Using Synthetic Data Safely in Classification

Jean Nonnemaker, Henry S. Baird

Computer Science & Engineering Dept
Lehigh University
19 Memorial Dr West
Bethlehem, PA 18017 USA

E-mail: jen2@lehigh.edu

ABSTRACT

When is it safe to use synthetic data in supervised classification? Trainable classifier technologies require large representative training sets consisting of samples labeled with their true class. Acquiring such training sets is difficult and costly. One way to alleviate this problem is to enlarge training sets by generating artificial, synthetic samples. Of course this immediately raises many questions, perhaps the first being "Why should we trust artificially generated data to be an accurate representative of the real distributions?" Other questions include "When will training on synthetic data work as well as - or better than training on real data ?".

We distinguish between sample space (the set of real samples), generator space (all samples that can be generated synthetically), and finally, feature space (the set of samples in terms of finite numerical values). In this paper, we discuss a series of experiments, in which we produced synthetic data in generator space, that is, by convex interpolation among the generating parameters for samples and showed we could amplify real data to produce a classifier that is as accurate as a classifier trained on real data. Specifically, we have explored the feasibility of varying the generating parameters for Knuth's Metafont system to see if previously unseen fonts could also be recognized.

1. INTRODUCTION

It has been widely accepted in pattern recognition research that the classifier trained on the most data wins Ho, T. K. and Baird, H. S.,[1] Simard, P. Y., LeCun, Y. A., Decker, J. S. and Victorri, B.,[2] Varga, T. and Bunke, H..[3] Of course, putting this strategy into practice can be troublesome, since large training sets are expensive or impossible to obtain, and may not be representative - or sets may be imbalanced, where one class is represented by too few samples, and others have too many. This is in the context of supervised classification in which classifiers are designed fully automatically by reading in files of labeled training samples so that the classifier can learn from example patterns.

Good results in pattern recognition have been achieved by the use of supervised classifiers such as nearest neighbors algorithms; however these results require large amounts of training data together with carefully labeled g*round truth* (the true classes of each sample) which can be even more expensive to provide than the data itself. Often the acquisition of such data becomes a problem, as much time and labor must be spent to locate existing training data, or alternatively, to classify new training data properly.

We believe that one way out of this impasse is to *amplify* the training data: that is to increase it artificially by generating more. We call such generated data *synthetic* in contrast to data collected in the field, *real* data. We conducted experiments to explore the validity and uses of such synthetically generated data.

We made the assumption, as is done classically in Bayesian decision theory, that we had a classifier technology which could be trained on a training set and tested on a separate set. Typically the training samples arise in a natural *sample space*. In our domain, the sample space was the set of all real images.

Usually the first step in crafting a classifier is to choose numerical features (say d of them) that can be algorithmically extracted so each sample is represented by a set of features: that is, as a point in d-dimensional space. In this sense, data also live in *feature space*.

However, since we know how images are generated, then the parameters that control the generation process are yet another way of describing the data – so synthetically generated data can be said to live in *parameter space* also.

It may be informative to give an example of a method of generating artificial data which 'lives' in each space. For example, in sample space, one might generate synthetic data by taking several real images, cutting them up, pasting them together in a random fashion, and so, making a collage of them.

To generate data synthetically in feature space, one might extract features for several samples, giving several points in feature space, then interpolate between the points to generate a new point. An example of feature space data generation is outlined in more detail in our small experiment described in the following pages. In this example, the features reddish, greenish and blueish have numerical values assigned to them, and new feature points are generated by interpolating between the values to obtain new reddish, greenish and blueish values.

Samples that are generated in parameter space were synthesized by varying the generating parameters. In the case of document images, these generating parameters were among the following; type, size, image degradations such as blur, threshold, additive noise, etc. and layout dimensions such as gutter width, line spacing. We can say that pairs of real images span ranges in parameter space and thus allow the generation of synthetic training data densely within that range.

We explored the relationship between parameter space, sample space and feature space which can be thought of as follows:

$$\text{parameter space} \rightarrow \text{sample space} \rightarrow \text{feature space}$$

2. PREVIOUS WORK

We are not the first to be interested in interpolated data, however, to the best of our knowledge we are the first to be interested in generating data by interpolating in parameter space.

