訓練事例の影響の軽量な推定

小林 颯介^{†,††}・横井 祥^{†,†††}・鈴木 潤^{†,†††}・乾 健太郎^{†,†††}

機械学習モデルの挙動の解釈において、各訓練事例がもたらす影響を理解することは重要である。単純にはデータセットから対象の訓練事例を除いて再訓練してモデルの変化を解析することもできるが、必要な計算量が非常に大きくなってしまい、特に膨大なパラメータのニューラルネットワークモデルへの適用が困難であった。本論文では、ニューラルネットワークモデルへの各訓練事例の影響の推定手法として、既存手法に比べて非常に効率的な方法を提案する。提案手法では、各事例の学習時に dropout を用いて、事例ごとに固有のサブネットワークのパラメータを更新せずに訓練を進めることで、訓練終了後には各訓練事例の影響を受けていないサブネットワークを自由に抽出し、それを影響値の推定に活かすことができる。実験では、提案手法を、文書分類と画像物体認識において、BERT および VGGNet に適用し訓練事例への紐付けを行うことで、解釈性の高い形でモデルの予測を解析できることを示した。また、サブネットワークの学習曲線の解析やデータフィルタリングの実験を通して、提案手法が事例間の関係性を適切に捉えていることを定量的に示した。

キーワード：影響関数, Dropout

Efficient Estimation of Influence of a Training Instance

SOSUKE KOBAYASHI^{†,††}, SHO YOKOI^{†,†††}, JUN SUZUKI^{†,†††} and KENTARO INUI^{†,†††}

Understanding the influence of a training instance on a machine-learning model is important for interpreting the behavior of the model. However, it has been difficult and inefficient to evaluate the influence, considering how the prediction of a model would be changed if a training instance were not used. This prevents the application of influence estimation in neural networks with a large number of parameters. In this paper, we propose an efficient method for estimating the influence for neural networks. The method is inspired by dropout, which zero-masks a sub-network and prevents the sub-network from learning each training instance. By switching between dropout masks, we can use sub-networks that learned or did not learn each training instance and estimate its influence based on the difference between the sub-networks. Through experiments with BERT and VGGNet on classification datasets, it was demonstrated that the proposed method can enhance the interpretability of error predictions. Quantitative evaluations were also performed by analyzing learning curves of sub-networks and applying the method to data filtering.

Key Words: *Influence Function, Dropout*

[†] 東北大学 情報科学研究科, Graduate School of Information Sciences, Tohoku University

^{††} 株式会社 Preferred Networks, Preferred Networks, Inc.

^{†††} 理化学研究所 革新知能統合研究センター, RIKEN Center for Advanced Intelligence Project

1 はじめに

機械学習モデルは、モデルの構造やその最適化手法に加え、実際に学習する事例集合によって性能と挙動が特徴づけられる。それを踏まえ、ある訓練事例が学習後のモデルに与える影響の解析は、これまで広く研究がなされてきた (Cook 1977; Koh and Liang 2017; Zhang et al. 2018; Hara et al. 2019)。事例ごとの影響を正しく把握することで、学習後のモデルの挙動や予測を解釈することに役に立つほか、事例の貢献度を考慮したデータフィルタリングなどの応用にもつながる。特に、自然言語処理などの分野で幅広く使われているディープニューラルネットワークモデルについては、性能の向上が著しい (Devlin et al. 2019)。その一方で、学習後のモデルの挙動や特性を理解することが極めて困難であるブラックボックス性が応用上の課題となっており、データに紐付けて説明を行うアプローチはその解決手段の一つとして有望視されている (Koh and Liang 2017; Plumb et al. 2018; Ouchi et al. 2020)。

この事例ごとの影響値の最も素朴な計算方法は、あるモデルをデータセット全体で訓練し、その後同様の訓練手続きを一事例だけ除いたデータセットの上でも実施することである。このような操作 (leave-one-out retraining) で作成した2つのモデルの予測の差を比較することは、除外対象の一事例の影響値を計測することに概ね相当する。しかし、この方法は解析したい事例ごとにモデルを再訓練して保存する必要がある、データセット全体への解析や解釈などを行う際には、非常に長い計算時間とモデルの保存容量を要する。例えば、予測に最も寄与した事例をユーザーに提示することでモデルの解釈性を高めたい場合を考える。このとき、訓練事例数 $|D|$ に依存した回数の訓練と個数のモデルの保存が予め必要となる。さらに、実際に推論する際にも、事例ごとの $|D|$ 回分の推論のたびに新たにモデルを読み込む必要があり、BERT (Devlin et al. 2019) のような巨大なモデルの場合、読み込みにも無視できない時間がかかる。この単純な leave-one-out の再訓練を行う他に、影響値を近似的に推定する手法がこれまで提案されてきたが (Koh and Liang 2017; Hara et al. 2019)、計算量の課題や適用可能なモデルの制約などが依然として存在するため、実用性を大きく損なっている。ニューラルネットワークを用いた実験も行われているが、典型的にモデルサイズが非常に小さく (Koh and Liang 2017; Hara et al. 2019)、また、より大きいモデルに適用を試みた実験でも上述の計算量の課題が報告されている (Han et al. 2020)。

そこで、本論文では、訓練事例の影響の計算を巨大なニューラルネットワークモデルにおいても実用的にするため、新たな推定手法 *turn-over dropout* を提案する。*turn-over dropout* はその名の通り dropout (Srivastava et al. 2014; Hinton et al. 2012) を活用した手法であり、dropout を事例ごとに決定的なサブネットワークのみを訓練する手法として新たに定式化したものである。訓練終了後には、各事例について、影響を受けていないサブネットワークと受けているサブネットワークの両方で予測が行えるため、その差分から事例ごとの影響を近似的に推定する

ことができる. この推定では, ニューラルネットワークの前向き計算による予測を 2 回実行するだけであり, (Hara et al. 2019) と (Koh and Liang 2017) の手法に比べて大幅に高速である. また, 訓練に関しても, 通常通りに一つのデータセット上で訓練を一度行ったモデルを用意するだけで, そこに含まれる全事例に関して推定を行うことができ, 再訓練も複数のモデルの保存も必要がない. 本論文では, 先行研究に対する計算量における優位性を議論するとともに, 文書分類タスクと画像物体認識のタスクにおいて推定の有効性を示す実験を行った. 提案手法を文書分類と画像物体認識における BERT (Devlin et al. 2019) および VGGNet (Simonyan and Zisserman 2015) に適用し訓練事例への紐付けを行うことで, 解釈性の高い形でモデルの予測を解析できることを示した. また, サブネットワークの学習曲線の解析やデータフィルタリングの実験を通して, 提案手法が事例間の関係性を適切に捉えていることを定量的に示した.

2 訓練事例の影響値

2.1 問題設定

まずはじめに, 本論文で扱う問題設定と定式化に関して説明する. 本論文では, (Hara et al. 2019) の定義にならない, ある訓練事例を学習に用いた際の, また一方のある事例 (典型的にはテストデータセット内の事例) の予測に関する影響値 (*influence*) について議論する. $z := (x, y)$ をある訓練事例とし, その内の入力を $x \in X$ および (正解) 出力を $y \in Y$ とする. また, $D := \{z_i\}_{i=1}^N$ で N 個の事例集合からなるデータセットを表す. データセット D を訓練事例集合として, SGD などの最適化手法を適用することで, 関数 $f_D: X \rightarrow Y$ を学習する. 特に本論文では学習されたパラメータを持つニューラルネットワーク f_D について議論する. $L(f, z)$ を損失関数とすると, 学習の目的は $\hat{f}_D := \operatorname{argmin}_f \mathbb{E}_{z_i \in D} L(f, z_i)^1$ を得ることである. 本論文の実験では, 特に分類問題の正解ラベルと予測の差に関するクロスエントロピーを損失関数として用いるが, 提案手法は一般的な他の損失関数に関しても適用可能である.

