DeepLearninig 勉強会

7 章後半

B4 T.Ochiai

June 17, 2019

Nagoya Institute of Technology
Takeuchi & Karasuyama Lab

目次

Next Section

パラメータ拘束

- これまでパラメータに制約を加える時は固定された領域に関して見てきた
  - 例) L^2 正則化の場合は重みパラメータの L2 ノルムに対して制約を課す
- 場合によってはパラメータの事前知識を制約に課すことが必要になる
  - パラメータの間に相関がある場合など

パラメータ同士が互いに近い関係を持つ場合

- パラメータ $\boldsymbol{w}^{(A)}$ を持つモデル A と $\boldsymbol{w}^{(B)}$ を持つモデル B を考える
- タスクが十分類似していて $\forall i$ で $w_i^{(A)}, w_i^{(B)}$ が近いと想定できるとする
- このような場合正則化には以下のような制約を組み込むことができる

$$\Omega\left(\boldsymbol{w}^{(A)}, \boldsymbol{w}^{(B)}\right) = \left\|\boldsymbol{w}^{(A)} - \boldsymbol{w}^{(B)}\right\|_2^2 \tag{1}$$

パラメータ共有

- 一部のパラメータを共有すること
- パラメータ共有の利点
  - パラメータの固有の集合だけをメモリに保存すれば良い
  - CNN などではこれによって大幅にメモリ使用料の削減が可能なケースがある

畳み込みニューラルネットワーク

- 畳み込みニューラルネットワーク (CNN) ではパラメータ共有が用いられている
- 画像は変換の前後で不変な統計的性質を多く保有している
  - 猫の写真は 1 ピクセル右に移動させても猫の写真のまま
- CNN では画像の中の複数の位置にわたってパラメータを共有することでこの性質を取り込む

Next Section

スパース表現

- ペナルティを課す方法
  - モデルパラメータに直接ペナルティを課す (重み減衰)
  - ユニットの活性化にペナルティを課し, スパースにする
- ノルムによる正則化は損失関数 J に正則化項 $\Omega(\boldsymbol{h})$ を足して表される

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{h}) \tag{2}$$

- $\alpha \in [0, \infty]$ は正則化項の寄与を重み付けしている
  - 大きくなるほどより正則化される
- パラメータの L_1 正則化によってスパース性が誘発される
  - スパース性をもたらすのは決して L_1 ノルムだけではない
- スパース性をもたらすその他の手法
  - スチューデントの t 事前分布から導かれたペナルティ
    - $\Omega(\boldsymbol{h}) = \alpha \sum_i \log(1 + \gamma h_i^2)$
  - KL ダイバージェンスペナルティ

直交マッチング追跡

- 活性化値に制約をもつ表現のスパース性が得られる手法
- 入力 $\boldsymbol{x}$ を以下の制約最適化問題を解く表現 $\boldsymbol{h}$ で符号化する

$$\underset{\boldsymbol{h}, \|\boldsymbol{h}\|_0 < k}{\arg\min} \|\boldsymbol{x} - \boldsymbol{W}\boldsymbol{h}\|^2 \tag{3}$$

- $\boldsymbol{x}$ に対するスパースな表現 $\boldsymbol{h}$ が得られる
- この手法は OMP-k とも呼ばれる
  - k は非ゼロ特徴量の数を指定するパラメータ

線形回帰でのスパース表現

$$\underbrace{\begin{bmatrix} 18 \\ 5 \\ 15 \\ -9 \\ -3 \end{bmatrix}}_{\boldsymbol{y} \in \mathbb{R}^m} = \underbrace{\begin{bmatrix} 4 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & -1 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & -4 \\ 1 & 0 & 0 & 0 & -5 & 0 \end{bmatrix}}_{\boldsymbol{A} \in \mathbb{R}^{m \times n}} \underbrace{\begin{bmatrix} 2 \\ 3 \\ -2 \\ -5 \\ 1 \\ 4 \end{bmatrix}}_{\boldsymbol{x} \in \mathbb{R}^n}$$

$$\underbrace{\begin{bmatrix} -14 \\ 1 \\ 19 \\ 2 \\ 23 \end{bmatrix}}_{\boldsymbol{y} \in \mathbb{R}^m} = \underbrace{\begin{bmatrix} 3 & -1 & 2 & -5 & 4 & 1 \\ 4 & 2 & -3 & -1 & 1 & 3 \\ -1 & 5 & 4 & 2 & -3 & -2 \\ 3 & 1 & 2 & -3 & 0 & -3 \\ -5 & 4 & -2 & 2 & -5 & -1 \end{bmatrix}}_{\boldsymbol{B} \in \mathbb{R}^{m \times n}} \underbrace{\begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \\ -3 \\ 0 \end{bmatrix}}_{\boldsymbol{h} \in \mathbb{R}^n}$$

