CMSC 828G Project: Using LBP features for multiclass SVM classification

Rohan Chandra

(In continuation to project 1))

Abstract

Good results on image classification and retrieval using support vector machines (SVM) [1] with local binary patterns (LBPs) as features have been extensively reported in the literature [3] where an entire image is retrieved or classified. The primary aim of this project is to understand the theory behind LBPs and why they are used. An implementation of LBP using MATLAB has been shown here and recognition has been done by training multi class SVM's using the CIFAR-10 dataset [2].

Keywords: LBP, SVM, CIFAR-10

1. Introduction

The LBP feature vector [3], in its simplest form, is created in the following manner:

1.) Divide the examined window into cells (e.g. 16x16 pixels for each cell).

2.) For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.

3.) Where the center pixel's value is greater than the neighbor's value, write 0. Otherwise, write 1. This gives an 8-digit binary number (which is usually converted to decimal for convenience).

4.) Compute the histogram, over the cell, of the frequency of each number occurring (i.e., each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector. Optionally normalize the histogram.

5.) Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.

The feature vector can now be processed using the Support vector machine or some other machine-learning algorithm to classify images. Such classifiers can be used for face recognition or texture analysis.

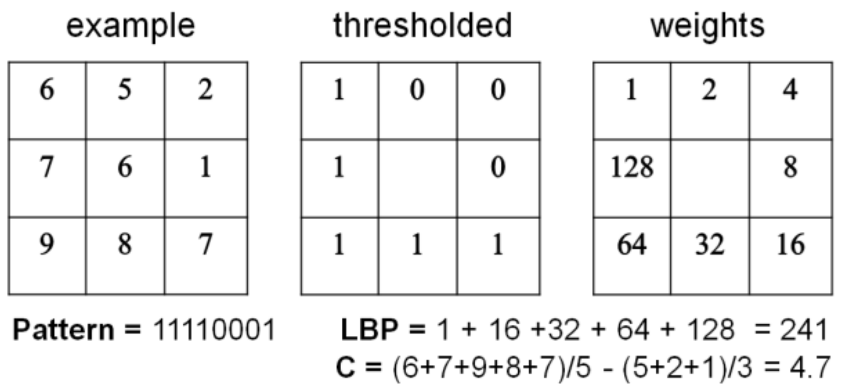

Figure 1: Extraction of LBP features

Support Vector Machines

Linear support vector machines are originally formulated for binary classification [1]. Given training data and its corresponding labels , (x_n, y_n), x=1,2..N, $x_n \in \mathbb{R}^D, t_n \in [-1, +1]$, SVM's learning consists of the following constrained optimization.

$$\underset{w,\xi_n}{\text{Min}} \quad 1/2 w^T w + C \sum_{n=1}^{N} \xi_n \quad s.t.$$

$$w^T x_n t_n \geq 1 - \xi_n \forall$$

$$\xi_n \forall n$$

ξ_n are slack variables which penalizes data points which violate the margin requirements. Note that we can include the bias by augment all data vectors

x_n with a scalar value of 1. The corresponding unconstrained optimization problem is the following:

$$\underset{w}{\text{Min}} \quad 1/2w^Tw + C\sum_{n=1}^{N} max(1 - w^Tx_nt_n, 0)$$

The objective of the above equation is known as the primal form problem of L1-SVM, with the standard hinge loss. Since L1-SVM is not differentiable, a popular variation is known as the L2-SVM which minimizes the squared hinge loss

For Kernal SVMs, optimization must be performed in the dual. However, scalability is a problem with Kernal SVMs, and in this paper we will be only using linear SVMs with standard deep learning models.

