DEEP LEARNING JP
[DL Papers]

Understanding Black-box Predictions via Influence Functions

Fukuta Keisuke Harada&Ushiku Lab.

書誌情報

- ICML 2017 best paper!!
- From Stanford University
- Deep Learning解析系

Background

「DLモデルは、何故その出力をしたのかがよくわからない」

「Black boxは使いづらい」

→ 何故モデルがこの出力したのか知りたい

- モデル改善、新たな知識発見、実アプリケーションへ向けて

関連研究

- あるdata点周りのみを考慮してよりsimpleなmodelでfittingさせて解析
  [Ribeiro, 2016]
- Data点にノイズを加えて予測がどう変化するか
  [Simyonyan et al., 2013], [Adler et al., 2016]

これらの研究は与えられたモデルがどう予測を変化させるかを解析したもの

本研究は、”なぜこのモデルに至ったのか”について考察するアプローチ

概要

- “Study models though the lens of their training data”
  - モデルがどうなっているか -> なぜモデルがそうなったのか
  - 学習サンプルに着目
  - もしこのサンプルを学習に使わなかったらどうなるのか

- 特定の学習サンプルがモデルパラメータに与える影響を定量的に解析
  - 一つずつ抜いて再学習すればいい話ではあるが時間がかかりすぎる
  - 統計で古くから利用されていた影響関数(influence function)を利用
    - Generalized linear modelsではすでに解析例があるが、比較的小さいモデルだった

Contribution

1. 影響関数(influence function)を用いて、特定の学習データの有無や、
   学習データに加える摂動が予測結果に与える影響を定式化
2. DNN等の複雑なモデルに対する影響関数の効率的な計算手法の提案
   - ナイーブに行うとパラメータ数の二乗のオーダーの計算となり、不可能
3. 実際の解析例、アプリケーションの検証
   1. モデルの挙動の理解
   2. ネットワークを混乱させる摂動の計算
      - Adversarial *training* samplesの作成
   3. ドメイン不適合の検知
   4. ノイズラベルの検出

影響関数

特定の学習データがモデルの予測結果に与える影響を定量的に知りたい

ただし、再学習無しで

1. 特定の学習データがモデルパラメータに与える影響を計算
2. それが予測結果に与える影響を計算

準備

- Training dataset $\{z_1, z_2, \ldots, z_n\}$ ， $z_i = (x_i, y_i)$
- データ点 z に対する, parameter θ の損失を $L(z, \theta)$とする

→ datasetの経験損失 $R(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$

- $\hat{\theta} \equiv argmin_{\theta} \ R(\theta)$
- 仮定 (あとでこの仮定についても検証します)
  - 損失$R(\theta)$は凸関数 ($\hat{\theta}$はglobal minimum)
  - 損失$R(\theta)$は二回微分可能

影響関数

損失を最小化する$\hat{\theta}$が計算できたとする

$$\hat{\theta} \equiv argmin_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$

ある学習データ$z = (x, y)$を**使わずに**学習を行った場合のパラメータを$\hat{\theta}_{-z}$とする

$$\hat{\theta}_{-z} \equiv argmin_{\theta} \frac{1}{n} \sum_{z_i \neq z} L(z_i, \theta)$$

→ $\hat{\theta}_{-z} - \hat{\theta}$を計算したい

＊当然、再学習すれば得られるがすべてのデータ点を一つずつ抜いて
行うのは現実的ではない

影響関数

そこで、損失を少しいじり、損失に対するzの影響をパラメータϵで管理
($\epsilon = -\frac{1}{n}$ で先程の式と一致)

$$\hat{\theta}_{\epsilon,z} \equiv argmin_{\theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon\, L(z,\theta)$$

ここで、Cook & Wisbergの導出から、

$$I_{up,params}(z) \equiv \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\, \nabla_{\theta} L\left(z,\hat{\theta}\right)$$

特定のデータの重みをupweight
した時のparametersの影響関数

$$H_{\hat{\theta}} \equiv \frac{1}{\mathrm{n}}\sum_{i=1}^{n} \nabla_{\theta}^{2}\, L(z_i,\hat{\theta})$$

