Machine Learning Final Project: Handwritten Sanskrit Recognition using a Multi-class SVM with K-NN Guidance

Yichang Shih
yichang@mit.edu

Donglai Wei
donglai@csail.mit.edu

Abstract

We develop an optical character recognition (OCR) engine for handwritten Sanskrit using a two-stage classifier. Inside the standard OCR pipeline, we focus on the classification problem assuming characters have been preprocessed decently. One challenge we face is that the language of Sanskrit has about a hundred core characters where model driven methods, like Support Vector Machine (SVM), have to search in the exponentially growth of the combinatoric model space during training, while data driven methods, like k nearest neighbor (kNN), becomes costly in computation during testing. To address this challenge, we propose a two-stage multiclassifier, using non-parametric to reduce the model space to search, and parametric models to relieve computation burden with better generalization. In the first stage, we apply kNN to coarsely assign the test data into the possible group of k classes, and a multiclassifier of k classes to label the sample in the second stage. Our method is fully automatic, highly accurate, and computational efficiently.

1. Introduction

Sanskrit scholars have been endeavoring to integrate the gathered Tegabytes of Sanskrit manuscript images into a digital library [1] for worldwide reach.

Besides merely displaying the pages, it is desirable to translate the image into unicode for further editing, where current commercial OCR software for Sanskrit are far from satisfactory, as shown in Fig.2.

We find Sanskrit OCR an interesting problem in three folds:

Firstly, OCR problem itself lies in between one dimensional natural language process (NLP) and three dimensional real world projected down to image in computer vision (CV).

Also Sanskrit characters lie in the middle of an alphabetical language like English and a stroke-based language like Chinese.

Lastly, these Sanskrit manuscript are written by well-trained ancient calligrapher and there is a high consistence in character shape between samples, as shown in Fig.1.

With the above properties in mind, we design an optical character recognition system (OCR) that can automatically map Sanskrit to Unicode.

Our database contains about one hundred different Sanskrit characters, as shown in Fig.3.

In machine learning community, there are 3 typical approaches to solve multi-class problems: generative-model-based, SVM-based, and non-parametric-model-base. In OCR, the physic relationship between feature and label is unclear, and so the generative model is less practical. A multiclassifier of one hundred classes has only accuracy of 71%, which cannot satisfy our requirement, and also computational consuming in the training phase. Non-parametric model, such k nearest neighbor (kNN), is good at separate visually very different character, but cannot separate samples of minor difference, such as the images in Fig.4.

We take advantage of SVM-based and kNN method, and propose a two stage multiclassifer for about one hundred classes. In the first stage, we use kNN to assign an input query sample to a group containing k possible labels. Character with label in this group are visually similar, and cannot be distinguished by non-parametric method, as illustrated in Fig.4. In the second stage, we use a SVM multiclassifier to classify between labels in the group. Because our database is written by well-trained calligrapher, so we can easily design a set of robust features for SVM. In our work, we use 26 features, ranging from image statistics to local features, such as corners on characters, to train 65 multiclassifier of k classes. Our recognition is 85%, which is better than using SVMs only.