影響値 (*influence*) $I(z_{\text{target}}, z_i; D)$ は, 事例 z_{target} に関する予測について, 訓練事例 z_i を学習したことの定量的な利得を表す. $f_{D \setminus \{z_i\}}$ をデータセット D から事例 z_i を抜いて訓練した場合のモデルとすると

$$I(z_{\text{target}}, z_i; D) = L(f_{D \setminus \{z_i\}}, z_{\text{target}}) - L(f_D, z_{\text{target}}). \quad (1)$$

と影響値は表される. 直観的には, 影響値が正に大きくなるほど, 訓練事例 z_i は事例 z_{target} の予測に関する損失の減少に大きく貢献していることになる. 典型的には, この推定時の参照事

¹ なお, 典型的には, 訓練データセット D 上で損失関数の最小化を達成するモデルは, その他の未知の事例に対して汎化した性能が出ないため, 訓練に用いられないデータセット D_{dev} について $\mathbb{E}_{z_{\text{dev}} \in D_{\text{dev}}} L(f, z_{\text{dev}})$ となる最適化を目指すことが多い.

例 z_{target} は、テストや開発用のデータセットに含まれている事例である。特に、あるテスト事例に対する誤り予測ラベルでの損失に関して、訓練事例ごとに式 (1) の値を計算し比較することで、その誤りに寄与した訓練事例たちを特定することができ、モデルの解析に有用であるとして研究されてきた (Koh and Liang 2017)。

2.2 関連手法

式 (1) の影響値を素朴に計算するためには、事例ごとに新たにモデルの訓練が必要になり、その計算量は非常に大きい。その代替として、これまで幾つかの推定手法が提案されてきた。

まず、Koh・Liang (Koh and Liang 2017) の推定手法は影響関数 (influence function) に基づいて提案されたものである。定式化としては強凸な関数のみに適用可能であり、また、解析対象のモデルについても、大域的最適解として損失の最小化が済んだパラメータを仮定している²。ニューラルネットワークを始めとする多くのモデルはその条件を満たさないが、(Koh and Liang 2017) や (Han et al. 2020) などは、手法に一部修正を施すことで簡易的にニューラルネットワークに適用した場合においても実験的には有用な推定が行えることを議論している。その一方で、実験対象のモデルサイズは非常に小さい (Koh and Liang 2017)。モデルサイズが大きい場合の検証はこれまで少ない他、その計算量の大きさについても実用上の観点から依然として大きな課題になっている (Han et al. 2020)。

次に、Hara ら (Hara et al. 2019) は上記のような制約のない推定手法を新たに提案した。しかし、その手法は訓練過程の多くのモデルスナップショットを保存するための大きな保存容量を必要とするほか、推定時にはモデルの更新回数に依存したスナップショットの読み出しと重い計算処理を行う。そのため、同様に、大きなモデルに対して適用することは計算量と保存容量の観点から現状困難である。

2.3 関連手法との計算量の比較

これらの既存手法に比べて、本論文での提案手法 (3 節で後述) は非常に効率的である。計算量の理論的な比較のため、訓練の時間計算量 (Training)、モデルの保存に要するディスク保存容量 (Storage)、影響値推定時の時間計算量 (Estimation) を表 1 にまとめた。影響値推定の時間計算量は、あるテスト事例 1 つの予測に関して訓練データセット D 内の各事例の影響値を推定する状況での計算量を記述した。例えば、あるテスト事例の予測に関して各訓練事例の寄与

² 厳密には、Koh・Liang (Koh and Liang 2017) が扱っている値は、我々や Hara ら (Hara et al. 2019) が扱う式 (1) の値とは似て非なるものである。単純に述べると、Koh・Liang (Koh and Liang 2017) の定式化では強凸な関数の最適パラメータに関してのみ $f_{D \setminus \{z_i\}}$ と f_D の差を議論している。一方で、我々と Hara ら (Hara et al. 2019) の定式化では、そのような条件や制約はなく、より広い問題として扱っている。本論文では、Hara ら (Hara et al. 2019) に従い、式 (1) の定義は f_D 及び $f_{D \setminus \{z_i\}}$ について、同じ初期値から事例 z_i を除いて同じ最適化手続きを行う限り、モデルの制約を付けない。例えば、確率的勾配降下法 (SGD) においては、ミニバッチの順序及び構成についても z_i を除き、完全に一致した最適化を行う設定で議論するが、その更新回数などにも制約はない。

Method	Training	Storage	Estimation
Re-train	$O(D ^2)$	$O(\theta D)$	$O(F D)$
Hara+	$O(D)$	$O(\theta T)$	$O(F D + (F+F')TB)$
Koh+	$O(D)$	$O(\theta)$	$O(F D + (F+F') \underbrace{rt}_{\approx D } b)$
Ours	$O(D)$	$O(\theta)$	$O(F D)$

表 1 提案手法, (Hara et al. 2019), (Koh and Liang 2017) ごとに必要な計算量と保存容量の比較. あるテスト事例 1 つの予測に関して, 訓練データセット D の全事例の影響値を計算した場合について示した. $|\theta|$ はモデルのパラメータ数, F は前向き計算あるいは後ろ向き計算の計算量, F' は二階微分の後ろ向き計算の計算量, T は訓練中の更新回数, B は訓練中のミニバッチ内の事例数. (Koh and Liang 2017) の b は近似計算時のミニバッチ内の事例数, rt はハイパーパラメータの一種であり, 典型的には $rt \approx |D|$ である. 詳細は本文を参照.

を解析したいケースはこの状況に相当する. 表 1 から明らかなように, 提案手法はどの項目においても最も効率的な水準であり, いずれの既存手法よりも総合的に効率的であると言える.

まず, モデルの訓練にかかる時間については, データごとに新たにモデルを用意する必要がある Re-train では大幅にコストがかかる. その他の手法 Hara+, Koh+, 本提案手法については, 訓練にかかる時間は基本的に同じである. 次に, 保存容量については, 除外対象のデータごとに個別のモデルを保存しておく必要がある Re-train や, 訓練中のモデルのスナップショットを頻繁に保存する必要がある Hara+ ではコストが大きい. そして, 推定においては, Re-train 及び本提案手法では, ニューラルネットワークの 2 度の前向き計算 (推論)³を行うだけで済むのに比べ, Hara+, Koh+ では大きな計算量がかかってしまう. さらに, Re-train, Hara+, Koh+ では推論時の計算の一部あるいは全部を, 1 つの対象データごとに別々のミニバッチ (すなわちバッチサイズが 1) で計算する必要がある. 一方で, 提案手法では異なる訓練事例に関する推定を同じミニバッチで一括処理することが可能であり, GPU での計算の効率性を大きく上げられる利点がある⁴. 実験的には, BERT-base を用いた含意関係認識タスクで, あるテスト事例 1 つの予測に関して 10,000 個の訓練事例の影響値を求めた場合, Koh · Liang (Koh and Liang 2017) の手法が約 10 分かかった (Han et al. 2020) 一方で, 提案手法では 同条件の 10,000 訓練

³ また, 本提案手法は, (Koh and Liang 2017) の手法及び (Hara et al. 2019) の手法とは異なり, 勾配の情報を用いずに, ニューラルネットワークの前向き計算自体を行って推定を行う特徴がある. そのため, 微分可能でない (損失) 関数についても適用が可能であり, また, 損失関数のようなスカラー値の出力以外にも, 途中の隠れ層ベクトル等に関する影響値も効率的に推定することが可能である. また, 前向き計算自体を行うため, 式 (1) のような (「差」に着目した) 影響値のみならず, 途中の隠れ層の個々の解析も可能である. これらのケースにおける検証は本論文では含めず, 今後の課題とする.

⁴ 例えば, BERT-base の前向き計算をバッチサイズ 32 で計算する場合は, バッチサイズ 1 で計算する場合に比べて, 同じ時間あたりに約 5 倍の量の事例を処理することができた. 計算には, 含意関係認識の Multi-Genre NLI dataset (Williams et al. 2018) のデータセットを使用した.