Training lossのヘッシアン

＊ つまり、$I_{up,params}(z)$は、微小なϵに対するパラメータの変化量を表す

また、$\hat{\theta}$は損失を最小にするものと仮定してるのでヘッシアン$H_{\hat{\theta}}$は正定値 -> 逆行列が存在

影響関数

$\epsilon = -\frac{1}{n}$が微小であると仮定すると、

$$\hat{\theta}_{-z} - \hat{\theta} \cong -\frac{1}{n} I_{up,params}(z)$$

これで再学習無しで **特定の学習データがモデルパラメータに与える影響** が計算できた

次に、それがモデルの予測に与える影響を計算

$$I_{up,loss}(z, z_{test}) \equiv \frac{dL\left(z_{test}, \hat{\theta}_{\epsilon,z}\right)}{d\epsilon}\Big|_{\epsilon=0}$$

特定のデータの重みをupweightした時のtest lossの影響関数

$$= \nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T \frac{d\hat{\theta}_{\epsilon,z}}{\mathrm{d}\epsilon}\Big|_{\epsilon=0}$$

Chain rule

$$= -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z, \hat{\theta}\right)$$

さっきの結果

影響関数

結局、特定の学習データがモデルの予測（ロス）に与える影響は、

$$I_{up,loss}(z, z_{test}) = -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z, \hat{\theta}\right)$$

$\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T$: あるtest dataのロスに対する勾配

$H_{\hat{\theta}}^{-1}$: ヘッシアンの逆行列

$\nabla_\theta L\left(z, \hat{\theta}\right)$: ある学習データのロスの勾配

これで計算できる！！

余談：Cook & Wisbergの導出

$\hat{\theta}_{\epsilon,z} = argmin_{\theta}\{\mathrm{R}(\widehat{\theta)} + \epsilon\, L(z,\theta)\}$ なので、$\theta_{\epsilon,z}$の点での$\mathrm{R}(\theta) + \epsilon\, L(z,\theta)$ の1次微分は0

$$0 = \nabla R\left(\hat{\theta}_{\epsilon,z}\right) + \epsilon \nabla L(z, \hat{\theta}_{\epsilon,z})$$

$\epsilon \to 0$だと仮定して1次までTaylor展開すると

$$0 \cong \left[\nabla R\left(\hat{\theta}\right) + \epsilon \nabla L\left(z, \hat{\theta}\right)\right] + \left[\nabla^2 R\left(\hat{\theta}\right) + \epsilon \nabla^2 L\left(z, \hat{\theta}\right)\right]\Delta_{\epsilon}$$

Δ_{ϵ}に関して解くと、

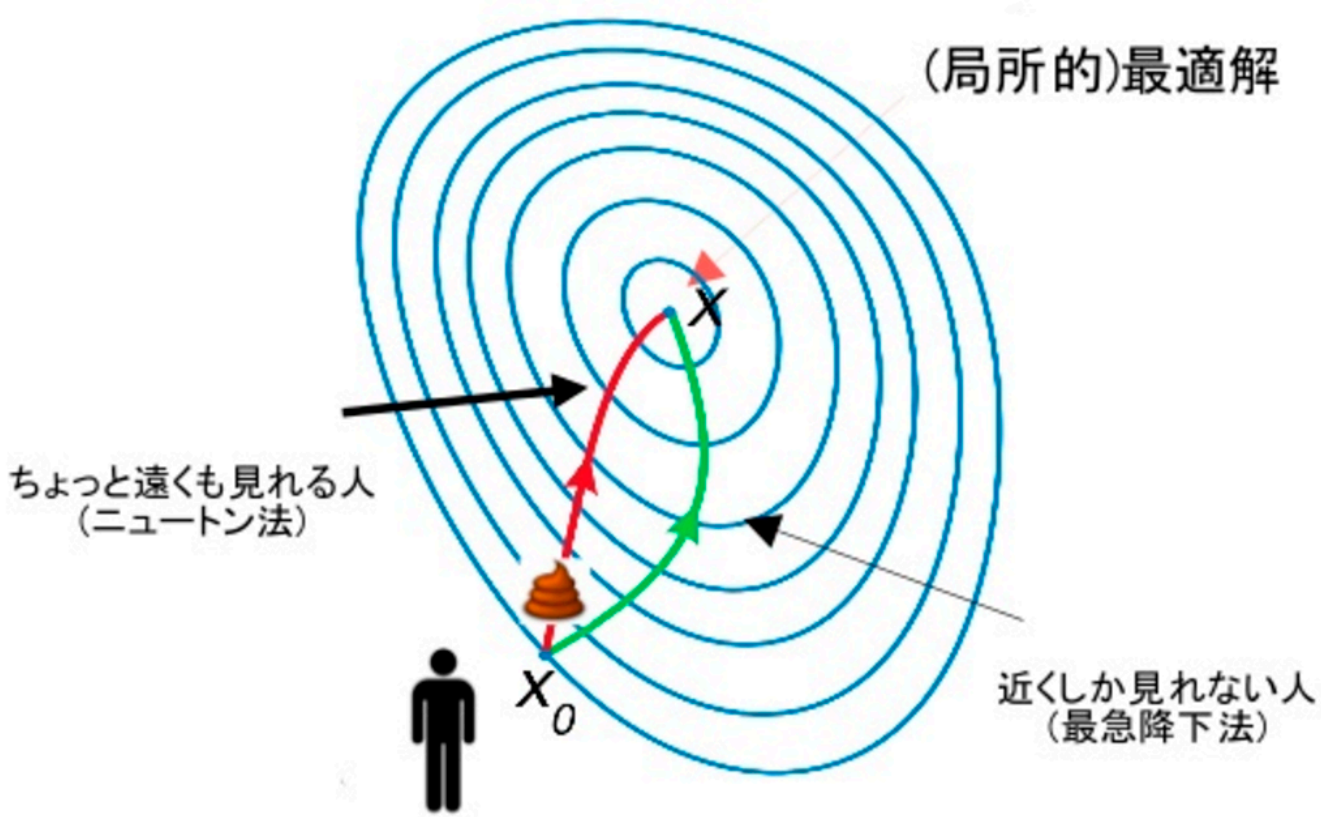
$$\Delta_{\epsilon} \cong \left[\nabla^2 R\left(\hat{\theta}\right) + \epsilon \nabla^2 L\left(z, \hat{\theta}\right)\right]^{-1}\left[\nabla R\left(\hat{\theta}\right) + \epsilon \nabla L\left(z, \hat{\theta}\right)\right]$$