Our paper makes the following contributions,

1. The first system that works for handwritten Sanskrit.

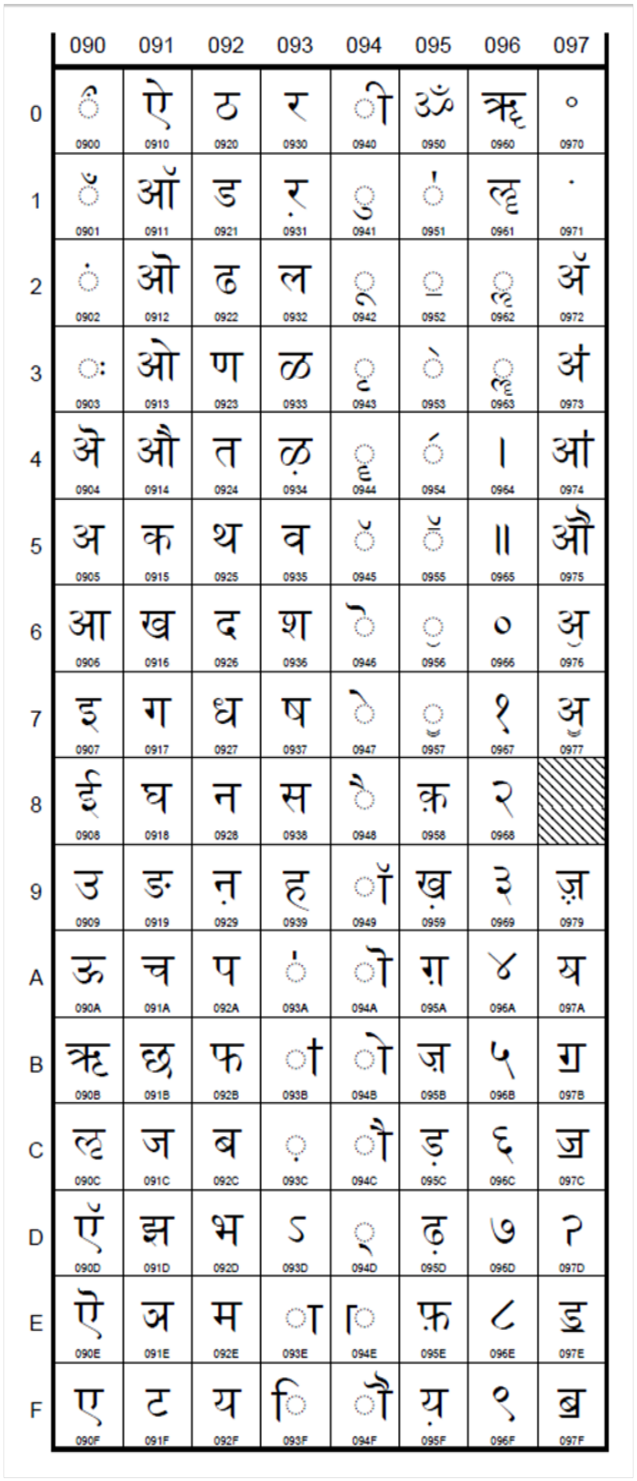
2. We applies a two-stage classifier that combines non-parametric method and SVM. This outperforms only using SVMs

2. Related Work

2.1. OCR Pipeline

Optical character recognition is a combination of natural language processing (NLP) and computer vision. In our application, language modeling is less important since we have more clues from the image likelihood, and the vision clues are more robust since it is an intrinsic 2D image.

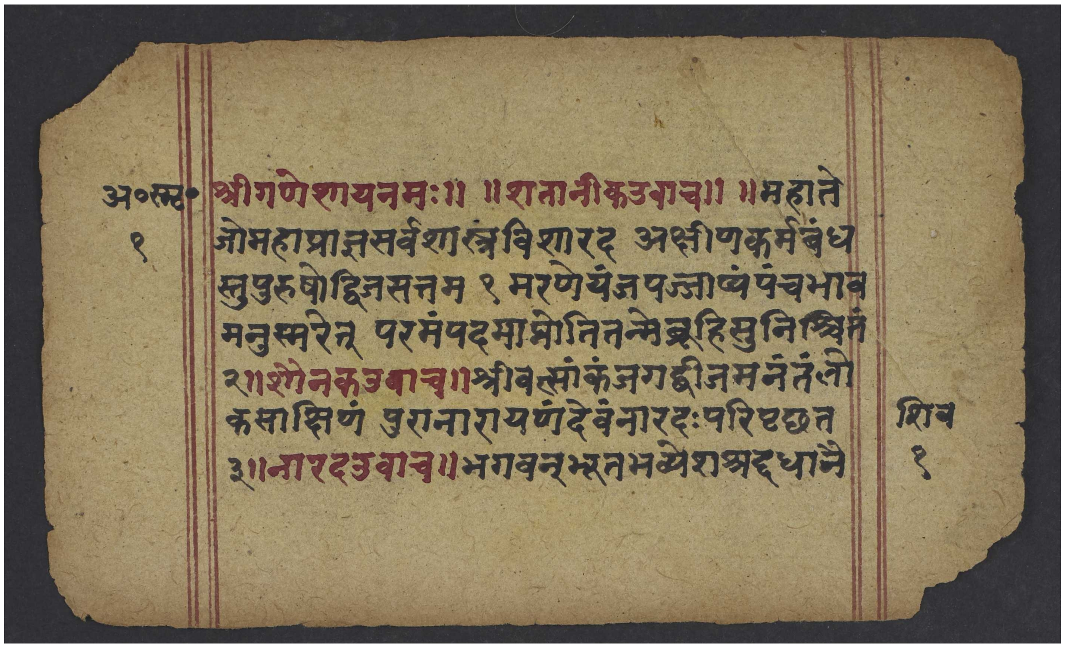
Below is the pipeline of a typical OCR system [4]:



	090	091	092	093	094	095	096	097
0	◌ऀ 0900	ऐ 0910	ठ 0920	र 0930	◌ी 0940	ॐ 0950	ॠ 0960	॰ 0970
1	◌ँ 0901	ऑ 0911	ड 0921	ऱ 0931	◌ु 0941	◌॑ 0951	ॡ 0961	ॱ 0971
2	◌ं 0902	ऒ 0912	ढ 0922	ल 0932	◌ू 0942	◌॒ 0952	◌ॢ 0962	ॲ 0972
3	◌ः 0903	ओ 0913	ण 0923	ळ 0933	◌ृ 0943	◌॓ 0953	◌ॣ 0963	ॳ 0973
4	ऄ 0904	औ 0914	त 0924	ऴ 0934	◌ॄ 0944	◌॔ 0954	। 0964	ॴ 0974
5	अ 0905	क 0915	थ 0925	व 0935	◌ॅ 0945	◌ॕ 0955	॥ 0965	ॵ 0975
6	आ 0906	ख 0916	द 0926	श 0936	◌ॆ 0946	◌ॖ 0956	० 0966	ॶ 0976
7	इ 0907	ग 0917	ध 0927	ष 0937	◌े 0947	◌ॗ 0957	१ 0967	ॷ 0977
8	ई 0908	घ 0918	न 0928	स 0938	◌ै 0948	क़ 0958	२ 0968	▨
9	उ 0909	ङ 0919	ऩ 0929	ह 0939	◌ॉ 0949	ख़ 0959	३ 0969	ॹ 0979
A	ऊ 090A	च 091A	प 092A	◌ऺ 093A	◌ॊ 094A	ग़ 095A	४ 096A	ॺ 097A
B	ऋ 090B	छ 091B	फ 092B	◌ऻ 093B	◌ो 094B	ज़ 095B	५ 096B	ॻ 097B
C	ऌ 090C	ज 091C	ब 092C	◌़ 093C	◌ौ 094C	ड़ 095C	६ 096C	ॼ 097C
D	ऍ 090D	झ 091D	भ 092D	ऽ 093D	◌् 094D	ढ़ 095D	७ 096D	ॽ 097D
E	ऎ 090E	ञ 091E	म 092E	◌ा 093E	◌ॎ 094E	फ़ 095E	८ 096E	ॾ 097E
F	ए 090F	ट 091F	य 092F	◌ि 093F	◌ॏ 094F	य़ 095F	९ 096F	ॿ 097F

Figure 3: In our database, nearly 70 different characters are used.

1. In low-level vision, the system finds characters by layout analysis and converts them



अ०स्मृ० श्रीगणेशायनमः॥ ॥शतानीकउवाच॥ ॥महाते
१ जोमहाप्राज्ञसर्वशास्त्रविशारद अक्षीणकर्मबंध
स्तुपुरुषोद्विजसत्तम १ मरणेयंजपज्जाप्यंपंचभाव
मनुस्मरेत् परमंपदमाप्नोतितन्मेब्रूहिमुनिश्चितं
२॥शौनकउवाच॥श्रीवत्सांकंजगद्बीजमनंतंलो
कसाक्षिणं पुरानारायणंदेवंनारदःपरिपृच्छत शिव
३॥नारदउवाच॥भगवन्भूतभव्येशश्रद्दधाने १

Figure 1: A page of Sanskrit Manucript in our database used for our OCR. The documents are written by well trained calligrapher, and so there is high consistence between the same characters, nearly as good as in printing version.

