Survey of Image Classification Methods

Anonymous Author(s)
Affiliation
Address
email

Abstract

Detecting background of images is one of the most important tasks in secret surveillance activity. This paper proposes several machine learning techniques for classifying surveillance footage taken from around the world, and analyzes the performances of those techniques. It also provides our methodologies for comparing and contrasting results, and finally our rationale for choosing the best classifier to be entered to an internal competition at the NSA.

1 Introduction

With the unearthing of surveillance images by our colleagues at Project S.U.N, we were given the task of identifying correct categories for those images to further advance the NSA's goals. We try various methods to find the most robust way to automatically detect the contents of each image by examining features extensively.

The paper is organized as following: in Section 2, we introduce the datasets and features available for training our machine learning techniques. In Section 3, we discuss the various implementation methods we tried. In Section 4, we present the results obtained from Section 3, and in Section 5, we analyze our findings and explain our rationale for choosing the best classifier.

2 Data and Features

With the advancement in Computer Vision technology and especially deep learning, we have access to deep-learned features of the images extracted from AlexNet. We were also given features extracted from bag-of-visual-word SIFT descriptors. In addition to AlexNet and SIFT features, we also have binary attributes, totaling 102, for the images.
We have training sets consisting of 3000 images that are labeled, and 10000 images that are unlabeled. For each of the unlabeled data, we have five English captions which describe the image. We also have 1000 test images on which we can test the accuracy of our algorithms. Both AlexNet and SIFT give each training image 4096 features.

3 Implementation

3.1 Data Preprocessing

The AlexNet and SIFT features are of the format .npy, a binary file format for storing a NumPy array. The binary attributes are provided in a .txt format. It is convenient to load and process these data types in Python using the Numpy module.

Figure 3.2.1 shows the distribution of each label in our AlexNet labeled training set. The data is uniformly distributed, and each label appears 15 times. Given that our features are 4096 dimensional

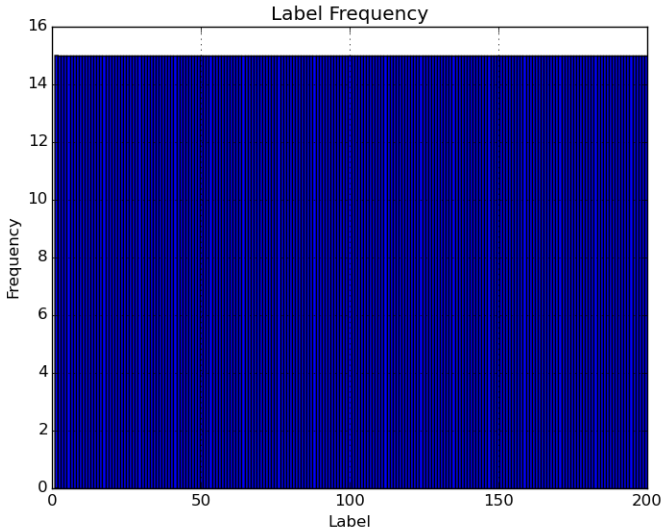


Figure 1: Histogram of label frequencies

vectors and we have 200 different labels, we are severely lacking data to properly train our machine learning algorithms with labeled data alone.

3.1.1 Dimensionality Reduction

In order to mitigate the high dimensionality of our dataset, we tried a few dimensionality reduction methods. By computing Singular Value Decomposition of the feature matrix, we realized the values of eigenvalues drop rapidly after the first two largest principal components. Thus, we chose to keep two principal components.

One disadvantage of PCA is that the basis function and coefficients can take on negative values. This is undesirable in reconstructing images, and thus we also explored NMF, which restricts values from having nonnegative values. We also chose to keep two principal components, and the result in the Figure 2 shows a dimension that has the most variance.

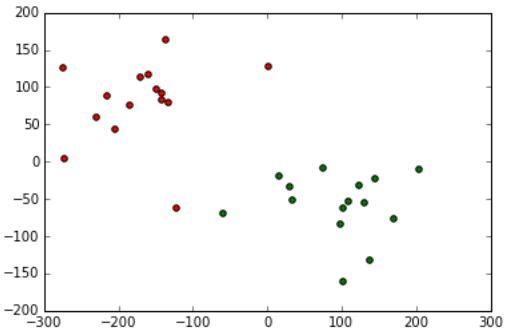


Figure 2: 2D Image feature

3.1.2 Standardization

For some classification methods, such as Naive Bayes and SVM, we make assumptions about the shape of data. Thus we standardized both the training and testing sets to fit the model. With standardization, we can fit the data so that it has a mean of 0 and a variance of 1.

