

@cvusk (/cvusk) 2018年06月23日に更新
(/cvusk)

...

誤ったラベルから正しいラベル分類を学習する弱教師あり学習

Python(/tags/python) 機械学習(/tags/%E6%A9%9F%E6%B0%E5%AD%A6%E7%BF%92)

Keras(/tags/keras) TensorFlow(/tags/tensorflow)

弱教師あり学習 (/tags/%E5%BC%B1%E6%95%99%E5%B8%AB%E3%81%82%E3%82%8A%E5%AD%A6%E7%BF%92)

▲ この記事は最終更新日から1年以上が経過しています。

誤ったラベルから学習する弱教師あり学習

教師あり学習で分類モデルを作るとき、通常の教師データに使うのは「正しく」ラベリングされたデータです。

教師あり学習では正しい教師データをもとにして、入力データを正しく分類できるモデルを生成します。

しかし、正しくラベリングされたデータをいつでも入手できるとは限りません。





たとえば新しく画像分類モデルを作ろうとする場合、まずは分類対象の教師データを収集し、それらに人手でラベリングをする必要があり、工数と費用がかかります。

さらにはラベルの種類が多くなると、そのラベリングの所要時間は増大します。

正しいラベルをつけることは面倒ですが、しかし誤ったラベルをつけることは比較的簡単です。

人手でやっても、10個のラベルから正しいものを選ぶより、間違っているものを選ぶほうが高速に楽にできます。

Choose the right one:
0. airplane
1. dog
2. penguin
3. cup
4. truck
5. sky
6. mouse
7. clock
8. chair
9. tree
10. house
11. door
12. cat
13. table
14. laptop
15. car



Choose a wrong one:
0. airplane
1. dog
2. penguin
3. cup
4. truck
5. sky
6. mouse
7. clock
8. chair
9. tree
10. house
11. door
12. cat
13. table
14. laptop
15. car

(<https://camo.qiitusercontent.com/66b8c1278b58a7401a4b13660b8298339cea09d6/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3535338342f37313361393038352d376465622d313536362d353566392d3932313737373065346439642e706e67>)

たとえば上記の場合、猫の正解ラベルは12番です。





ラベル付け作業は以下のようになり、誤りラベルを付けるほうが大した注意も必要なく、簡単だと思います。

- 正解ラベルを付ける場合：猫画像にはラベル12を探して、12番と付ける。
- 誤りラベルを付ける場合：12番以外で、適当な数字を選ぶ。

すべての教師データがランダムに誤ってラベリングされている場合、その特徴量をもつデータはラベリングされていないものが正しいラベルになります。

機械学習においても、抽出された特徴量に合致するものがラベリングされていないものを消去法的に分類するように学習すれば、結果として正しい分類ができるようになります。

Choices:
0. airplane
1. dog
2. penguin
3. cup
4. truck
5. sky
6. mouse
7. clock
8. chair
9. tree
10. house
11. door
12. cat
13. table
14. laptop
15. car



0. airplane

2. penguin

6. mouse

11. door

wrong labels

(<https://camo.qiitusercontent.com/32234791fd9a1598d1a551493db5def7d5cb2fd2/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f35353338342f66376465383561302d356434322d313366662d366332632d3230393461653938616633302e706e67>)

こうした誤ったデータ（またはデータが不十分な状態）から教師あり学習でモデルを生成する手法を半教師あり学習（semi-supervised learning）、または弱教師あり学習（weak supervised learning）と呼びます。

今回は誤ったラベルの教師データのみから、正しい分類モデルを生成してみます。
Cifar10のデータを使い、Convolutional Neural Network（CNN）でモデルをつくります。
まずはCNNで正解ラベルで分類モデルをつくり、同じ構造のCNNで誤ったラベルから分類モデルを生成して正解率を比較します。