ここで、$\hat{\theta} = argminR(\theta)$なので$\nabla R\left(\hat{\theta}\right) = 0$であること、$\mathrm{O}(\epsilon^2) \to 0$を利用して

$$\Delta_{\epsilon} \cong \nabla^2 R\left(\hat{\theta}\right)^{-1} \nabla L\left(z, \hat{\theta}\right)\epsilon \;\rightarrow\; I_{up,params}(z) \equiv \frac{d\widehat{\theta}_{\epsilon,z}}{d\epsilon}\Big|_{\epsilon=0} = -H_{\widehat{\theta}}^{-1}\, \nabla_{\theta} L\left(z, \hat{\theta}\right)$$

余談：ニュートン法

最急降下法は1次の勾配しか利用してないが、二階微分まで考慮する勾配法



余談：ニュートン法

$f(x)$を最適化したい関数として、二次のTaylor展開で近似すると

$$f(x+\Delta x) \cong f(x) + \Delta x^T \nabla f(x) + \frac{1}{2}\Delta x^T \nabla^2 f(x) \Delta x$$

$$= f(x) + \Delta x^T g + \frac{1}{2}\Delta x^T H\, \Delta x$$

$f(x+\Delta x)$を最小化するΔxを求めないのでΔxで微分すると、

$$\frac{\partial f(x+\Delta x)}{\partial \Delta x} = g + H\Delta x$$

この偏微分が0になるときに$f(x+\Delta x)$は最小値を取るので、$g + H\Delta x = 0$から、

$$\Delta x = H^{-1} g$$

余談：直観的理解

影響関数（再掲）

$$I_{up,params}(z) = -H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z, \hat{\theta}\right)$$

$$I_{up,loss}(z, z_{test}) = -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z, \hat{\theta}\right)$$

