CS 6347 Statistical Methods in AI & ML

Final Project Report

Optical Character Recognition using Conditional Random Fields

Presented By

Varun Kumar Reddy Bankiti (vxb150830)

Sreenivas Venkitachalam (sxv163530)

Mahesh Narayan Krishnamurthy (mxk166930)

INDEX

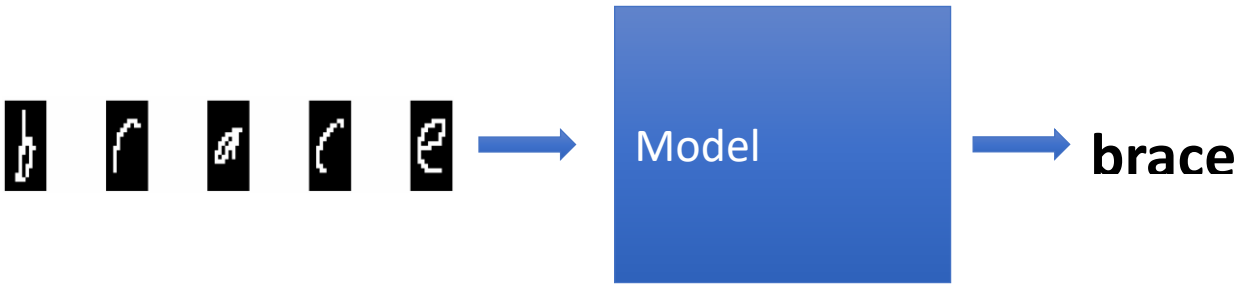
1. PROBLEM DESCRIPTION & OVERVIEW

Goal of this project is to implement a conditional random field for optical character recognition. In other words, if we are given a sequence of images consisting of hand written English alphabets corresponding to a word, our model should predict the word. The images of word have been segmented into letters with each letter, being represented as a 16*8 small image.

Mathematically, given a word $X = (X_1, X_2, \ldots, X_m)$ as a sequence of images, we predict the corresponding $y^* = (y_1, y_2, \ldots, y_m)$ using conditional random field model as

- $y* = argmax_y p(y|X)$
- $X_m \in \mathbb{R}^{(h \times w)}$
  - h=height of image
  - w= width of image
- $y_m \in \mathbb{R}$

This problem is of great significance in the field of Natural Language Processing for designing systems that accurately label or parse sequential data.

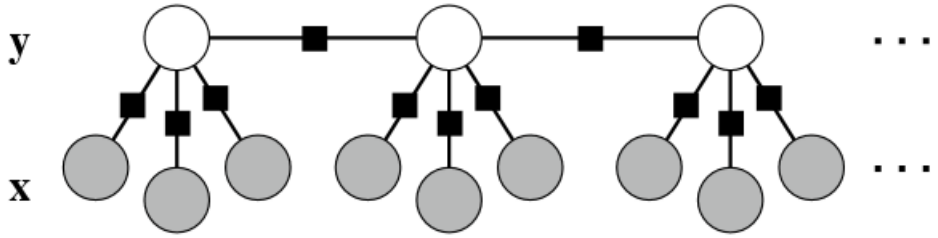


2. CONDITIONAL RANDOM FIELDS

Conditional random fields (CRFs) are a class of statistical modelling method often applied in pattern recognition and machine learning and used for structured prediction. CRFs fall into the sequence modelling family. A discrete classifier predicts only the label for a single sample without considering its neighbouring samples, but a CRF takes context into account. We are interested in Linear chain CRFs, where we predict the sequences of labels for sequences of input samples.

CRF is a discriminative undirected probabilistic graphical model, whose nodes can be divided into exactly two disjoint sets: X and Y, the observed and output variables, respectively. And the conditional distribution P(Y|X) is modelled. We usually assume a log linear form for P(Y|X).