元ネタ

Learning from Complementary Labels (<https://arxiv.org/abs/1705.07541>)という論文で、誤ったラベルデータ（complementary labels、補ラベル）から学習する手法が論じられています。
具体的な内容は論文をご一読いただくのが良いですが、誤ったラベルから学習する際に重用なのは、誤った教師データに対して誤った分類をするように学習することです。
入力データが誤っている前提なので、入力データのラベルどおりに学習してはいけません。
ランダムに誤っている入力データに対して誤るように学習することで、結果的に正確な分類モデルを生成します。

そのための手法として、損失関数を工夫します。
通常の学習（正しいラベルの学習）では、正しく分類するとクロスエントロピー誤差は小さくなります。
学習の目的はクロスエントロピー誤差を最小化することです。

誤ったラベルからの学習では、通常の学習とは逆に、誤った分類をするとクロスエントロピー誤差が小さくなるように損失関数を組み、誤差を最小化するように学習します。

論文では誤ると誤差が小さくなる関数として以下が説明されています。

Zero-one loss: $\ell_{0-1}(z) = \begin{cases} 0 & \text{if } z > 0, \\ 1 & \text{if } z \leq 0, \end{cases}$

(12)

Sigmoid loss: $\ell_S(z) = \frac{1}{1 + e^z},$

(13) (ht

Ramp loss: $\ell_R(z) = \frac{1}{2} \max \left(0, \min \left(2, 1 - z \right) \right).$

(14)

[tps://camo.qiitusercontent.com/0272459c3c36b34684bdee6123586659c4934cc9/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f35353338342f6636135326164662d613365392d303437622d336266332d3434336532386664623133342e706e67](https://camo.qiitusercontent.com/0272459c3c36b34684bdee6123586659c4934cc9/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f35353338342f6636135326164662d613365392d303437622d336266332d3434336532386664623133342e706e67))

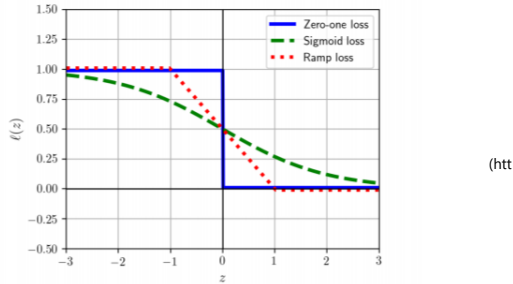


Figure 1: Examples of binary losses that satisfy the symmetric condition (11).

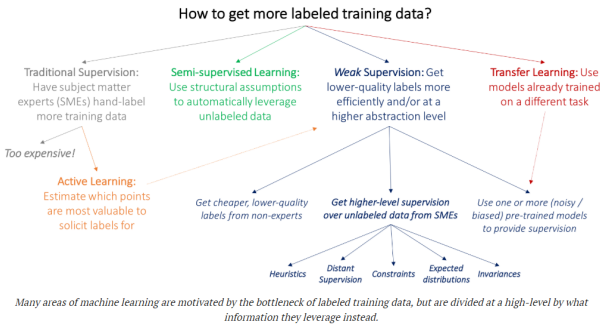
ps://camo.qiitusercontent.com/c92adab95acadb9b89c9a011df8aac62ff2c020/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f35353338342f63656432383565302d373164632d363766342d653439652d3661643162373738326338652e706e67)

今回はSigmoid lossを使って検証してみます。

教師あり学習の種類

学習手法に応じた教師あり学習の種類は以下で説明されています。
参考 (https://dawn.cs.stanford.edu/2017/07/16/weak-supervision/)