要は、

- 上の式は、学習データを入れた時に、ニュートン法によって得られる更新方向
- 下の式は、その更新方向と、testエラーの勾配方向の内積

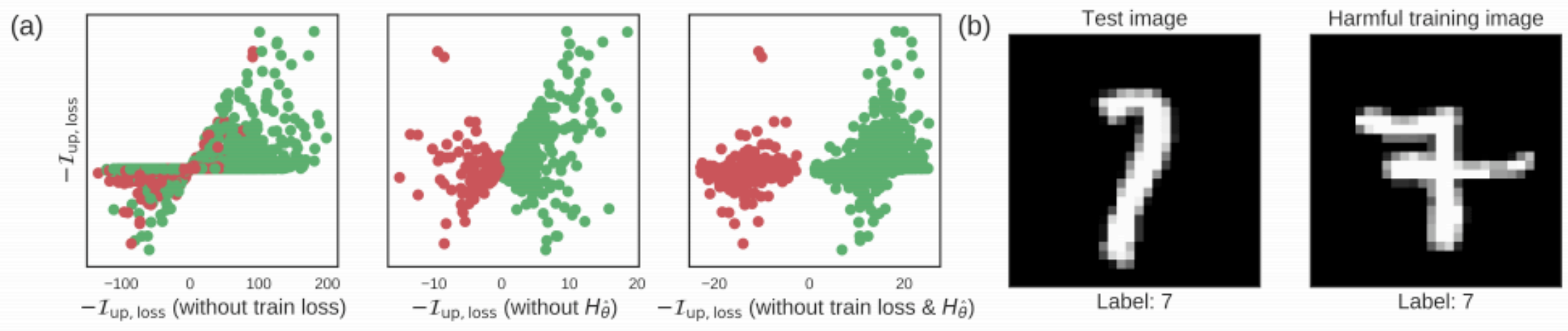
つまり

- それらが同じ向き向いてたら、その学習データはtest errorを直す方向に更新してくれてたことに
- 逆向きだったら、その学習データはむしろtest dataに対する予測を悪くしてたことに

実際どうなのか

$$I_{up,loss}(z, z_{test}) = -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z, \hat{\theta}\right)$$

- 直感的には、test dataと一番似てるtraining dataが影響を与えているっていうだけの可能性もある
- ヘッシアンまで見る必要あるのか



Trainingデータへ加えられたノイズの影響

先程は学習データzを抜いたらどうなるかを考えた

ここでは学習データにノイズが加えられた場合($z_\delta = \mathrm{z} + \delta$)のモデルの出力結果への影響を考える

大体導出は同じなので割愛

$$I_{pert,params}(z) = -H_{\hat{\theta}}^{-1}\left[\nabla_x \nabla_\theta L\left(z, \hat{\theta}\right)\right]\delta$$

$$I_{pert,loss}(z, z_{test}) = -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1}\left[\nabla_x \nabla_\theta L\left(z, \hat{\theta}\right)\right]\delta$$

これで、学習データにノイズを加えたことによるモデルの影響関数がわかった

あとで出て来るが、これを使うとtraining dataに混ぜると、
あるtestデータを間違えさせることができるノイズを計算できる

影響関数の効率的な計算

$$I_{up,loss}(z, z_{test}) = -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z, \hat{\theta}\right)$$

この計算は二つの理由で計算量がとても大きい

1. 損失関数のヘッシアン 逆行列 $H_{\hat{\theta}}^{-1}$の計算 $O(np^2 + p^3)$
2. 全訓練データに対する$I_{pert,loss}(z, z_{test})$の反復計算

普通に考えたらDLモデルは$10^6 \sim 10^8$オーダーなので、10^{20}とか無理

影響関数の効率的な計算

$$I_{up,loss}(z, z_{test}) = -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z, \hat{\theta}\right)$$

Idea

- $s_{test} \equiv H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T$ を予め計算しておけば全訓練データに対する計算は楽になる
(二次形式は順番変えて大丈夫)
- 正定値行列Aの逆行列とあるベクトルの積 $A^{-1}v$は、Avが計算可能ならば共役勾配法によってA^{-1}を陽に計算することなく計算可能
- また、ヘッシアンとベクトルの積は、HVP (Hessian Vector Product)と呼ばれ$O(p)$でヘッシアンを陽に計算することなく計算可能 (CG法）
- 共役勾配法は毎回全サンプルを利用してHvを計算する必要があって若干遅いので、[Agarwal et al]らの近似手法でもs_{test}を計算してみた。CG法よりも高速に動いた。

影響関数の効率的な計算

Algorithm

1. $v \leftarrow \nabla_\theta L(z_{test}, \hat{\theta})$
2. $\widetilde{H}_0^{-1} v \leftarrow v$
3. $\widetilde{H}_i^{-1} v \leftarrow v + (\mathrm{I} - \nabla_\theta^2 L(z_{sampled}, \hat{\theta}))\, \widetilde{H}_{i-1}^{-1} v \;\; (i = 1,2, \ldots, t)$
4. $s_{test} \leftarrow \widetilde{H}_t^{-1} v$