事例から1テスト事例への影響値推定に35秒しかかからなかった⁵。よって、このケースでは、提案手法では先行研究の (Koh and Liang 2017) の手法に比べて約17倍高速に処理を終えることができた。このように提案手法の優位性はその非常に高速に処理を終えることができる点であり、訓練事例の影響の推定を実応用に向けて進展させることができたと考えられる。なお、大量の事例に対して同様に推定を行いたい場合では、複数GPUを用いて並列計算を行うことで高速化を行うことができるが、それは表1に示した4つの手法全てで適用可能であり、速度の比自体に大きな変化はない。

3 提案手法

3.1 Dropout

Dropout (Hinton et al. 2012; Srivastava et al. 2014) は、ニューラルネットワークの訓練において頻繁に使われる正則化手法である。訓練中には、ある層の隠れ状態ベクトルについてその次元数 d と同じサイズの d 次元のマスクベクトル $\mathbf{m}$ をサンプリングし、隠れ状態の値に要素積をとる。マスクベクトル $\mathbf{m} \in \{0, \frac{1}{p}\}^d$ の各要素 m_j は、確率 p のベルヌーイ分布から独立にサンプリングされて係数 $\frac{1}{p}$ によって定数倍された値である ($m_j := \frac{m'_j}{p}, m'_j \sim \text{Bernoulli}(p)$)。 p は要素が非ゼロとなる確率を表すハイパーパラメータであるが、本論文では0.5とする。すなわち、約半数の要素がゼロとなる設定を考える。dropoutでは、学習後の推論時にはマスクを使わないモデルで素朴に計算を行う。

典型的な dropout は上記のように隠れ状態に対する要素積として実施されるが、より一般化すると、マスクベクトルによる dropout はモデルのパラメータへ適用されているものとして解釈できる。例えば、あるパラメータ行列 W をかける前の隠れ状態にマスクベクトル $\mathbf{m}$ で要素積をとった場合、隠れ状態内の各次元に対応する W の行に一括で積をとっていることに相当する。特に、あるマスクベクトルの要素が0であるとき、対応する W 内のパラメータに0がかかるため、モデルの計算においてそのパラメータは全く寄与しなくなる。言い換えると、このように dropout をニューラルネットワークの各層で行う場合、あたかもマスクベクトル $\mathbf{m}$ に応じて枝刈りされたサブネットワーク $f^{\mathbf{m}}$ を用いた演算を行っているように見なせる。訓練時には、各更新ごとに異なるマスクベクトルが使われるため、様々なサブネットワークがランダムに選ばれ学習していくことになる。これまでの研究 (Hinton et al. 2012; Srivastava et al. 2014;

⁵ Han ら (Han et al. 2020) は、Koh・Liang (Koh and Liang 2017) の手法を BERT-base モデルに適用し、含意関係認識の Multi-Genre NLI dataset (Williams et al. 2018) において、10,000 個の訓練事例からある1事例への影響値の推定を行う場合、NVIDIA GeForce RTX 2080 Ti GPU 一つを用いて10分かかったと報告している。我々も同様に、同データセットと同じGPUの製品を用いて実験を行ったところ、提案手法の turn-over dropout では、35秒で1事例への推定を終えることができた。なお、どちらの実装も同じライブラリ (Wolf et al. 2019) から派生したもので、実装による速度の差異は軽微であると考えられる。

Bachman et al. 2014; Baldi and Sadowski 2014; Bulò et al. 2016) でも, 上記のように dropout が多数のサブネットワークの訓練をし, 推論時の全体ネットワークはそれらのアンサンブルとして演算されていると言われている.

3.2 提案手法: Turn-over Dropout

上で述べた典型的な dropout では, マスクベクトルは訓練中の更新ごとに独立にランダムサンプリングされる. 一方, 本提案手法では, 各訓練事例 z ごとに異なる事例固有マスク $m(z)$ を決定的に使い続けるものとする. より正確には, 事例固有のマスクは同様にベルヌーイ分布からサンプリングされたランダムな値をとるが, それを事例ごとに固定させ, ある事例を用いた訓練時には必ず決定的に同じ値を使い続けるものとする. そのため, 図 1 の上部に示すように, ある事例 z で訓練を行う際には同じ $m(z)$ が使われるため, その前向き計算及び後ろ向き計算も常に固有のサブネットワーク $f^{m(z)}$ (赤) によってのみ行われることになる. 言い換えると, サブネットワーク $f^{m(z)}$ のパラメータが更新される一方で, そのマスクを反転, すなわちゼロ・非ゼロ要素の位置を逆転させたようなマスク (以下, 反転マスク) $\tilde{m}(z) = \frac{1}{p} - m(z)$

図 1 提案手法の概略図. Dropout は各訓練事例 z ごとにサブネットワークを生成することができる. そして, その残ったパラメータのみ (赤; 上部) を更新する. 反対に, 反転したサブネットワーク (青; 下部) はその事例 z の影響を受けないままになる. 提案手法ではこの二つのサブネットワークの差を推定に用いる.

(図 1 の中央参照) を使った相対するサブネットワーク $f^{\tilde{m}(z)}$ のパラメータは一切 z の影響を受けて更新されることがないと言える。したがって、このようなサブネットワーク $f^{\tilde{m}(z)}$ は、本来のモデルのネットワーク f に比べて表現力が弱まっているものの、まさしく「ある事例 z を使わずに訓練したモデル」となっている。そして、サブネットワーク $f^{\tilde{m}(z)}$ は同等の表現力を持つ「ある事例 z も使って訓練したモデル」である。本論文では、この 2 つのサブネットワークを類推的に式 (1) の f_D と $f_{D \setminus \{z_i\}}$ の近似として活用し、その出力を推定に使うことを提案する。すなわち、影響値 $I(z_{\text{target}}, z_i; D) = L(f_{D \setminus \{z_i\}}, z_{\text{target}}) - L(f_D, z_{\text{target}})$ を次のように推定する。

$$\hat{I}(z_{\text{target}}, z_i; D) := L(f_D^{\tilde{m}(z_i)}, z_{\text{target}}) - L(f_D^{m(z_i)}, z_{\text{target}}), \quad (2)$$

これは z_{target} の予測に関して、 $f_D^{\tilde{m}(z_i)}$ の代わりに $f_D^{m(z_i)}$ を使った場合の利得 (損失の減少) を表す。本提案手法を、以後 *turn-over dropout* と呼ぶ。turn-over dropout の利点は影響値の推定のために必要なコストが非常に小さいことであり、以下のようにまとめられる。

- 計算時間が少ない: 提案手法は、ある事例ペアの影響値推定に関して、モデルの前向き計算を 2 回行うだけで実行可能である。また、一つのミニバッチ内で別の訓練事例に関する処理をまとめて行うことができる⁶。
- モデルの保存容量が少ない: 訓練時に提案手法を組み込んでおくことで、1 度の訓練でデータセット中の全事例に関して推定が行える。更新ごとのスナップショットや再訓練は不要である。
- 実装が簡易: 提案手法の仕組みは極めてシンプルであり、実装の修正が最小限で済む。実装の例を付録 B に示した。

本提案手法では、サブネットワークへの 1 訓練事例の影響を推定し、それを全体のネットワークへと転用して推定している。(Hinton et al. 2012; Srivastava et al. 2014; Bachman et al. 2014; Baldi and Sadowski 2014; Bulò et al. 2016) が論じたように、ニューラルネットワークの全体のネットワークは各サブネットワークのアンサンブルに相当している。全体のネットワークの性質はパラメータ集合に分散的に学習されているため、そのパラメータの部分集合を用いる各サブネットワークもその全体のネットワークと非常に近い挙動を示す。例えば、一例として、学習後の全体のネットワークを枝刈り (Blalock et al. 2020) した場合も同様の精度を達成する。また、全体のネットワークの出力 (分類層の直前) と dropout によるサブネットワークを比べた

⁶ dropout は、ニューラルネットワークの各層の出力結果である隠れ層ベクトルに対して、dropout のマスクベクトルを要素積で掛け合わせることで実現できるため、ミニバッチ内の各事例について各マスクベクトルをミニバッチとして用意・保持し、そのままバッチ演算を行うことが可能である (付録 B 参照)。なお、本論文の dropout とは違い、パラメータ行列自体に行列形式の dropout マスクをかける dropconnect (Wan et al. 2013) を用いたい場合は、異なるマスクを用いたバッチ演算を行うことができない。

り, サブネットワーク同士の出力を比べてたりしてみても, 経験的にはそれらの出力は非常に近い値になる⁷. 本提案手法では, 上記の「異なるサブネットワーク間の挙動は平均的には同じである」及び「サブネットワークと全体ネットワークの挙動は平均的には同じである」というニューラルネットワークの特性を利用している. これは, 典型的に性能を保持したまま枝刈りが行えるような, 冗長なパラメータを持つネットワークではしばしば成り立つと考えられる. 一方で, もし十分なパラメータを持たない小さなモデルや低次元空間で計算を行うような場合では, サブネットワークごとの挙動に違いが生まれやすくなり, 提案手法の有効性も下がりやすい. 本論文では, そのような小さいネットワークのモデルは扱わず, 提案手法の高速という利点が特に役に立つ BERT などの巨大なネットワークでのみ実験を行った.