教師あり学習は以下のように分類できるようです。



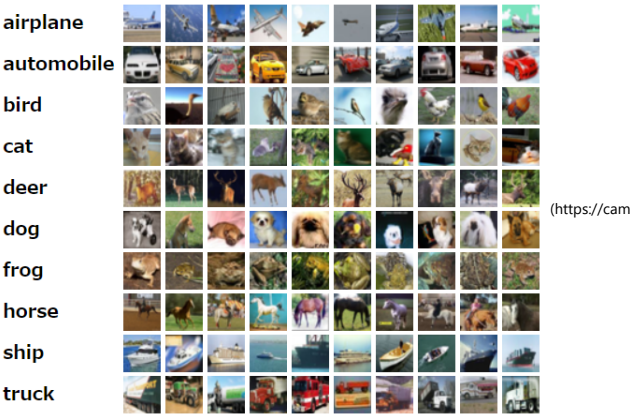
(https://camo.qiitusercontent.com/402b24f337a38248b8f012bde69a62d2d8087f76/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f35353338342f63656432383565302d373164632d363766342d653439652d3661643162373738326338652e706e67)

上記のうち、Active Learningや半教師あり学習は以前検証してみたことがある (https://qiita.com/cvusk/items/f411dadd0e464b7d5471)ので、興味があればご参照ください。

やってみた

データ

Cifar10 (https://www.cs.toronto.edu/%7Ekriz/cifar.html)から3ラベル、4ラベル、10ラベルを抜粋して使います。



o.qiitausercontent.com/a18c540d4f6dfdc0899a1eb93e5c90343773caa6/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f3533338342f33376462306135632d666139622d353863392d646632622d3363356165313364653133382e706e67)

- 3ラベル：Cifar10のうち、飛行機（0）、自動車（1）、鳥（2）を教師データとして使用します。
- 4ラベル：Cifar10のうち、飛行機（0）、自動車（1）、鳥（2）、猫（3）を教師データとして使用します。
- 10ラベル：全データを使用します。

方針

Cifar10のラベルデータに対し、ランダムに誤ったラベルをつけて教師データとします。
誤ったラベルからの学習は正しいラベルからの学習よりもデータ量がシビア（たとえば3ラベルで分類しようとするとき、1枚の誤ったラベルは、そのラベルではないという情報しか持たない（どれが正解かわからない））ため、検証では3ラベル、4ラベルと少ないラベル数で分類モデルを生成します。

手順：

- データを用意する
 - 正しいラベルのデータ（Cifar10）を用意する
 - 正しいラベルのデータをもとに、誤ったラベルのデータを用意する
 - 誤ったラベルはランダムに選択する
- CNNで学習する
 - 正しいラベルの分類器をCNNで学習し、正解率を計測する
 - 誤ったラベルの分類器をCNNで学習し、正解率を計測する

環境はGoogle Colaboratory (<https://colab.research.google.com>)、ライブラリはKerasで、GPUとしてNvidia K80を使用しています。

コードは以下にあります。
https://github.com/shibuiwilliam/complementary_labels_keras (https://github.com/shibuiwilliam/complementary_labels_keras)