Table 1: Distance Measures Comparison

	Time (s)	Accuracy
Cosine	0.691	0.37
Correlation	0.582	0.22
Cityblock	0.472	0.34
Euclidean	0.679	0.36

3.2 Method

3.2.1 Method 1: K-means Clustering with SVM classifier

The process of using k-means clustering is shown in Figure 3. The 10K image set's AlexNet features were fitted to k-means clustering algorithm to generate 1000 centroids, and then we used the centroids to extract feature using the AlexNet feature of 3000 training set data.

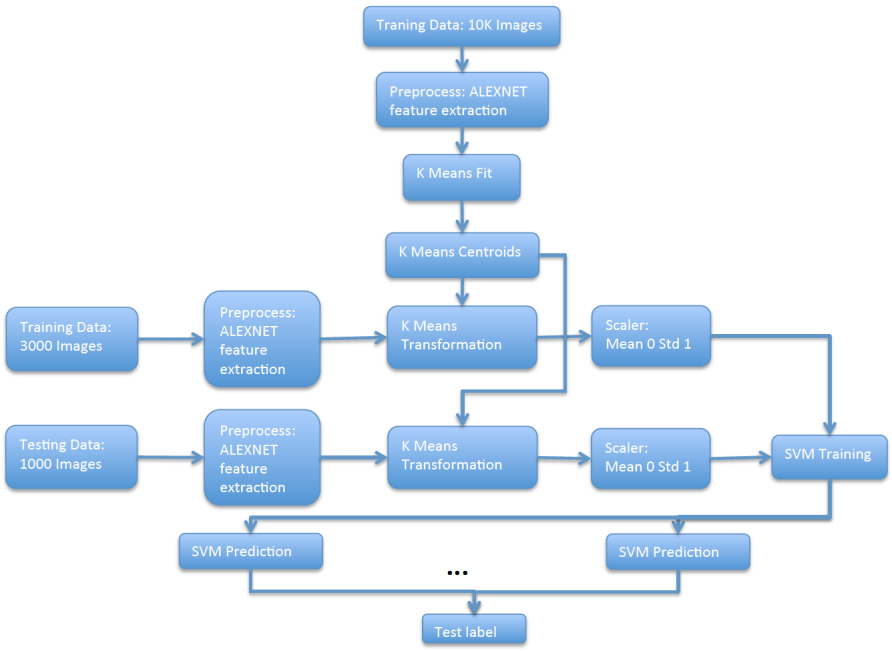


Figure 3: K-means Clustering and SVM Process

By implementing k-means clustering, we generated centroids according to the visual features of the images. In order to extract features, we calculated the cosine distance between each image and the centroid, and then used the k-means distance values as the new feature. By doing so, we reduced the features' size to $3000 \times k$ without losing the original features. This idea was inspired by the Stanford Visual Cortex project, where they extracted features that use only the centroid that most resembles the image by using average squared Euclidean distance. We interpreted the process creatively, and revised it accordingly. [1]

When we enter a high dimensional space, it is difficult to tell two distances apart. Thus, we compared the performances of different distance measures, including cosine distance, correlation distance, city block distance and euclidean distance. As shown in Table 1, cosine distance has the best accuracy, thus we extracted features from k-means clustering according to the cosine distance.

Next, before applying SVM, we standardized the feature vectors so that they have a mean of 0 and a standard deviation of 1.

Then we fitted the SVM using the "one vs rest" strategy. That is, we trained a SVM classifier for each category, which only give a binary result (yes or no). Then we combined the result from all SVM classifiers linearly, and assigned label that had the highest confident score. The idea is explained in the equation below:

$$y = \arg \max_{x \in 1...K} f_k(x) \tag{1}$$

The accuracy of this method from cross validation is 0.37 in average, and the accuracy on Kaggle is 0.43.