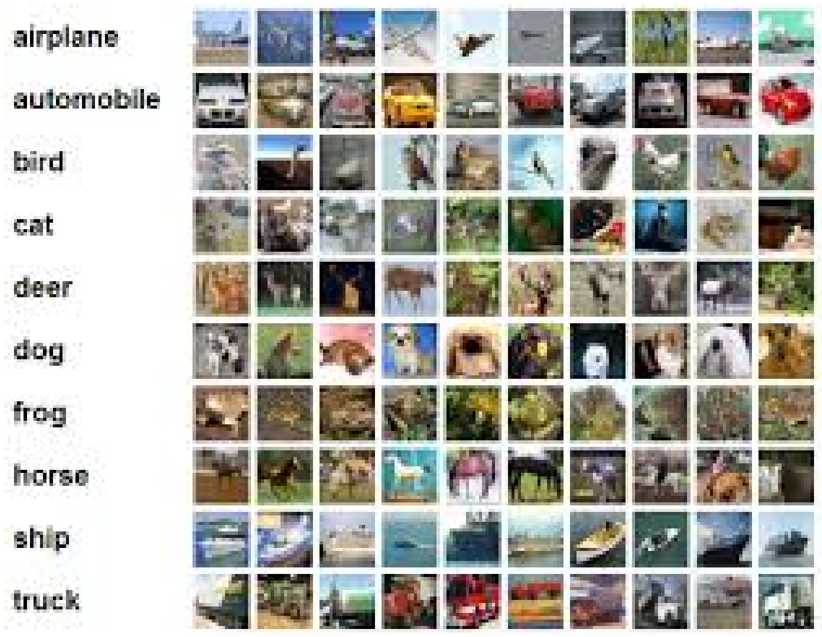

Figure 2: Multi-Class CIFAR-10 Dataset

Multi-Class SVM's

The simplest way to extend SVMs for multiclass problems is using the so-called one-vs-rest approach (Vapnik, 1995). For K class problems, K linear SVMs will be trained independently, where the data from the other classes form the negative cases. Hsu and Lin (2002) discusses other alternative multiclass SVM approaches, but we leave those to future work. Denoting the output of the k-th SVM as

$$a_k(x) = w^T x$$

The predicted class is

$$\arg\max_k \quad a_k(x)$$

Note that prediction using SVMs is exactly the same as using a softmax. The only difference between softmax and multiclass SVMs is in their objectives parametrized by all of the weight matrices W. Softmax layer minimizes cross-entropy or maximizes the log-likelihood, while SVMs simply try to find the maximum margin between data points of different classes.

The LBP code read in the images and produced feature vectors for each image. Various settings of the LBP code were allowed to answer the following questions:-

1. Is a higher dimensional feature vector more efficient than a lower dimensional one?
2. Would a normalised histogram perform better than an unnormalized one.

We used the LIBSVM MATLAB package for classification. This work addresses the following questions:

1. Which kernel would perform better for higher dimensional features and which kernel for lower dimensional?
2. How much cross-validation is allowed before over-fitting occurs?
3. Figuring out the appropriate cost value for our ideal model?

2. Experiments

The CIFAR-10 dataset consists of 60,000 images out of which 50,000 are training images and the rest are test data.The number of classes is 10. The images are 32 by 32 and packed in 5 training batches and 1 test batch each of 10,000 images with 1,000 images of each class. This batch format makes it easier in the pre-processing stage. However, much pre=processing of this dataset was still needed; the images were packed in 1-D vectors for compression and they needed to be blown up into proper images to be read by the LBP code.

The LBP code is maintained to be rotationally invariant. However, to answer the questions put forth in section 1, we changed the arguments for various settings. The LBP code takes in jpg, png, bmp file formats and the dataset was packed in 1-D uint8 vectors. So we wrote a script to facilitate easy parsing of the dataset for the LBP code.

Additionally, LIBSVM only accepts sparse training data of type double. Thus further preprocessing was achieved and the output of the LBP code was packaged in a format easily read by the LIBSVM code. We experimented with numerous settings of the code which are explained below:-

- -t kernel type : set type of kernel function (default 2)
  0 – linear: $U^T * V$
  1 – polynomial: $(*\gamma * U^T * V)$
  2 – radial basis function: $\exp(-\gamma * |u - v|^2)$
  3 – sigmoid: $\tanh(gamma * u' * v + coef0)$
  4 – precomputed kernel (kernel values in training_set_file)

- -c cost : set the parameter C of C-SVC, ϵ_SVR (default 1)
- -v n: n-fold cross validation mode
- -g gamma : set gamma in kernel function (default 1/num_features)