データを用意する

Cifar10のデータはKerasで入手可能です。
まずは3ラベル（飛行機、自動車、鳥）だけで分類モデルを生成します。
誤ったラベルはランダムにつけます。

例：ターゲットデータとして、飛行機（ラベル0）に対しては、ラベル1または2をつける。これはトレーニング、テスト両方のデータに対して実行する。

```
import keras
from keras.datasets import cifar10

import os
import numpy as np
import math

num_classes = 3

# load dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# generate complementary labels for training targets
y_train3_pos = np.where(y_train<=2)[0]
x_train3 = x_train[y_train3_pos]
y_train3 = y_train[y_train3_pos]

y_ctrain3 = np.zeros(len(y_train3)).reshape(len(y_train3), 1)
for i,v in enumerate(y_train3):
    if v == 0:
        y_ctrain3[i] = np.random.choice([1,2],1)
    elif v == 1:
        y_ctrain3[i] = np.random.choice([0,2],1)
    elif v == 2:
        y_ctrain3[i] = np.random.choice([0,1],1)

# generate complementary labels for test targets
y_test3_pos = np.where(y_test<=2)[0]
x_test3 = x_test[y_test3_pos]
y_test3 = y_test[y_test3_pos]

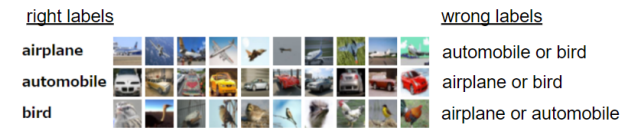
y_ctest3 = np.zeros(len(y_test3)).reshape(len(y_test3), 1)
for i,v in enumerate(y_test3):
    if v == 0:
        y_ctest3[i] = np.random.choice([1,2],1)
    elif v == 1:
        y_ctest3[i] = np.random.choice([0,2],1)
    elif v == 2:
        y_ctest3[i] = np.random.choice([0,1],1)

# prepare x dataset
x_train3 = x_train3.astype('float32')
x_test3 = x_test3.astype('float32')
x_train3 /= 255
x_test3 /= 255

# Convert ordinary class vectors to binary class matrices.
y_train3 = keras.utils.to_categorical(y_train3, num_classes)
y_test3 = keras.utils.to_categorical(y_test3, num_classes)

# Convert complementary class vectors to binary class matrices.
y_ctrain3 = keras.utils.to_categorical(y_ctrain3, num_classes)
y_ctest3 = keras.utils.to_categorical(y_ctest3, num_classes)
```

これで正しいターゲットデータ（y_train3, y_test3）と誤ったターゲットデータ（y_ctrain3, y_ctest3）ができました。



(<https://camo.qiitusercontent.com/5e037d5a08f8c7a00e397e2ea14dbd1c41b9cf02/68747470733a2f2f71696974612d696d6167652d73746f72652e73332e616d617a6f6e6177732e636f6d2f302f35353338342f37343534363833372d376634652d323632622d623235322d3766626434383862393761332e706e67>)

学習する

正しいラベルで学習する

正しいターゲットデータで学習し、CNNの分類モデルを生成します。

```

import keras

from keras.layers import Dense, Conv2D, BatchNormalization, Activation, MaxPooling2D
from keras.layers import Input, GlobalAveragePooling2D, Dropout
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras import backend as K
from keras.models import Model
from keras.datasets import cifar10
from keras.callbacks import EarlyStopping, CSVLogger
from keras import metrics

def vgg(x_train, classes):
    inputs = Input(shape=x_train.shape[1:])

    x = Conv2D(64, (3,3), padding="same")(inputs)
    x = Activation('relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)
    x = Conv2D(64, (3,3), padding="same")(x)
    x = Activation('relu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2,2))(x)

    x = Conv2D(128, (3,3), padding="same")(x)
    x = Activation('relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)
    x = Conv2D(128, (3,3), padding="same")(x)
    x = Activation('relu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2,2))(x)

    x = Conv2D(256, (3,3), padding="same")(x)
    x = Activation('relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)
    x = Conv2D(256, (3,3), padding="same")(x)
    x = Activation('relu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2,2))(x)

    x = GlobalAveragePooling2D()(x)
    x = Dropout(0.5)(x)
    x = Dense(classes, activation='softmax')(x)

    model = Model(inputs, x)
    return model

# load model for ordinary class
model = vgg(x_train3, num_classes)

batch_size = 32
epochs = 100

es_cb = EarlyStopping(monitor='val_loss', patience=3, verbose=1, mode='auto')
csv_log = CSVLogger("Cifar3_normal.csv", separator=',', append=True)

model.compile(loss='categorical_crossentropy',
               optimizer=Adam(lr=0.0001, decay=1e-5, amsgrad=True),
               metrics=['accuracy'])

# define image generator for data augmentation
datagen = ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False)

datagen.fit(x_train3)

# train
history = model.fit_generator(datagen.flow(x_train3, y_train3,
                                           batch_size=batch_size),
                             steps_per_epoch=x_train3.shape[0] // batch_size,
                             epochs=epochs,
                             validation_data=(x_test3, y_test3),
                             callbacks=[es_cb, csv_log])

# evaluate
evaluation = model.evaluate(x_test3, y_test3)
print("TEST Loss: {} \t Accuracy: {}".format(evaluation[0], evaluation[1]))