3.2.2 Method 2: Majority Vote

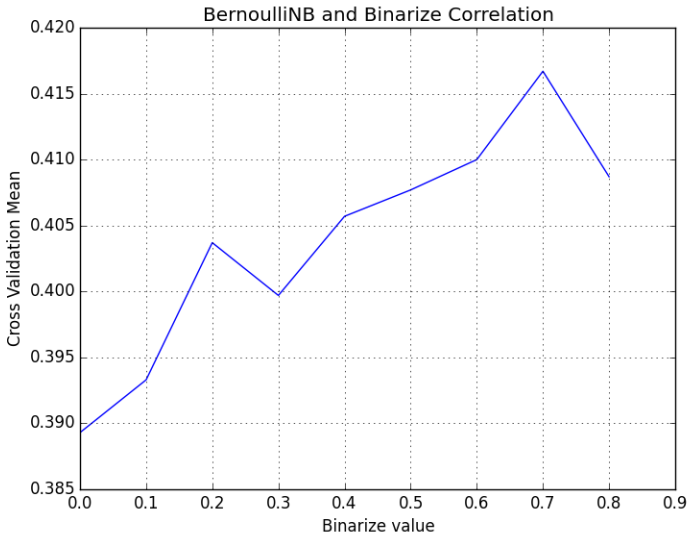
We tried various classification methods, but the performance of each classifier was not particularly noteworthy. Therefore, we used ensemble method to improve the performance of classification results. The goal of ensemble methods is to combine the predictions of several base estimators and therefore improve the generalizability and robustness over a single estimator.

Similarly, since we have three datasets which represent the same images from different perspectives, we also tried to leverage the results over the 3 datasets. The idea is that each classifier casts a vote for the prediction of the datasets and we pick the most popular one.

Naive Bayes Classifier

Naive Bayes makes the assumption that features in a class are conditionally independent. With this assumption, we can escape curse of dimensionality. Since we have a high dimensional yet sparse vector space with a large number of data points, it spares extensive computing power and time. We tried all three class prior models, namely Gaussian, Multinomial and Bernoulli. Naive Bayes has the assumption that each feature is i.i.d and it also require data looks like normally distributed data. So before we run the classifier, we standardize the data first.

We finally picked Bernoulli Naive Bayes because its cross validation accuracy was 42%, which outperformed the other two methods.



Support Vector Machines

Support Vector Machine is a supervised learning method which try to find a margin between categories and make the margin as wide as possible. The advantages of Support Vector Machines

is that it is effective in predicting in high dimensional spaces, and works even when the number of dimensions is greater than the number of samples.

Additionally, Support Vector Machines uses a subset of training points in the decision function called support vectors, which makes it memory efficient.

SVM classifier can be extended multi-class classification. SVM uses the "one vs rest" strategy, which means that it trained a SVM classifier for each category, and the classifiers only give a binary result. Then it combines the result from all SVM classifiers linearly, and give the label with the highest confident score or lowest loss.

Ensemble Methods

Majority Rule Ensemble Classifier

For cross-classifiers majority vote, Python's scikit-learn library currently does not support its implementation of VotingClassifier, thus we referred to Majority Rule Ensemble Classifier developed by Sebastian Raschka [1]. The classifier predicts labels in two different ways. It bases its decision on a majority voting rule or average of predicted probabilities.

And for cross-dataset majority vote, we used our own algorithm to combine the result of Bernoulli over Alexnet,Sift and Attribute.

3.3 Semi-supervised Learning

Semi-supervised learning uses both labeled and unlabeled data for training algorithms, usually with a small number of labeled data and a large number of unlabeled data. Since we have 3000 labeled data points and 10000 unlabeled data points, we deemed the method would augment our data set, which in turn would result in a more robust model.

3.3.1 Contrastive Pessimistic Likelihood Estimation (CPLE) and Self learning (self training)

There is an open source Python package for CPLE and Self-learning techniques written by Tamas Madl [2]. It picks a classifier with which we train the semi-labeled dataset, and then try to predict the labels for the unlabeled data. We used these methods to label the unlabeled portion of the data.