In an effort to determine if performance of a recognition system affected by the size and quality of the training data, Varga, T. and Bunke, H.,[43] examine under what circumstances a larger training set improves the accuracy of handwriting recognition systems. They show how synthetic generation of new training samples can be achieved, e.g. through perturbation of, or interpolation between the original samples. They tested the use of continuous nonlinear functions that control a class of familiar geometrical transformations applied on an existing handwritten text line.

Chawla et al.[5] propose that undersampling of the majority (normal) is a good means of increasing the sensitivity of a classifier to the minority class. The authors use a combination of over-sampling the minority class and under-sampling the majority class to achieve better classifier performance.

Ho T. K. and Baird H. S.[1] present three studies that rely on synthetic data generated pseudo-randomly in accordance with an explicit stochastic model of document image degradations.

Sun J. et al.[6] generate synthetic degraded character images based on a simplified video degradation model, common in facial recognition systems. They generate the images themselves based on the Dual Eigenspace construction using multiple clusters per category.

3. METAFONT INTERPOLATION

For our experiments, We generated training samples in parameter space and tested if a set of images generated from interpolated typeface styles performed as well as a same-size set of images generated from original metafont type styles on previously seen data. This tested the safety of our algorithm. Next we tested if our set of interpolated training images performed better on previously unseen data. This tested the effectiveness of our algorithm.

A Metafont[8] program was created to interpolate between two or more fonts from the Computer Modern Roman families of fonts. I will discuss the case of a two-font interpolation. First, the parameter values were listed for each font and differences between the same parameter in each font were calculated. Since we were

creating nine interpolations, the difference was divided by 10 and one tenth of this value was added to or subtracted from the first font until we arrived at the second font. Boolean values were either True or False. One example of a Boolean parameter would be the Serifs parameter (Serifs = True or Serifs = False). If both fonts had True, the interpolations were also True. If one was False and one was True, the first half of the interpolations was set to False, while the second half were set to True and vice versa. A Metafont program was used to create the fonts and once they were created, LaTeX was used to create an image of the letters c and e (or i and j) as a .png file.

4. TEST SET CREATION

The IDMGEN [7] program created by Henry Baird as an image defect model generator was used to generate the images of a letter in a selected font. The program is given an image of an isolated character in PNG format, from which it generates a series of the same character in text-line format in an ascii file. The text characters are pseudo-randomly distorted using a quantitative model of the printing and imaging process. Alternatively the program will read/write HSLC, an enhanced HSL format developed by the DICE project at Lehigh. The details of this file format are beyond the scope of this dissertation, however the interested reader is referred to the website at snake-eyes.cse.lehigh.edu for details. The HSLC files are easily converted to PNG files for better display of the character images by using the DICE project utilities.

A parameter for seed (-S) is input at the start of each letter generation run to set the pseudo-random number generator and produce an image of a character. Input parameters are used to generate scalar random variables with defined distributions which are applied to the generated images. Three of the parameters we used to introduce variation in the images were blurring, a value which represents the point-spread (or impulse response) function of the combined printing and imaging process, threshold, a parameter which models binarization as a test on each pixel and sensitivity, a parameter which randomizes each pixel's photo-receptor sensitivity in two stages. The interested reader may find the details of these parameters in the IDMGEN documentation referenced above.

The end result was a series of images with variations in blur, brightness and intensity, some very readable, and some greatly distorted. Each of the experiments within a series of fonts used different values of certain parameters to control distortion. Samples of images generated by our process are shown in figure 1.

5. OUR INTERPOLATIONS

Interpolations between CMR (Computer Modern Roman) and CMSS (Computer Modern Sans Serif) were constructed by smoothly interpolating the parameters used in creating both CMR and CMSS. Below is an example showing the letters in CMR, nine interpolations, and finally CMSS.

We also tested a similar interpolation between CMR and CMFF (Computer Modern Funny Font) on the letters C and E as well as on the letters I and J. Lastly we tested a three-way interpolation between CMR, CMFF, and CMSSI (Computer Modern Sans Serif Italic) fonts on the letters C and E. An example of the three-way interpolation is shown in figure 2.

