Explorations in Handwritten Digit Classification Using the MNIST Dataset

Goutham, Revant, Abhimanyu, Piyush, Kishan
under the guidance of
Prof. Piyush Rai

IIT KANPUR

18-11-2016

Outline

Problem Statement

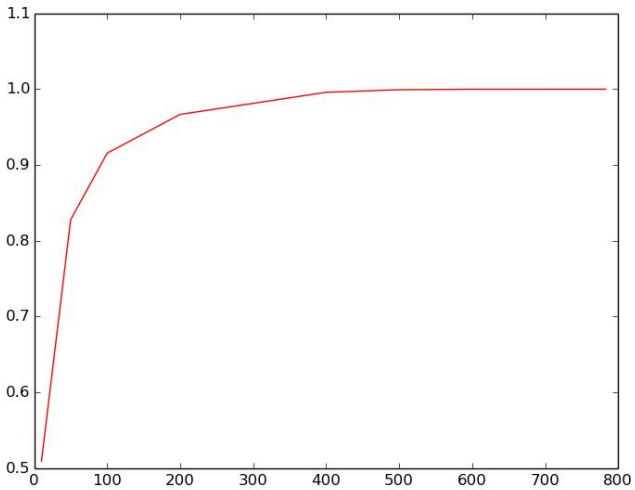
Explorations in Handwritten Digit Classification: Try out different learning algorithms and/or combinations thereof, to see what's the best classification accuracy you can achieve on the test data of Handwritten Digit Recognisation problem using the MNIST dataset.

Outline

Dimensionality Reduction: PCA

After doing eigen decomposition of the covariance matrix we can see from the figure that **82.8%** of variance is covered by **top 50 eigen vectors** and **91.5%** by **top 100 eigen vectors**. Further **we ran 1-NN** on reduced dimensions and saw the accuracies. We can observe from the table that after the first 50 eigen values the remaining values are negligible compared to the top 50 values. So we are reducing to 50 dimensions.

Here onwards when we do PCA, that means we are reducing it to 50 dimensions from 784



Dimensions	Accuracy
50	97.73%
75	97.51%
100	97.4%

Outline

Methods considered/tried

The methods that we considered include

- SVM
- K-NN
- Logistic Regression
- Random Forest
- LeNet
- Disturbed LeNet

On the first 4 models we did parameter estimation by using cross validation or doing random sampling. The last two methods use neural networks for their working. In last method we implemented a recent research paper, and analysed the problem of overfitting in LeNet's CNN.

Methods: SVM and K-NN

SVM :We worked on **Linear, RBF kernels** and we also did hyperparameter tuning. To create the model we normalized the data such that **mean=0** and **co-variance=1** for each feature. Since top 50 eigen values are the major ones, hyperparameter tuning is done on the pre-processed dimensions, hyperparameters are tuned by cross-validation.

K-NN :Here we are finding the best K by testing the model on the **10%** random data from the training data. Model is trained with the remaining **90%** data.

Results: SVM

- **Without PCA**
  - **Linear Kernel**: Classifier took **482** seconds to run with the default c value(c = 1) and we got an accuracy of **91%** on test data.
  - **RBF Kernel**: Classifier took **782** seconds to run with the default c value(c = 1) and got an accuracy of **96.5%** on test data.
- **With PCA**

Table: Accuracy and time for different values of c(penalty for wrong prediction) on training data using linear kernel

c	Accuracy	Time	Accuracy(RBF)	Time(RBF)
1	89.5%	50.5 seconds	98.38%	79.66 seconds
2	89.5%	59.1 seconds	98.56%	70 seconds
5	89.5%	68.6 seconds	98.55%	76 seconds
10	88.5%	78.25 seconds	98.5%	73 seconds
15	87.16%	82 seconds	98.53%	78 seconds

Results: K-NN

- **Without PCA**:The value of K that gives the best results is **K = 5** and this K gives an accuracy of **96.8%**.
- **With PCA**:The value of K that gives the best results is **K = 5** and this K gives an accuracy of **97.55%**.