これで、全サンプルの影響関数が$O(np + rtp)$で計算可能

仮定について検証

影響関数の導出に当たって置いた仮定について

1. 損失$R(\theta)$は凸関数 ($\hat{\theta}$はglobal minimum)
   - 実際、$\hat{\theta}$が本当にglobal minimumである保証はどこにもない
     → そうでなかった場合、Hが正定値行列でなくなるので逆行列が計算できない
     (検証によると、CNNを500k iterationほど学習させてもHは正定値にならなかったらしい)

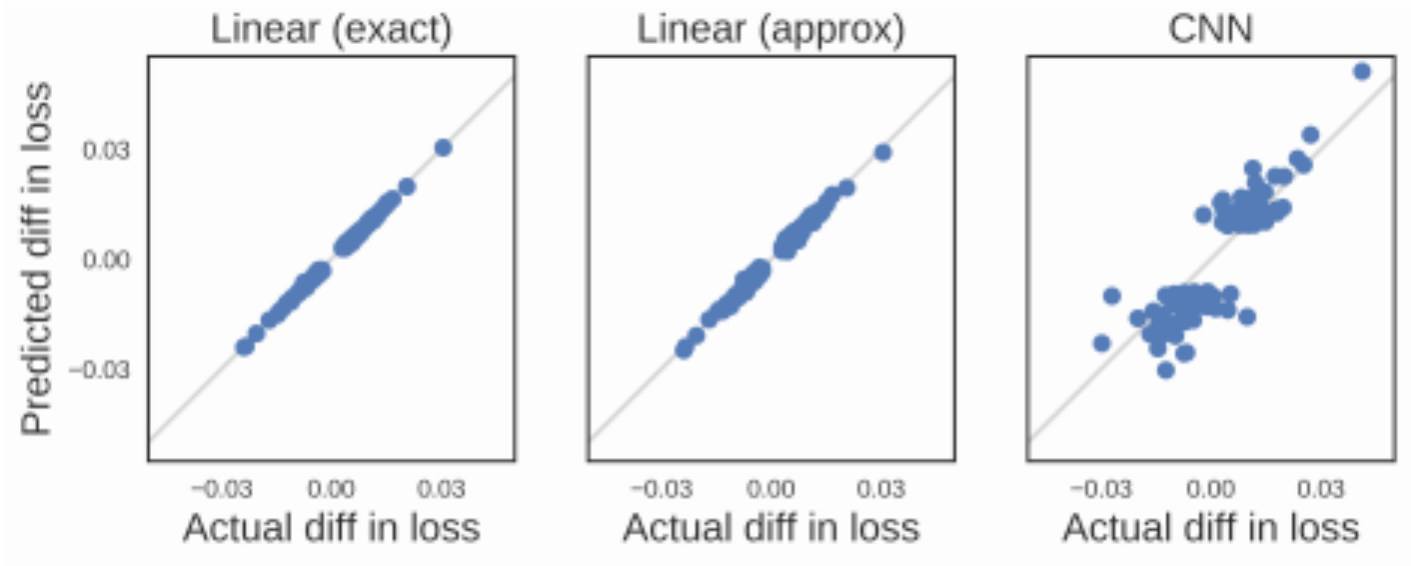
   → だが、無理やりHを正定値にする方法がある $H \leftarrow H + \lambda I$

2. 損失$R(\theta)$は二回微分可能
   - Softmax Cross Entropyなら出来るが、SVMのようにHinge lossを使ってると二階微分できない