4 Analysis

We used k-Fold Cross Validation to assess the performance of each classifier, with k = 3 and 5. The result can be seen in Table 2:

The best result overall is the Majority Vote method with AlexNet features. If comparing between three features, the cross validation accuracy values are always higher for AlexNet features, and followed by Attributes, and SIFT. We believe it is because the SIFT data is very sparse, and SIFT is more about the edges and points information, while AlexNet in Theano comes from deep convnet.

The reason for k-means clustering's low accuracy could be attributed to the fact that extracting features from 1000 centroid distance was not theoretically proved to be the best way, and the number of centroids could be tested given more time.

With the semi-supervised learning method, we saw that the cross validation accuracy was way higher than performance on Kaggle, which we believe stems from possible overfitting. With the sheer number of data points in the augmented dataset and high dimensional nature of the data, the classifier was optimized for the training data, but did not perform well on the test data.

[1] http://rasbt.github.io/mlxtend/docs/classifier/scikit-learn_ensemble_classifier/

[2] https://github.com/tmadl/semisup-learn

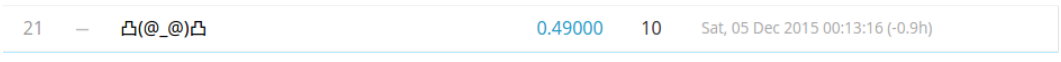
Table 2: Cross Validation Results

	AlexNet	SIFT	Attributes
Logistic Regression (LR)	0.37	0.14	0.30
k Nearest Neighbors	0.30	0.01	0.29
Random Forest	0.30	0.05	0.28
Naive Bayes	0.43	0.14	0.31
Support Vector Machine (SVM)	0.39	0.15	_
Extra Trees	0.37	0.13	0.29
Ensemble - SVM	0.375	_	_
K-means - SVM	0.38	_	_
Bagging (Naive Bayes)	0.36	_	_
Majority Vote (GNB,SVM,LR)	0.49	_	_

Another possible reason for having not very high accuracy is that the size of training set is not very large, and the given image features could be better. We've also trained a CNN network using Keras Cifa10, but we couldn't finish it because of the limitation of laptop's computing ability and time. It is also possible to use the pre-trained online CNN to extract features, and the result is expected to be better.

5 Summary

- Bernoulli Naive Bayes, Logistic Regression and SVM with linear kernel give the best result of this classification task. It makes sense that SVM perform good in this way because SVM has an advantage on high-dimensional dataset. And Bernoulli Naive Bayes works well on attribute data because it is binarized.
- Multiclassification strategy "one-vs-rest" works well to improve the performance of classification but also increases the running time because it is training separate classifiers per each category.
- Ensemble strategy seems to work well on improving accuracy. We got our best performance 49% by combining the result from different datasets. But in our case, the MajorityVote method does not seems to increase the performance significantly than a single classifier. Perhaps it is because we did not explore deep enough each method about ways to combine the results of different classifiers since we spent our time trying multiple methods.
- Changing parameter has a good effect on improving the performance of classifiers. We used the Gridsearchcv method and cross validation to test the results of tuning parameters.
- We failed in trying semi-supervised method. The problem is we had a really good performance in cross validation, 69%. But turned out to be poor as 46% after submitting to Kaggle. The cause behind this might be overfitting. Since semi-supervised method use labeled data to predict the labels of unlabeled data and then merge all this data to train a model, this model is bound to work well on the same training dataset.
- In this experiment, we gained more insight into solving practical problems in the real world. We started by trying various classifiers, tuning parameters and picked up one with the best performance. Through this project, we learned how to evaluate different models and select parameters. And we also discovered strategies such as voting when the single classification did not perform well enough and tried semi-supervised method when the training dataset is not big enough. Machine learnin is a very involved process, which not only entails thorough understanding of the fundamentals of algorithms, and the structure of data, but also trying various practical methods to keep making the model better.



Acknowledgments

We would like to acknowledge the extensive field work done by our fellow spies at Project S.U.N.

References

[1] Awni Hannun & Chris Piech (2013) Project 3: Visual Cortax. Stanford CS221.

[2] Dibya Jyoti Bora & Dr. Anil Kumar Gupta (2014) Effect of Different Distance Measures on the Performance of K-Means Algorithm: An Experimental Study in Matlab. *International Journal of Computer Science and Information Technologies* Vol. **5**(2), 2014, 2501-2506