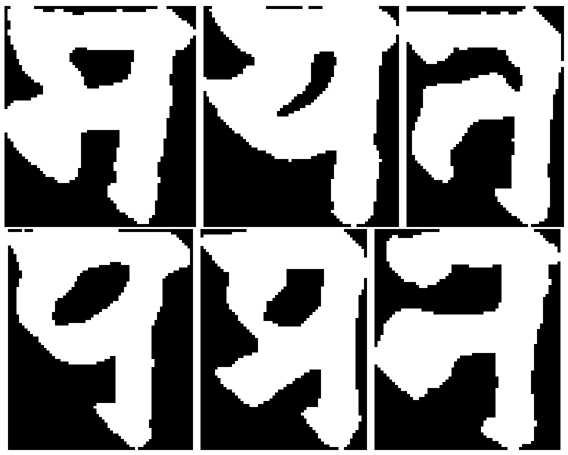


Figure 4: A group of 6 characters that kNN cannot classify. They have very similar visual shapes.

into grayscale (preprocessing).

2. In mid-level vision, the article is separated to unconnected characters.
3. In high-level vision, test characters are classified based on the model from training data. This is referred to as recognition.
4. In NLP, a n-gram HMM model is fitted to correct certain ambiguity.

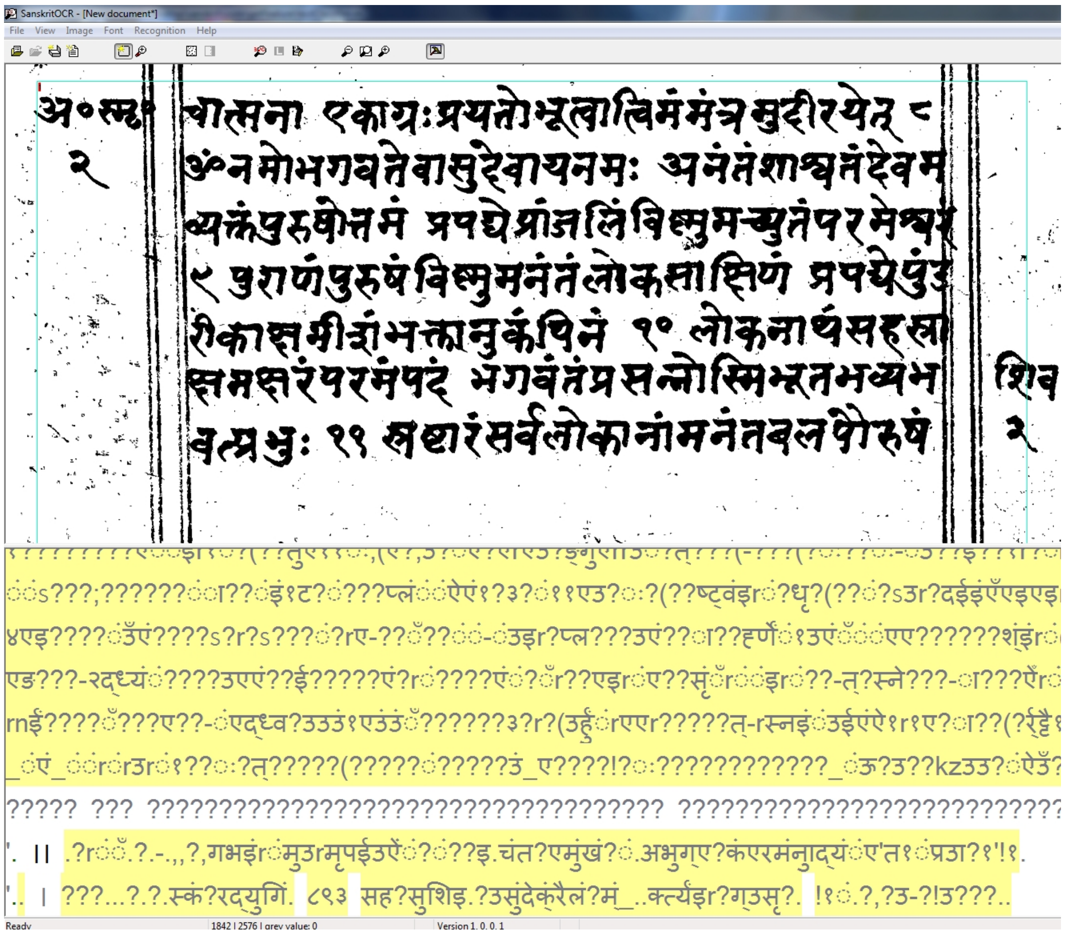
Here, we focus on the classification problem, assuming all training and testing characters have already been correctly cropped out, and we leave out the NLP part to further improve the performance.