3.3 効率的な事例固有のマスクの実装

愚直には, 事例固有のマスクを用いるということは, データセット内の事例数に比例して新たにパラメータを用意する必要がある, メモリ空間やディスク容量増加の原因になりうる. しかし, このコストは本節で提案する **揮発性マスク生成** によって $O(|D|)$ から $O(1)$ へと劇的に減らすことができる.

まず注目すべき点として, 提案手法の turn-over dropout では, マスクはランダムなベクトルであればよく, かつ訓練中に更新されることはない. また, 一般的なランダム配列生成の実装ではシード値とサイズを与えることで決定的な挙動をし, 常に固定のベクトルを生成することができる. 例えば, 事例ごとのインデックス番号をシード値として与えることで, 常にその事例に応じたベクトルを生成することができる. したがって, マスクのベクトルそのものは保存しておく必要がなく, それらベクトルを生成する手続き (生成アルゴリズムとシード値) さえ保持しておいて必要な際に生成を行えばマスクのパラメータが得られることになる. 保存や保持のための容量や空間計算量を減らす代わりに, ランダム値の再生成の時間計算量がかかることになるが, 一般にランダム値の生成は非常に高速なためほとんど無視することができる⁸.

4 実験

本論文の貢献は, 式 (1) の値を推定するための高速な手法を提案することである. これによ

⁷ 例えば, 4 節の実験で扱った, CIFAR-10 (Krizhevsky 2009) の訓練データセットで学習した VGGNet (Simonyan and Zisserman 2015) において, テストデータセットの画像それぞれについて, 全体のネットワークの出力と dropout によるサブネットワークの出力を比較すると, そのコサイン類似度は平均で 0.9897 となった. また, 異なるサブネットワーク同士でのコサイン類似度も 0.9804 となった.

⁸ 本論文の実験では, 揮発性マスク生成法によって, 事例固有のマスクを利用する場合に保持する必要があるパラメータ数を減らし, 容量とメモリの問題を回避できた. しかし, さらに膨大なデータを使う場合などの特有のケースでは, 問題の回避のために実装にさらなる工夫が必要となる場合がある. そのようなケースの説明とさらなる解決方法については付録 C に詳述した.

り、先行研究の (Koh and Liang 2017) や (Hara et al. 2019) の手法では膨大な計算コストがかかってしまう、モデルの予測結果ごとの解析を短時間で実行できる。提案手法の計算量についての優位性は、2.3 節で議論した。本節では、提案手法 turn-over dropout を幾つかの分類タスクで学習するニューラルネットワークモデルに適用し、速度面以外のその有効性を示す。まず、4.2 節では、turn-over dropout の適用時に、サブネットワークがどのような振る舞いをしているのかをその学習過程を観察することで検証する。そして、4.3 節では、実際にモデルが誤り予測をした場合についてその原因となった訓練事例を解析し、解釈性の高い予測根拠 (Jacovi and Goldberg 2020) を提示できていることを示す。4.4 節では、推定された値が予測に紐付いており信頼性のあることを確認するために、影響値を用いたデータフィルタリングの実験を行い、“悪い” 影響をもたらしていると推定された訓練事例を除外した場合にモデルの汎化性能が向上することを示す。最後に、4.5 節では、dropout やスパースなネットワークでの訓練がときとしてモデルの性能を悪化させうる問題について議論する。

4.1 データセットとモデル

まず1つ目のデータセットとして、二値極性分類タスクの Stanford Sentiment TreeBank (SST-2) (Socher et al. 2013) を用いた。SST-2 では、映画のレビューテキストについてそのテキストが “positive” なのか “negative” なのかの二値ラベルが付与されている。元の訓練データセットからランダム抽出した 5,000 事例を訓練データセットとして使い、開発データセットとしては元の 872 事例をそのまま用いた。モデルは、事前訓練済みの BERT (Wolf et al. 2019) に基づく分類器を、adapter modules (Wolf et al. 2019) によって構築した。これは BERT のネットワークの各ブロックについて、新たに枝分かれネットワークを挿入し、元のネットワーク出力に対する残差として和を取るような構造を持つ。元々の BERT のネットワークは事前訓練後のパラメータを固定し、新たに追加される枝分かれネットワークと分類層のみを SST-2 で fine-tuning する。また、turn-over dropout も同様に fine-tuning が行われる箇所のみに適用する。

2つ目に、さらに複雑なタスクとして、Yahoo! Answers Dataset (Zhang et al. 2015) をデータセットとして用いた。これは、投稿者の質問のトピックが 10 種類の分類のうちどれに属するかを予測するタスクであり、単語だけでは判別できないような抽象度の高いトピックへ分類する必要がある。訓練事例数やモデル等の設定は上の SST-2 と同様にした。

3つ目のデータセットとして、自然言語処理だけでなく画像認識への応用として、犬・飛行機などからなる 10 クラスの画像分類タスクである CIFAR-10 (Krizhevsky 2009) の 50,000 事例の訓練データセットと 10,000 事例の開発データセットを用いた。畳み込みニューラルネットワークからなる VGGNet19 (Simonyan and Zisserman 2015) による分類器を訓練した。turn-over dropout は、11 層目以降に適用した。

BERT と VGGNet19 の両分類器は共に分類の正解ラベルと予測の差に関するクロスエントロ

ピーの最小化で最適化した. より詳細な設定は付録 A に示した.

4.2 学習曲線

turn-over dropout では, 事例固有マスクを適切に切り替えて生まれるサブネットワークが各事例を見ていないサブネットワークとして使えることを前提に置いている. 本節では, そのような特性が turn-over dropout によって本当に獲得されているのかを直接的に検証するために, サブネットワークごとの損失の値について学習曲線を観察する. 具体的には, 各訓練事例を見ていない反転マスク $\widetilde{\mathbf{m}}(z_{\text{train}})$ のサブネットワーク $f_D^{\widetilde{\mathbf{m}}(z_{\text{train}})}$ が, その対応する訓練事例 z_{train} について過剰適合 (overfitting) しないことを確認する. BERT を SST-2 で訓練した際の損失関数の変化を図 2 に示した.

まずはじめに, 2つの点線が示すのはテストデータセットで計算された損失であり, 下に行けば行くほど性能は改善しているとみなせる. 2つの点線はどちらも訓練が進むにつれて, 一定の値まで下がっていった後に概ね停滞し, 最後にはわずかに上昇していく. 赤い点線が示すのは, 各訓練事例の固有マスク $\mathbf{m}(z_{\text{train}})$ のサブネットワーク $f_D^{\mathbf{m}(z_{\text{train}})}$ を用いた場合の平均で, 青い点線はその反転マスク $\widetilde{\mathbf{m}}(z_{\text{train}})$ のサブネットワーク $f_D^{\widetilde{\mathbf{m}}(z_{\text{train}})}$ を用いた場合の平均である. 訓練に使っていないテストデータセットにおける性能は一般に汎化性能を表すことから, 事例固有マスクによるサブネットワーク $f_D^{\mathbf{m}(z_{\text{train}})}$ とその反転マスクによるサブネットワーク $f_D^{\widetilde{\mathbf{m}}(z_{\text{train}})}$ の間では未知のデータに対しては大きな性能差はないと言える.

次に, 2つの実線が表す訓練データセット上での損失に注目する. 赤い実線が示すのは, 事例固有マスク $\mathbf{m}(z_{\text{train}})$ のサブネットワーク $f_D^{\mathbf{m}(z_{\text{train}})}$ で計算した訓練データセット上での損失 $L(f_D^{\mathbf{m}(z_{\text{train}})}, z_{\text{train}})$ である. すなわち, 事例固有のマスクで訓練していく中で計算されて最小化される損失そのものである. この値は下降し続け, 通常の訓練でよく見られる最適化の傾向をた

図 2 BERT を SST-2 で訓練した場合の損失関数の学習曲線.