The svmtrain function returns a model which can be used for future prediction. It is a structure and is organized as Parameters, nr_class, totalSV, rho, Label, ProbA, ProbB, nSV, sv_coef, SVs

The function svmpredict has three outputs. The first one, predictd_label, is a vector of predicted labels. The second output, accuracy, is a vector including accuracy (for classification), mean squared error, and squared correlation coefficient (for regression). The third is a matrix containing decision values or probability estimates (if -b 1 is specified). If k is the number of classes in training data, for decision values, each row includes results of predicting k(k-1)/2 binary-class SVMs. For classification, k = 1 is a special case. Decision value +1 is returned for each testing instance, instead of an empty vector. For probabilities, each row contains k values indicating the probability that the testing instance is in each class. Note that the order of classes here is the same as Label field in the model structure.

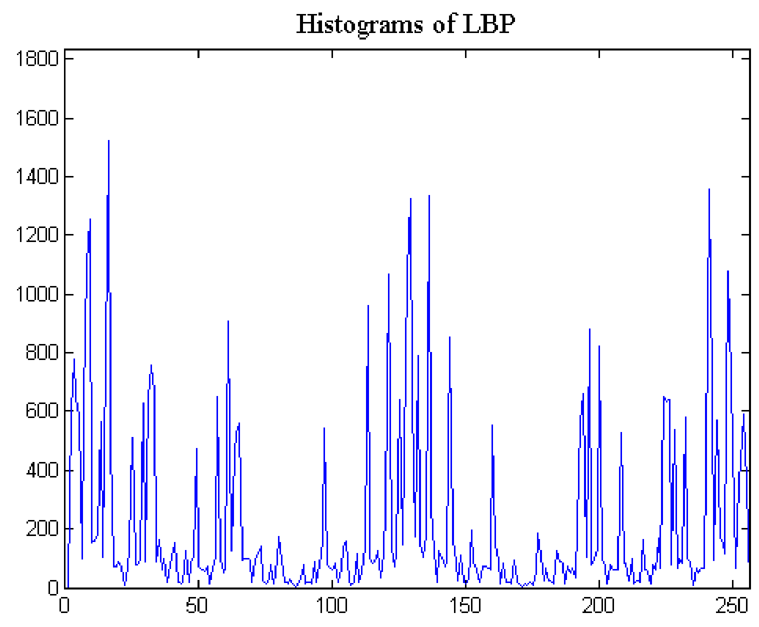

Figure 3: Format of output Histogram

Unnormalised histograms

Various LBP settings were employed. In order to answer the question on normalisation, we trained two models for an unnormalised LBP histogram and 3 models for a normalised histogram. In the unnormalised case, we used an RBF kernel for the first model at a cost of 1 and a gamma of 0.07. The radius was set to 1 and the number of sampling points was 8. For the second model, we switched to a linear kernel with all the LBP parameters remaining same except for the cost which we set to 50. Additionally, we used a 5-fold cross-validation model which took a lot of time.

Normalised histograms

Testing was switched using normalised histograms. Parameter palettes of 8-point, 16-point and 20-point LBP's were used to check for efficiency of higher dimensional feature vectors. After usage of the RBF kernel in the unnormalised experiment, We tried a sigmoidal (tanh) kernel here but performance was found to be similar. Different cost values were assumed and a range was calculated and experimented with.

