Sign Language Recognition System using HMMs

Devansh Gupta*[1]

[1]Department of Electrical Engineering and Computer Science, Case Western Reserve University

April 29, 2019

Abstract

HMM is nothing but Markov process with hidden states. And this hidden parameters are obtained from observable parameters. Basically HMM has capability of modeling spatio-temporal information. Each gesture is modeled by a different HMM and given unknown gesture is tested by each HMM that is main advantage of HMM and The HMM output with maximum probability matching is nothing but the final recognized result.

The overall goal of this project is to build a word recognizer for American Sign Language video sequences, demonstrating the power of probabilistic models. In particular, this project employs hidden Markov models (HMMs) to analyze a series of measurements taken from videos of American Sign Language (ASL) collected for research (see the RWTH-BOSTON-104 Database). In this video, the right-hand x and y locations are plotted as the speaker signs the sentence.

*dxg397@case.edu

1 Introduction

While there are many different types of gestures, the most structured sets belong to the sign languages. In sign language, each gesture already has assigned meaning, and strong rules of context and grammar may be applied to make recognition tractable. American Sign Language (ASL) is the language of choice for most deaf people in the United States. ASLs grammar allows more flexibility in word order than English and sometimes uses redundancy for emphasis. Another variant, English Sign Language, has more in common with spoken English but is not in widespread use in America. ASL uses approximately 6000 gestures for common words and finger spelling for communication of obscure words or proper nouns. Conversants in ASL may describe a person, place, or thing and then point to a place in space to store that object temporarily for later reference.

Hidden Markov models have intrinsic properties which make them very attractive for sign language recognition. Explicit segmentation on the word level is not necessary for either training or recognition. Language and context models can be applied on several different levels, and much related development of this technology has already been done by the speech recognition community. Consequently, sign language recognition seems an ideal machine vision application of HMM technology, offering the benefits of problem scalability, well defined meanings, a pre-determined language model, a large base of users, and immediate applications for a recognizer.

Conversant in ASL may describe a person, place or thing and then point to a place in space to temporary store that object for later reference.[4]

Part of Speech	Vocabulary
Pronoun	I you he we you(pl) they
Noun	box car book table paper pants bicycle bottle can wristwatch umbrella coat pencil shoes food magazine fish mouse pill bowl
Verb	want like lose dontwant dontlike love pack hit loan
Adjective	red brown black grey yellow

Table 1: ASL Vocabulary used

In this system, sentences of the form "personal pronoun, verb. noun, adjective" are to be recognized. This sentence structure emphasizes the need for a distinct grammar for ASL recognition and allows a large variety of meaningful sentences to be randomly generated using words from each class. Table 1 shows the words chosen for each class. Six personal pronouns,nine verbs, twenty nouns, and five adjectives are included making the total lexicon number forty words.

2 Data-set

The National Center for sign language and Gesture Resources of the Boston University published a database of ASL sentences. Although this database has not been produced primarily for image processing research, it consists of 201 annotated video streams of ASL sentences.

The signing is captured simultaneously by four standard stationary cameras where three of them are black/white and one is a color camera. Two black/white cameras, placed towards the signers face, form a stereo pair and another camera is installed on the side of the signer. The color camera is placed between the stereo camera pair and is zoomed to capture only the face of the signer. The movies published on the internet are at 30 frames per second and the size of the frames is 312*242 pixels. We use the published video streams at the same frame rate but we use only the upper center part of size 195*165 pixels because parts of the bottom of the frames show some information about the frame and the left and right border of the frames are unused.

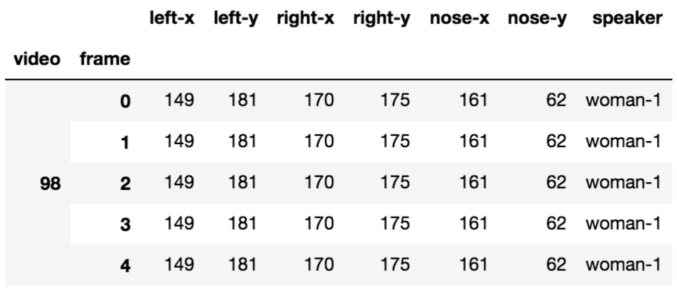
To create the RWTH-BOSTON-104 sign language database for ASL sentence recognition, we separated the video streams to training and test set which thetraining set consists of 161 sign language sentences and the test set includes 40 remaining sign language sentenses.