どっている。そして、青い実線が示すのは、反転マスク $\widetilde{m}(z_{\text{train}})$ のサブネットワーク $f_D^{\widetilde{m}(z_{\text{train}})}$ を用いて計算した訓練データセット上での損失 $L(f_D^{\widetilde{m}(z_{\text{train}})}, z_{\text{train}})$ である。注目すべき特徴として、青い実線では、今述べた赤い実線とは異なり、点線によるテストデータセットでの損失に近い値を取り続けている。すなわち、反転マスクに基づくサブネットワークは、訓練データセット上に過剰適合することなく、対応する訓練データを見なかったかのように振る舞っていることを示唆している。これにより、本実験では、turn-over dropout が各訓練事例に過剰適合しないサブネットワークを生み出せていることを確認できた。また、この観察は、開発用データセットを別途用意したり交差検定を行わずとも訓練データセットのみから汎化性能を計測することができる応用の可能性も示唆しており、turn-over dropout のさらなる活用先として今後の研究領域となりうると考えられる。

4.3 訓練事例への紐付けによる誤り予測の解釈

ニューラルネットワークによる予測モデルは、挙動の仕組みが人間にとって理解し難いブラックボックス性を持ち、それが信頼性や使い勝手を害しているとしばしば批判されている (Ribeiro et al. 2016)。その課題に関しても、訓練事例の影響値の推定を用いれば、ある予測に関して大きな影響を及ぼした訓練事例を特定できることを (Koh and Liang 2017) が示した。ある誤り予測に際して影響力のあった訓練事例を探し出すことができれば、それは予測を誤った原因を解釈するのに役に立つ。本節の実験では、本提案手法が、十分に解釈性 (plausibility) (Jacovi and Goldberg 2020) の高い「モデルの誤った予測に最も貢献した訓練事例」を提示できることを示す。モデルを訓練し終えた後に、開発用データセットの全事例について予測を行い、その中から誤った分類を与えていた事例だけを集めた。そして、それらの誤分類事例について、誤ったラベルの損失を低くする方向に最も大きい影響値を持つ訓練事例、すなわち、誤った分類に最も貢献してしまったような事例を特定した。

まず、図 3 に、BERT を SST-2 で訓練した場合の結果を 1 つ示した。上に示したテキストは本来

Test input: if s ##te ##ven so ##der ##berg ##h ' s ` solar ##is '
is a failure it is a glorious failure .
(if steven soderbergh ' s ` solaris ' is a failure it is a glorious failure .)
[positive → **negative**]
Influential: un ##fo ##cus ##ed , ex ##c ##ru ##ciating ##ly te ##dious
(unfocused , excruciatingly tedious)

図 3 BERT を SST-2 で訓練した場合における、テストセットの中で誤った予測がなされたテキストと、その誤ったラベルを持つ訓練事例の中で最も影響値が大きかったテキスト。誤って予測されたラベルを赤、正しいラベルを黒で示した。

“positive” と予測されるべきところ、誤って “negative” と予測した。最も影響を及ぼしたのは、その下に示した事例である。両者の共通点として、一般的な単語単位よりさらに細かく subword (Schuster and Nakajima 2012) として分解されたようなトークンを多く含んでいることが目を引く。特に固有表現 (steven, soderbergh, solaris) やその他の低頻度の単語 (excruciatingly) は細かく多くの subword に分割されやすく、このケースではそのような多くに散逸した subword の処理に失敗し、同様に多くの subword を含んでいた訓練事例からの影響で誤ったラベルを予測してしまったと推察できる。

さらに複雑なタスクとして、Yahoo! Answers Dataset (Zhang et al. 2015) を用いて同様の実験を行った。図 4 に Yahoo! Answers Dataset での結果を示した。誤分類された入力テキストは、“ch ##rist” というフレーズを含み、それは影響値の高い訓練事例にも含まれている。このタスクでのトピック分類は “Society & Culture” や “Business & Finance” などであり抽象度は高い。そのため、本来であれば “ch ##rist” のフレーズだけではトピックを分類できないような低次の特徴であるが、その箇所に引っ張られて、フレーズを共有している訓練事例の影響から誤って “Business & Finance” と予測してしまっていると考えられる。

次に、VGGNet を CIFAR-10 で訓練した場合の結果について、図 5 に示した。画像ではテキストよりもさらに直感的に解釈できる結果となった。例えば、左端の 2 列では、上の “bird” の画像をモデルは “airplane” として誤って予測した。その誤り予測に最も影響した訓練事例は、対応列の下側に示してある。どちらについても鳥の向きや羽の広げ方が飛行機の状態に類似しており、ほとんどシルエットしか見えないテスト事例については、誤って予測してしまうこと

Test input:

- why do some people assume they know who the ask ##er is , based on the question he asks ? from reading back the answers that i get on my questions , i am now officially a m ##us ##lim - ch ##rist ##ian - b ##udd ##his ##t , a straight ... it ok with you that i still don ' t have an identity - crisis ?

[Society & Culture → **Business & Finance**]

Influentials:

- is ch ##rist ##mas eve in the evening or is it just the day before ch ##rist ##mas ? my sister thinks that ch ##rist ##mas eve ...
- how i can be ch ##rist ##ain ? i want to be real ch ##rist ##ain , how i can be ch ##rist ##ain ask ch ##rist into ...

図 4 BERT を Yahoo! Answers Dataset で訓練した場合における、テストセットの中で誤った予測がなされたテキストと、その誤ったラベルを持つ訓練事例の中で最も影響値が大きかったテキスト。誤って予測されたラベルを赤、正しいラベルを黒で示した。

図 5 VGGNet を CIFAR-10 で訓練した場合における，テストセットの中で誤った予測がなされた画像（上の行）と，その誤ったラベルを持つ訓練事例の中で最も影響値が大きかった画像（下の行）．誤って予測されたラベルを赤，正しいラベルを黒で示した．

にも納得が行く結果となった．右側の残る 3 事例ペアについても “bird” → “dog”，“dog” → “deer”，“deer” → “bird” のような誤り予測について，誤り原因と考えられる学習特徴（形やレイアウトや色など）が推察できるような訓練事例を見つける事ができている．

4.4 データフィルタリング

影響値（式 (1)）は，前節の代表的な応用の他に，データフィルタリングにも応用できることがこれまで示されている (Koh and Liang 2017; Hara et al. 2019)．本節では，提案手法による影響値の計算結果の妥当性・信頼性 (faithfulness) (Jacovi and Goldberg 2020) を，データフィルタリングの実験を通して検証する⁹．

開発用データセット全事例に対する平均損失に関して，訓練事例ごとに式 (1) の値を計算し比較することで，どの訓練事例が汎化を妨げてしまうのか予想することができる．前節の誤り予測の解釈で論じたように，訓練データセットの中では，他の事例への予測に害をなす事例も存在する．例えば，ある訓練事例がテスト時の様々な未知事例に対して平均的に誤分類を誘発するとき，その訓練事例は汎化性能を損なう事例だといえる．本実験では，そのような訓練事例を特定し除外（フィルタリング）したのちに，モデルをフィルタリング後のデータセットで

⁹ 本節の実験は，提案手法による影響値の妥当性に関する一つの検証実験であり，データフィルタリングのタスクにおいて提案手法が（影響値推定以外にも含めた）他の手法に比べて有用であると主張するためのものではないが，参考のためにデータフィルタリングに必要な計算量について，先行研究の (Koh and Liang 2017) の手法との比較を付録 D に記載した．

訓練し直すことで、汎化性能を向上させることができることを示す。一方で、「汎化性能を損なう事例」がそもそも訓練データセットに含まれていなければフィルタリングで性能が上がることはない。今回は、訓練データセットの中に「汎化性能を損なう事例」が十分含まれていることが担保されているような応用上のケースとして、訓練時とテスト時でデータのドメインが変わるドメインシフトの設定で実験を行う。

訓練データセットとして映画レビューである SST-2 を用いて、一方で最終的な性能評価のテストデータとしては電化製品レビューである Multi-Domain Sentiment Dataset (Blitzer et al. 2007) の electronics サブセット (Elec) を用いた。Elec データセットは 200 事例を開発用データ、1,800 事例をテスト用データとしてランダムに分割した。なお、性能を最大化したい場合には最後の訓練時に開発用データも混ぜて訓練するなどが考えられるが、今回はデータフィルタリングの効果のみを検証するために、Elec のデータセットは訓練データセットには一切含めずに性能を比較した。

