Project-II by Group SanDiego

Juniong Wang
EPFL
junxiong.wang@epfl.ch

Zixuan Liang
EPFL
zixuan.liang@epfl.ch

Abstract

In this project,we analyze these two kinds of feature and create many model to classify the data. In general,we analyze the extraction process of HOG and CNN feature and state the reason why we choose CNN feature to train data. Also we use different kinds of methods to train data set and we compare BER as performance measurement among these methods. We find that for binary classification, SVM based on linear kernel is the best while for multi-class classification autoencoding MLP is the best method.

1 Data Description

Our train data is a 1×1 struct with 3 fields. One field X_cnn is the CNN feature of X. It is a 6000×36865 single matrix. This matrix is a sparse matrix. The non-zero elements of the matrix are mostly within the range of $[0, 10]$. The second field X_hog is the HOG feature of X. It is a 6000×5408 single matrix. The elements of the matrix are all within the range of $[0, 1]$. It indicates that we have 6000 samples. The dimension of CNN feature is 36865 and the dimension of HOG feature is 5408. The third field y is the category vector of the 6000 samples. They fall into 4 categories. These 4 categories are represented as $1, 2, 3, 4$. As for the CNN and HOG features, we will discuss them in the following section.

2 Feature Analysis

2.1 HOG

Histograms of Oriented Gradients is a feature extraction technique, which is described in [1]. The main idea of this paper is summarized as 1. The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. As it is shown in the 2(a), the HOG feature shows the contour of original picture. Left-hand side is the original picture, in which there is a car. Right-hand side is the picture after HOG feature extraction. We can see from the HOG feature picture the approximate shape of the original car. However from HOG feature picture we cannot see other details of original pictures. In their method, they extracted the feature with gradient computation firstly.Then each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. The cells themselves can either be rectangular or radial in shape, and the histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is 'unsigned' or 'signed'.

2.2 CNN

Convolution Neural Networks is described in [2]. The architecture of CNN is illustrated in Figure 2(b). CNN is biologically-inspired variants of MLPs, they have different kinds of layers and each

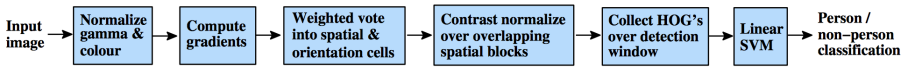
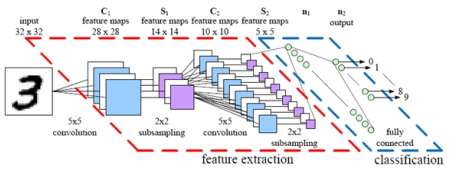


Figure 1: An overview of HOG feature extraction and object detection chain

(a) Original Car Picture and HOG Feature Picture



(b) Original Car Picture and CNN Feature Picture

Figure 2: Original Picture and Feature Picture

different layer works different than the usual MLP layers. Generally there are two layers, one is convolution layer and the other is subsample layer. The input to a convolutional layer is a $m \times m \times r$ image where m is the height and width of the image and r is the number of channels. The convolutional layer will have k filters (or kernels) of size $n \times n \times q$ where n is smaller than the dimension of the image and q can either be the same as the number of channels r or smaller and may vary for each kernel. The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size $mn + 1$. Each map is then subsampled typically with mean or max pooling over $p \times p$ contiguous regions where p ranges between 2 for small images (e.g. MNIST) and is usually not more than 5 for larger inputs. Either before or after the subsampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map. After the convolutional layers there may be any number of fully connected layers. The densely connected layers are identical to the layers in a standard multilayer neural network.
After reading the orginal paper [2], we realized that the the dimension of given X_cnn feature is 36864 rather than 36865. After observing all the elements of the 36865's column is 0, we remove this deliberate noise.