Link to the Dataset: [1]

[1]http://www-i6.informatik.rwth-aachen.de/ dreuw/databaserwth-boston-104.php

3 Feature Engineering

The database contains a number of videos which give the location of the left hand, right hand and nose of the signer. For example, in video 98, we have: The objective of feature engineering is to design features which contain



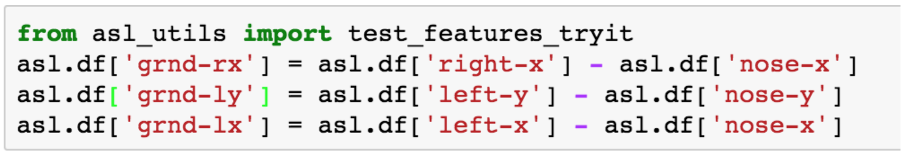
video	frame	left-x	left-y	right-x	right-y	nose-x	nose-y	speaker
98	0	149	181	170	175	161	62	woman-1
	1	149	181	170	175	161	62	woman-1
	2	149	181	170	175	161	62	woman-1
	3	149	181	170	175	161	62	woman-1
	4	149	181	170	175	161	62	woman-1

Figure 1: Data for frames 0,1,2,3,4 in video 98

highly relevant information, while also keeping the number of features as small as possible. Keeping the model as simple as possible also reduces training time. I will present 5 possible feature engineering techniques. A priori, its very difficult to determine which technique will be most promising. We will use a development set along with appropriate evaluation metrics to determine the effectiveness of each approach.

3.1 Grounded Positions

We could ground the right and left hand positions relative to the nose position, as opposed to an arbitrary origin. This new feature would capture the difference between the hand and nose position. The computation of these new features would be as follows:



```
from asl_utils import test_features_tryit
asl.df['grnd-rx'] = asl.df['right-x'] - asl.df['nose-x']
asl.df['grnd-ly'] = asl.df['left-y'] - asl.df['nose-y']
asl.df['grnd-lx'] = asl.df['left-x'] - asl.df['nose-x']
```

3.2 Normalized Positions

To account for speakers with different heights and arm lengths, we could use the standard score equation to normalize these differences. The computation would be as follows:

```
# adding features for normalized by speaker values of left, right, x, y
# using Z-score scaling (X-Xmean)/Xstd

asl.df['norm-lx']= (asl.df['left-x']-asl.df['speaker'].map(df_means['left-x']))/ asl.df['speaker'].map(df_std['left-x']]
asl.df['norm-ly']= (asl.df['left-y']-asl.df['speaker'].map(df_means['left-y']))/ asl.df['speaker'].map(df_std['left-y']]
asl.df['norm-rx']= (asl.df['right-x']-asl.df['speaker'].map(df_means['right-x']))/ asl.df['speaker'].map(df_std['right-
asl.df['norm-ry']= (asl.df['right-y']-asl.df['speaker'].map(df_means['right-y']))/ asl.df['speaker'].map(df_std['right-

features_norm = ['norm-rx', 'norm-ry', 'norm-lx','norm-ly']
```

3.3 Polar Coordinates

Another possible representation of the features would be to use polar coordinates. We swap the x and y axes in order to move the discontinuity to 12 oclock instead of 3 oclock. This moves the discontinuity directly above the speakers head, an area not generally used in signing.

```
# adding features for polar coordinate values where the nose is the origin
# Name these 'polar-rr', 'polar-rtheta', 'polar-lr', and 'polar-ltheta'
# Note that 'polar-rr' and 'polar-rtheta' refer to the radius and angle

asl.df['polar-rr']= (asl.df['grnd-rx'].apply(np.square)+asl.df['grnd-ry'].apply(np.square)).apply(np.sqrt)
asl.df['polar-rtheta']=  np.arctan2(asl.df['grnd-rx'], asl.df['grnd-ry'])
asl.df['polar-lr']= (asl.df['grnd-lx'].apply(np.square)+asl.df['grnd-ly'].apply(np.square)).apply(np.sqrt)
asl.df['polar-ltheta']= np.arctan2(asl.df['grnd-lx'], asl.df['grnd-ly'])

features_polar = ['polar-rr', 'polar-rtheta', 'polar-lr', 'polar-ltheta']
```