2.2. Sanskrit Recognition

Some commercial software, such as has implemented Sanskrit OCR. [2], but their performance, as shown in Fig.2 is not satisfactory in our data set.

2.3. Classification

We review the state-of-art classification algorithm in machine learning, and state the reason we propose a two-stage jointly non-parametric and discriminative classifier.


SanskritOCR - [New document*]
File View Image Font Recognition Help

```
◌◌s???;??????◌◌ı??◌इ१ट?◌???प्लं◌◌ऐएं१?३?◌११एउ?◌ः?(??ष्ट्वंइr◌?धृ?(??◌?s3r?दईइएँएइएइ
४एइ????◌उँएं????s?r?s???◌?rए-??◌ँ??◌◌-◌उइr?प्ल???उएं??◌ı??ह्र्णे◌१उएं◌ँ◌◌एए??????शुंइr◌
एङ???-२दृध्यं◌????उएएं??ई?????एं?r◌????एं◌?◌ँr??एइr◌ए??सृं◌ँr◌◌इr◌??-त्?स्ने???-◌ı???ऐंr◌
rnई????◌ँ???ए??-◌एद्ध्व?उउउं१एउंउं◌ँ??????३?r?(उर्हुं◌rएएr?????त्-rस्नइं◌उईएंऐ१r१ए?◌ı??(?र्टुं१
_◌एं_◌◌r◌r3r◌१??◌ः?त्?????(?????◌?????उं_ए????!?◌ः????????????_◌ऊ?उ??kzउउ?◌ऐउँ?
????? ??? ????????????????????????????????????? ?????????????????????????????
'. ।। .?r◌◌ँ.?.-.,,?,गभइंr◌मुउrमृपईउऐं◌?◌??इ.चंत?एमुंखं?◌.अभुग्ए?कंएरमंनुाद्यं◌ए'त१◌प्रउा?१'!१.
'.. । ???...?.?.स्कं?रद्युगिं. ८९३ सह?सुशिइ.?उसुंदेकंरैलं?मं_..क्त्यंइr?गुउसृ?. !१◌.?,?उ-?!उ???..
```

Ready
1842 | 2576 | grey value: 0
Version 1. 0. 0. 1

Figure 2: Recognition result of Indsez, a commercial Sanskrit OCR software. The upper black-and-white image is the input article, and the recognition result are marked in yellow. The question mark means the character cannot be recognized. We cannot test the recognition rate because the source code is not open.

2.3.1 Definition

We first review the general goal of Machine Learning Algorithms: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

For classification problem, the task to is to tell things different apart and the experience E is to learn the suitable representation of the true data distribution based on the performance measured by P on training data. It can be viewed as an optimization over hypthesis space$\mathcal{H}$. In the parametric setting, the objective function is given by $\hat{f}(w;\theta)$ where w is the model parameter, θ a given sample. An empirical objective: $\hat{F}(w)$ can be written as:

$\hat{F}(w) = \mathbf{E}[f(w;\theta)] = \frac{1}{m}\sum_{i=1}^{m} f(w;\theta_i)$

here $\theta_1, \theta_2, ..., \theta_m$ are a sequence of observed samples. It is normally assumed that these samples are i.i.d. drawn from some unknown distribution D, and therefore the stochastic objective F(w) is often more desirable:

$\hat{F}(w) = \mathbf{E}_{\theta \approx D}[f(w;\theta)]$

For example, if we take $f(w;\theta) = max\{0, 1y(w,x) + \frac{\lambda}{2}\|w\|_2^2$, we will arrive at the famous SVM, with a weighted L2 norm regularizer.