```

テストデータに対する正解率が約91.43%の分類モデルが生成されました。

誤ったラベルで学習する

続いて、新たにモデルをつくり、誤ったラベルで学習します。
ニューラルネットワークの構造は正しいラベルのものと同じものを使います。

```
# load model for complementary label
cmodel = vgg(x_train3, num_classes)

batch_size = 32
epochs = 100

def sigmoid_loss(target, output):
    return 1 / (1 + math.e ** K.categorical_crossentropy(target, output))

def caccuracy(target, output):
    return 1 - metrics.categorical_accuracy(target, output)

es_cb = EarlyStopping(monitor='val_loss', patience=3, verbose=1, mode='auto')
csv_log = CSVLogger("Cifar3_complementary.csv", separator=',', append=True)

cmodel.compile(loss=sigmoid_loss,
                optimizer=Adam(lr=0.0001, decay=1e-5, amsgrad=True),
                metrics=[caccuracy])
```

損失関数 (sigmoid_loss) は論文 (<https://arxiv.org/abs/1705.07541>)のSigmoid Lossを参考にして
います。
この関数では、誤りラベルに対して同じラベルを分類すると誤差が大きくなり、違うラベルを分
類すると誤差が小さくなります（誤りラベル0を0と分類すると誤差が大きくなり、1と分類する
と誤差が小さくなる）。
評価関数は 1-(誤ったラベルに対する正解率) で、参考値程度のつもりです。

損失関数や評価関数を独自に実装する方法は以下公式ドキュメントをご参照ください。
損失関数 (<https://keras.io/ja/losses/>)
損失関数の実装 (<https://github.com/keras-team/keras/blob/master/keras/losses.py>)
評価関数 (<https://keras.io/ja/metrics/>)

この設定で学習してみます。

```
datagen = ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False)

datagen.fit(x_train3)

# train
history = cmodel.fit_generator(datagen.flow(x_train3, y_ctrain3,
                                             batch_size=batch_size),
                              steps_per_epoch=x_train3.shape[0] // batch_size,
                              epochs=epochs,
                              validation_data=(x_test3, y_ctest3),
                              callbacks=[es_cb, csv_log])

cpred = cmodel.predict(x_test3)

score = 0
for i in range(len(y_test3)):
    if np.argmax(y_test3[i]) == np.argmax(cpred[i]):
        score += 1
print("TEST Accuracy: {}".format(score / len(y_test3)))
```

結果

3ラベルでの誤ったラベル学習では、テストデータに対して約85.90%の正解率の分類モデルが生
成されました。
正しいラベルでの学習が正解率91.43%だったので、悪くない分類モデルが生成されていると思
います。

同様に検証して4ラベル、10ラベルの成績は以下のとおりです。

	正しいラベルの正解率	誤ったラベルの正解率
3ラベル	91.43%	85.90%


	正しいラベルの正解率	誤ったラベルの正解率
4ラベル	84.80%	62.12%
10ラベル	81.84%	16.76%


10ラベルではだいぶ低い正解率となりました。

10ラベルでは、1つのラベルに対して誤りラベルが9つあるため、正解ラベルを特定する特徴量を集約するのが難しくなるのだと思います。


誤りラベルでも教師データを増やせば正解率を上げることができると思われます。

ただし、誤りラベルも工数0で無料で付けられるわけではないので、正解ラベルを付ける工数や費用との兼ね合いだと思います（ということを書き出すと元も子もないのかもしれませんが・・・）。


編集リクエスト (https://qiita.com/cvusk/items/66252eb48e89435824f5/edit) 