Each set of experiments had tests involving differing degrees of blur, noise and variance. Each test compared the performance of pure versus mixed (interpolated) training sets on both pure and mixed test sets. Some of the tests in each set took test data from within the entire range of interpolated samples, while some took all the interpolated data from the midpoint between the two fonts.

6. EXPERIMENTAL DESIGN

Eight hundred samples of the letter c and e were generated using the idmgen program. For the first group of experiments, the ideal prototype of each of the training samples was a machine print type form of these letters in Knuth's CMR (Computer Modern Roman) typeface. (800 samples total)
Additionally, eight hundred samples of the letter c and e were generated using the idmgen program and a machine

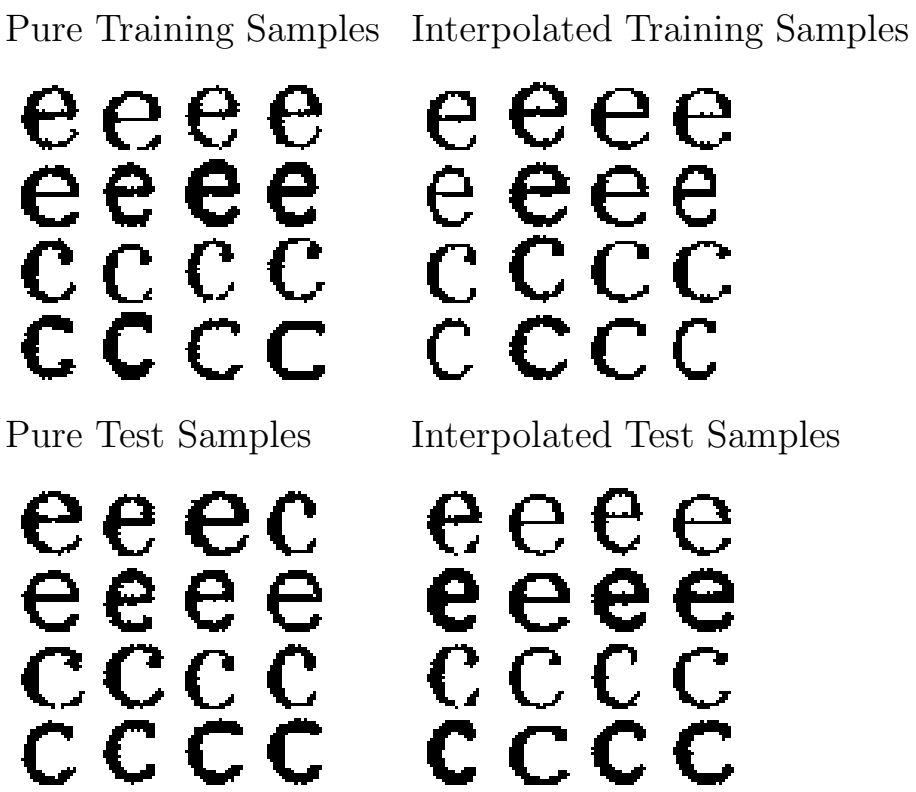

Figure 1. Samples of Generated Images

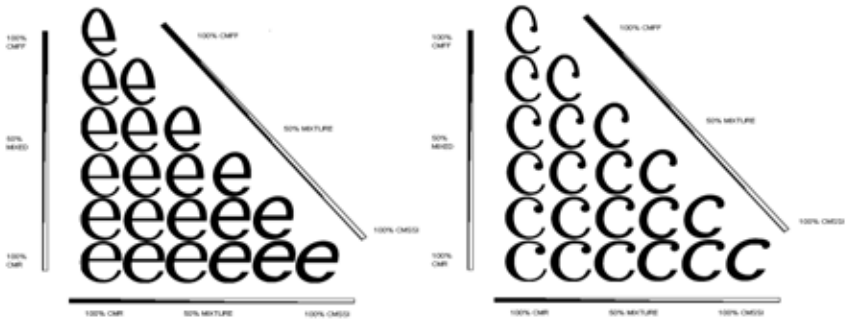

Figure 2. Example of Three-Way Interpolation

print type form of the letter e and the letter c in Knuth's CMSS (Computer Modern Sans Serif) typeface as the ideal prototype (800 samples total). This provided a training set of 1600 samples, equally divided between CMR and CMSS. We call this training set A.