まず初めに SST-2 データセットで BERT を turn-over dropout を用いた fine-tuning で訓練し、その後に各訓練事例から Elec の開発用データに対する平均的な影響値を計算した。そして、その平均影響値が最も悪い（負に大きい）訓練事例を上位 1%取り除いたような、フィルタリング済みの訓練データセットを抽出し、再度 BERT を turn-over dropout なしで訓練した。そして、Elec のテストデータで性能を計測し、結果を表 2 に示した (1% Filtering)。比較対象として、ランダムに同量の 1%だけデータを取り除いた場合 (1% Random Filtering) とデータを取り除かずに訓練した場合 (No Filtering) の性能も並べて示した。turn-over dropout はデータフィルタリングの段階でのみ用いており、表で比較されている数値はすべて turn-over dropout なしで訓練された BERT のものである。実験は異なるシード値によって 4 回行い、それらの結果の平均と標準偏差を載せた。結果、分類精度についてもロスの値についても、データフィルタリングを行った場合に性能が向上することが確認できた。単にランダムにデータを取り除いた場合にはやはり性能が向上することではなく、turn-over dropout の推定量に基づくフィルタリングのみが効果的であった。この結果により、影響値の推定のターゲットを一事例でなく事例集合に設定して平均的な影響値を見ることで、対象ドメインでの汎化性能への影響を考慮し、データフィルタリングを行えることを示した。

	Accuracy (%)	Loss
1% Random Filtering	76.8 $\pm$ 1.1	0.521 $\pm$ 0.030
No Filtering	77.0 $\pm$ 0.9	0.536 $\pm$ 0.063
1% Filtering	78.3 $\pm$ 0.2	0.484 $\pm$ 0.008

表 2 データフィルタリングによる性能の変化。Loss はクロスエントロピーの値。4 回の実験の平均と標準偏差を示した。

4.5 Dropout の適用箇所

提案手法 turn-over dropout は対象となるモデルに dropout を導入し、訓練後に効率的に訓練事例の影響値を推定できるようにする手法である。dropout 自体は様々なニューラルネットワークで用いられており汎化性能の向上にも役立つことが多いが、過剰に適用した場合には、スパースなネットワークの訓練と同様にモデル性能が悪化してしまうことも考えられる¹⁰。そのような訓練の不安定性については未だ明らかになっていない点も多い (Evci et al. 2020)。本節では、どのように dropout を適用すれば「性能劣化の防止」と「訓練事例の影響値の推定」を両立できるかを最後に議論する。

まず、SST-2 で BERT を adapter module 付きで訓練する場合の、性能低下（テストデータにおける分類精度 (accuracy) の低下）の可能性について論じる。実験では、事例数が 5,000 の小さなデータセットで訓練した場合において、ベースラインの性能が 90.0% だったのに対して turn-over dropout を用いた場合に 88.3% へとわずかに精度が低下した。事例数がもう少し多い 20,000 事例での実験では、それぞれ 90.5% と 90.2% の精度となり、ほぼ性能劣化が見られなくなった。したがって、効率的な turn-over dropout が特に有用であるような大規模なデータセットを扱う場合においては性能の劣化が十分に小さくなり、実用性も十分保てる可能性を示唆している。一方で、adapter module を用いずに、元の BERT をそのまま使って全層で turn-over dropout を用いた場合には、訓練が不安定になり性能も劣化した。これは turn-over dropout 特有の新たな問題ではなく、通常の動的な dropout を $p = 0.5$ で用いた場合にも同様である。これは事前訓練の段階で扱っている dropout の非ゼロ要素率 $p = 0.9$ よりもかなり小さな $p = 0.5$ を使うことで情報の伝播を大きく妨げてしまい、BERT が安定した計算を行えていない可能性がある。adapter なしの fine-tuning でこのような小さい p を使えるようにする解決策としては、事前訓練の段階から小さい p を予め使っておくことが考えられる。また、事前訓練直後のモデルのパラメータを固定しておき、fine-tuning ではそのパラメータからの差分値のみを別途学習し、その差分値パラメータにのみ dropout をかけるようにすると、情報の伝播を大きく途切れさせることなく turn-over dropout を適用できる可能性がある。これは dropout の亜種である mixout (Lee et al. 2020) に turn-over dropout を応用したケースとしてみなせる。

画像分類の CIFAR-10 データセットで CNN ベースの VGGNet を全層への turn-over dropout で訓練した場合にも、訓練の不安定性や性能劣化が観測されたが、全層に適用する代わりに一部の層のみに適用した場合には訓練が安定するようになった。そこで我々は、各画像固有の影響を強く受ける層にのみ適用することで、訓練の安定性を上げながら推定を行えるようにする

¹⁰ $p = 0.5$ の dropout 自体は様々なニューラルネットワークで用いられており、汎化性能の向上にも役立っている。しかし、全結合層や限られた層のみに適用されている場合も多く、BERT や CNN の全ての層に $p = 0.5$ で適用した場合の影響については未知数である。また、本提案手法で新たに導入された、事例ごとに紐付いた静的なマスクを使った場合についても、性能への影響もこれまで明らかにはなっていない。

ことを考えた. (Asano et al. 2020) は, CNN の序盤の層は画像群の統計値について限られた情報のみを保持しており, その低次の統計量についてはたった一枚の画像からですら学習可能であることを示している. この知見に基づき, 序盤の層は各画像ごとに固有の情報を多く学習することは少ないと仮定し, 後半の層 (11 層以降) のみに turn-over dropout を適用した. その結果, 分類精度は 86.2% となり, ベースラインの 92.0% よりやや悪い程度の訓練を行えるようになった. ここにさらに, 正則化としてのランダムな dropout も合わせて適用することで, 91.3% まで性能が回復し, ベースラインに匹敵する性能を出すことができた.

BERT と VGGNet に基づくこれらの検証により, いくつかのケースでは, dropout を適用しつつ性能を保持するために適用方法を修正・探索する必要性やその対策方法の例が明らかになった. dropout やスパースなネットワークにおける学習の不安定性 (Evcı et al. 2020) については, 現状明らかになっていない点も多く, 提案手法の枠内に閉じない今後の重要な研究課題である.

5 結論

本論文では, 旧来の手法が直接的に適用しづらいようなパラメータ数と計算量が膨大なニューラルネットワークを用いる場合についても, 非常に小さい実用的な計算コストで訓練事例の影響値を推定する方法を提案した. 提案手法では, 訓練時に dropout が従来用いている確率的なマスクベクトルの代わりに訓練事例ごとに紐付いたマスクを決定的に用いる. これにより, 訓練後のネットワークは, 各事例の影響を受けていないパラメータだけからなるサブネットワークを個別に併せ持つことができ, そのサブネットワークの出力を比較することで影響値を推定できる. また, 事例とサブネットワークの紐付け情報を保存しなくてはいけないコストについても, マスクのランダムベクトルそのものでなくそれを生成するためのランダムシードのみを保存することで, 劇的に保存容量や空間計算量を抑えられる. 実験では, 実際に各サブネットワークは過適合していないことを検証し, さらに提案手法が誤り予測の事例ベースの解釈に活用できることを, データフィルタリングによる定量評価も用いて示した. 今後のさらなる研究課題としては, 提案手法のみならずスパースなネットワークを訓練した際に見られる性能低下現象の解析が考えられる. また, さらなる応用の模索としても, 4.2 節で示唆された汎化性能を訓練データセットのみから計測できる性質に基づいた交差検定の代替としての活用法や, オンラインに高速に影響値を推定できる点を生かして各事例の学習自体を改善する方向性も有望だと考えられる.

謝 辞

本研究の一部は, JST, CREST, JPMJCR20D2 (JPMJCR20D2) の助成を受けたものである.

また、本研究の一部は、JSPS 科研費 JP19H04162, 人工知能研究振興財団研究助成事業 30AI036-8 の助成を受けたものである。また、本研究の一部は、JST, ACT-X, JPMJAX200S の支援を受けたものである。

本論文の一部は、First Workshop on Simple and Efficient Natural Language Processing で発表されたものである (Kobayashi et al. 2020)。

参考文献

- Asano, Y. M., Rupprecht, C., and Vedaldi, A. (2020). “A Critical Analysis of Self-supervision, or What We Can Learn from a Single Image.” In *International Conference on Learning Representations (ICLR)*.
- Bachman, P., Alsharif, O., and Precup, D. (2014). “Learning with Pseudo-Ensembles.” In *Advances in Neural Information Processing Systems 27*, pp. 3365–3373.
- Baldi, P. and Sadowski, P. (2014). “The Dropout Learning Algorithm.” *Artificial Intelligence*, **210** (C), pp. 78–122.
- Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., and Gutttag, J. (2020). “What is the State of Neural Network Pruning?” In Dhillon, I., Papailiopoulos, D., and Sze, V. (Eds.), *Proceedings of Machine Learning and Systems*, Vol. 2, pp. 129–146.
- Blitzer, J., Dredze, M., and Pereira, F. (2007). “Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification.” In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 440–447, Prague, Czech Republic. Association for Computational Linguistics.
- Bulò, S. R., Porzi, L., and Kotschieder, P. (2016). “Dropout distillation.” In *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48, pp. 99–107.
- Cook, R. D. (1977). “Detection of Influential Observation in Linear Regression.” *Technometrics*, pp. 15–18.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186.
- Evci, U., Ioannou, Y. A., Keskin, C., and Dauphin, Y. N. (2020). “Gradient Flow in Sparse Neural Networks and How Lottery Tickets Win.” *CoRR*, **abs/2010.03533**.
- Han, X., Wallace, B. C., and Tsvetkov, Y. (2020). “Explaining Black Box Predictions and Unveiling Data Artifacts through Influence Functions.” In *Proceedings of the 2020 Annual*

Conference of the Association for Computational Linguistics (to appear).