   → smoothに近似すれば行けた！！

解析：再学習との比較

一個ずつ抜いて再学習した場合と影響関数を使った場合でどれだけ差がでるか

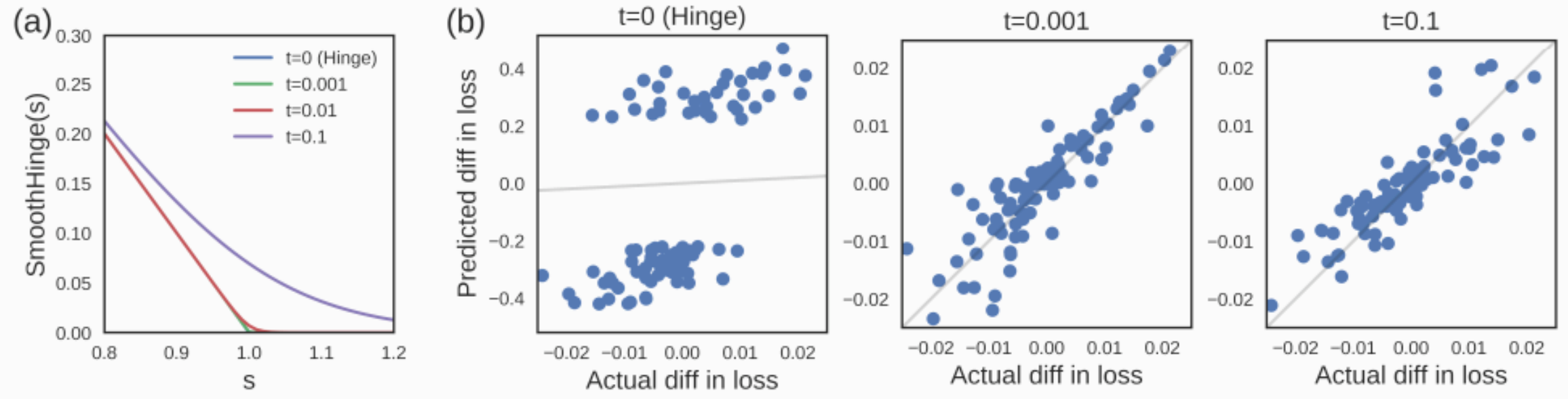


近似してもでかいネットワークでも大体OK！！！

CNNは $H \leftarrow H + \lambda I$ のトリックを利用

解析：微分不可能なロス

- ヒンジで学習したパラメータに対して、smooth hingeで近似して影響関数計算

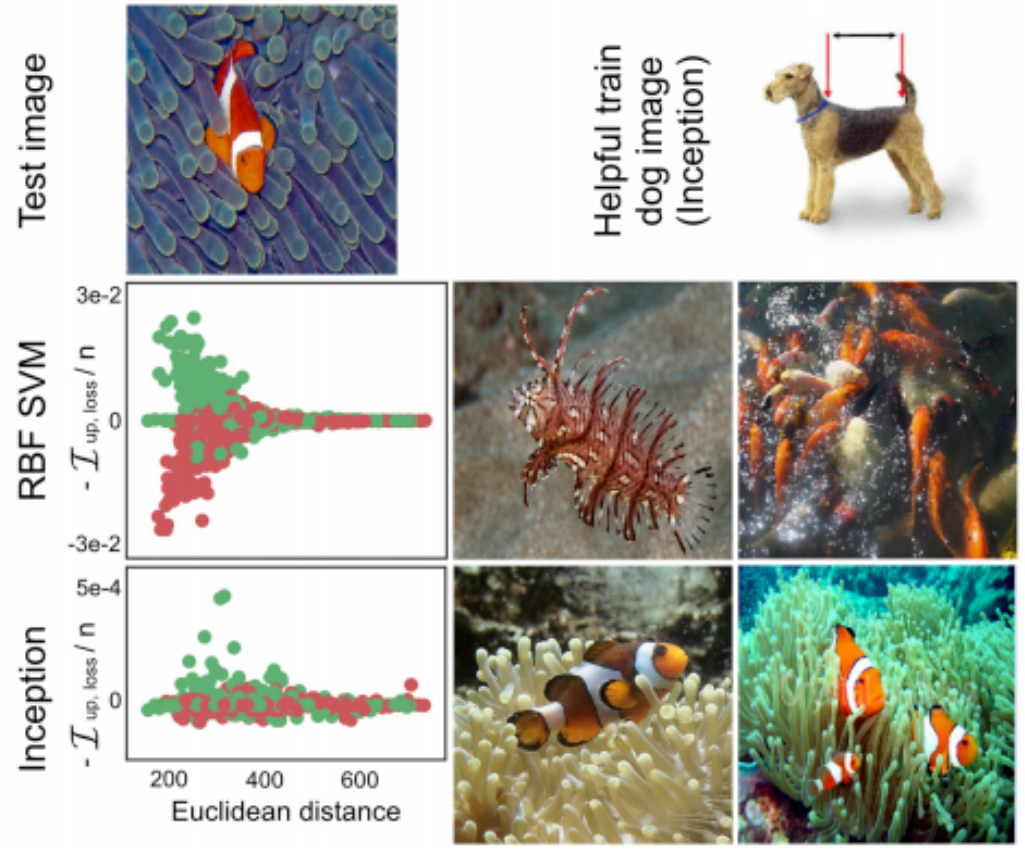


Use case 1: モデルの挙動の理解

- 犬vs魚の二値分類問題をInception-v3、RBF-SVMで学習

- SVMの方はユークリッド距離に大きく影響
- Inceptionはその傾向はあまり見られず、
  より高次な特徴に影響されているとわかる

- Inceptionの方は、犬も魚も両方
