【高精度な画像分類器作りに
精度100%

tmtk

Topics

こんにちは。データサイエンスチーム tmtkです。
この記事では、桜とコスモスの写真を分類する機械学習モデルを、**転移学習を用いて作成**します。

はじめに

前回の記事では、VGG16をファインチューニングすることで、桜とコスモスの写真を分類する機械学習モデルを作成しました。
今回は、学習済みのVGG16を特徴量抽出に使い、SVMと組み合わせることで、桜とコスモスの写真を分類する機械学習モデルを作成します。このような方法を、**転移学習**といいます。

実験

前々回の記事と同じ準備の下で実験をします。
IPythonを起動します。

```
ipython3
```

訓練データを読み込みます。

```
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from sklearn.model_selection import train_test_split
import numpy as np
import os
input_shape = (224, 224, 3)
batch_size = 128
epochs = 12
num_classes = 2
x = []
y = []
for f in os.listdir("sakura"):
    x.append(image.img_to_array(image.load_img("sakura/"+f, target_size=input_shape[:2])))
    y.append(0)
for f in os.listdir("cosmos"):
    x.append(image.img_to_array(image.load_img("cosmos/"+f, target_size=input_shape[:2])))
    y.append(1)
x = np.asarray(x)
x = preprocess_input(x)
y = np.asarray(y)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=3)
```

VGG16のモデル・パラメータを読み込みます。畳み込み層、プーリング層の出力を特徴量として使用するために、末尾の全結合層は除き、最後に平均値でプーリングします。

```
from keras.applications.vgg16 import VGG16
base_model = VGG16(weights='imagenet', include_top=False, pooling="avg")
```

訓練データをVGG16に処理させたものを、新しい特徴量とします。

```
x_train_vgg16 = base_model.predict(x_train)
x_test_vgg16 = base_model.predict(x_test)
```

最後に、線形SVMに訓練データを学習させます。

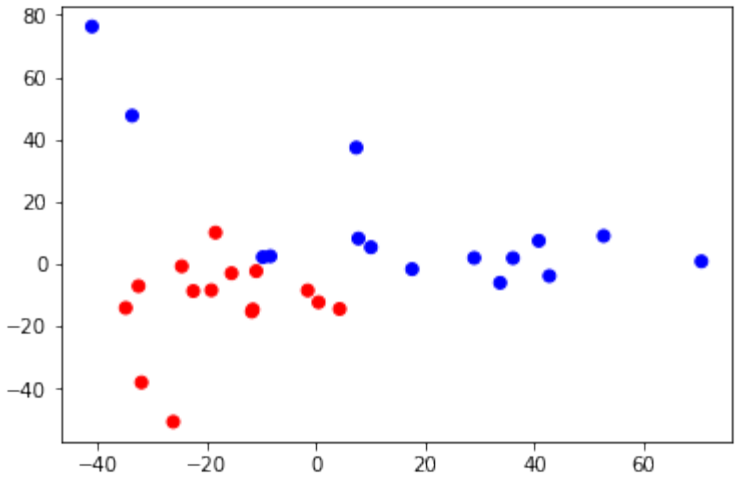
```
from sklearn.svm import SVC
clf = SVC(kernel="linear").fit(x_train_vgg16, y_train)
```

精度を確かめると、

```
from sklearn.metrics import accuracy_score
print("acc = {}".format(accuracy_score(clf.predict(x_train_vgg16), y_train)))
print("val_acc = {}".format(accuracy_score(clf.predict(x_test_vgg16), y_test)))
```

```
acc = 1.0
val_acc = 1.0
```

となり、**訓練データ、バリデーションデータともに、100%の精度を達成**できています。これで、Microsoft AzureのCustom Vision Serviceの性能に追いつくことができました。



（VGG16で変換した特徴量を主成分分析にかけ、可視化したもの。桜とコスモスを見分けるための特徴量が抽出できていることが推察できる。）

ファインチューニングのコツについて

スタンフォード大学の講義資料 によると、十分な訓練データを用意することが難しいことが多いため、畳み込みニューラルネットワークを**ゼロから学習させることは少ない**そうです。かわりに用いられるのがこれまで紹介してきた**ファインチューニング**や**転移学習**といった技法ですが、それにもいくつかの**ノウハウ**があるようです。それを以下で引用・翻訳し紹介します。

When and how to fine-tune? How do you decide what type of transfer learning you should perform on a new dataset? This is a function of several factors, but the two most important ones are the size of the new dataset (small or big), and its similarity to the original dataset (e.g. ImageNet-like in terms of the content of images and the classes, or very different, such as microscope images). Keeping in mind that ConvNet features are more generic in early layers and more original-dataset-specific in later layers, here are some common rules of thumb for navigating the 4 major scenarios:

1. *New dataset is small and similar to original dataset.* Since the data is small, it is not a good idea to fine-tune the ConvNet due to overfitting concerns. Since the data is similar to the original data, we expect higher-level features in the ConvNet to be relevant to this dataset as well. Hence, the best idea might be to train a linear classifier on the CNN codes.
2. *New dataset is large and similar to the original dataset.* Since we have more data, we can have more confidence that we won't overfit if we were to try to fine-tune through the full network.
3. *New dataset is small but very different from the original dataset.* Since the data is small, it is likely best to only train a linear classifier. Since the dataset is very different, it might not be best to train the classifier form the top of the network, which contains more dataset-specific features. Instead, it might work better to train the SVM classifier from activations somewhere earlier in the network.
4. *New dataset is large and very different from the original dataset.* Since the dataset is very large, we may expect that we can afford to train a ConvNet from scratch. However, in practice it is very often still beneficial to initialize with weights from a pretrained model. In this case, we would have enough data and confidence to fine-tune through the entire network.

いつどのようにファインチューニングするか？新しいデータセットに対してどんなタイプの転移学習をするか、どのように決めるか？ これはいくつかの因子からの関数ですが、新しいデータセットの大きさ（小さいか大きいか）と、新しいデータセットと元のデータセットの類似度（たとえば、内容と種類がImageNetに似ている画像か、あるいは顕微鏡写真のようであるか）が二つのもっとも重要な因子です。畳み込みニューラルネットワークでは入力側のレイヤーほど一般的な特徴量を出力し出力側のレイヤーほど元のデータセットに特有の特徴量を出力することを考えると、4つの主要な筋書きに対する一般的な指針は以下のようになります。

1. 新しいデータセットが小さく、元のデータセットに似ている場合 データが小さいので、畳み込みニューラルネットワークをファインチューニングするのは過学習が懸念され、よくありません。データが元のデータに似ているので、畳み込みニューラルネットワークの高次の特徴量が新しいデータセットとの間にも関係していると期待できます。したがって、CNN符号（訳注：畳み込みニューラルネットワークで、最後の層の直前の層から出力される特徴量のこと）を使って線形分類器を訓練するのが最善でしょう。
2. 新しいデータセットが大きく、元のデータセットに似ている場合 より多いデータがあるので、ネットワーク全体をファインチューニングしようとしても過学習しない可能性が高いといえます。
3. 新しいデータセットが小さいが、元のデータセットとぜんぜん違う場合 データが小さいので、線形分類器を訓練するだけが一番いいでしょう。データセットがぜんぜん違うので、特徴量がデータセット特有のものになっているネットワークの出力側レイヤーから分類器を訓練するのはよくないでしょう。かわりに、SVMをネットワークのもっと入力側のどこかの特徴量を使って訓練するのがよいでしょう。
4. 新しいデータセットが大きく、元のデータセットとぜんぜん違う場合 データセットが非常に大きいので、畳み込みニューラルネットワークをゼロから訓練しても大丈夫です。しかし実際には、訓練済みモデルの

重みでネットワークを初期化すると有効であることが頻繁にあります。この場合では、ネットワーク全体をファインチューニングするのに十分なデータと自信が持てるでしょう。

（拙訳）
3回の記事にわたる今回の実験では、ゼロからCNNを学習させたりCNNをファインチューニングするよりも、学習済みCNNから特徴量を抽出し線形識別させたほうが性能がよくなりました。これはいま紹介したノウハウを裏付ける結果となっています。
ファインチューニングのコツとしては他にも、Kerasの作者がブログで書いているように 、「**ランダムに初期化された層を後ろにつけた状態で、学習済みのレイヤーごと学習させようとしない**」「RMSPropのような最適化アルゴリズムを使わずに、**SGDなどで小さい学習レートで学習させる**」などがあるようです。

まとめ

この記事では、VGG16のモデル・パラメータを使って画像の特徴量を抽出し、SVMで学習することで、**ついに100%の精度をもつ桜-コスモス分類器を作成**することができました。
第1回の記事ではゼロからCNNを学習させて過学習に苦しみ、第2回の記事ではVGG16をファインチューニングすることで過学習をある程度克服しました。最後に、第3回のこの記事ではニューラルネットワークを学習させることをあきらめ、SVMで線形識別させることで、**過学習を克服**し精度100%を達成しました。

参考

- 竹内一郎、烏山昌幸『サポートベクトルマシン』
- CS231n Convolutional Neural Networks for Visual Recognition
- Building powerful image classification models using very little data

#Amazon EC2#Custom Vision Service#Keras#Microsoft Azure#Python#TensorFlow#ニューラルネッ
データ分析と機械学習とソフトウェア開発をしています。 アルゴリズムとデータ構造が好きです。

GCPの利用料が安くなる|GCPの請求代行・運用代行・導入移行支援AWSアの

2020.5.18 **GCP**2020.3

About us会社情報
セミナー・イベント
採用情報
執筆者への取材依頼
フォトギャラリー

Category AWS
Data Science
Tech
Event
Column
Tags
Members

商標について
個人情報保護方針
ISMS認証