2.3 Feature Compare

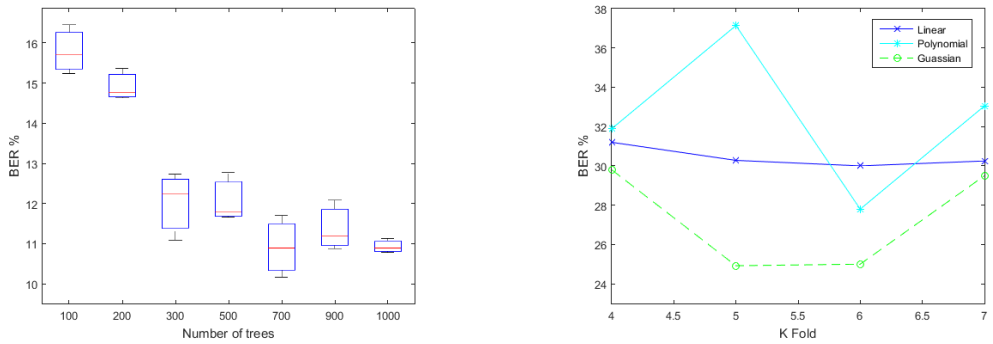
Accurately, we prefer the CNN method than the HOG. The disadvantage of HOG is obvious. The background of the image is still exist and it will be noise in the training process. Useful work like bounding object is totally ignored. Given HOG feature doesn't consider texture of images, and it has only shape characteristic. In the computer vision, we need to combine SIFT and other object recognition algorithm to get a correct recognition And this is proved by the experiment, comparing the Hog feature, there are more likely to gain a better accuracy rate using CNN feature.

3 BER Error

Balanced Error Rate is computed as

$$BER = \frac{1}{C}\sum_{c=1}^{C}[\frac{1}{N_c}\sum_{n=1}^{N}(y_n = c)(y_n \neq \hat{y_n})]$$

Where C is the number of classes, N_c is the number of examples in class c, y_n is the ground truth for sample n and $\hat{y_n}$ is its prediction. What does BER means is that it is the average of the proportion wrong classifications in each class. In this way we can avoid being biased by the large difference in number of samples between classes in the training data.



(a) BER over different number of trees

(b) BER over different K fold

Figure 3: BER of random forest and different kernel functio and K fold Using Hog feature

4 Logistic Regression

In multiclass logistic regression, we use softmax function to do the classification. The softmax function is the gradient-log-normalizer of the categorical probability distribution. Specifically, in multinomial logistic regression and linear discriminant analysis, the input to the function is the result of K distinct linear functions, and the predicted probability for the $j'th$ class given a sample vector x is:

$$P(y = j|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}}$$

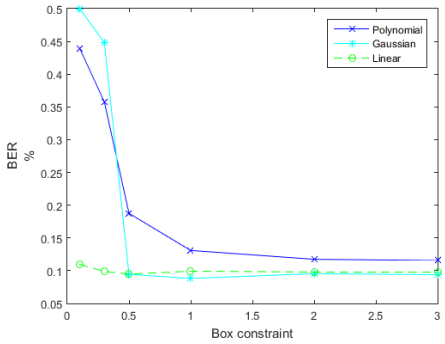
This can be seen as the composition of K linear functions and the softmax function (where x^TW denotes the inner product of x and W). In our experiment, we apply this method in the training data, and get a relatively poor model with BER is around 23.45%
In the binary classification, we use the non-penalized logistic regression model to train the data. With the learning rate is 2 and the k fold is 5, the BER of this model is around 14.45%.

5 Random Forest

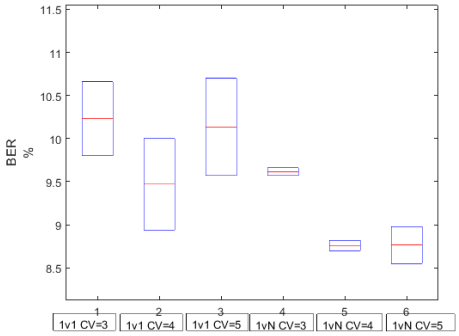
In machine learning, random forest works as a large collection of decorrelated decision trees. Its output classes are decided by voting of individual trees. Compared with single decision tree algorithm, it is better at classification and prediction, and it is less possible to overfitting.
In our experiment, we continuously add the number of trees from 100 to 1000, and observe the change of BER. It is more likely we can get the better result form increasing the result. However, the improvement decreases as the number of trees increases, i.e. at a certain point the benefit in prediction performance from learning more trees will be lower than the cost in computation time for learning these additional trees. As Figure 3(a) shows, if the number of trees is too small, the error is very high for the overfitting of most tree. When the number is 100, the BER is as high as up to 15% . As the number of trees grows, overfitting is reduced and the BER Error decreases. When the number reaches 700, the BER already drops to 11%. Afterwards it is meaningless to increase the number of trees as the BER doesn't decrease even the number of trees reaches 1000 for the limitation of random forest model.