A kNN Classifier was trained on the CMR and CMSS training sets. The classifier was then tested on a test set consisting of 200 CMR samples and 200 CMSS samples, similarly generated, which we call test set A. Rates

of accuracy for the 400 test samples were recorded.

Next, 1600 mostly interpolated samples of the letter c and the letter e were generated using the idmgen program. The ideal prototype of each of the ten sets was as follows; a machine print type form of the letter e or the letter c in Knuth's CMR (Computer Modern Roman), CMSS or one of our interpolated typefaces. This was called training set B.

The classifier was trained on the interpolated samples (training set B) and then tested the pure test samples (test set A). This tested if the classifier which has been trained on interpolated data performed equally well on samples in previously known fonts as a classifier trained on "real" data.

A test set consisting of 400 mostly interpolated samples of the letter c and the letter e were generated using the idmgen program using CMR and CMSS typefaces and nine interpolations between them. This became Test Set B.

We then trained a classifier on the known font type samples (Training Set A) and tested it on the interpolated samples (Test Set B) to test how well it performed when it was tested on previously unseen fonts.

Finally, the classifier trained on the interpolated samples (Training Set B) was tested on the interpolated test set (Test Set B) to see how well the classifier performed when it had been trained on interpolated data and tested on previously unseen data.

Two hypotheses were proposed and tested using the χ^2 statistic

6.1 Hypothesis 1:

AB is trained on only pure data and tested on mostly interpolated data. BB is trained and tested on interpolated data. We expect that BB will perform significantly better than AB. Therefore our null hypothesis is that AB will perform better than BB. This part of the experiment speaks to the performance or strength of our algorithm.

The areas of interest in this part of our experiment are the differences between AB and BB. If the null hypothesis is rejected then our classifier trained on mostly interpolated data has performed better than the classifier trained on only pure data and we can say that our interpolated classifier is better at classifying interpolated data.

6.2 Hypothesis 2:

AA is trained and tested on pure data. BA is trained on mostly interpolated data and tested on pure data. We would hope that BA would not perform significantly worse than AA. If this is the case, we have shown that in our experiment, training on interpolated data does not "hurt" the classifier in the identification of pure data. In this way we are testing if our algorithm is "safe". Our null hypothesis is that BA and AA have the same accuracy and we would hope that the null hypothesis holds.

The areas of interest in this part of the experiment is the difference between AA and BA. If our null hypothesis is not rejected then we can say that training on interpolated data does not appear to "hurt" the classifier in the identification of pure data.

- A = previously known data (CMR and CMSS fonts)
- B = previously unseen data (interpolated fonts)

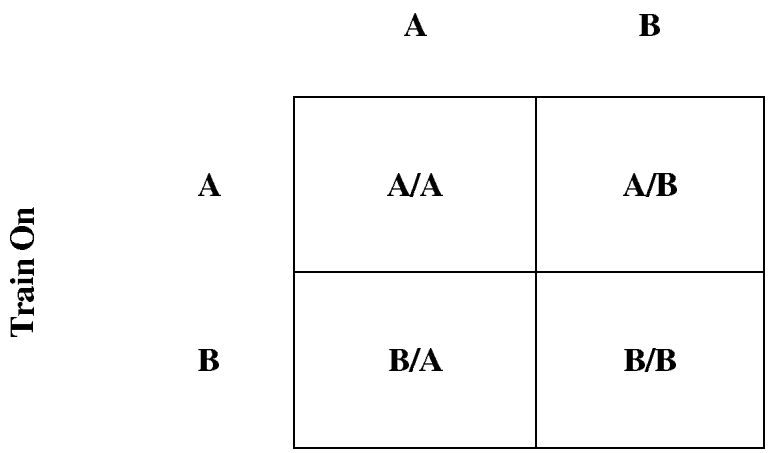

Figure 3. Design of Experiment

7. RESULTS

We have found that training on interpolated data is for the most part safe, that is to say never produced more errors, when tested on the pure samples. Furthermore, the classifier trained on interpolated data often but not always improved (about one third of the time) classification when tested on previously unseen interpolated samples.

I refer the reader to the following table 1 for a concise graphical summary of our results.