  魚を認識するのに有効であったと言うことがわかる



Use case 1: モデルの挙動の理解

- 数字 4 vs 7

Use case 2: Adversarial training samples

- 学習データにノイズを加える影響関数を利用すると、あるtest dataに対して予測を間違えるような、training dataを作成することができる

  -> adversarial training samples

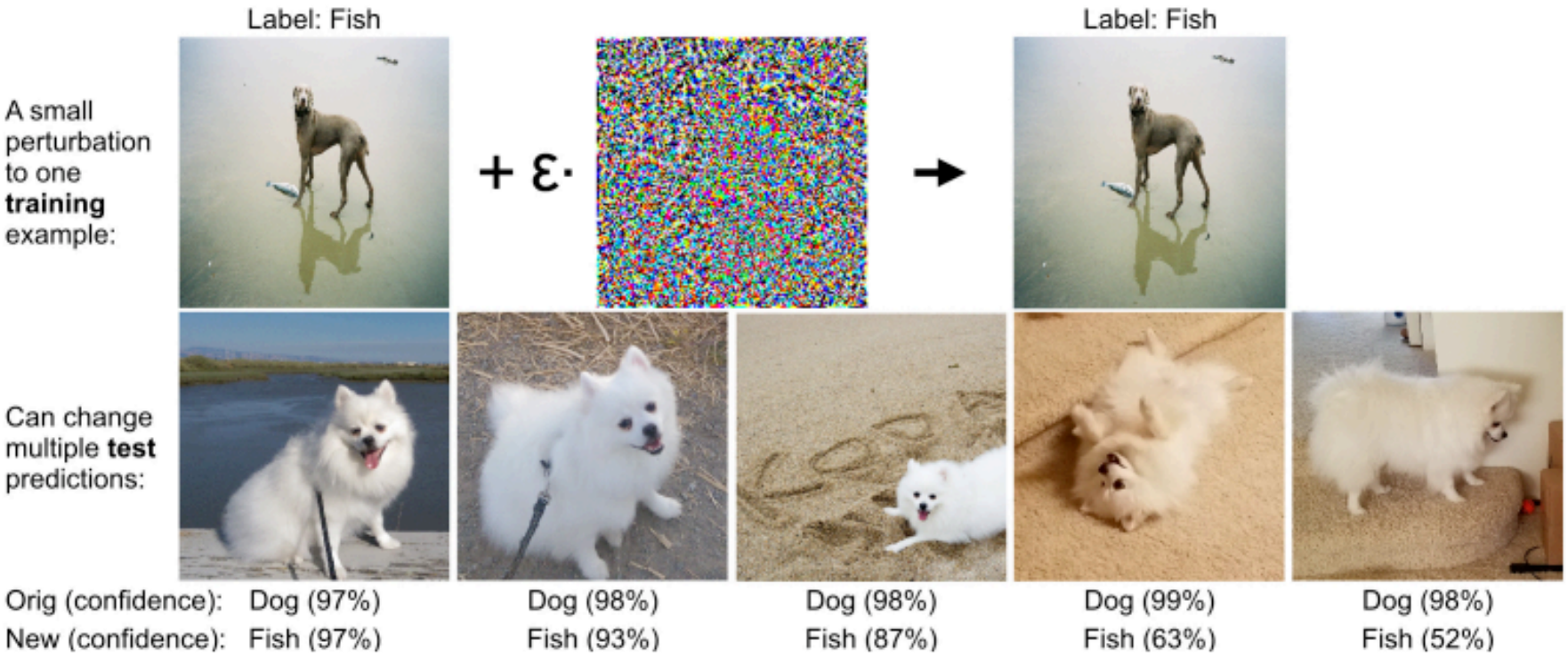
  - (今まであったのは識別を間違えるような test dataだった)

- あるモデルを学習　→ test dataをpick up -> その予測を最も間違えさせるようなノイズを、最も影響が大きいサンプルに対して足す -> 再学習
  を繰り返して、ほとんどtest dataに対して実際に間違えさせることができた