3.4 Delta Differences

Yet another possibility is to use the difference in values between one frame and the next frame as features. The idea is that relative movements are more important than grounded movements. The implementation is simply:

```
# Name these 'delta-rx', 'delta-ry', 'delta-lx', and 'delta-ly'
#.fillna(method='backfill').fillna(method='ffill')
asl.df['delta-rx'] = asl.df['right-x'].diff(periods=1).fillna(method = 'backfill')
asl.df['delta-ry'] = asl.df['right-y'].diff(periods=1).fillna(method = 'backfill')
asl.df['delta-lx'] = asl.df['left-x'].diff(periods=1).fillna(method = 'backfill')
asl.df['delta-ly'] = asl.df['left-y'].diff(periods=1).fillna(method = 'backfill')

features_delta = ['delta-rx', 'delta-ry', 'delta-lx', 'delta-ly']
```

3.5 Normalized Polar Coordinates

Finally, I combine elements of normalization and polar coordinates in the obvious way. The basic idea here is that polar coordinates can move the discontinuity to a more favourable location and normalization scales the values appropriately.

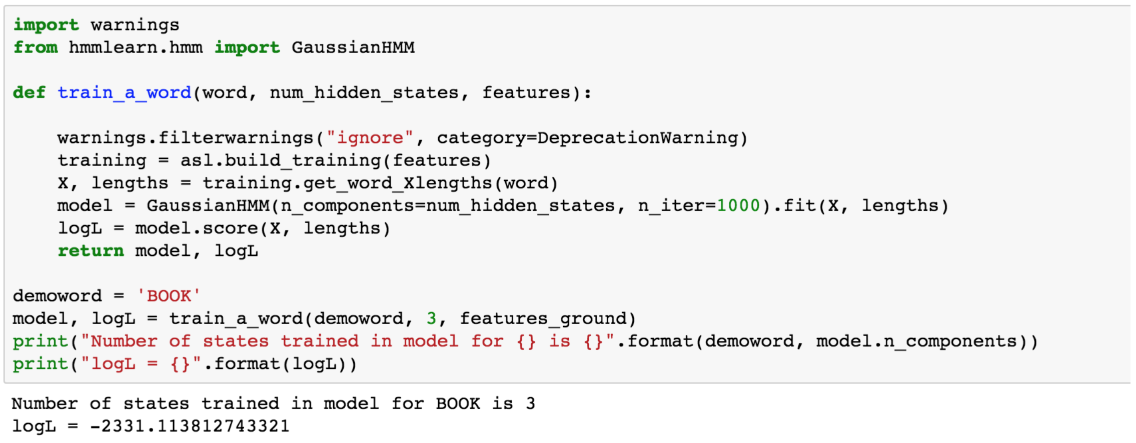
```
# adding features of your own design, which may be a combination of the above or something else,
#I tried Norm-Polar combination
# Name these whatever you would like
# diff in polar coord
df_means = asl.df.groupby('speaker').mean()
df_std = asl.df.groupby('speaker').std()
asl.df['custom2-rr'] = (asl.df['polar-rr']-asl.df['speaker'].
                        map(df_means['polar-rr']))/ asl.df['speaker'].map(df_std['polar-rr'])

asl.df['custom2-rtheta'] = (asl.df['polar-rtheta']-asl.df['speaker'].
                            map(df_means['polar-rtheta']))/ asl.df['speaker'].map(df_std['polar-rtheta'])

asl.df['custom2-lr'] = (asl.df['polar-lr']-asl.df['speaker'].
                        map(df_means['polar-lr']))/ asl.df['speaker'].map(df_std['polar-lr'])

asl.df['custom2-ltheta'] = (asl.df['polar-ltheta']-asl.df['speaker'].
                            map(df_means['polar-ltheta']))/ asl.df['speaker'].map(df_std['polar-ltheta'])

# defining a list named 'features_custom' for building the training set

features_custom_2 = ['custom2-rr', 'custom2-rtheta', 'custom2-lr', 'custom2-ltheta']
```

4 Model Selection

Model selection is a process of seeking the model in a set of candidate models that gives the best balance between model fit and complexity [5]. I have always used AIC for that. But you can also do that by cross-validation. Specifically, Stone [6] showed that the AIC and leave-one out cross-validation are asymptotically equivalent. I wanted to experience it myself through a simple exercise. The objective here is to choose the appropriate number of states in the HMM model for each word. The evaluation metrics we will consider are:

1. Log likelihood using cross-validation folds (CV)
2. Bayesian Information Criterion (BIC)
3. Discriminative Information Criterion (DIC)

In order to fit a HMM to a single word we will use the hmmlearn library in Python. The fit function will invoke Baum-Welch Expectation-Maximization to iteratively find the best estimate for the model for a given number of hidden states. We could train the word BOOK as follows:



```
import warnings
from hmmlearn.hmm import GaussianHMM

def train_a_word(word, num_hidden_states, features):

    warnings.filterwarnings("ignore", category=DeprecationWarning)
    training = asl.build_training(features)
    X, lengths = training.get_word_Xlengths(word)
    model = GaussianHMM(n_components=num_hidden_states, n_iter=1000).fit(X, lengths)
    logL = model.score(X, lengths)
    return model, logL

demoword = 'BOOK'
model, logL = train_a_word(demoword, 3, features_ground)
print("Number of states trained in model for {} is {}".format(demoword, model.n_components))
print("logL = {}".format(logL))
```

```
Number of states trained in model for BOOK is 3
logL = -2331.113812743321
```

For this example, we assume the correct number of hidden states is 3, but that is just a guess. How do we know what the ”best” number of states for training is? We will need to find some model selection technique to choose the best parameter.

The HMM model is trained and information can be pulled from the model, including means and variances for each feature and hidden state.

The log likelihood for any individual sample or group of samples can also be calculated with the score method.

4.1 Evaluation Metrics

4.1.1 Log likelihood using cross-validation folds (CV)

This approach simply loops over all possible folds of the data and computes an average probability of the data fitting the model. We would simply choose the number of states which maximizes this average probability.

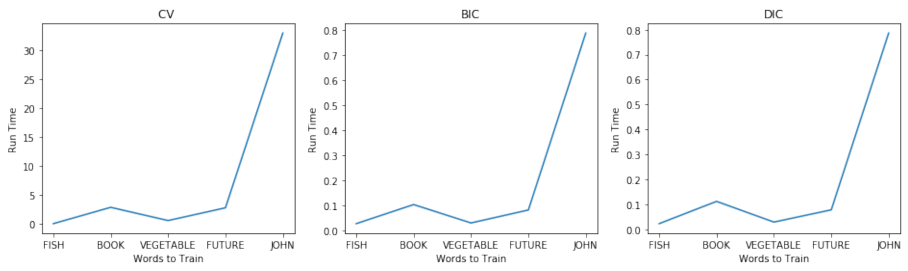
4.1.2 Bayesian Information Criterion (BIC)

We choose the number of states which gives the lowest Bayesian information criterion. The formula for BIC is -2 * $\log L$ + p * $\log N$, where p is the number of parameters in the model and N is the number of hidden states. As seen in the formula, model complexity applies a penalty in the form of $p * \log N$.

4.1.3 Discriminative Information Criterion (DIC)

Here we choose the number of states which gives the highest DIC. The formula for DIC is $\log(P(originalworld))$—average($\log(P(otherwords))$). The idea of DIC is that we are trying to find the model that gives a high likelihood to the original word and a low likelihood to the other words. In other words we are maximizing the difference between the probability that the model fits the original word and the average probability that the model fits the other words.

4.1.4 Which metric should I use?



When it come to speed SelectorCV is definitely the loser. It is a whole magnitude higher than the BIC Selector and DIC Selector times. The outcomes of the BIC Selector and DIC Selector are identical. However, the DIC Selector is slightly faster than the BIC Selector. So DIC Seems to be the optimal selection algorithm.

Cross validation scoring works by choosing the model with the highest average likelihood by moving over all possible hold-out possibilities. It is not biased in the sense that it considers all possible testing groups in computing a likelihood. It is simple to interpret as well. BIC does not use a hold out set. It penalizes generalization by accepting for model complexity in the formula.

BIC is a method for choosing a model with the best trade-off between fit and complexity, and cross-validation is a method for choosing a model with the best out-of-sample predictive accuracy.