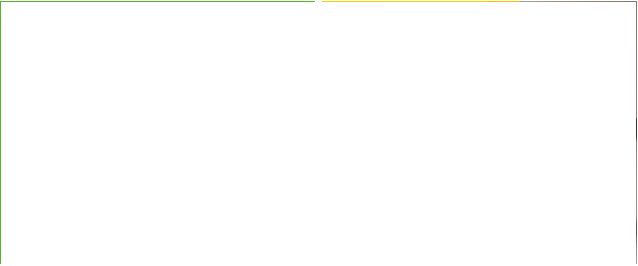
 15 (https://qiita.com/cvusk/items/66252eb48e89435824f5/likers)

(https://qiitadon.com/share?;E6%95%99%E5%B8%AB%E3%81%82%E3%82%8A%E5%AD%A6%E7%BF%92%20by%20cvusk%20https%3A%2F%2Fqiita.com%2Fcvusk%2Fitems%2F66252eb48e89435824f5%20%23Qiita%20%2



@cvusk (/cvusk)
#Python #Golang #C++ #Bash #Linux #MachineLearning #DeepLearning #Keras #Tensorflow #Docker #Kubernetes #JupyterNotebook #AWS #Azure #GCP #Bluemix #SAS #MENZA #Unity
https://github.com/shibuiwilliam (https://github.com/shibuiwilliam)





コミュニティスポンサー広告 (http://blog.qiita.com/post/176483510744/community-sponsor)

関連記事 Recommended by (https://www.logly.co.jp/privacy.html)

- 

Jupyter Notebookでいじって学ぶTensorFlow - MNIST For...
(https://qiita.com/oimou/items/4a4258a7f7cc2bd70afe)
by oimou
- 


DBNとLabelSpreadingを使った半教師ありラベル学習
(https://qiita.com/shima_x/items/1899ccc769a29559679f)
by shima_x
- 


今さら聞けないGAN（１） 基本構造の理解
(https://qiita.com/triwave33/items/1890ccc71fab6cbca87e)
by triwave33
- 

ロジスティック回帰 [TensorFlowでDeep Learning 1]
(https://qiita.com/kumonkumon/items/53e3121e7031b9402205)
by kumonkumon
- **アイシン精機のAI・人工知能エンジニアにインタビュー**

(https://dsp.logly.co.jp/click?ad=G8GmQLsVMLY2zbNgg0Hv6trhcugTqGIWhRjzFvni7dzNF-B5dwfSA65BnF2MPhGRqXs9JQ9AZOJfpkIRGNauic_KaolqfjAyLINrYjZNdMSACeaG_pcisfG5jWyTsY-DunLP5LjVnKZLvif8OX3ZjQsaTABOWZ0KEuyHYOWCowgscvPPcUOM80YthzZCcRYNNdclRaltPa7igqCJarThzAMy4amYqVjCnzl2EvGeMsZeSrMY8Kv6d8fj1Bp97Q4xKJ8kftMBiZL5-SchkhZiWFOwQl446NzPY0MGoedQNL4dqDjfkKMDJv_mtmWH40i8Rt4KwVAfNkWRGDp0QT_PlyUPaUKSkRgRYuc_5qa9sTKT8PoiOOW6WqaUZY-ivhGpWVSEJF4JMSGy2uOLJ7ns1qi3ehrGdsrf3u4S0h2LnpBcpmPyUpdxwe5Taxh3wvyj1lu538phwDOzRb1M6GnPk8mopkxwpOTbWPYNOcbl-AgWo38FoT1R1f_6teVqx_cwOxnXw)
PR アイシン精機

この記事は以下の記事からリンクされています

- 

弱教師あり学習は何を見ているのか？ (/cvusk/items/52724ca5c7e286678fa7#_reference-397c49bc12a53c0e1d37)
からリンク 約1年前
- 

CIFAR-10で「天然物か人工物かどうか」のフラグを入れて分類すると精度が上がるか？ (/koshian2/items/f1e86fb5f979f525c23#_reference-d5475503b0f2e0c98ff3) からリンク 8ヶ月前