Handwritten Digits Classification by Support Vector Machine

Huihuang Zheng
UTEID: hz4674
huihuang@utexas.edu

Abstract

Support Vector Machine (SVM) [2] is a powerful technic in machine learning. In this work, we explore what factors can influence the accuracy of SVM in handwritten digit classification accuracy. We found that Principal Component Analysis (PCA) [4], applying image features (e.g. Histogram of Histograms of Oriented Gradients, HOG [3]), increasing the number of Eigenvectors, and increasing size of training data can significantly improve the accuracy. We also find training images used for PCA doesn't effect the SVM accuracy much. Finally, we found SVM kernel types can effect the SVM accuracy and number of images need to be used for training.

1. Implementation

1.1. Pre-required Software and Tools

In this work, we used Matlab. Our SVM tool is from LIBSVM [1] and we used library VLFeat [5] for computing image feature Histograms of Oriented Gradients (HOG) [3]. LIBSVM has executable files and we attached them in **./src** so you don't need to install them. But you need to have Matlab in your computer and install VLFeat to your Matlab.

1.2. Implementation Details

1.2.1 Pre-computing Image Features

see **./src/precomputeImageFeatures.m**. Pre-compute image features and store them into file will make experiments faster because you don't need to re-compute image features every time you run an experiment. This function just calls interface of VLFeat [5].

1.2.2 Baseline Experiments

see **./src/baselineExperiment.m**. We use SVM on raw image data, without PCA, linear kernel: u'*v to run experiment as baseline. The function changes the sizes used for training SVM and call LIBSVM interface: libsvmtrain, libsvmpredict to train SVM and get accuracy.

1.2.3 SVM Experiments interface

see **./src/svmExperiment.m**. The function firstly converts the image into features (by loading pre-computed features or compute now, in this work, we computed feature of HOG [3] using cell size 4), then pick exact number of images as SVM training set. In the SVM data, the function pick specific number of images as PCA training set. The pick method is change first n images. Then, it's PCA step. We call the code in homework1 to get eigenvectors and mean of PCA training data. Next, we project training and testing data to eigenvectors space (subtract mean from training and testing data and then multiply the eigenvectors). After we have done PCA, we convert training data into LIBSVM input format and rescale training and testing data to range $[0, 1]$. Finally, we call LIBSVM interface: libsvmtrain, libsvmpredict to train and get accuracy and write results to output file.

1.2.4 Main Function

see **./src/main/m**. The function specified the path for data files, the different parameters we used for experiments and run the **baselineExperiment.m** and **svmExperiment.m** with different parameters.

1.3. Run my code

see **./readme.txt** for running instructions. Note that it takes a few minutes to train and test by SVM. Running all experiments in **./src/main/m** costs about 1.5 hour. (not including pre-computing of features). Pre-computing features take about 3 hours. I attached pre-computing features in uploaded code so you don't need to run pre-computing feature step.

2. Experiment

To see how different parameters influence accuracy of SVM classification, we fixed some parameters and changed some parameters.

Using PCA	Img Feature	Train Size	Accuracy
no	raw	1000	0.0974
no	raw	10000	0.0974
no	raw	60000	0.0974
yes	raw	1000	0.8587
yes	raw	10000	0.9082
yes	raw	60000	0.9244
yes	hog	1000	0.9474
yea	hog	10000	0.9737
yes	hog	60000	0.9796

Table 1. Impact of PCA and HOG feature

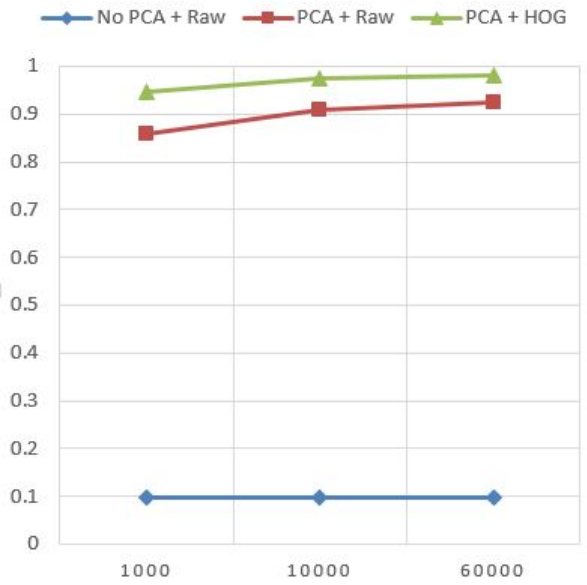

Figure 1. Impact of PCA and HOG feature

2.1. How do PCA and Image Features and Training Size of SVM influence the SVM accuracy?

Our experiment result is presented in the following table. The Img Feature can be hog (HOG [3]) or raw (raw image). The experiment uses linear kernel of SVM ($x' * y$), 600 images to train PCA and use 50 eigenvectors. Table 1 and Figure 1 show impact of PCA and HOG features. We could see that without PCA and HOG feature, we simple apply SVM in the images just got 9.74% accuracy. Even random guess can get 10% accuracy, so SVM without PCA cannot be applied to images. Then, by comparing PCA + Raw and PCA + HOG, we could find the HOG feature improve a lot of SVM accuracy. Since PCA and HOG can help a lot, all my following experiments will use PCA + HOG.