Chain CRF is of the form:



3. SPECIFIC FORM OF MODEL

The model we are using is a Linear Chain CRF and we assume a log linear model. Each word is represented as $X = \{x_1, x_2, \dots, x_m\}$where m is the number of letters in the word and x_j is a 128-dimensional vector that represents its j-th letter image. The sequence label of a word is encoded as $y = (y_1, y_2, \dots, y_m)$ where $y_j \in Y \coloneqq \{1,2, \dots,26\}$ represents label of j-th letter. So, we can write the probabilistic model for a word $y = (y_1, y_2, \dots, y_m)$ given its image $X = \{x_1, x_2, \dots, x_m\}$ can be written as

$$p(y|\boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})} \exp\left(\sum_{j=1}^{m} < w_{y_j}, x_j > + \sum_{j=1}^{m-1} T_{y_j, y_{j+1}}\right)$$

Where

$$Z(\boldsymbol{X}) = \sum_{y \in Y^m} exp\left(\sum_{j=1}^{m} < w_{\hat{y}_j}. x_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j y_{j+1}}\right)$$

Inference is easier and more efficient as $Z(\boldsymbol{X})$ is a function of X.

Two groups of parameters are used here:

- Letter-wise discriminant weight vector $w_y \in R^{128}$ for each possible label $y \in Y$(State parameter)
- Transition weight matrix T which is sized 26-by-26. T_{ij} is the weight associated with the letter pair of the i-th and j-th letter in the alphabet. (Transition parameter)

4. DATASET

Dataset is downloaded from http://ai.stanford.edu/~btaskar/ocr/. It contains the image and label of 6,877 words collected from 150 human subjects, with 52,152 letters in total. To simplify feature engineering, each letter image is encoded by a 128 (=16*8) dimensional vector, whose entries are either 0 (black) or 1 (white). The 6,877 words are divided evenly into training and test sets, provided in data/train.txt and data/test.txt.

Fields of dataset:

1. id: each letter is assigned a unique integer id
2. letter: a-z
3. next_id: id for next letter in the word, -1 if last letter
4. word_id: each word is assigned a unique integer id (not used)
5. position: position of letter in the word (not used)
6. fold: 0-9 -- cross-validation fold
7. p_i_j: 0/1 -- value of pixel in row i, column j

Sample Dataset is of the form:

5. INFERENCE

Two types of inference problems that we are interested in are

- Predict the labels of new input X using most likely labelling
  $y^* = argmax_y p(y|X)$
- Computing node marginal $p(y_t|X)$ and edge marginal $p(y_t, y_{t-1}|X)$
  In the case of linear-chain CRFs, both inference tasks can be performed efficiently and exactly by variants of the standard dynamic programming algorithms for HMMs. Two algorithms- forward backward algorithm for computing marginal distributions and Viterbi algorithm for computing most probable assignment. These algorithms are special case of more general belief propagation algorithm for tree structured graphical models.

Two issues must be resolved for efficient inferencing in CRF. First issue is that inference subroutine is called repeatedly during parameter estimation. Second, when approximate inference is used, there can be complex interactions between the inference procedure and the parameter estimation procedure. This two factors play a major role in choice of inference algorithm.

5.1 Forward Backward Recursion

Basic idea behind forward backward recursion is to first rewrite the naïve summation $p(X) = \sum_y p(x, y)$ using distributive law

$p(X) = \sum_y \prod_{t=1}^{T} \varphi_t(y_t, y_{t-1}, x_t)$

$= \sum_{y_T} \sum_{y_T-1} \varphi_T(y_T, y_{T-1}, x_T) \sum_{y_{T-2}} \varphi_{T-1}(y_{T-1}, y_{T-2}, x_{T-1} \cdots$

We define a set of forward variables α_t each of vector of size M(M is the number of states) which represents the intermediate sums.

$$\alpha_t(j) = p(x_{<1,\ldots,t>}, y_t = j) = \sum_{i \in S} < w_{y_t}, x_t > \times T_{y_{t-1}, y_t} \times \alpha_{t-1}(i)$$

$\varphi_T(y_T, y_{T-1}, x_T) = < w_{y_t}, x_t > \times T_{y_{t-1}, y_t}$ where $< w_{y_t}, x_t >$ denotes inner product between vectors w and x.

Initialization of forward variable is $\alpha_1(j) = \varphi_1(j, y_0, x_1)$

Backward recursion is exactly same as forward recursion except summation is done in reverse order.

$$\beta_t(j) = p(x_{<t+1,\ldots,T>}, y_t = j) = \sum_{i \in S} < w_{y_t}, x_t > \times T_{y_{t-1}, y_t} \times \beta_{t+1}(i)$$

Initialization of backward variable is $\beta_T(i)$=1

Partition function $Z(X) = \sum_{i \in S} \alpha_T(i)$ or $\sum_{i \in S} \beta_1(i)$

5.2 Marginal Distribution

To compute edge marginal $p(y_{t-1,}\, y_t | \boldsymbol{X})$, which is used for parameter estimation combine the results of forward and backward recursions.

$$p(y_{t-1,}\, y_t | \boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})} * \alpha_{t-1}(y_{t-1}) * \varphi(y_{t,}\, y_{t-1}, x_t) * \beta_t(y_t)$$

Z(X) can be calculated using forward variable or backward variable as mentioned above.