2.3.2 $\mathcal{H}$: Parametric vs Nonparametric

In terms of modeling the hypothesis space, classification algorithms can be divided into parametric ones and nonparametric ones. In the parametric setting where the hypothesis space is a family of parametrized function, no matter Frequentist, Bayesianist or Learning Theorist, no matter Discrimitive, Generative or Algorimitic, we end up with an object function composed of the bonus of the likelihood of the data under the model and the penalty of the complexity of the model to optimize. For example, SVM is trying to

In the nonparametric setting however, we are trying to make use of the empirical probability distribution of the data itself.

2.3.3 $\mathcal{H}$: Single vs Structured

Instead of searching one best function f in one subspace of $\mathcal{H}$ with model selection, researchers have been seeking efficient ways to combine different functions instead of . We can consider the single function approach as the 0th order approximation in $\mathcal{H}$, where the voting/bagging/boosting scheme that linearly combining each weighted function as the 1st order approach. Second order methods, like Decision Tree/Random Forest/Feed-forward Neural Nets(Mulitple Layer Perceptron), are trying to come up with a 2D decomposition of $\mathcal{H}$.

2.3.4 $\mathcal{H}$: Binary-class vs Multi-class

In the practice of binary classification problem, boosting and SVM generally works well enough with carefully chosen kernel/feature. However, so far, there is no satisfactory methodology on Multi-class problems. One natrual approach is to reduce the multi-class problem to binary ones, but neither one-vs-one or one-vs-all scheme gives strong discriminative power. Even if we try more tasks to fill out a coding matrix, introducing redundancy to reduce errors, the problem becomes how to efficiently choose powerful set of tasks, which is proved to be NP hard. Another approach is to jointly model all classes Instead, we of Hierachical SVM [5], Boosting SVM [6]

2.3.5 Besides $\mathcal{H}$: Feature Space

So far, we are focusing on different construction in the Hypothesis Space $\mathcal{H}$, which takes in the data generically.

In different application however feature design: Meta Knowledge of the task selection/combination:

3. Algorithm

In our work, most characters are well-structured and well-separated, which is different from general vision recognition tasks where objects have much variation. On MNIST handwritten digit recognition task, people find surprisingly that some easy kNN matching algorithm perform much better than most crafted SVN/Boosting/Neural Nets. Thus quick explicit nonparametric methods work better than parametric ones doing optimization costly while implicitly trading-off among joint object function. But in order to further improve the performance by resolving the deformation problem, we need to add some ingredients of SVM to learn from ambiguous cases. Thus, We here plan to use kNN as an initial guess and build ensembles of SVMs around set of classes that are close to each other. We not only try to combine the robustness of nonparametric modeling and the expressiveness of parametric modeling, but also the computation efficiency during training and testing respectively.

3.1. kNN

Though conceptually easy, designing an effective distance metric for kNN can be hard. One natural approach is to use explicit distance functions (like L1, L2, correlation) in the feature space. The baseline kNN tested here uses the pixel value as the feature and transforms the two dimensional image into one dimensitional vector. Apparently such metric does not capture the two dimensional structure of the image and is consequently not robust to the deformation of the characters, which may cause huge distance with simple affine transformation. One way to work around, as tangent distance [8] does, is to incorporate invariance into the metric on the feature space. But the increase of the complexity of the metric function still does not help to capture non-affine deformation of the image, which is common in practice.

Recently, instead of searching an explicit analytic form in the hypothesis space, people are looking into metric based on matching score. [3] samples points on the image boundary and combine the bipatrite matching score with other penalties into the final distance. However, the weight parameters in the distance has to be identified with cross-validation with huge computational cost. Surprisingly, [7] reports that simply matching local windows of two image contexts works extremely well on digit recognition task. Below is a simple description of the algorithm. For each point x in the test image context, we search over a window with fixed size s_1, say 5 by 5,around the corresponding point y (matched up in position) in the reference image. Then the local matching cost for this point is defined as the minimum distance computed above with an explicit function f(x,y). To make it more robust to outliers, we replace the distance of two points with the sum of the distance of matched up points within a small window around them. Now, we get a local matching cost for each point in the test image context and the distance between two image contexts is defined as the sum over all local matching cost. On MNIST dataset, [7] uses the gradient field of the image, which captures local two dimensional structure of the image as the image context. And the simple choice of f(x,y) as L2 norm produces the third best testing result so far.