2.2. How does training size of PCA influence the SVM accuracy?

In this experiment, we use PCA + HOG, 60000 images to train SVM with linear kernel ($x' * y$), different numbers of images to train PCA and using first 50 eigenvectors. The result is presented in Table 2. From the result, we could see that with increasing sizes of PCA training, the accuracy of SVM can increase or decrease. So once you train PCA enough, you don't need a lot of images to run PCA because increasing number of training doesn't always have positive effect.

PCA Training Size	Accuracy
250	0.9815
300	0.9786
350	0.9781
400	0.9725
450	0.9775
500	0.977
550	0.9823
600	0.9796

Table 2. Impact of PCA training size

Number of Eigenvectors	Accuracy
10	0.9241
20	0.9666
30	0.9769
40	0.978
50	0.9796
60	0.9822
70	0.982
80	0.9841

Table 3. Impact of number of eigenvectors

2.3. How does number of eigenvectors influence the SVM accuracy?

In this experiment, we use PCA + HOG, 60000 images to train SVM with linear kernel ($x' * y$), 600 images to train PCA but use different number of eigenvectors. We increased number of eigenvectors we used from 10 to 80 with increment of 10. We could see using more eigenvectors in PCA can increase the SVM accuracy.

2.4. Different Kernels

In this experiment, we compare four types of kernels. 1, linear: $x' * y$. 2, polynomial: $(x' * y)^3$. 3, radial basis: $\exp(-\gamma * |x - y|^2)$. 4, sigmoid: $\tanh(\gamma * x' * y)$. In radial basis and sigmoid kernels $\gamma = 1/number\_of\_features$. The result was presented in Table 4 and Figure 3. Although when we use all 60000 training images, those kernels can all achieve accuracy between 96% and 98%, the polynomial and sigmoid kernels can not got high accuracy when we only use 1000 training data. So you need more data to let polynomial and sigmoid kernels get high accuracy. So we can see different kind of kernels not only influence accuracy

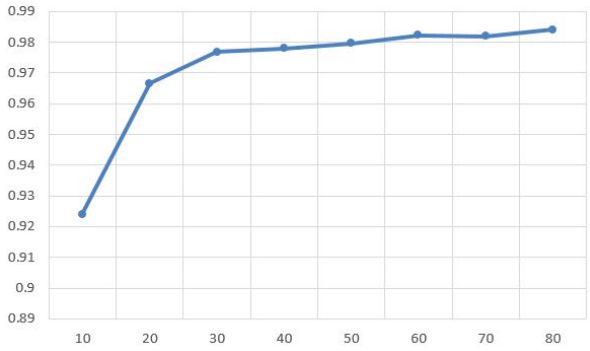

Figure 2. Impact of number of eigenvectors

Kernel	SVM Train Size	Accuracy
linear	1000	0.9474
linear	10000	0.9737
linear	60000	0.9796
polynomial	1000	0.1028
polynomial	10000	0.8965
polynomial	60000	0.9642
radial	1000	0.9105
radial	10000	0.9654
radial	60000	0.9755
sigmoid	1000	0.4359
sigmoid	10000	0.9611
sigmoid	60000	0.9729

Table 4. Impact of Different Kernels

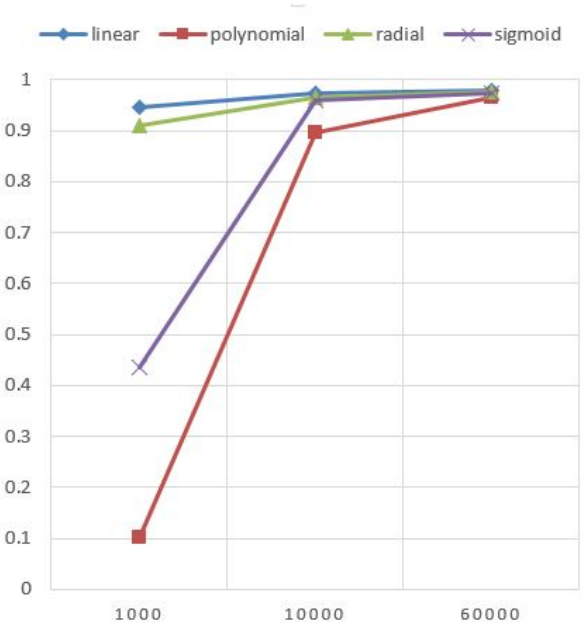

Figure 3. Impact of Different Kernels

but also how many data you need to train the SVM.

3. Conclusion

In this work, we explore what factors can influence the accuracy of SVM in handwritten digit classification accuracy. We found that Principal Component Analysis (PCA) [4], applying image features (e.g. Histogram of Histograms of Oriented Gradients, HOG [3]), increasing the number of Eigenvectors, and increasing size of training data can significantly improve the accuracy. We also find training images used for PCA doesn't effect the SVM accuracy much. Finally, we found SVM kernel types can effect the SVM accuracy and number of images need to be used for training.

References

[1] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[4] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[5] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2008.