Figure 1: 線形回帰でのスパース表現．上の表現はスパースにパラメータ化された例．下の表現はデータ $\boldsymbol{x}$ のスパース表現 $\boldsymbol{h}$ をもつ線形回帰の例．

Next Section

バギング

- ブートストラップサンプリングを繰り返して生成した判別器を合成して，より判別精度の高い判別器を生成する方法
  - ブートストラップサンプリング: 重複を許してサンプリングして新たなサンプル集合を作ること (平均的には元のデータ集合と 2/3 が共通する)
- 複数モデルで別々に訓練させてテスト事例に対する出力を投票させる
  - モデル平均化と呼ばれる手法の一種
  - この手法を用いた方法はアンサンブル手法と呼ばれる
- 通常はモデルが異なれば同じテスト事例でも全てが同じ間違いをすることはない

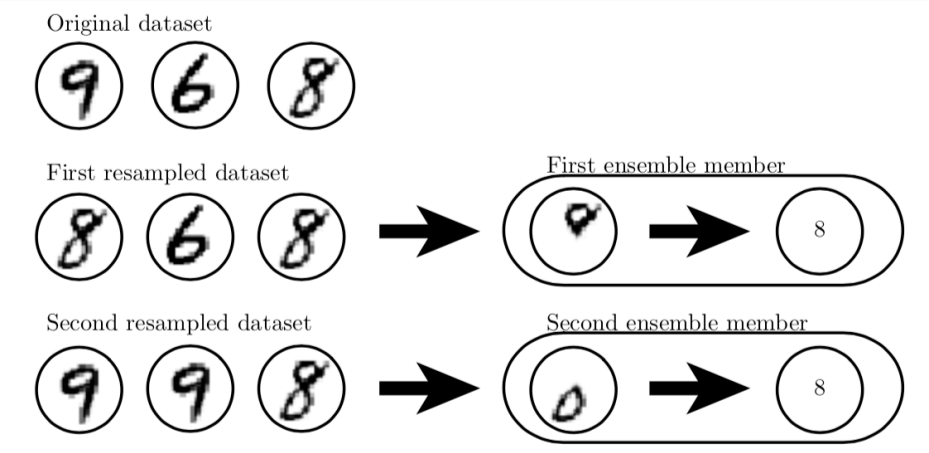

Figure 2: バギングの動作．8, 6, 9 を含むデータ集合から 8 を見つける検出器を訓練する．上のデータ集合からは 8 の上側の輪を学習し，下のデータ集合では 8 の下の輪を学習する

アンサンブル学習の例

k 個の回帰モデルからなる例

- 各モデルが各事例に対して誤差 ϵ_i を出力する
- 誤差は平均 0 の多変量正規分布から得られる
  - 分散 $\mathbb{E}[\epsilon_i^2] = v$, 共分散 $\mathbb{E}[\epsilon_i \epsilon_j] = c$
- 全てのアンサンブルモデルの予測平均で得られる誤差は $\frac{1}{k}\sum_i \epsilon_i$
- アンサンブル予測器の平均二乗誤差は以下のようになる

$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i \left(\epsilon_i^2 + \sum_{j\neq i} \epsilon_i \epsilon_j\right)\right] = \frac{1}{k}v + \frac{k-1}{k}c \qquad (4)$$

- 誤差が完全に相関していて $c = v$ の場合
  $\rightarrow$ 平均二乗誤差は v となり , モデル平均化は役に立たない
- 誤差に相関がなく $c = 0$ の場合
  $\rightarrow$ 平均二乗誤差は $\frac{1}{k}v$ だけになる

ニューラルネットワークにおけるアンサンブル学習

- ニューラルネットワークでは全てのデータ集合で訓練されているとしても十分に幅広い多様な解に到達する
- モデル平均化は汎化誤差を減少させるには非常に優れた手段
  - アルゴリズムのベンチマークにはモデル平均化は推奨されない
  - 計算量の増加量と引き換えにモデル平均化によって大きな利益が得られるため
- 機械学習コンテストでは多数のモデルに対してモデル平均化を行うのが当たり前になってきている
- アンサンブルでは必ずしも汎化誤差を減少させる手段ではない
  - ブースティングはやりすぎると過学習する