- Hara, S., Nitanda, A., and Maehara, T. (2019). “Data Cleansing for Models Trained with SGD.” In *Advances in Neural Information Processing Systems 32*, pp. 4215–4224.
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). “Improving Neural Networks by Preventing Co-adaptation of Feature Detectors.” *CoRR*, **abs/1207.0580**.
- Jacovi, A. and Goldberg, Y. (2020). “Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness?” In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4198–4205, Online. Association for Computational Linguistics.
- Kobayashi, S., Yokoi, S., Suzuki, J., and Inui, K. (2020). “Efficient Estimation of Influence of a Training Instance.” In *Proceedings of SustainNLP: Workshop on Simple and Efficient Natural Language Processing*, pp. 41–47, Online. Association for Computational Linguistics.
- Koh, P. W. and Liang, P. (2017). “Understanding Black-box Predictions via Influence Functions.” In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1885–1894.
- Krizhevsky, A. (2009). “Learning Multiple Layers of Features from Tiny Images.” Tech. rep..
- Lee, C., Cho, K., and Kang, W. (2020). “Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models.” In *International Conference on Learning Representations (ICLR)*.
- Mosbach, M., Andriushchenko, M., and Klakow, D. (2021). “On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines.” In *International Conference on Learning Representations*.
- Ouchi, H., Suzuki, J., Kobayashi, S., Yokoi, S., Kuribayashi, T., Konno, R., and Inui, K. (2020). “Instance-Based Learning of Span Representations: A Case Study through Named Entity Recognition.” In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6452–6459.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc.
- Plumb, G., Molitor, D., and Talwalkar, A. S. (2018). “Model Agnostic Supervised Local Explanations.” In *Advances in Neural Information Processing Systems 31*, pp. 2515–2524.
- Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). “Stanza: A Python Natural

- Language Processing Toolkit for Many Human Languages.” In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 101–108.
- Ribeiro, M., Singh, S., and Guestrin, C. (2016). ““Why Should I Trust You?”: Explaining the Predictions of Any Classifier.” In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pp. 97–101, San Diego, California. Association for Computational Linguistics.
- Schuster, M. and Nakajima, K. (2012). “Japanese and Korean Voice Search.” In *International Conference on Acoustics, Speech and Signal Processing*, pp. 5149–5152.
- Simonyan, K. and Zisserman, A. (2015). “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In *International Conference on Learning Representations*.
- Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank.” In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” *Journal of Machine Learning Research*, **15**, pp. 1929–1958.
- Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). “Regularization of Neural Networks using DropConnect.” In Dasgupta, S. and McAllester, D. (Eds.), *Proceedings of the 30th International Conference on Machine Learning*, Vol. 28 of *Proceedings of Machine Learning Research*, pp. 1058–1066, Atlanta, Georgia, USA. PMLR.
- Williams, A., Nangia, N., and Bowman, S. (2018). “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference.” In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). “HuggingFace’s Transformers: State-of-the-art Natural Language Processing.” *arXiv*, **abs/1910.03771**.
- Zhang, T., Wu, F., Katiyar, A., Weinberger, K. Q., and Artzi, Y. (2021). “Revisiting Few-sample BERT Fine-tuning.” In *International Conference on Learning Representations*.
- Zhang, X., Zhao, J., and LeCun, Y. (2015). “Character-level Convolutional Networks for Text Classification.” In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett,

R. (Eds.), *Advances in Neural Information Processing Systems 28*, pp. 649–657. Curran Associates, Inc.

Zhang, X., Zhu, X., and Wright, S. (2018). “Training Set Debugging Using Trusted Items.” In *AAAI Conference on Artificial Intelligence*, pp. 4482–4489.

付録

A 実験設定

本節では、4 節で行った実験についてより詳細に設定を記述する。

本論文では、BERT として参照した場合にはすべて、現在最も広く使われている Transformers ライブラリ (Wolf et al. 2019) に含まれる *bert-base-cased* の事前訓練済みモデルを用いた。タスクごとの fine-tuning のハイパーパラメータは、予備実験を行い、開発用データセットにおける精度を元にして調整した。その他の設定は Transformers library (Wolf et al. 2019) のデフォルトに則った。最終的に使用した及び探索したハイパーパラメータの範囲 ([]) は次の通りである。学習率 = $5e-5$ [$2e-5$, $5e-5$], バッチサイズ = 32, 訓練データセット周回数 = 3 [3, 6, 10]. turn-over dropout を使う場合には周回数を 10 とした。3 や 6 の周回数でも同様の精度が出る場合はあったものの、これまでの研究 (Zhang et al. 2021; Mosbach et al. 2021) で報告されている通り、長めの 10 とした場合が最も安定した。訓練事例数は文書分類のいずれの実験でも 5,000 個としたが、図 2 で学習曲線を示す際には、視認しやすくするために、事例数を 25,000 個へと増やした設定で訓練を行った。Multi-Domain Sentiment Dataset (Blitzer et al. 2007) は <http://www.cs.jhu.edu/~mdredze/datasets/sentiment/> のデータを用いた。そこから ‘electronics’ に含まれるデータのみを抽出し、Stanza (Qi et al. 2020) を用いてトークンを分割し、既に分割済みである SST-2 とフォーマットを揃えた。Yahoo! Answers Dataset (Zhang et al. 2015) は <https://github.com/LC-John/Yahoo-Answers-Topic-Classification-Dataset> のデータを用いた。

VGGNet19 は momentum SGD (momentum = 0.9, weight decay = $5e-4$) を用いて最適化を行った。学習率は、0.1 から始め、150 周後の時点と 225 周後の時点でそれぞれ 0.1 倍にした。データ拡張からは左右反転を除いた。実装は、<https://github.com/kuangliu/pytorch-cifar> に基づいた。

B 事例固有マスクを用いた Dropout の実装例

事例固有マスクを用いた dropout を, Python による深層学習フレームワーク PyTorch (Paszke et al. 2019) によって実装する場合の例を示す.

```
1 import torch
2 batchsize = 8
3 dimension = 100
4 linear = torch.nn.Linear(dimension, dimension)
5
6 def make_instance_masks(instance_indices, p=0.5):
7     seed = 777
8     max_instance_idx = 999
9     all_masks = torch.bernoulli(torch.full((max_instance_idx, dimension), p),
10                                     generator=torch.Generator().manual_seed(seed)).float()
11     / p
12     # shape = (999, 100)
13     extracted_masks = all_masks[instance_indices]
14     # shape = (8, 100)
15     return extracted_masks
16
17 # prepare input
18 x_input = torch.rand(batchsize, dimension)
19 x_instance_indices = torch.tensor([1, 22, 333, 44, 5, 66, 777, 88]).long()
20 # run hidden layer
21 hidden = torch.nn.functional.sigmoid(linear(x_input))
22 # generate dropout mask
23 dropout_mask = make_instance_masks(x_instance_indices)
24 # apply dropout
25 hidden_dropped = hidden * dropout_mask
```

Listing 1 PyTorch における事例固有マスクを用いた dropout 実装の例.

C ハッシュに基づくマスクの合成

揮発性マスク生成 (3.3 節) は, モデルの保存容量と推論時のメモリ使用量 (空間計算量) を最小にして問題を解決するために役立つ方法である. 一方で, 実装によっては, その方法を用いたとしても一時的なメモリ使用量の増加が無視できない量になってしまう可能性がある.

