Category Classification With Animal Image Data

Chudan Liu
Harvey Mudd College
Claremont, CA 91711
iliu@g.hmc.edu

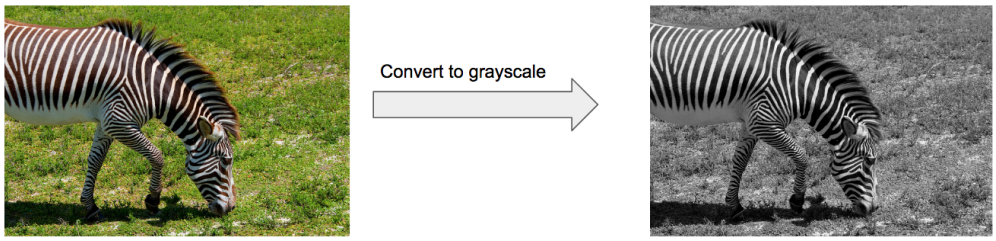
Xinyu Yang
Harvey Mudd College
Claremont, CA 91711
xiyang@g.hmc.edu

Abstract

1 Introduction

In this paper we are dealing with image classification problem, which is the job of assigning an input image one label from a fixed set of categories. This is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications. Moreover many other seemingly distinct Computer Vision tasks (such as object detection, segmentation) can be reduced to image classification.

The task in Image Classification is to predict a single label (or a distribution over labels as shown here to indicate our confidence) for a given image. Images are 3-dimensional arrays of integers from 0 to 255, of size Width x Height x 3. The 3 represents the three color channels Red, Green, Blue. For the sake of our future implementation, we transfer the RGB into grayscale image.



For this project, we selected the Common Object in Context (COCO) dataset http://mscoco.org/dataset, a popular image recognition, segmentation, and captioning dataset to perform image classification. We chose this dataset because it has over 80,000 images with multiple objects and clear labels. Before discussing the classifier used in this project, we would first introduce our data construction. In this project, we chose the logistic regression model and a multi-layer perceptron neural network model to classify our data and make the prediction.

2 Data Processing

We used some open-source software libraries to facilitate the implementation, including sklearn http://scikit-learn.org/stable/, opencv (Open Source Computer Vision Library) opencv.org/, numpy http://www.numpy.org/, and matplolib https://matplotlib.org. For algorithm, we used the logistic regression model, support vector machine(svm) and neural network module in sklearn to train and test our data with different algorithms. To analyze the performance of our prediction, we plot the confusion matrix using the matrix class in sklearn.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2.1 Introduction to the COCO dataset

COCO is a popular image recognition, segmentation, and captioning dataset with more than 30,000 images, 2 million instances, and 80 object categories. In this project, we used the 2014 dataset for detection challenge with object instance annotations provided as our primary dataset. For this dataset, there are total 82,782 images of over 90 different categories.

The dataset are in json format, providing comprehensive information. Each annotation instance contains a series of fields, including the category id and segmentation mask of the object, which provides information including the information of the dataset, the categories and supercategories each image belong to and information about segmentation, etc.

In this project, we looked into each annotation instance and created our final dataset using images with supercategory 'animal.' For this supercategory in particular, there are ten categories shown as below:

```
{'id': 16, 'name': 'bird', 'supercategory': 'animal'},
{'id': 17, 'name': 'cat', 'supercategory': 'animal'},
{'id': 18, 'name': 'dog', 'supercategory': 'animal'},
{'id': 19, 'name': 'horse', 'supercategory': 'animal'},
{'id': 20, 'name': 'sheep', 'supercategory': 'animal'},
{'id': 21, 'name': 'cow', 'supercategory': 'animal'},
{'id': 22, 'name': 'elephant', 'supercategory': 'animal'},
{'id': 23, 'name': 'bear', 'supercategory': 'animal'},
{'id': 24, 'name': 'zebra', 'supercategory': 'animal'},
{'id': 25, 'name': 'giraffe', 'supercategory': 'animal'}
```

2.2 Assigning training and testing data sets

Considering the enormous size of the whole dataset, we found it infeasible to train the whole dataset based on the current device we have. Instead, we selected one supercategory 'animal', which contains ten different categories or species shown above. We first extracted all images labeled with $category\_id$ that belongs to the supercategory 'animals,' which consist of over 19,000 images. After we obtained all animal images, we randomly sample 5,000 images as our final full dimension dataset.