6 SVM

SVM is a efficient classification model based on maximum margin method.



(a) BER over different box-constraint



(b) BER of 1 versus 1 and 1 versus N

Figure 4: BER of 1 versus 1 and 1 versus N and BER over different box-constraint

6.1 Binary Classification

SVM can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. Basically, three kinds of kernel is widely used in Support Vector Machine. The first kernel is the linear kernel. The second is the polynomial kernel, which represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models. Intuitively, the polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these. In the context of regression analysis, such combinations are known as interaction features. The third is the gaussian kernel. The gaussian kernel is an example of radial basis function kernel

$$k(x, y) = \exp(-\frac{||x - y||^2}{2\sigma^2})$$

which can map input to infinity dimension.
Using the Hog feature, as we can see from Figure 3(b), the BER of guassian kernel is lower than linear for guassian kernel can produce more complex boundary, thus it can fit into more situations and reduce error. Among gaussian, polynomial and linear kernels, gaussian is the best and its BER can be as low as 25%.
Using CNN feature, we compare the effort of box constraint to BER. From Figure 4(a) we see that as we increase the box constraint which is related to the penalty for the data that falls in marginal range, it is very obvious the BER of gaussian kernel and polynomial kernel drop significantly. While the box constrain almost has no obvious impact on the BER of linear kernel. The main reason is that when the box constrain is smaller, gaussian kernel and polynomial kernel is more likely to fit untypical data and outliers to form completed boundary. When the box constraint increase, the penalty of the point that lie in boundary will increase, then gaussian kernel and polynomial kernel will change the boundary to reduce the impact of outliers.

6.2 MultiClass Classification

We also use the multiclass support vector machine model. We test the different kernels, such as linear kernel, polynomial kernel and gaussian kernel. There are two strategies to train multi-classification problem. The most common technique in practice has been to build K one-versus-rest classifiers. Another strategy is to build a set of one-versus-one classifiers, and to choose the class that is selected by the most classifiers. While this involves building $\frac{K(K-1)}{2}$ classifiers. We use "fitcecoc" which uses $\frac{K(K-1)}{2}$ binary support vector machine models and the one-versus-one coding design in matlab, where K is the number of unique class labels. And we implement the one versus rest by ourselves. We will compare these two method.

6.2.1 1 Versus 1

Firstly we split data into 5 fold and normalized it, then we use the function "fitcecoc" provided by matlab. The process of 'fitcecoc' as fellows, it trains $K(K-1)/2$ binary classifiers for a K-way multiclass problem. Each receives the samples of a pair of classes from the original training set, and must learn to distinguish these two classes. At prediction time, a voting scheme is applied: all $K(K-1)/2$ classifiers are applied to an unseen sample and the class that got the highest number of "+1" predictions gets predicted by the combined classifier.

6.2.2 1 Versus rest

We implements this method by ourselves. The one versus rest strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a class label. Discrete class labels alone can lead to ambiguities, where multiple classes are predicted for a single sample. Although this strategy is popular, it is a heuristic that suffers from several problems. Firstly, the scale of the confidence values may differ between the binary classifiers. Second, even if the class distribution is balanced in the training set, the binary classification learners see unbalanced distributions because typically the set of negatives they see is much larger than the set of positives. But the model is less time consuming compared the 1 versus 1 strategy.

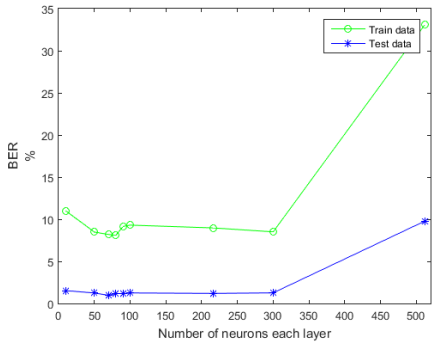
While in our experiment, the result of this model is show in 4(b). The 1 versus N has better performance that 1 versus 1 for the k fold is 3,4,5. One possible reason for this is the implements of 'fitcecoc' do not choice the suitable parameter, such as the kernel scale and this function do not standardize the input data.