Next Section

ドロップアウト

- 幅広いモデル族を正則化する計算量の小さい手法
- バギングを多くのニューラルネットワークに対して実用的にする
  - バギングは大規模なニューラルネットにおいてはコストがかかりすぎる
- 指数的に多くのニューラルネットワークを集めたアンサンブルの評価が可能
- 出力層ではないユニットを削除することで部分ネットワークからなるアンサンブルモデルを学習
  - ユニットの出力値に 0 をかけることで実質的にネットワークからユニットを削除

ドロップアウトの方法

- ドロップアウトの訓練のためにミニバッチ的なアルゴリズムを導入

ドロップアウトのアルゴリズム

1. ミニバッチに事例を導入する
2. 全ての入力と隠れ層に適応する 2 値マスク μ を無作為にサンプリング
   - マスクに 1 が選ばれる確率はハイパーパラメータ
3. 2 で得られたマスクを用いてネットワークを学習
4. 損失 $J(\theta, \mu)$ を計算
5. 1 から 4 を繰り返す
6. $\mathbb{E}_{\mu} J(\theta, \mu)$ を最小化する

ドロップアウトの例

- $\mu_\star$ はドロップアウトの 2 値マスクを表す
- 無作為に μ を抽出し，入力，隠れユニットにベクトルの要素を 1 つ対応させる.
- 各ユニットに対応するマスクをかけ，その後は通常通りに順伝播を行う

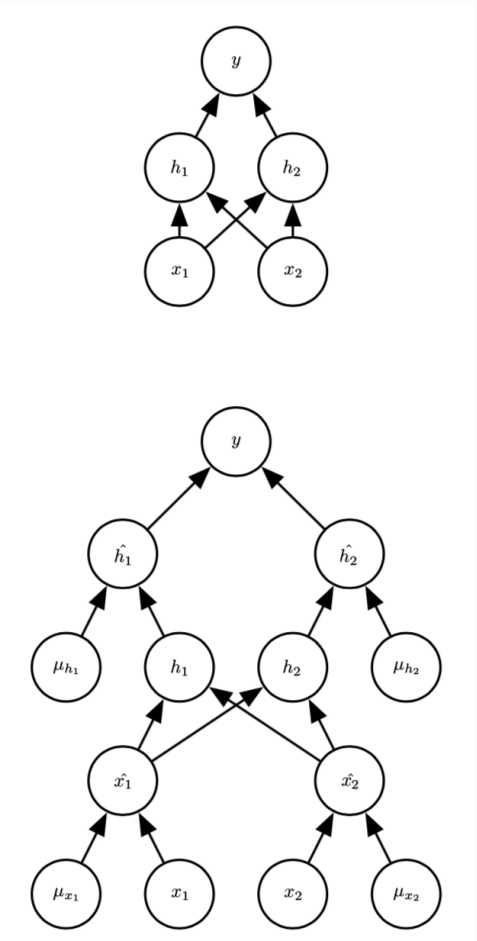

Figure 3: ドロップアウトの例．上は通常の順伝播，下は上のネットワークにドロップアウトを施したもの.

ドロップアウトとバギングの違い

- ドロップアウトとバギングの学習は同じではない
- モデルの独立性
  - バギングは全てのモデルが独立
  - ドロップアウトはパラメータを共有する
- モデルの訓練方法
  - バギングはそれぞれの訓練集合で学習する
  - ドロップアウトは明示的に訓練されることはほとんどない
    - 可能性のある部分ネットワーク全てをサンプリングしようとすると指数爆発
- ドロップアウトでは部分ネットワークの小さな部分が 1 ステップで訓練される
- パラメータ共有によって残りの部分ネットワークのパラメータが良い設定となる

推論

- アンサンブルでは構成する全モデルの出力を統合する
  - この処理のことを推論と呼ぶことにする
- モデルのタスクが確率分布を出力することだと仮定する
- バギングでは各モデル i は確率分布 $p(y|\boldsymbol{x})$ を出力する
  $\rightarrow$ アンサンブルの予測は $\frac{1}{k}\sum_{i=1}^{k} p^{(i)}(y|\boldsymbol{x})$
- ドロップアウトではマスクベクトル $\boldsymbol{\mu}$ で定義されるモデルは確率分布 $p(y|\boldsymbol{x}, \boldsymbol{\mu})$ を出力する
  $\rightarrow$ アンサンブルの予測は $\sum_{\mu} p(\boldsymbol{\mu})p(y|\boldsymbol{x}, \boldsymbol{\mu})$

