HOMEWORK 2 EXPERIMENT RESULTS

SJSU Students

1. You should keep your testing and training sets separate. Your test set can be relatively small, say 1000 items.
2. In one experiment I want you to vary just the amount of data trained on from say 1000, 5000, 10000, 15000, 20000, while training only for Clubs, and fixing everything else. What is the effect of the training set size on accuracy of the trained model?
3. The second experiment should compare for the same amount of training data how accurate an SVM for C, D, H, S will be. Which symbols are the easiest/hardest to recognize? Why?
4. The third experiment should try to vary the tweakable parameters at the top of *suits_generator.py* and see the effect of increasing distortion on the training.
5. The last experiment should vary ε and see the effect of this on the accuracy of the trained models.

EXPERIMENT 1:

To vary just the amount of data trained on from say 1000, 5000, 10000, 15000, 20000, while training only for Clubs, and fixing everything else. What is the effect of the training set size on accuracy of the trained model?

DESCRIPTION:

We have used variable training set sizes from 1000 to 20000. The test set is of size 1000 for every run. The epsilon value is set to 0.01 and the number of update steps is 10000. The model used is clubs and the model file is named clubs_model.txt.

RUNS:

Training Size	Test Size	Class	Final Result
1000	1000	C	Correct: 74.2% False positive:0% False negative:25.8%

5000	1000	C	Correct: 75.1% False positive:0.0% False negative:24.9%
10000	1000	C	Correct: 75.3% False positive:0% False negative:24.7%
15000	1000	C	Correct:23.6% False positive:76.4% False negative:0%
20000	1000	C	Correct:75.3% False positive:0% False negative:24.7%

INFERENCE:

The accuracy was around 75% for all the training set sizes but an exception to this trend was the training set size of 15000. For the 15000 training set, we also found false positive results which were not found in other experiments. Sometimes increasing the training size affects the SVM performance negatively and the model may have undergone overfitting for that particular training set. Increasing the training size did not seem to have any effect on the accuracy of the model.

EXPERIMENT 2:

To compare for the same amount of training data how accurate an SVM for C, D, H, S will be. Which symbols are the easiest/hardest to recognize? Why?

DESCRIPTION:

We have used a fixed training set of size 10000 and a test set of size 1000 for every run. The epsilon value is set to 0.01 and the number of update steps is 10000. We ran the training program for classes C, D,H and S.

RUNS:

Training Size	Test Size	Class	Final Result
10000	1000	C	Correct:74.9% False positive:0% False negative:25.1%

10000	1000	D	Correct:74.5% False positive:0 % False negative:25.5%
10000	1000	H	Correct:75.2% False positive:0% False negative:24.8%
10000	1000	S	Correct:75.4% False positive:0% False negative:24.6%

INFERENCE:

The accuracy for all the symbols is roughly around 75% but it is slightly less for diamonds making it difficult to identify. Spades symbol has shown the highest accuracy of 75.4% and is the easiest to detect.Diamond is hard to detect because it is a simple shape and it's vertex portions resemble the top portion of spades and bottom portions of hearts. Spade is the easiest to detect, although it is odd since spade does resemble heart in a way.It might have been easier to tell apart a spade from a heart because we never inverted any shape in our experiment.

EXPERIMENT 3:

To vary the tweakable parameters at the top of *suits_generator.py* and see the effect of increasing distortion on the training.

DESCRIPTION:

The data size used for this experiment is 1000. There are 300 images in the test folder. The epsilon value is set to 0.01 and the number of update steps is set to 20000. The model used is clubs_model.txt. We have conducted four different runs with varied parameters.

RESULTS:

1. For the first case, the tweakable parameters are varied as follows ->

   position=1
   size = 5
   thickness = 10

angle_min=-20
angle_max=20
stray_number=2
stray_size=3

These are the initial parameters set. The accuracy obtained is 76%, with 24% of the data being classified as false negative and 0% as false positive.

2. For the second case, the tweakable parameters are varied as follows ->

   position=1
   size = 5
   thickness = 10
   angle_min=-40
   angle_max=40
   stray_number=2
   stray_size=3

   For this case, we have increased the range of the orientation of the symbol from 20 degrees to 40 degrees.The accuracy obtained is 71%, with 29% of the data being classified as false negative and 0% as false positive.

3. For the third case, the tweakable parameters are varied as follows ->

   position=1
   size = 7
   thickness = 10
   angle_min=-20
   angle_max=20
   stray_number=3
   stray_size=4

   For this case, the number and size of stray marks are increased. The size of the symbol is also increased. The accuracy obtained is 78.34%, with 21.66% of the data being classified as false negative and 0% as false positive.

4. For the last case, the tweakable parameters are varied as follows ->

   position=2
   size = 7

thickness = 15
angle_min=-40
angle_max=40
stray_number=3
stray_size=5

For this case, every parameter is increased by a certain value. The accuracy obtained is 22.6%, with 77.33% of the data being classified as false negative and 0% as false positive.

INFERENCE:

From the experimental results it can be inferred that, increasing the orientation range resulted in decreased accuracy of 71%. However, increasing the size of the symbol and the number and size of the stray marks had a positive impact on the accuracy. It increased to 78.34%. Increasing all the parameters by some value had a drastic negative impact on the accuracy of the trained model, as it dropped to 22.6%. Changing these parameters have both positive and negative effects on the accuracy of the trained model.

EXPERIMENT 4:

Vary ε and see the effect of this on the accuracy of the trained models.

DESCRIPTION:

The data size used for this experiment is 1000. There are 300 images in the test folder. The number of update steps is set to 20000. The model used is clubs_model.txt. We have conducted four runs and the epsilon value is varied from 0.01 to 0.8 in different increments.

RESULTS:

1. For the first case, the epsilon value is set to 0.01. The accuracy obtained is 76%, with 24% of the data being classified as false negative and 0% as false positive.

2. For the second case, the epsilon value is set to 0.1. The accuracy obtained is 76%, with 24% of the data being classified as false negative and 0% as false positive.

3. For the third case, the epsilon value is set to 0.5. The accuracy obtained is 76%, with 24% of the data being classified as false negative and 0% as false positive.

4. For the fourth case, the epsilon value is set to 0.8. The accuracy obtained is 76%, with 24% of the data being classified as false negative and 0% as false positive.

INFERENCE:

As seen in the experimental results, any change in the epsilon value does not have any effect on the accuracy of the trained model as it is 76% for every case. Hence, it can be inferred that changing the epsilon value does not impact the accuracy of the model.