  - ノイズを足すtraining dataを増やせば増やすほど簡単に間違えさせることが

Use case 2: Adversarial training samples

- 著者の犬画像にノイズを混ぜてめっちゃ魚に間違えさせるようなdataを量産して学習させた結果



Use case 2: Adversarial training samples

知見

1. Inputに加えたノイズはとても小さいが、最終層ではオーダーとして100倍のL2距離に
   - すなわち見た目では気づかなくても、feature spaceを調べればわかるかも？

2. ノイズの方向が比較的一貫性がある
   - ある特定の方向にoverfitさせるのが一番簡単なのかも

3. 元々曖昧な画像にノイズを加えるのが最も影響が大きい

Use case 3 : train, testのdomain mismatchの検知

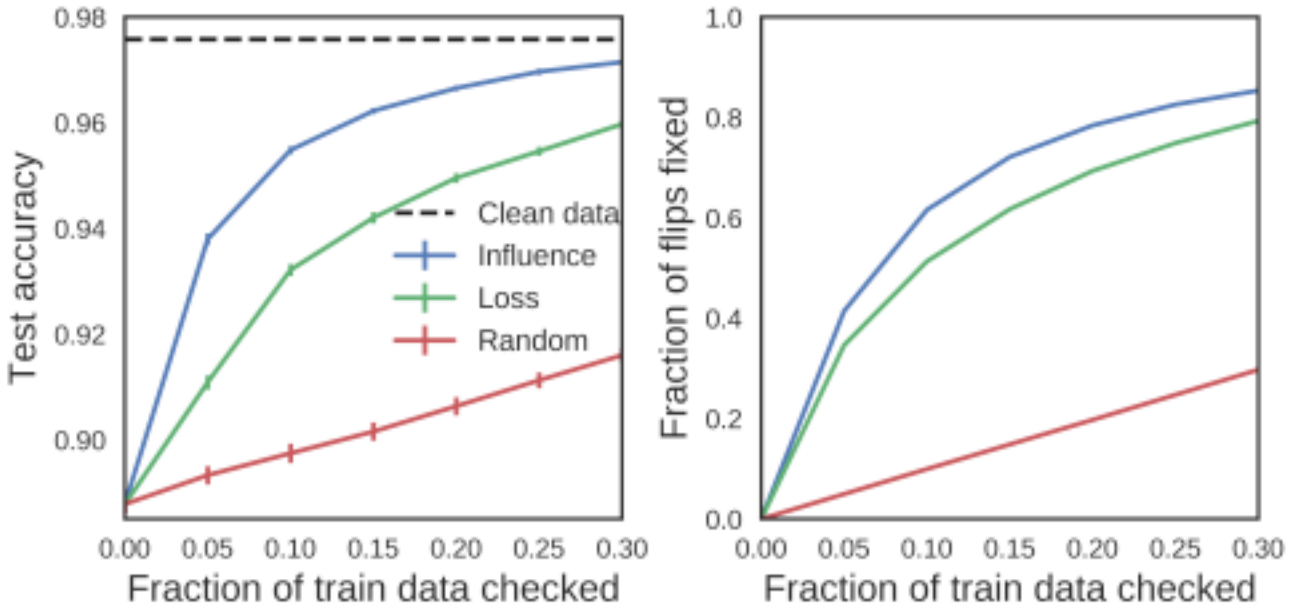
- 127の属性から再入院するかしないかを予測する問題
- 元々のTrain dataには、10歳以下の子供24人いてそのうちの3人が再入院
  → そのうちの再入院しなかった20人をtrain dataから抜く (test と分布が異なる)

- 学習させると、当然train errorとtest errorに大きな乖離
- そこで、間違えたsampleに対して影響関数が大きかったsampleを探すと、その4人の子供がhighlightされていた

→ 原因が特定できたことに

Use case 4 : ラベルノイズの検知

- Datasetには基本的にラベルノイズがある
- 人手で直すのはかなり大変だが、間違えたサンプルに対して影響関数を使って並べ直すと楽に直せた！



まとめ・感想

- Best paperだけあって面白かった
- 長年言われてきたDLは結局black boxだよねという批評に対して現状では最も説得力のありそうな解析
- しかし、現在のparameterの近傍しか考慮してないので、globalに見た時のtraining dataの影響は実際はわからないのがネック
- 説明を省いたが、training dataのノイズに関しては、小さい連続的なものに限って導出してるわけではないので離散ノイズや大きめのノイズでも足すことが出来る