To compute the globally most probable assignment $y^* = argmax_y p(y|X)$ summations are replaced by maximization. This yields the Viterbi algorithm.

$$\delta_t(j) = max_{i \in S} < w_{y_t}, x_t > \times T_{y_{t-1}, y_t} \times \delta_{t-1}(i)$$

Once the δ variables have been computed, the maximizing assignment can be computed by a backwards recursion

$$y_{T^*} = argmax_{i \in S} \delta_T(i)$$

$$y_{t^*} = argmax_{i \in S} \varphi_t({y_{t+1}}^*, i, x_{t_{+1}}) \delta_t(i)$$

6. PREDICTION

Basically, given the state and transition parameters, our model can be used to predict the sequence label for a new word image, $X^* := (x_1^*, \dots, x_m^*)$ using:

$$y^* = argmax_{y \in Y^m}\{\sum_{j=1}^{m} < w_{y_j}, x_j^* > \ + \sum_{j=1}^{m-1} T_{y_j, y_{j-1}}\}$$

y^* is calculated using Viterbi algorithm.

Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states called the Viterbi path, that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models.

7. PARAMETER LEARNING

There are two types of parameters in our specific conditional random field model, they are:

- **State parameters (W)**
  - Letter-wise discriminant weight vector $w_y \in R^{128}$ for each possible label $y \in Y$
- **Transition parameters (T)**
  - Transition matrix T which is sized 26-by-26. T_{ij} is the weight associated with the letter pair of the i-th and j-th letter in the alphabet $T_{ij} \neq T_{ji}$

Given a training set $\{X^i, y^i\}_{i=1}^n$, State parameters(W) and Transition parameters(T) are estimated by minimizing the following negative log likelihood function with regularization terms

$$min_{W,T} -\frac{C}{n}\sum_{i=1}^{n} logp(y^i|X^i) + \frac{1}{2}\sum_{y \in Y}\|w_y\|^2 + \frac{1}{2}\sum_{ij} {T_{ij}}^2$$

Where C > 0 is the trade-off weight (or) regularization constant that balances log-likelihood and regularization.

The above equation can be minimized using Limited-memory BFGS algorithm. Limited memory BFGS (L-BFGS or LM-BFGS) is an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden Fletcher Goldfarb Shanno (BFGS) algorithm using a limited amount of computer memory. It is a popular algorithm for parameter estimation in machine learning.

Now, gradients of likelihood part of the minimizing function are calculated as:

For a single data point $< X, \vec{y} > \in D$

$$P(\vec{y}\,|X) = \frac{1}{Z(X)} \exp\left\{\sum_{j=1}^{m} < w_{\vec{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\vec{y}_j, \vec{y}_{j+1}}\right\}$$

$$Z(X) = \sum_{\hat{y} \in y^m} \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j, \hat{y}_{j+1}}\right\}$$

Log probability:

$$\log\, P\,(\vec{y}\,|X) = \sum_{j=1}^{m} < w_{\vec{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\vec{y}_j, \vec{y}_{j+1}} - \log \sum_{\hat{y} \in y^m} \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j, \hat{y}_{j+1}}\right\}$$

Gradients:

$$\nabla_{w_Y} \log\, P\,(\vec{y}\,|X) = \nabla_{w_Y} \sum_{j=1}^{m} < w_{\vec{y}_j}, X_j > - \frac{1}{Z(X)} \nabla_{w_Y} \sum_{\hat{y} \in y^m} \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j, \hat{y}_{j+1}}\right\}$$

$$= \sum_{j=1}^{m} I\,[Y = \vec{y}_j]\vec{x}_j - \frac{1}{Z(X)} \sum_{\hat{y} \in y^m} \nabla_{w_Y} \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j, \hat{y}_{j+1}}\right\}$$

$$\nabla_{w_Y} \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j, \hat{y}_{j+1}}\right\} = \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j, \hat{y}_{j+1}}\right\} \nabla_{w_Y} \sum_{j=1}^{m} < w_{\hat{y}_j}, X_j >$$

$$= \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j, \hat{y}_{j+1}}\right\} \sum_{j=1}^{m} I\,[Y = \hat{y}_j]X_j$$