7 Neural Network

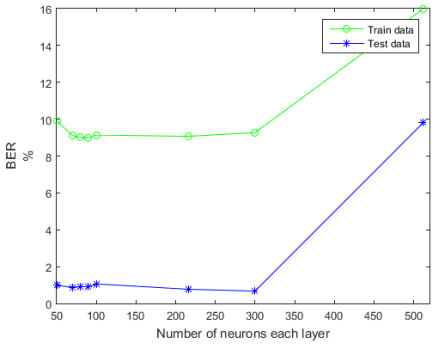
In the section, we will use two kind of neural network, MLP and CNN which is widely used in image recognition. And we use the autoencoder technique to get a better model. We will discuss the effect of different parameters. Admittedly, there are not solid mathematical skills about how to choose the number of layer and the neuron of each layer. But from testing the different parameters of the network, we can observe the trend of the error. And it will give us some clues about how to adjust these parameters for a better accuracy.

7.1 Multilayer Perceptron

Multilayer perception contains multiple layers of calculation. Compared with single layer perceptrons, there are multiple output ends and there is a hidden layer between input end and output end. We use Hyperbolic Tangent function as our activation functions. It is a feed-forward network, in which information flows left-to-right. We have input observed features, then we compute hidden nodes with activation functions, then we compute next layer and so on. The update method of this network can be divided into two phases: propagation and weight update.
Phase a: propagation. Each propagation involves the following steps: 1. Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations. 2. Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas (the difference between the input and output values) of all output and hidden neurons.
Phase b: weight update. For each weight-synapse follow the following steps: 1. multiply its output delta and input activation to get the gradient of the weight. 2. subtract a ratio of the gradient from the weight. This ratio influences the speed and quality of learning: it is called the learning rate. The greater the ratio, the faster the neuron trains; the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates where the error is increasing, this is why the weight must be updated in the opposite direction.
Repeat phase 1 and 2 until the performance of the network is satisfactory. Accurately, there are many high efficient optimal method than SGD, such as ADAdelta[3], ADAgrad and Stochastic Spectral Descent[4]. But with the limited of deep learning tools in matlab, we can only use the SGD.



(a) BER of training data and test data for 20 iteration times



(b) BER of training data and test data for 50 iteration times

Figure 5: BER of training data and test data for 20 iteration times and 50 iteration times

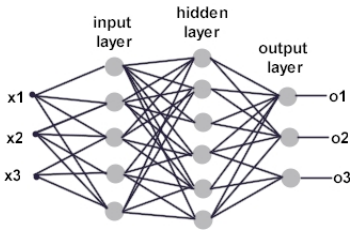


Figure 6: Multilayer Perceptron Architecture

First we setup the neural network, the first layer needs to have number of feature neurons. We set the iteration number to be 20. And the number of samples for mini-batch gradient descent is 100. Learning rate is 2. Our activation function is sigmoid function. We change the different parameters and record the BER with different parameters.

As Figure 5(a) and Figure 5(b) show, the number of hidden layer is 1, and we change the number of neurons in this layer, the result of 20 epoch is 5(a) while the result of 50 epoch is 5(b). We can see the BER drop down in the begin and increase later with the continuously increase of the neural numbers. And we can see when the number of the neurons is between 80 to 100, we can get a model with better performance. That mainly because the cnn feature extracted by the deep convolution network, and the dimension of feature is about 36864. The computation our computer without GUP is limited. So it unlikely get a correct model when training them in network with large number neurons in the hidden layer. The more weird thing is that when we increase the number neurons to more than 1024. All the test data is predicated with a same label. One possible reason is the limited of our computer and the lose of computation precise. So when the huge computation in matlab, some iteration is not executed correctly.
Also there is a significant drop when the number of epoch increase to 50 for much neuron number, the reason is that when the hidden layer become more completed, they always need more iterations to convergence.
We are eager to use the smaller learning rate and increase the number of epoch to get a better model. But it seems that the improvement of our model is very slight. Also the BER is not changed so much when we change our different activation function, from sigmod to tanh. About the output function, the model using sigmod is slightly better than the model using softmax as the output function.