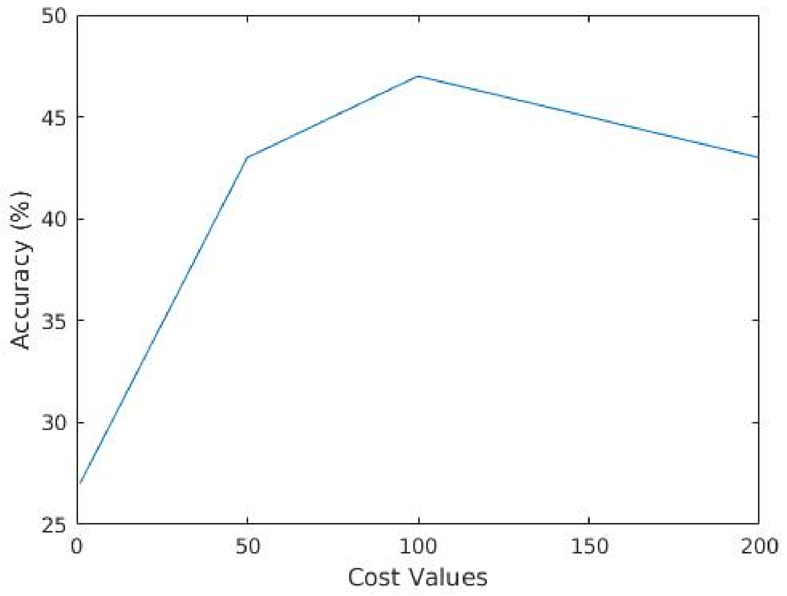

Figure 4: Accuracy vs Cost values

3. Results

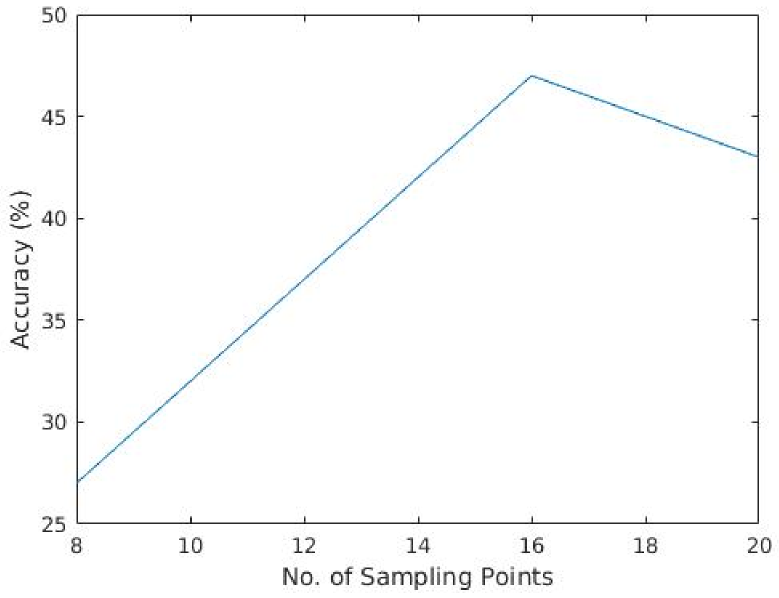

Figure 5: Accuracy vs No. of Sampling Points

It is found that a normalised feature vector performs better in general and gives a much wider testbed for further improvements. It gives similar accuracies of 43% for tanh and sigmoidal kernel functions as well as for the linear kernel with a sampling point of 8. But we have a clear winner with a 16-point sampling range and a linear kernel with a cost of 100 genrating an accuracy of 47% and a mean squared error of 9.8% as opposed to the increased 16.55% that comes with using an RBF kernel. A cost of > 100 and a sampling point parameter of > 16 performances sub-optimally at 43/46% accuracy.

Kernel	**Cost_opt/Points_opt**	**Accuracy/MSE/Sq Correlation Coefficient**
RBF (gamma 0.07)	1/8	27%/16.55/0.03
Sigmoidal	50/8	43%/-/-(Cross-Validation)
Linear	100/16	47%/9.85/0.15

Table 1: Performance of various kernels used

Additionallly, table one shows that at the specified settings, the linear kernel outperforms the RBF and the sigmoidal kernels. Further testing is required to determine if a larger radius could have a positive effect on the RBF kernel with an improved function for γ

Comparison with deep learning techniques shows that CNN's outperform SVM's using LBP features by $\tilde{3}0\%$. This is evident by the fact that even the most sub-optimally trained model at its worse performs at an accuracy of 53%. With the most optimally trained CNN performing at 73%, it is clear that CNN's are the way to go and will dominate the field of computer vision for the forseeable future.

4. References

[1] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[2] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[3] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.