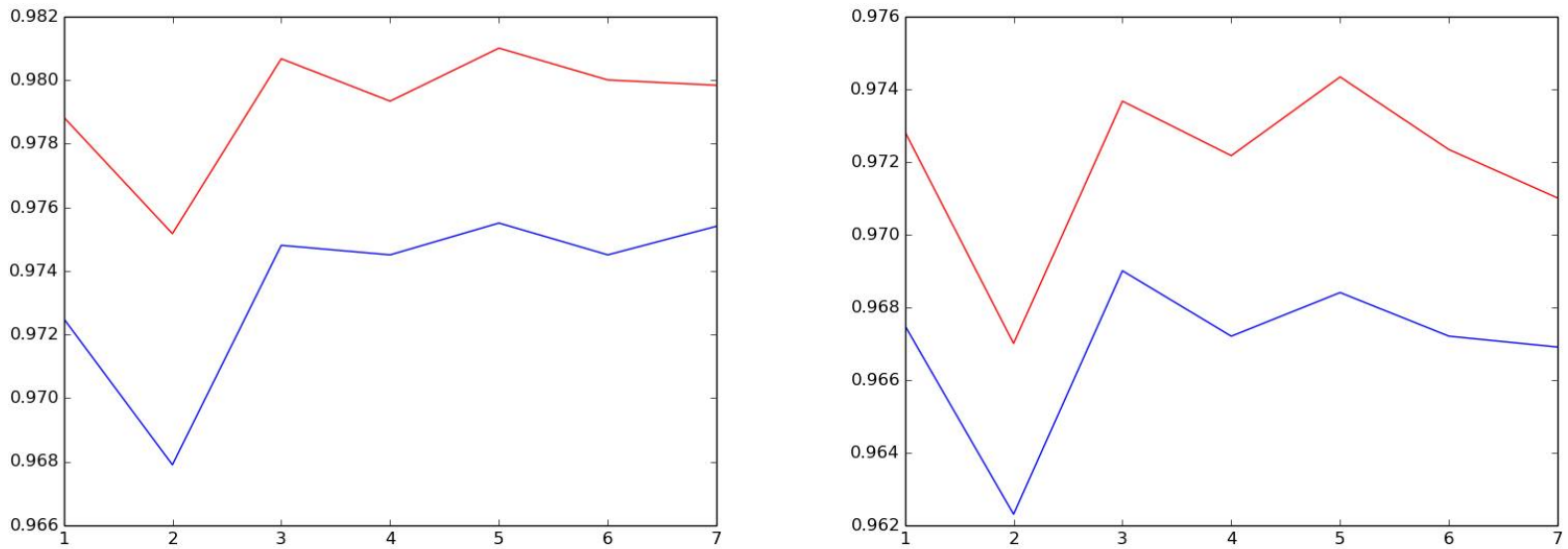


Figure: Accuracy versus value of K

Methods: Random Forest and Logistic Regression

Random Forest : In this we first run the algorithm using **cross_validation** to first find the number of trees that we would like to uses, the number of optimal trees are selected both on the basis of the accuracy that they give on the training data and time taken.

Logistic Regression : Logistic Regression is pretty straight forward and we don't need to do much parameter estimation, so here we just do Logistic Regression with all the training data and also we use PCA for dimensionality reduction and then run logistic regression again.

Results: Random Forest

- **Without PCA**: Number of tree that we used for is **50** and the accuracy that we got is **96.79%**
- **With PCA**: Number of tree that we used for is **50** and the accuracy that we got is **95.12%**

Table: Accuracy and time for different values of trees for both with and without PCA

No.of trees	Accuracy	Time	Accuracy	Time(PCA)
10	94.75%	45.00 seconds	92.22%	53.25 seconds
15	95.47%	64.85 seconds	93.32%	78.18 seconds
20	95.91%	149.04 seconds	93.89%	100.04 seconds
30	96.32%	290.62 seconds	94.53%	150.87 seconds
50	96.70%	433.87 seconds	94.97%	254.66 seconds

Results: Logistic Regression

- **Without PCA**: In this case we get an accuracy of **91.75%** and the total time taken by the algorithm to fit the training data and run on the test data is **3557.36** seconds
- **With PCA**: this model we get an accuracy of **90.48%** and the time taken to execute the program is **155.00** seconds.

Method: LeNet

- Applying neural networks to MNIST generally faces the problem of overfitting the data, even LeNet without any kind of regulaliser or other method, overfits the MNIST dataset
- We implemented two variations of LeNet, in the first implementation we handled the overfitting using the dropout method and in the second implementation we handled the overfitting using a recent method call Disturb Label.
- Disturb Label randomly chooses a small subset of data and corrupts its labels. This leads to better generalization of the model.

Method: LeNet

- We first experimented with LeNet(Simple CNN) with a dropout and for this we got an accuracy of around 99.2%
- One of the tweaks that we made was to normalize the data so that we have mean $= 0$ and standard deviation $= 1$
- Further we changed the optimiser to use a modified version of SGD
- We also worked on a modified version of LeNet called the Disturb LeNet.
- In disturb lenet we randomly selects a small subset of samples and randomly sets their ground-truth labels to be incorrect

Results: LeNet

Table: Accuracy for different values of epochs and methods

epochs	Basic 128	tweaks 64	tweaks 128	tweaks 256
5	97.94%	99.12%	99.23%	98.99%
10	98.65%	99.39%	99.31%	99.18%
15	98.96%	99.36%	99.32%	99.26%
20	99.03%	99.46%	99.45	99.41%
25	99.16%	99.46%	99.52%	99.43%

Results: Disturb LeNet

Table: Accuracy for different values of epochs

epochs	Accuracy
5	98.99%
10	99.23%
15	99.36%
20	99.28%
25	99.40%

Outline

Inferences and Comparisons

- By far the best accuracy is given by the CNN method, LeNet, and it's modification, Disturb LeNet. This is because CNN models kind of hard codes all the training data which makes it fit the model really well and help in making really accurate predictions.
- KNN is very sensitive to bad features so feature selection is also important. This is mostly the reason why KNN performs better when we use PCA.
- In the case of SVM we see that RBF kernel gives way better accuracy than linear kernel. This might be because the border might not be linear but more complicated and this complicated border is found out by RBF kernel

Inferences and Comparisons

- We observe that among all the methods that don't use neural networks RBF kernel fits the data best and gives the best predictions, this is to be expected because RBF kernel is able to learn really complicated borders
- We observe that SVM perform better with PCA this is justified because on applying PCA we avoid the directions in which we have low variance which in turn makes the points more separable and hence we can find a border which better fits the model.
- In general, random forests can run on large data sets without problems. So doing PCA is not going to be of much advantage. In Random Forest we observe a pattern where the accuracy decrease with PCA.

Outline

What we learned

- We implemented and analysed various methods discussed in class.
- In the process of deciding upon a state of the art method to implement, we learnt about several methods in the literature.
- We were introduced to the basics of Deep Learning.
- We learnt in detail about CNNs and LeNet.
- Finally, we learnt about the various methods used to avoid overfitting and implemented DisturbLabel.