Therefore,

$$\begin{aligned}
&\nabla_{w_Y} \log\, P\,(\vec{y}\,|X) \\
&= \sum_{j=1}^{m} I\,[Y = \vec{y}_j]X_j - \frac{1}{Z(X)} \sum_{\hat{y} \in y^m} \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j, \hat{y}_{j+1}}\right\} \sum_{j=1}^{m} I\,[Y = \hat{y}_j]X_j \\
&= \sum_{j=1}^{m} I\,[Y = \vec{y}_j]X_j - \sum_{\hat{y} \in y^m} P\,(\hat{y}|X) \sum_{j=1}^{m} I\,[Y = \hat{y}_j]X_j \\
&= \sum_{j=1}^{m} I\,[Y = \vec{y}_j]X_j - E_{\hat{y}|X}\left[\sum_{j=1}^{m} I\,[Y = \hat{y}_j]X_j\right]
\end{aligned}$$

Similarly,

$$
\begin{aligned}
&\nabla_T \log\ P\left(\vec{y}\,|X\right) \\
&= \nabla_T \sum_{j=1}^{m-1} T_{\vec{y}_j,\vec{y}_{j+1}} - \nabla_T \log \sum_{\hat{y}\in y^m} \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j,\hat{y}_{j+1}}\right\} \\
&= [I(< i,j > \in \vec{y})]_{<i,j>\in 26\times 26} - \frac{1}{Z(X)} \sum_{\hat{y}\in y^m} \nabla_T \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j,\hat{y}_{j+1}}\right\} \\
&= [I(< i,j > \in \vec{y})]_{<i,j>\in 26\times 26} - \frac{1}{Z(X)} \sum_{\hat{y}\in y^m} \exp\left\{\sum_{j=1}^{m} < w_{\hat{y}_j}, X_j > + \sum_{j=1}^{m-1} T_{\hat{y}_j,\hat{y}_{j+1}}\right\} \nabla_T \sum_{j=1}^{m-1} T_{\hat{y}_j,ty_{j+1}} \\
&= [I(< i,j > \in \vec{y})]_{<i,j>\in 26\times 26} - \sum_{\hat{y}\in y^m} P\left(\hat{y}|X\right)[I(< i,j > \in \hat{y})]_{<i,j>\in 26\times 26} \\
&= [I(< i,j > \in \vec{y})]_{<i,j>\in 26\times 26} - E_{\hat{y}|X}[I(< i,j > \in \hat{y})]_{<i,j>\in 26\times 26}
\end{aligned}
$$

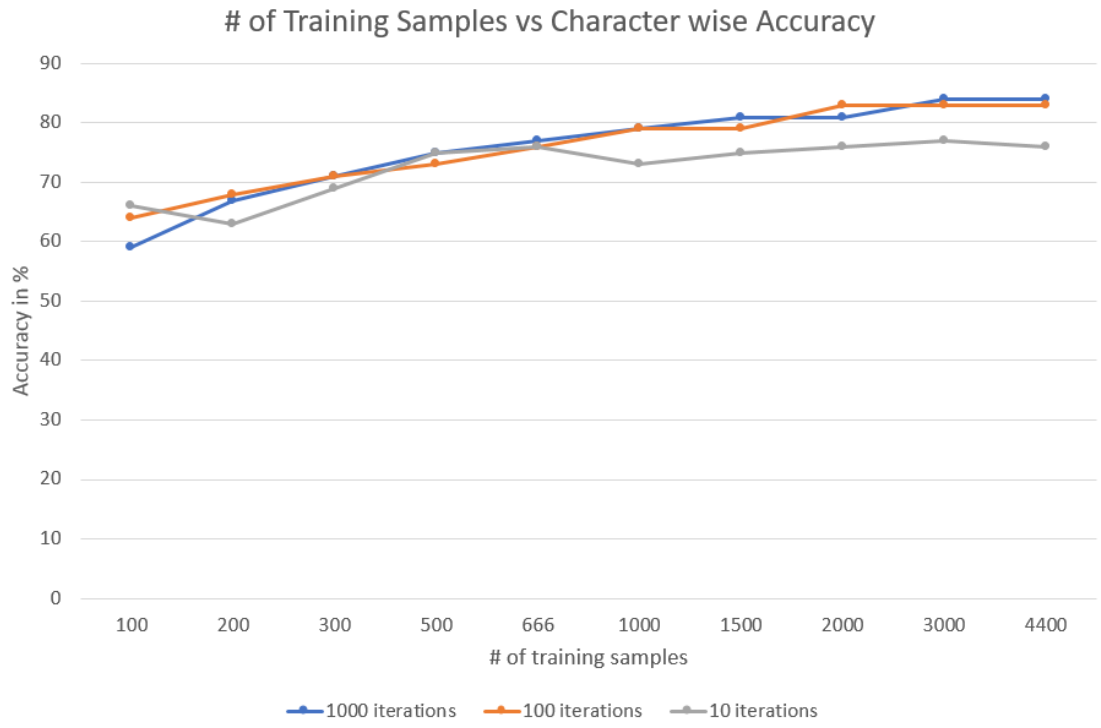
where $[I(< i,j > \in \vec{y})]_{<i,j>\in 26\times 26}$ denotes a 26×26 matrix whose value on coordinate $< i,j >$ is 1 if $\vec{y}$ contains transition $< i,j >$,and 0 otherwise.