7.2 Convolution Neural Network

In machine learning, convolution neural network is a type of feed-forward artificial network where the individual neurons are tiled in such a way that they respond to overlapping regions in the visual

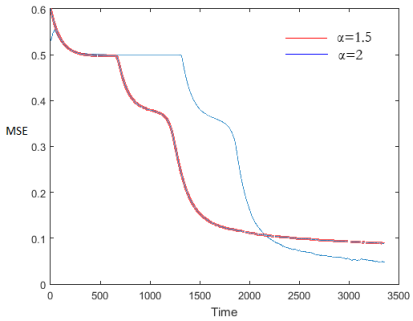


Figure 7: BER over time

field. First we have the input matrix of the image, then we apply different filters to the input matrix. Filters are just kernels of the matrix. Thus we get the layer 1. Different filters have different usages. For example, there are filters to detect vertical edges and horizontal edges. The size of matrix is reduced. Then we apply subsampling and the size of matrix reduces again. After multiple layers we have the final classification output.
In our experiment, our convolution neural network has two convolution layers and two sub-sampling layers. For the first convolution layer, the size of filter is 5×5. For the second convolution layer, the size of filter is 3×3. Limited by our computers, we set the channel number to be 1. From our experiment result we can see that the number of layers affect the prediction accuracy. As Figure 7 shows, for learing rate $\alpha = 2$ at the 45^{th} iteration, the MSE for full batch drops significantly. TAnd the BER of our CNN framework is around around 8.56%.

7.3 Autoencoder

In MLP network, the number of the layer and the neuron of each layer is random. Using autoencoder technique, we can train the number of neuron in each hidden layer firstly to reduce the information loss between two closed layer and then use the MLP network to train data. The process of this technique is as below, firstly we train SAE network to minimize the loss of information in each hidden layer. Then we use this SAE network to initialize each hidden layer of MLP.

Compared the MLP when the number of hidden layer is only 1 and the neurons of this hidden layer is small, the model which using auto encoding is not satisfied, for the BER is larger than normal MLP. Because the error of training SAE is too large, amostly not change in each iteration. So when we use the hidden layer in SAE to initial MLP, we choose a unsuitable one which is more likely to sink into local minimum. While we change many parameters for a better performance of MLP, such as changing the learning rate, epoch number and mini-batch size. So for simple hidden layer and small number of neurons, using auto encoder to decrease the loss of MLP is limited. While autoencoder method is especially useful when the number of hidden layer increase to 3 and more each neural applied in each hidden Layer.
Another important thing is that when the model is more complicated, the BER of the model is not always decrease, as it shown in fig 8. The model which with less BER use two hidden layer [512 32].

8 Model Compare

As the fig 9 shows, for binary classification, we find that the BER of SVM based on linear kernel is the lowest, which is 7.5%. Follows are MLP(8.5%) and random forest(13.57%). For multiclass classification, autoencoding MLP has the lowest BER(7.54%). Then CNN and MLP has approximately the same BER which is about 8%. Compared them, the BER is large in SVM model(9.3%), random forest(11.68%) and softmax. All of these model is the best model with lowest BER we have found in this method. And the cross validation of this model is 5 fold.

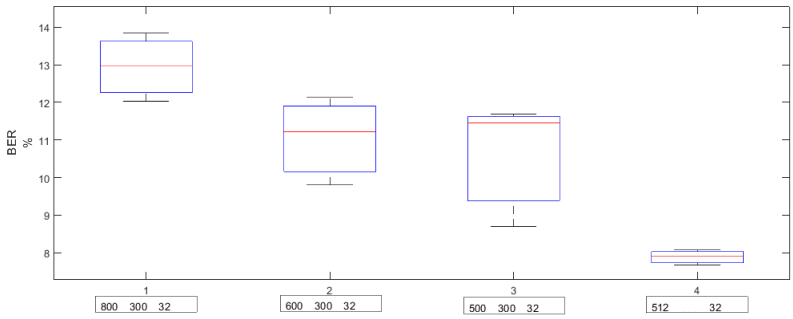


Figure 8: BER for different hidden layers

Model	SVM	MLP	Random Forest	Autoencoding MLP	CNN	Logistic Regression
Binary Classification	7.6%	8.58%	13.57%			14.45%
Multiclass Classification	9.39%	8.08%	11.69%	7.54%	8.29%	23.4%

Figure 9: BER for different models

Acknowledgments

References

[1] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[2] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[3] Duchi, J., Hazan, E., Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12, 2121-2159.

[4] Carlson, D. E., Collins, E., Hsieh, Y. P., Carin, L., Cevher, V. (2015). Preconditioned spectral descent for deep learning. In Advances in Neural Information Processing Systems (pp. 2953-2961).