PythonとKerasによるディープラーニング

第4章　機械学習の基礎

4.3 データ前処理、特徴エンジニアリング、表現学習

16T4063F 結城洸太

4.3.1 ニューラルネットワークでのデータ前処理

生のデータをニューラルネットワークに適したものにする。

- ベクトル化

  画像、音声、テキストなどのデータを、浮動小数点のテンソルに変換する。

  (例)整数のリストとしてあらわされたテキストを、one-hotエンコーディングを使って、float32型のテンソルに変換する。

4.3.1 ニューラルネットワークでのデータ前処理

- 値の正規化

  ネットワークの学習を容易にするため、データを変換する。

  - 小さな値をとる(一般に、0～1)
  - 種類が同じである(ほぼ同じ値をとるようにする)
  - 平均が0になるように各特徴量を個別に正規化する。
  - 標準偏差が１になるように各特徴量を個別に正規化する。

  (例)0~255の整数データ

  →float型にキャストし、255で割ることで、0～1の浮動小数型にする。

4.3.1 ニューラルネットワークでのデータ前処理

- 欠損値の処理

  データによっては欠損値が含まれていることがある。

  一般に、入力データの欠損値を0にする。(0が意味のある値としてすでに使用されていない前提)

  →ネットワークは0が欠損値であることを学習し、無視するようになる。

  欠損値が含まれないデータは、人工的に欠損値を作成し、「欠損値を無視すること」を学習させるべき。

4.3.2 特徴エンジニアリング

- 特徴エンジニアリング

  使用しているデータと機械学習アルゴリズムに基づいて、そのアルゴリズムの性能を向上させるプロセス。

  データをモデルに供給する前に、ハードコーディングされた変換を適応する。

4.3.2 特徴エンジニアリング

(例)時計の画像から時刻を返すモデル

入力データとして画像のピクセル
を用いる場合、難しい機械学習を
取り組むことになる。

畳み込みニューラルネットワーク
と膨大な計算リソースが必要。

4.3.2 特徴エンジニアリング

(例)時計の画像から時刻を返すモデル

機械学習アルゴリズムに合わせて
もっと入力特徴量を考え出す。

時計の針を表す黒いピクセルを追
跡。針の位置を(x,y)座標で出力。

単純な機械アルゴリズムで、座標
と該当する時刻の関係を学習する。

生のデータ: ピクセルグリッド		
より改善された 特徴量: 時計の針の座標	{x1: 0.7, y1: 0.7} {x2: 0.5, y2: 0.0}	{x1: 0.0, y2: 1.0} {x2: –0.38, y2: 0.32}

4.3.2 特徴エンジニアリング

(例)時計の画像から時刻を返すモデル

さらに、座標返還を行う。

画像の中心を基準に極座標を表現。
各時計の針の角度θを出力。

機械学習が必要ないほど簡単な問題に。

生のデータ: ピクセルグリッド		
より改善された 特徴量: 時計の針の座標	{x1: 0.7, y1: 0.7} {x2: 0.5, y2: 0.0}	{x1: 0.0, y2: 1.0} {x2: –0.38, y2: 0.32}
さらに改善された 特徴量: 時計の針の角度	theta1: 45 theta2: 0	theta1: 90 theta2: 140

4.3.2 特徴エンジニアリング

特徴量をより単純な方法で表現することで、問題を容易に。
→問題を深く理解していることが要求される。

特徴エンジニアリングは、ディープラーニング登場前は重要だったが、ニューラルネットワークは、生のデータから有益な特徴を自動で抽出できる。
→ほとんどの特徴エンジニアリングは必要なくなっている。
しかし、考えなくてよいということではない。

4.3.2 特徴エンジニアリング

ディープラーニングにおいて特徴エンジニアリングを考える理由

- よい特徴量があると、リソースの消費を抑え、問題をより的確に解決できる。(時計の文字盤を読み取る問題を解決するのに畳み込みニューラルネットワークを使うのははばかげている)

- よい特徴量があると、問題をより少ないデータで解決できる。(ディープラーニングのモデルが特徴量から学習できるかどうかは、データの量にかかっている。サンプルが少ししかない場合、特徴量の情報的価値はきわめて高くなる。)