ドロップアウトにおける推論

- ドロップアウトの予測の総和の中には指数的な数の項が含まれる
  - モデルの構造が単純でないと評価するのが難しい
- 今の所は深いニューラルネットワークで扱いやすくするための方法はわかっていない
- 代わりに多数のマスクを用いたモデルの出力を平均化して推論を近似できる
  - 10 から 20 のマスクがあれば良い性能を得るのには十分
- さらに良いアプローチとしてたった 1 回の順伝播でアンサンブル全体の予測を行う方法 (重みスケーリング規則) がある
  - 算術平均ではなく幾何平均を用いる方法
  - 詳細はテキストで

ドロップアウトの利点

- 計算量が非常に小さい
  - 各ユニットで n 個の二値の数字を作り出し各状態と掛け合わせる
  - 訓練中にドロップアウトを行う場合は 1 個の事例あたり $O(n)$
- 使えるモデルの種類に重大な制限がない
  - 他の正則化手法ではモデルに厳しい制約を課すものが多い

ドロップアウトの欠点

- 完全なシステムとしてドロップアウトを使うのはコストが大きくなる可能性がある
- 正則化の手法のためモデルの表現力を削減してしまう
  - 解決するにはモデルのサイズを大きくしなくてはいけない
- 検証誤差はドロップアウトによって低くなるが, それには以下ことが必要になる
  - モデルサイズを非常に大きくする
  - 訓練アルゴリズムの反復を大幅に増やす
- 非常に大きなデータ集合では汎化誤差の減少が小さい

高速ドロップアウト

- ドロップアウトにおいて全ての部分モデルに対する総和を近似する手法
- 幾何平均ではなく算術平均に則っている
- 勾配の計算における確率性を削減することで収束までの時間を短くする
- 小規模なニューラルネットワークでは標準的なドロップアウトと同様の性能
- 大規模なものに適応できるほどにはまだ改善はされていない

Next Section

敵対的事例

- ニューラルネットワークの性能は人間と同じ程度に到達することが多い
  $\rightarrow$ 本当に人間と同じレベルでタスクを理解しているのか
- ネットワークの理解レベルを調べるためにモデルが誤分類した例を考えることができる
- x と x' が近い場合人間はその敵対的事例を判別できないがネットワークでは全く異なる予測が可能