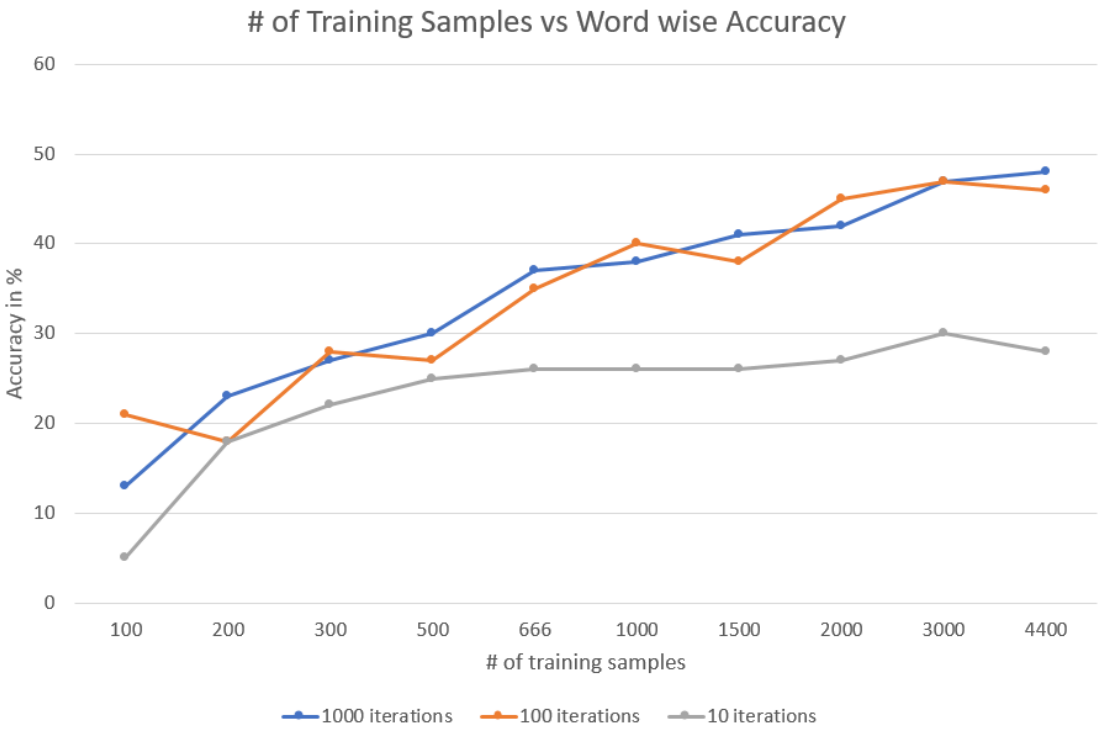
8. RESULTS

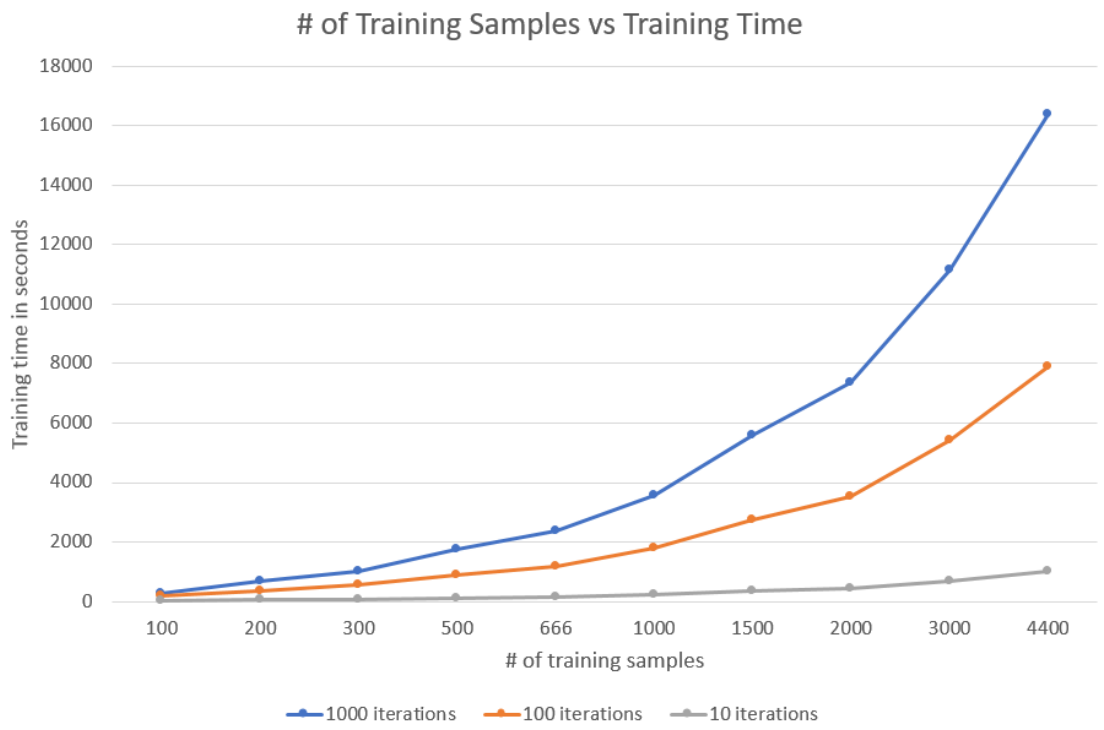
We use a Training vs Testing Ratio of 66:33 in order to determine the accuracy of the predicted results.

- Training Data: 4400 words, ~34000 characters
- Testing Data: 2200 words, ~17500 characters
- C = 1000 (Regularization constant)
- Accuracy
  - Character wise accuracy =~ 84 %
  - Word wise accuracy =~ 48 %

For, C = 1000 (Regularization constant), and using, count of Testing samples = Half the count of training samples and different values of iterations of optimization algorithm we have plotted the following graphs:


# of Training Samples vs Character wise Accuracy
Accuracy in %
90
80
70
60
50
40
30
20
10
0
100
200
300
500
666
1000
1500
2000
3000
4400
# of training samples
1000 iterations
100 iterations
10 iterations


# of Training Samples vs Word wise Accuracy
Accuracy in %
60
50
40
30
20
10
0
100
200
300
500
666
1000
1500
2000
3000
4400
# of training samples
1000 iterations
100 iterations
10 iterations



From the above plots, it is evident that the accuracy increases as we increase the number of iterations of the optimization algorithm and it increases with the number of training samples.

9. CONCLUSION

We have designed a model that helps us in Optical Character Recognition based on Linear Chain Conditional Random Fields. We have implemented the Learning and Inference Algorithms using the Python language. We measured the accuracy of individual characters and words, which were recorded and reported.

10. FUTURE SCOPE

- We can introduce Hidden / Latent variables in the model to model more dependencies / independencies, which will make the model closer to the real-world situations.
- Predicting the character of a word currently depends only on previous character and feature values, but we can try to model a general CRF where there will be edges between y_j , y_{j+1} and y_j, y_{j+2} and more.

11. REFERENCES

1. Introduction to Conditional Random Fields
   - http://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf
2. Object Recognition using Conditional Random Fields
   - http://people.csail.mit.edu/ariadna/mthesis.pdf
3. Character Recognition Using Conditional Random Field Based Recognition Engine
   - http://dl.acm.org/citation.cfm?id=2549401