In the experiment, we implement the algorithm in [7] in a slightly more efficient way by avoiding repetitive computation in the for loops. Instead of costly cross-validation, we set the parameters to be the same as that mentioned above.

Though kNN can achieve amazing performance with little knowledge representation of the task, it has the bottleneck of generalibility, which requires huge amount of data to gain further improvement. This is where we need SVM to infer the hidden discrimitive rules to generalize beyond limited training data after narrowing down the search range.

3.2. Multiclass SVM

Feature design We design 26 features for classification in this phase. The 26 features can be roughly divided into 2 categories, global statistics features and local shape features. We have 9 global features, as described in the below,

1. Number of horizontal lines.
2. Number of vertical lines.
3. Euler number of the image.
4. Center of mass along x direction.
5. Center of mass along y direction.
6. Standard deviation along x direction.
7. Standard deviation along y direction.
8. The height to width ratio of the bounding box of the image.
9. Area of the written part in the image

To compute the number of horizontal lines, we project the input character image to y-axis to generate a one-dimensional array P. Then we binary threshold P. For each element i in P, if $P(i)$ is smaller than 0.6*(image width), then $P(i) = 0$, else $P(i) = 1$. 0.6*(image width) here represents

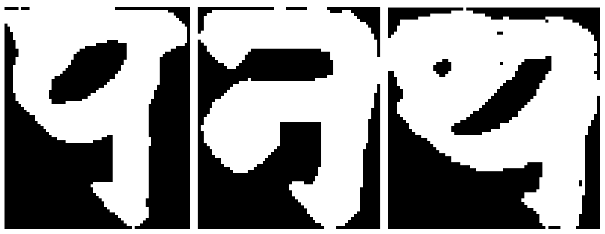


Figure 5: Euler number of this 3 images, from left to right, are 0,1,-1.

the minimal length of a horizontal line. Then we count the number of nonzero element in P, and divide a parameter w that represents the width of a stroke, to calculate the number of horizontal line. The same of vertical lines. In our work, w is chosen to be 15.

The definition of Euler number, e, is as below,

$$e = \text{\# of objects} - \text{\# of holes in the objects} \quad (1)$$

Fig.5 shows some examples of euler number using our data set. Euler number is a measure of topology, which is invariant to handwriting.

Our system uses 16 Local, shape-aware features. For each input character image, we segment the image into 4 non-overlap sub-images: top-right, top-left, bottom-left, and bottom right. For each sub-images, we compute the normalized cross correlation (NCC) between the sub-image and the 4 templates in Fig.6, which detects corners toward the 4 directions, respectively. NCC is defined as,

$$NCC(u,v)_{s,t} = \frac{\Sigma[s(x,y)-\bar{s}][t(x-u,y-v)-\bar{t}]}{\{\Sigma[s(x,y)-\bar{s}]^2\Sigma[t(x-u,y-v)-\bar{t}]^2\}^{0.5}}$$

where s is the sub-image and t is the template. If the maximal NCC is greater than the threshold, then the corresponding feature is 1, otherwise is 0. For example, if the maximal NCC between the s^{th} sub-image and the t^{th} template is greater than the threshold, the $(4s+t)^{th}$ feature is 1, which physically means that s^{th} sub-image contains the t^{th} pattern. In our work, we set the threshold to be 0.9 and works very well.

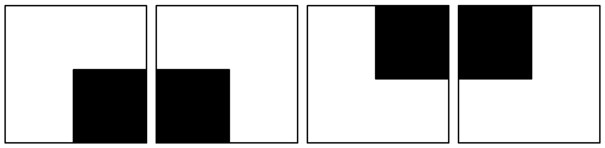


Figure 6: Four templates used for corner detection in our system.