We divided the dataset by assigning $3/4$ of the data, 3750 images, to our training set and the rest of $1/4$, 1250 images, to the testing set. When we sampled the dataset, we shuffled the dataset so that images of most categories under this supercategory are included in the dataset.

2.3 Dimension analysis

3 Learning Algorithms

In our project, we used two algorithms to classify the data: multinomial Logistic Regression and muli-layer perceptron neural network(MLP). Below are details about the algorithm we used in this project.

3.1 Multinomial Logistic Regression

Multinomial Logistic Regression is the first learning algorithm we used. Given a set of independent categorical variables, which are the pixels of animal images from our dataset in this case, we trained a logistic regression model to predict the probabilities of the different possible animals of the independent categorical variables.

As most other statistical classification models, the foundation of the multinomial logistic regression model is to construct a linear predictor function that computes a score from a set of regression coefficients that indicate the relative effect of variables on the y. In particular,

$$Score(k, X_i) = \beta_k \cdot X_i,$$

where k indicates the k^{th} outcome of , X_i is the vector of explanatory variables, which are all the pixels of an image describing observation i, β_k is a vector of regression coefficients corresponding to outcome k.

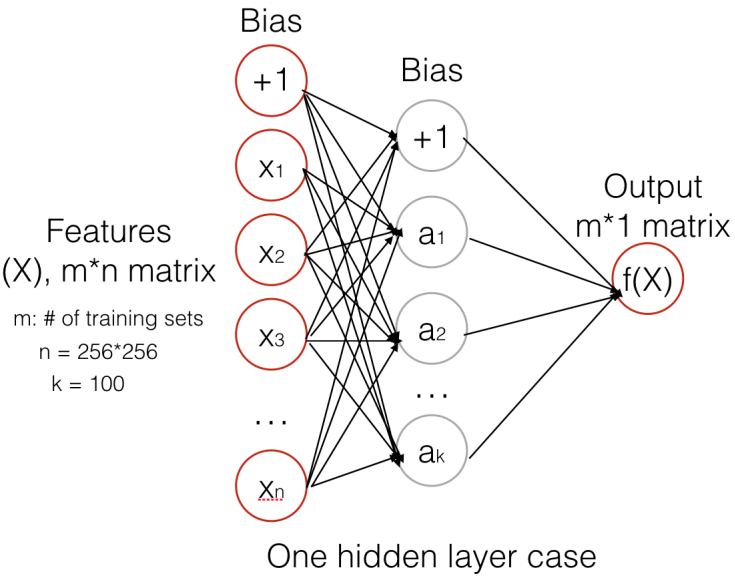
In terms of actual implementation, we added a regulation parameter $reg = 0.2$ to prevent the case of overfitting. To analyze the model's performance, we computed the accuracy and generated confusion matrix based on the training set and the test set.

3.2 Multi-layer Perceptron(MLP) Classifier

Multi-layer Perceptron (MLP) `http://scikit-learn.org/stable/modules/neural_networks_supervised.html` is a supervised learning algorithm with multiple hidden layers. It inputs a set of m features

$$X = x_1, x_2, ..., x_m$$

and an output function $f(X)$, learns a non-linear function approximator for either classification or regression. In our implementation, we used the class MLP Classifier in sklearn library. We set the solver as 'adam,' a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba `https://arxiv.org/abs/1412.6980`. We originally set the hidden layer size as (100,100,100), which included three hidden layers, each has size 100, and we also added L2 penalty (regularization term) parameter.



As someone may ask, given the logistic regression model, why do we need multi-layer perceptrons to classify dataset. Comparing with logistic regression, the advantage of multi-layer perceptrons is the capability to learn non-linear models. However, different random weight initializations can have different test accuracy because MLP with hidden layers have a non-convex loss function where there exists more than one local minimum.

4 Performance and Analysis

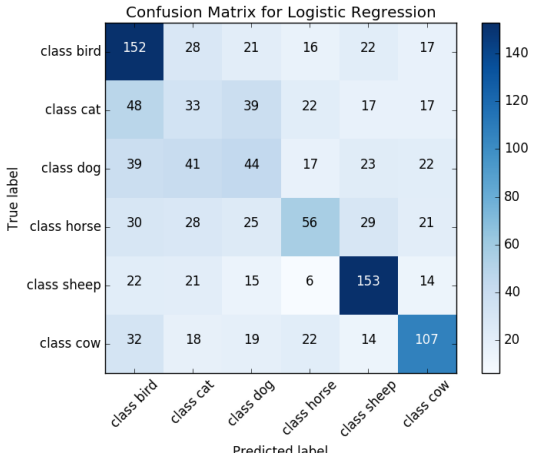
4.1 Evaluate Performance: Confusion Matrix