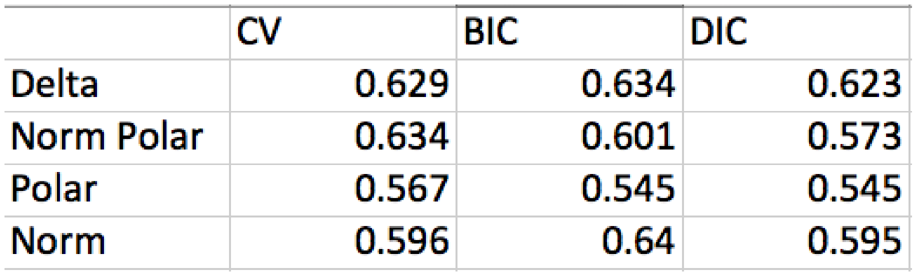
The disadvantage of BIC is that there is no guarantee that the complexity penalty will exactly offset the overfitting property. If you only care about model generalization, then cross-validation is the best option. On the other hand, DIC does not consider model complexity. The idea behind DIC is that you are trying to find the number of components which gives a high score to the current word and a low average score to the words scored against this model.

Its basically saying that you want your model to give a high probability to the word you are currently scoring, and a low probability to the other words.

There's a lot of discussion going on which metric is the best. I would argue that cross validation is the most effective approach due to its simplicity and unbiased scoring metric, which ultimately maximizes generalization potential.

5 Results

I used a development set containing 180 sentences to test all 12 combinations of the features and evaluation metrics discussed above. The results are as follows:



	CV	BIC	DIC
Delta	0.629	0.634	0.623
Norm Polar	0.634	0.601	0.573
Polar	0.567	0.545	0.545
Norm	0.596	0.64	0.595

The best combination was (Polar, DIC) and (Polar, BIC) because these have the lowest word error ratio (WER) on words not already seen by the model. The polar coordinates are shown to have the highest predictive power out of the features I considered. In general, DIC seems to be the best overall and BIC seems to be the worst. After running the tests a few times, I decided to implement (Polar, DIC) as my final model because it was slightly better on average. Note that the average sentence length was about 5 words. If we just guessed randomly we would get the correct sentence with probability $(1/500)^5$ since there are 500 words in the database. Thats a WER of 1.0.

6 Conclusion

I built a system which converts ASL video sequences into text sentences. Feature engineering was used to maximize generalization potential. Both the quality and quantity of the features had great influence on the effectiveness of the model.

Finally, appropriate evaluation metrics were used for hyperparameter tuning. The results indicate that polar coordinates and the DIC metric were most effective at model generalization capabilities.

For additional predictive power, one should implement a temporal model which considers the time series nature of sentences. We are currently using a 1-gram model. But one should use an n-gram model, where n ¿ 1.

Real time systems can be implemented using the algorithms, as stated in these papers [9,10].

References

[1] Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, douard Duchesnay. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830 (2011)

[2] Comaniciu D. and Meer P. (2002) Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[3] Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56

[4] George Sperling, Michael Landy, Yoav Cohen, and M. Pavel. 1986. Intelligible encoding of ASL image sequences at extremely low information rates. In Papers from the second workshop Vol. 13 on Human and Machine Vision II, Azriel Rosenfeld (Ed.). Academic Press Professional, Inc., San Diego, CA, USA, 256-312.

[5] Burnham, K.P. and Anderson, D.R. (2002) Model Selection and Inference: A Practical Information-Theoretic Approach. 2nd Edition, Springer-Verlag, New York. http://dx.doi.org/10.1007/b97636

[6] Stone, Mervyn . 1977. An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaikes Criterion. Journal of the Royal Statistical Society, Series B 39:44-47.

[7] Lunds university. Model Selection http://www2.imm.dtu.dk/courses /02433/doc/ch6_slides.pdf

[8] Alain Biem. 2003. A Model Selection Criterion for Classification: Application to HMM Topology Optimization. In Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 1 (ICDAR '03), Vol. 1. IEEE Computer Society, Washington, DC, USA, 104-.

[9] Chuan C.-H., Regina E., Guardino C. American Sign Language Recognition Using Leap Motion Sensor; Proceedings of the 13th International Conference on Machine Learning and Applications; Detroit, MI, USA. 35 December 2014;

[10] Chong T.-W., Lee B.-G. American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach. Sensors. 2018;18:3554. doi: 10.3390/s18103554.