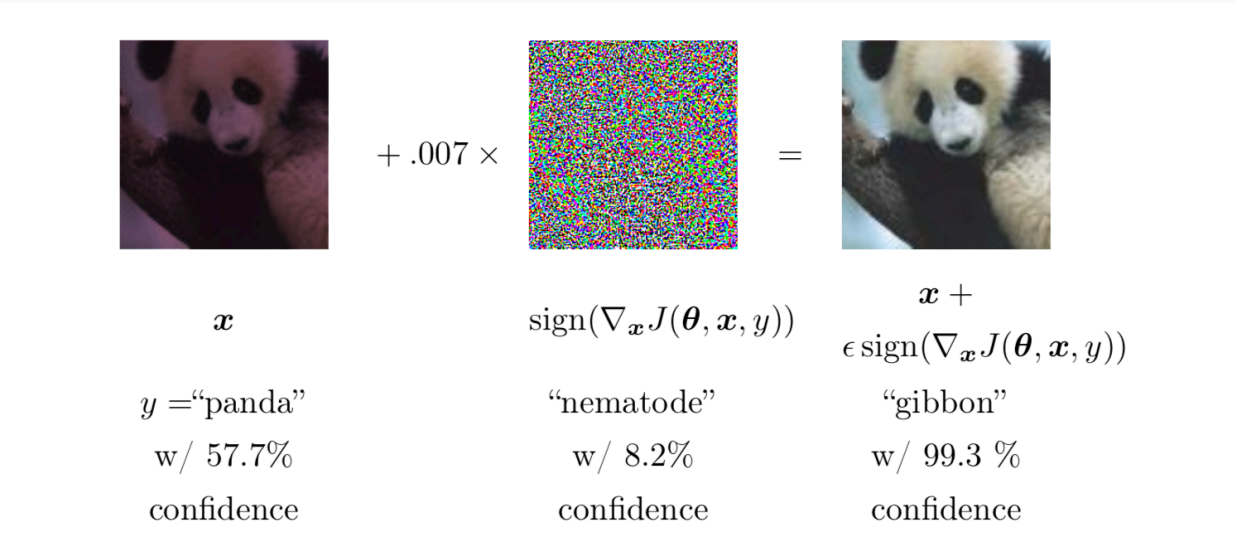

Figure 4: 敵対的事例の説明の図．知覚できないような小さなベクトルを画像に追加することで画像分類結果を変えることができる

敵対的学習

- 訓練集合に敵対的な加工をした事例の学習を敵対的学習と言う
- 敵対的学習のよって元のテスト集合での誤り率を削減できる
- 敵対的事例の原因は過度な線形性である
  - ニューラルネットワークは主に線形性に関連した要素で構成される
  - 入力の数が大きいと線形関数の値は急激に変化する可能性がある
- 敵対的学習ではノイズを考慮することでネットワークを安定化する
- 敵対的事例は半教師あり学習にも適応できる
  - ラベルが付与されていない点 $\boldsymbol{x}$ においてモデルはラベル $\hat{y}$ を割り当てる
  - モデルは $y' \neq \hat{y}$ を出力させる敵対的事例 $\boldsymbol{x}'$ を探すことができる
  - その後モデルは $\boldsymbol{x}$ と $\boldsymbol{x}'$ で同じラベルを割り当てるように学習する
- 真のラベルではなく訓練モデルから提供されたラベルを用いて生成される敵対的事例は仮想敵対的事例と呼ぶ

Next Section

接距離アルゴリズム

- 機械学習アルゴリズムの多くはデータが低次元多様体の近傍にあると仮定することで次元の呪いを克服しようとする
- 多様体仮説を活用した方法の一つに接距離アルゴリズムがある
  - ノンパラメトリックの最近傍アルゴリズム
  - ユークリッド距離ではなく近傍で確率が集中している多様体の知識から得られるもの
  - 同じ多様体上の事例は同じカテゴリを共有していると仮定
- 点 x_1 と x_2 の最近傍距離としてそれぞれが属する多様体 M_1 と M_2 の距離を使うのが妥当
  $\rightarrow$ 計算上困難だが，妥当な方法として M_i を x_i での接平面で近似し，2 つの接平面の距離を測る方法がある
  - これは低次元線形系で解くことが可能

接線伝播法

- ニューラルネットワークの各出力 $f(\boldsymbol{x})$ を既知の変動要因に対して局所的に不変にするペナルティを加える
  - 変動要因は同じクラスの事例が集中している点の近傍の多様体に沿った動きに対応している
- 局所不変性を実現するための方法
  - $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ が $\boldsymbol{x}$ における既知の多様体の接ベクトルに対して直交する
  - 等価的な正則化ペナルティ $\Omega(f) = \sum_i \left(\left(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \right)^\top \boldsymbol{v}^{(i)} \right)^2$ を追加する
- データ拡張などに関連している
  - 特定の変換 (画像でいうと回転や平行移動) に対してモデルが不変となる
- 多様体の接ベクトルを知っている必要がある

多様体接分類器

- 事前に接ベクトルを知る必要がない
- 自己符号化器では多様体接ベクトルを推定できる

多様体接分類器のアルゴリズム

1. 自己符号化器を使って教師なし学習で多様体の構造を学習する
2. 1 で学習した接ベクトルを用いて接線伝播法で正則化する

- 詳細は 14 章

接線伝播法と多様体接分類器

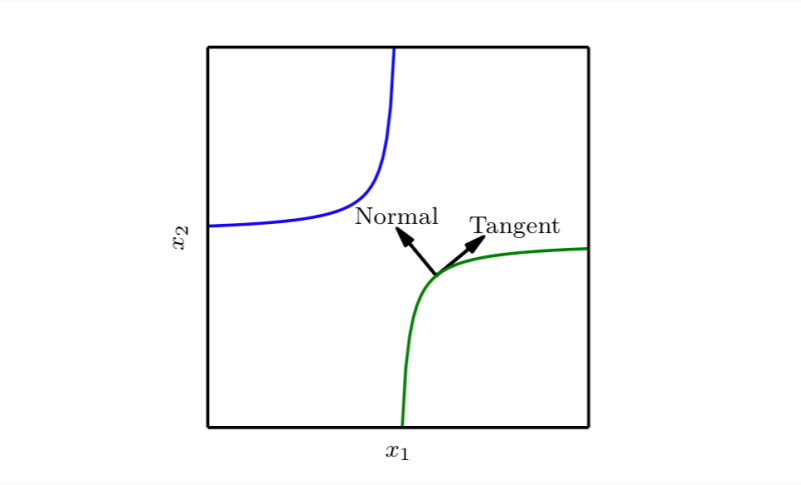

Figure 5: 接線伝播法と多様体接分類器の主要な考えを示す図．各曲線は異なるクラスの多様体を表し，ここでは 2 次元平面に埋め込まれた 1 次元多様体を図示する．

- 分類関数は多様体に対して垂直に動くときは急激に変化
  - クラス多様体に沿って動くときは変化しない
- 接線伝播法と多様体接分類器は多様体に沿って x が動いてもあまり変化しないように $f(x)$ を正則化する