SVM classifier We use a multiclassifer of k classes to finely classify the input character image. We tried both one-against-all and one-against-one methods for classification, and obtain the same performance, about 85% recognition rate. The first stage generates 65 groups, and so we train 65 multiclassifier in the second stage.

4. Experiment

4.1. Training Set

Starting from the raw Sanskrit manuscript image, we first do standard page layout analysis to find the position and statistics of potential characters. Then we binarized the image making characters stand out from background by contrived thresholding method based on the information from the analysis above. Taking advantage of the property of the manuscript that characters tend to disjoint, we crop out the characters by finding connected components. We also threshold the size the image patch for fewer false alarms and segment the image patch when several characters accidentally connected to each other. So far, we have the image bank of the Sanskrit characters.

Unlike OCR for printing or for handwritten English which are much studied, there is no public datasets for Sanskrit characters with ground truth labels. As can be imagined, there is no single trick to build up the whole dataset around human knowledge about discriminative patterns of shapes and structure from scratch. We here managed to do so by using the combination of a Top-Down approach to decompose problems into smaller ones and a Bottom-Up method to group characters conservatively. On the first stage, we manually pick discriminative features, like structural pattern to roughly seperate different groups of characters apart. Since none of the features are robust enough to help us recursively build the noiseless training data, we develop an interactive GUI for quick relabeling/merging same groups of characters with binary choices on the second stage. After the traininig data first built, we run unsuperised algorithms like K-means and hierarchical clustering for anormaly detection. Within a few steps of refinement, the training dataset generally has the acceptable quality for conducting experiment.

In the end, we have around 6,000 image pathes of characters which can be grouped into around 200 groups. We then manually choose 64 of the character classes with around 2,500 instances as the training data and create the test data with around 600 instances.

4.2. kNN training

Firstly, for each character class, we tried to figure out its k confusion neighbours (closest character classes) in order to train SVM in advance. One way is to do the leave one out cross-validation on the training data and see which wrong labels may be returned by the rest of the training data. In practice however, we find that character classes are well grouped in nature and we may simply use the mean of a character class to figure out its k confusion neighbours.

Also, during testing, in order to have a coarse initial guess, we may not need to search over all the training data for each test image. Thus we need to select representative images inside each character class which cover most of the variation of the class. One way to do is to greedily stage-forward fitting, adding the image that minimize the mutal information among the selected set of images.

4.3. SVM training

We tried both one-against-all and one-against-one methods for classification, and obtain the same performance. In the training phase, we use both polynomial and gaussian as kernel function, and also obtain the same performance. We set the soft margin parameter to be 1000. The training takes about 1 minutes for each classifier, using MATLAB in a 8-cores machine, and so about 1 hour for the whole 70 classifiers.

5. Result

Our training data contains 65 characters, and totally about 2500 training samples. Testing data contains 300 different characters. The correct recognition rate is 85.6% using one-against-all and 84.57% using one-against-one multiclassifier, respectively. Compared with only using SVM, which results accuracy of 72%, our method has significantly improvement. But for practical use, this is still not enough.

6. Discussion

The proposed method outperform simply using SVM about 10%. However, our method is still not good enough for practical use. Constraint to the difficulty of obtaining ground truth data, we here only conduct experiment on a relatively small dataset. In future work, we wish we can train this methods with more data and have further testing.

References

[1] http://sanskritlibrary.org/. 1

[2] http://www.indsenz.com/int/. 3

[3] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. *Advances in neural information processing systems*, pages 831–837, 2001. 6

[4] R. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7):690–706, 1996. 2

[5] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Hierarchical classification: combining bayes with svm. In *Proceedings of the 23rd international conference on Machine learning*, pages 177–184. ACM, 2006. 5

[6] T. Gao and D. Koller. Multiclass boosting with hinge loss based on output coding. 2011. 5

[7] D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(8):1422–1435, 2007. 6

[8] P. Simard, Y. LeCun, J. Denker, and B. Victorri. Transformation invariance in pattern recognitionxtangent distance and tangent propagation. *Neural networks: tricks of the trade*, pages 549–550, 1998. 6