4.2 Performance of Multinomial Logistic Regression

Testing Accuracy analysis:

```
-- Time elapsed for training: 1815.84 seconds
==> Testing the logistic regression model...
-- Training accuracy: 0.9880
-- Testing accuracy: 0.4360
================
==> Plotting confusion matrix...
Confusion matrix, without normalization
[[152  28  21  16  22  17]
 [ 48  33  39  22  17  17]
 [ 39  41  44  17  23  22]
 [ 30  28  25  56  29  21]
 [ 22  21  15   6 153  14]
 [ 32  18  19  22  14 107]]
********************************
```

Confusion matrix analysis:

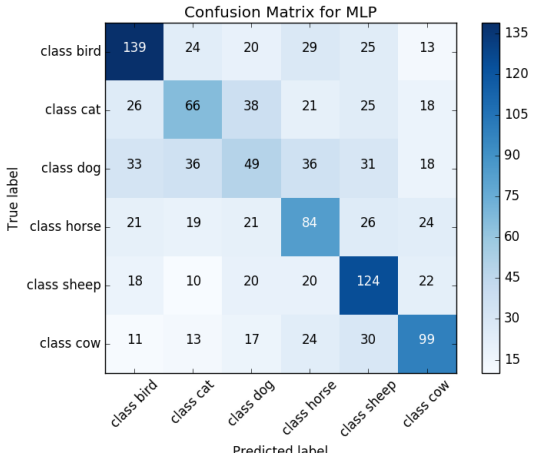


4.3 Performance of Multi-layer Perceptrons

Testing Accuracy analysis:

```
==> Setting up MLP model
==> Start training the MLP model...
/Users/IvyLiu/anaconda/lib/python3.5/site-packages/sklearn/neural_network/multilayer_perceptron.py:904: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example usi
ng ravel().
  y = column_or_1d(y, warn=True)
===============
-- Actual number of iterations: 13
-- Time elapsed for training: 121.34 seconds
==> Testing the MLP model...
-- Training accuracy: 0.1971
-- Testing accuracy: 0.1952
===============
==> Plotting confusion matrix...
Confusion matrix, without normalization
[[  0   0 180  80   0   0]
 [  0   0 114  51   4   0]
 [  0   0 143  65   2   0]
 [  0   0  90  99   0   0]
 [  0   0 122 102   2   0]
 [  0   0  94 101   1   0]]
*******************************
```

Confusion matrix analysis:

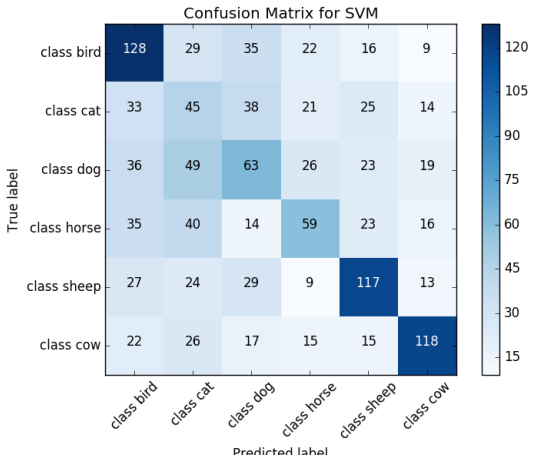


4.4 Performance of Linear SVM

Testing Accuracy analysis:

```
==> Setting up model for linear support vector machine...
==> Start training the linear SVM model...
/Users/IvyLiu/anaconda/lib/python3.5/site-packages/sklearn/utils/validation.py:526: DataConversionWarning: A column-vector y
 was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
===============
-- Actual number of iterations: 1000
-- Time elapsed for training: 3691.60 seconds
==> Testing the linear SVM model...
-- Training accuracy: 0.9896
-- Testing accuracy: 0.4240
================
==> Plotting confusion matrix...
Confusion matrix, without normalization
[[128  29  35  22  16   9]
 [ 33  45  38  21  25  14]
 [ 36  49  63  26  23  19]
 [ 35  40  14  59  23  16]
 [ 27  24  29   9 117  13]
 [ 22  26  17  15  15 118]]
********************************
```

Confusion matrix analysis:



5 Conclusion

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. **Remember that you can use a ninth page as long as it contains *only* cited references.**

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.