Our first set of experiments was performed on interpolations between the Computer Modern Roman (CMR), a serif font, and the Computer Modern Sans Serif (CMSS), a sans-serif font, both from the Computer Modern (CM) family of fonts. The fonts were fairly close to each other and results of testing on a set of pure test samples proved that the classifier trained on the interpolated training samples performed as well as the one trained on the pure (50

On all the experiments, both classifiers performed equally when tested on the pure test sets. However, the classifier trained on interpolated data performed better than the one trained on only pure data about half the time when CMR-CMFF were involved and in five out of the seven tests for the three-way interpolation.

The last set of tests was performed on the two fonts, CMR and CMFF with the letters i and j. Here, once again, both classifiers performed equally when tested on the pure test sets. In every case, the interpolated classifier was at least as good as the pure classifier. However, when tested on the interpolated test images, there was also no difference between the recognition of the images. We think that there was either a great enough difference between the i's and j's that the classifiers were able to correctly identify them.

8. CONCLUSION

In our systematic family of tests, we have demonstrated that the use of interpolated data in the training sets has never worsened the results, and has frequently improved the results. The improvement is greater when the fonts being interpolated are most different from each other and when the images are greatly blurred with little variance. The three-way interpolation tests showed the most number of significant improvements for the interpolated training sets. Note that the results shown are for the two easily confused pairs of characters e/c and i/j.

Table 1. Overall Results

CHARS	FONT STYLES	IMAGE QUALITY	TEST SET	SAFE?	BETTER?
e and c	CMR-CMSS	normal	full range	yes	–
		slightly blurred	full range	yes	–
		greatly blurred, high variance	full range	yes	–
		greatly blurred, some variance	full range	yes	–
		greatly blurred, little variance	full range	yes	yes
		greatly blurred, little variance	midpoint	yes	–
		slightly blurred, little variance	midpoint	yes	yes
e and c	CMR-CMFF	normal	full range	yes	–
		slightly blurred	full range	yes	yes
		greatly blurred, high variance	full range	yes	–
		greatly blurred, some variance	full range	yes	–
		greatly blurred, little variance	full range	yes	yes
		greatly blurred, little variance	midpoint	yes	yes
		slightly blurred, little variance	midpoint	yes	yes
e and c	CMR-CMFF-CMSSI	normal	full range	yes	yes
		slightly blurred	full range	yes	yes
		greatly blurred, high variance	full range	yes	–
		greatly blurred, some variance	full range	yes	yes
		greatly blurred, little variance	full range	yes	yes
		greatly blurred, little variance	midpoint	yes	yes
		slightly blurred, little variance	midpoint	yes	–
i and j	CMRCMFF	normal	full range	yes	–
		slightly blurred	full range	yes	–
		greatly blurred, high variance	full range	yes	–
		greatly blurred, some variance	full range	yes	–
		greatly blurred, little variance	full range	yes	–
		greatly blurred, little variance	midpoint	yes	–
		slightly blurred, little variance	midpoint	yes	–

REFERENCES

[1] T. K Ho and H. S. Baird. Large-scale simulation studies in image pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1067–1079, 1997.

[2] Simard, P., Le Cun, Y., Denker, J., and Victorri, B. Transformation invariance in pattern recognition - tangent distance and tangent propagation. In G. B. Orr Miller and K-R, editors, *Neural Networks: Tricks of the Trade*, volume Chapter 12. Springer, 1998.

[3] T. Varga and Bunke H. Comparing natural and synthetic training data for off-line cursive handwriting recognition. In IEEE, editor, *9th International Workshop on Frontiers in Handwriting Recognition*, 2004.

[4] T. Varga and Bunke H. Effects of training set expansion in handwriting recognition using synthetic data. In *11th Conf. of the International Graphonomics Society*, pages 200–203, Scottsdale, Arizona, USA, 2003.

[5] N. et al Chawla. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[6] J. et al Sun. Low resolution character recognition by dual eigenspace and synthetic degraded patterns. In ACM, editor, *HDP '04*, pages 15–22, Washington, DC, USA, 2004. ACM.

[7] Baird, H. Document Image Defect Models. In H. S. Baird, H. Bunke, K. Yamamoto, editors, *Structured Document Image Analysis*, Springer-Verlag, 1992.

[8] Knuth, D. *Computer Modern Type Faces*. Adedison Wesley Publishing Company, 1986.