ means\ weight(R^{m_i \times n_i})\\
b_i &\quad is \quad a\ vector\ which\ means\ bias(R^{m_i})\\
y_i &\quad is \quad a\ vector(R^{n_i})
\end{aligned}
$$

この式は簡単な深層学習の一層を表しています。 f_i については事前に与えられており固定されたものとすると、この式の最適化は、 y_k への入力を x_k としてその際の正解出力を z_k 、予測出力を $\tilde{z_k}$ とすると、パラメータ W_i, b_i に関する損失関数 $d(\cdot, \cdot)$ について $\Sigma_k d(\tilde{z_k}, z_k)$ の最小化とも言えます。

しかしここで最適化しなければならないパラメータ (trainable parameter 訓練パラメータ) は一般に W_i, b_i のすべての要素の数 n_W で、これはとても大きな値であると言えます。n_W が大きいと、１反復にかかる計算が多くなること、~~反復回数が多くなること (読んだ感想、を参照)~~ が予測され、つまり訓練時間が長くなってしまう可能性があります。

つまり本論文では **この n_W を減らせるような普遍的な構造を提案したい** 、ということになります。

この２つ目を達成する一般的手法としては、CNN に対して Convolutional layer を導入することです（なんのこっちゃと思いますが、そのままです [2]）。Convolutional Layer が持つ効果として、Shift invariant というものがあります。これはシフト不変性と訳されることもあるように、ちょっとズレた入力画像も同等に扱うことができるという機能を示しています (要：考察)。これによって訓練パラメータを劇的に減少させることが出来ます。これは shift invariant が必要となるような問題（例えば画像識別）ではうまく機能します。別の手法としては密行列 (dense weight matrices) を巡回行列 (circulant matrices) に置き換えるものが知られています [3]。　[2]: 原 (The most well known model reduction technique is the introduction of the convolutional layer in convolutional neural networks.)

[3]: CirCNN (Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai Qian, Yu Bai, Geng Yuan, Xiaolong Ma, Yipeng Zhang, Jian Tang, Qinru Qiu, Xue Lin, Bo Yuan)

4　どういう手法を提案したの？

この論文のアイデアの根幹は以下の式になります。

$$
\begin{aligned}
W &= \prod_{j=1}^{p} \Sigma_{k=1}^{s_i} g_{jk}(a_j k) M_{jk} \qquad (2)\\
\textit{where } a_{jk} &\ \textit{are} \ \textit{the trainable parameters, also denoted as the variable parameters}\\
j &= 1, \ldots, p\\
k &= 1, \ldots, s_j\\
M_{jk} &\ \textit{are} \ \textit{the fixed parameters (matrices)}\\
g_{jk} &\ \textit{are} \ \textit{the fixed nonlinear functions}
\end{aligned}
$$

これは重み行列 W を計算するために提案された式です。

a_{jk} というのが訓練パラメータで、論文中では可変パラメータとも呼称されるものです。

M_{jk} は固定された行列であり、このサイズは、M_{jk} と $M_{jk'}$ に関しては同じ次元数、 M_{jk} と $M_{j+1,k}$ に関してはこの間で加算、乗算が可能であるように制限されます。

非線形関数である g_{jk} は微分可能な関数であれば、簡単に導出することが出来るので、本論文中ではすべて $g_{jk}(x) = x$ としています。

この手法によって W のための訓練パラメータの総数 n_W は、 $\Sigma_j s_j$ となります。つまり最適な重みである W_{opt} がもし a_{ij} を用いて近似分解できる場合 (かつ $\Sigma_i s_i$ が小さい場合)、この手法によってモデルのパラメータ数は劇的に減らすことが出来ます。またもし先に W_{opt} の近似が得られている場合（例えば転移学習など）では、パラメータ数を減らすためにその重み行列を上式の形に低ランク分解することが出来ます。

またこの手法が convolutional layer や KFC, curculant matrices, Toeplitz matrices Handel matrices などでも有効であること (重み行列を分解することができること) は明らかです。

但しこの手法は、 n_W を小さい値で抑えながら、最適な固定パラメータと分解を見つけることですが、本手法では訓練可能なパラメータを任意に変更できるという利点を持っています (他の手法を用いた分解方法だと、行列サイズを固定する必要があることが差異)。

しかもこの提案手法は、現在ある深層学習用のアーキテクチャで用意に実装できます。この証拠として、先程の分解の式の偏微分は以下のように導出することが出来ます。

5 どのくらいの精度が出たの？

いくつかのテーマに別れて実験が行われているので、それぞれについて簡潔に紹介します。

5.1 一つの行列和の分解 (提案手法の $\mathbf{p=1}$)

5.2 複数の行列和の内積の分解 (提案手法の $\mathbf{p>1}$)

5.3 Convolutional Layer のない画像認識タスク

6 読んだ感想とか

この論文、とてもおもしろい研究だと思うんですが、**どこの学会にも出されていない** という不思議なことになっています。多分出したところから落とされたのかな？と思っているんですが、どうなんでしょう？

また再現実験をしているレポジトリとかも見当たらないのが気になったりしています。 ~~どこかに実装ないかな…~~

また、Pervasive Attention でわかったように、必ずしもパラメータ数が計算量に比例するわけではなさそうなので、この手法を用いることの利点とされる、訓練回数が少なくなる、というのは少し難しいと思われますね。　　（あとこの論文中で言われていた、CirCNN という論文がとてもパワーがあるのでそのうち目を通したいです。）