まず大枠として, ネットワークの前向き計算の中では, ある層においてマスク生成が行われ, 処理を行い, その層の処理を終えたらマスクは揮発し, メモリ使用量はゼロになる. あるミニバッチ内の各インスタンスに対するマスク生成は, 典型的な効率の良い実装において, 次のような数段階の操作に分けて実行される. まず, 始めにランダムシードによってデータセット内の全事例についてマスクを生成する. これは, 一般的なランダムベクトルの生成においては, 一括で生成する方がより効率が良いためである (例えば for-loop によってミニバッチ内の各事例ごとに生成命令を呼ぶ場合, 所要時間は大幅に伸びる). そして, その一括で生成した巨大なラン

ダムマスクの中から, ミニバッチ内の事例に関する箇所だけをインデキシングなどによって抜き出して, 所望のランダムマスクをミニバッチ化したものを得る. このとき, 最初の一括生成の段階において一時的に総事例数に依存した大きなメモリ使用量を必要とする. 本論文の実験では, その一時的なメモリ使用量でメモリがあふれることはなかったが, モデルサイズやデータサイズに応じて問題が生じる可能性がある. そのような場合の解決策として, ハッシュに基づくマスクの合成を活用する方法をここで提案する. 端的には, $K (\ll |D|)$ 個のランダムマスクから $k (< K)$ 個選んで組み合わせて作ったようなマスクを各事例ごとに用いるという枠組みである. まず始めに K 個の d 次元バイナリベクトルをランダムに $\text{Bernoulli}(1 - p^{1/k})$ からサンプリングして用意し, *codebook* とする. また, ある事例を受け取り, k 個のインデックス ($1 \sim K$) を出力するようなハッシュ関数を用意する. これらによって, ある事例ごとに *codebook* から k 個のランダムベクトルを決定的に選択できるようになる. そして, それら k 個のランダムベクトルについて, 要素ごとに総乗 (あるいはバイナリに関する論理和) を取る. このとき, その結果生成されるバイナリベクトルの各要素が 1 である確率は p であるため, ここに dropout のスケール値である $\frac{1}{p}$ をかけることで, 結果的に当初欲しかったランダムベクトルと同様の分布に従っているランダムベクトルを構成的にかつ決定的に生成することができる. このアルゴリズムが一時的に要する最大の空間計算量は, サイズ $K \times d$ の *codebook* に加えて, ミニバッチ (サイズ b) 内の各事例について k 個のベクトルを保持したときのものになり, $O(Kd + kbd)$ まで抑えられることになる. 実際の実装上は, kb 個のベクトルは複製されず, *codebook* への参照のみで済むことも多く, その場合はさらに *codebook* のみのコストである $O(Kd)$ まで抑えられる. この *codebook* についても揮発性マスク生成の方法を適用することがもちろん可能である. それにより, データサイズは増えた場合においても, 事例固有マスクを使うことによる計算量は十分に小さく抑えることができる.

D データフィルタリングにかかる計算量

本論文の焦点はデータフィルタリングへの応用性ではないが, 先行研究の (Koh and Liang 2017) と提案手法それぞれでデータフィルタリングを行う場合の理論計算量と実時間の概算についても参考のために記載する.

今回のそれぞれの実験で求めたい値は, (a) テストデータセット内の各事例への予測結果を解釈・解析する場合 (式 (1) の問題設定) と (b) 開発用データセット内の平均損失を用いるデータフィルタリングを行う場合とでは, 異なっている. (a) と (b) の各問題設定で, ある訓練事例 $z_i \in D$ に対して求めたい値は, 具体的には次のとおりである (以下, 単純化のために開発用デー

タについても dev ではなく test と表記する).

$$(\text{各 } z_{\text{target}} \in D_{\text{test}} \text{ について}) \quad L(f_{D \setminus \{z_i\}}, z_{\text{target}}) - L(f_D, z_{\text{target}}) \quad (\text{a})$$

$$\frac{1}{|D_{\text{test}}|} \sum_{z_{\text{target}} \in D_{\text{test}}} L(f_{D \setminus \{z_i\}}, z_{\text{target}}) - L(f_D, z_{\text{target}}) \quad (\text{b})$$

(b) の値は (a) の方法から間接的に推定でき, 各開発用事例 $z_{\text{target}} \in D_{\text{test}}$ について (a) を求めたのちに平均を取ることで (b) を求められる. そのため, 提案手法では, どちらの場合についても計算量は $O(F|D||D_{\text{test}}|)$ となる. 一方で, (Koh and Liang 2017) の手法では, 式 (1,a) の各項の値を求めてから差を得るのではなく, 損失の勾配情報を用いて直接その差を推定する. その処理は, 損失に対するパラメータ勾配を計算する処理 A と, その勾配それぞれを用いて訓練事例全てに対して計算する処理 B に分けられる. したがって, (a) の影響値推定の場合では, 損失に対するパラメータ勾配計算の A がテスト事例の数だけ必要になり, 処理 B も同様の回数行われる. 一方で, (b) のデータフィルタリングでは, 開発用データセット全体に対する平均損失に対するパラメータ勾配を 1 つ求めて, 処理 B を一度だけ行えば済むため, 開発用データのサイズに対して効率的である. 具体的には, 計算量オーダーについては, (a) では $O(F|D||D_{\text{test}}| + (F+F')|D_{\text{test}}| \underbrace{rt}_{\approx |D|} b)$ であり, (b) では $O(F(|D|+|D_{\text{test}}|) + (F+F') \underbrace{rt}_{\approx |D|} b)$ となる. なお, キャッシュを活用することにより (a) の場合でも高速化を行える可能性はあるが, パラメータ勾配のデータサイズが非常に大きいため, ハードウェアや実装の都合上実現は困難だと考えられる. なお, 訓練事例ごとにそのみを抜いて訓練したモデルを用意する方法 (Re-train) では, フィルタリングのための推定に $O(F|D||D_{\text{test}}|)$, 訓練に $O(\text{epoch 数} \times F|D|^2)$ の計算量が必要となる.

計算時間の概算としては, 例えば, 含意関係認識の Multi-Genre NLI dataset (Williams et al. 2018) のデータセットにおいて開発用事例 200 個と訓練事例 10,000 個を使う場合, (Koh and Liang 2017) の手法では約 12 分で終わる¹¹一方で, 提案手法で上記のナイーブな計算を行う場合には約 110 分程度かかると考えられる.

E 訓練事例自身への影響値

訓練による影響値の解析について簡単な実験を行った結果を示す. 教師あり学習において, ある事例で訓練した場合にはその事例の入力からは出力を予測できるようになることが期待される. この実験では, ある訓練事例のその事例自身への影響値 (自己影響値) を計測し, 上で述

¹¹ 推定の精度と速度のトレードオフを調整するハイパーパラメータが存在し, それにより速度が大きく変わるため一概には言えないが, 今回は (Han et al. 2020) の実装のデフォルト値を用いた場合について試算した.

図 6 BERT を SST-2 で訓練した場合の、訓練データセットに対する自己影響値のヒストグラム

べた期待に一致することを確認した。BERT を SST-2 で訓練した際の、訓練事例 5,000 個について自己影響値を計測し、図 6 にヒストグラムとして示した。ほぼ全ての事例について自己影響値が正になっていることから、ある事例での訓練がその事例に対して正の影響を及ぼしているという期待に沿った計算が行えていることがわかる。また、CIFAR-10 で訓練した VGGNet でも同様の傾向を示した。

略歴

小林 颯介：2016 年東北大学大学院情報科学研究科博士前期課程終了。同年、株式会社 Preferred Networks に入社。2018 年東北大学同研究科博士後期課程入学。現在在学中。

横井 祥：2020 年東北大学大学院情報科学研究科博士課程修了。同年より東北大学大学院情報科学研究科助教。理化学研究所 AIP センター数理統計学チーム客員研究員兼任。

鈴木 潤：2001 年から 2018 年まで日本電信電話株式会社コミュニケーション科学基礎研究所研究員（主任研究員／特別研究員）。2005 年奈良先端科学技術大学院情報科学研究科博士後期課程修了。現在、東北大学データ駆動科学・AI 教育研究センター教授。

乾 健太郎：1995 年東京工業大学大学院情報理工学研究科博士課程修了。2010 年より東北大学大学院情報科学研究科教授。2016 年より理化学研究所 AIP センター自然言語理解チームリーダー兼任。

(2020 年 11 月 1 日 受付)

(2021 年 2 月 8 日 再受付)

(2021 年 3 月 11 日 採録)