

Hidden Markov Models Experiment Report



Nisarg Negi

Department of Computer Science and Engineering

University at Buffalo, Buffalo, NY 14260

[nisargne@buffalo.edu](mailto:nisargne@buffalo.edu)

1. Introduction

Hidden Markov model(HMM) is a discrete sequential data modelling method which works on the assumption that the observed events depend on a trait which are not directly observable. This is where the hidden part of the model comes from.

As for the Markov part, it comes from how we model the hidden states. We consider the Markov property from probability theory which is also known as the memoryless, i.e. the next hidden states only depends on the current state and is not affected by any other states.

HMM’s are applied in speech recognition, activity recognition from video, gesture tracking, gene tracing.

HMM is used as a tool to represent probability distributions of sequential data. In HMM we take a stochastic process which has an observation Xt at a time t, here we have an unknown hidden state Zt. We assume that Zt satisfies the Markov property and Zt at time t is hence only dependent on the previous state Zt-1 at time t-1. This is the first order of Markov model. Hence, nth Markov model would depend on the previous n states.

Below is an image of a Bayesian network representing the first order HMM. Shaded states are the hidden states.

A picture containing text, clock

Description automatically generated

Hence, Joint distribution of a sequence of states and order for first order HMM:

Text

Description automatically generated

2. Dependence

HMM is characterized by the below five elements:

1. Number of states(N): The number of states that a hidden Markov model has is also related to the problem being modeled. We will encode the state Zt at time t as a K x 1 vector of binary numbers.
2. Number of Distinct observations(Ω): Observations are the number of disctinct observations in the experiment. We will encode Xt at a time t as Ω x 1 vector of binary numbers, where the only non-zero element is the only lth element.
3. State transition model(A): This is a N x N matrix with elements Aij containing the probability of transitioning from Zt-1,i to Zt,j in one time steps. It is written as:



Row sum of each row in the State transition matrix sums to 1, hence it is a stochastic matrix.

 A drawing of a pair of glasses

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidenceAbove is a 3 state stochastic matrix for which

If in one jump a state can be reached from another state the value of Aij will be > 0

else it will be zero. Hence, the state transition model for the above diagram will be

A picture containing text

Description automatically generatedConditional probability can be written as

Text

Description automatically generated with medium confidence

We can take its log and write it as :

1. Observation model(B): It is a matrix that describes the emission probabilities of the observable process(with elements Xt,k. Elements of this matrix are given as Bk,j at state Zt.j, and the matrix B is calculates as B = Ω× K.



Elements of B i.e. Bk,j can be written as:

A picture containing text

Description automatically generatedConditional probabilities can be written as:

We can rake its log and write it as: Text

Description automatically generated

A picture containing text

Description automatically generated

1. Initial state distribution(π): Initial distribution of the states(π) is used for modeling where πi = P(Z1i=1| π) =

These 5 parameters can be used to specify the HMM model as, λ = (A, B, π).

3. Three classes of Problems that can be solved using HMM

a. Known a set of observations Ω and 3 model parameters 𝝅, A and B, we can find out the probability of occurrence of X. We can find this by using forward filtering.

b. Known a set of observations Ω and 3 model parameters 𝝅, A and B, we can find out the optimal set of hidden states Z that result in X. We can find this by using Viterbi algorithm.

c. Known a set of observations Ω we can find out the optimal parameters 𝝅, A and B using Baum-Welch algorithm.

4. Algorithms

Graphical user interface, application, Word

Description automatically generateda. Forward Backward Algorithm: Forward-backward algorithm is a dynamic programming algorithm which uses belief propagation. It computes the filtered and smoothed marginals that are used to perform inference, sequence classification, MAP estimation, anomaly detection and clustering.

Text

Description automatically generatedb. Viterbi Algorithm: To compute the set of most probable sequence of hidden states(problem b) we will use the Viterbi algorithm. It uses the trellis diagram of HMM to compute the shortest path. Trellis diagram connects each state of the next time step in the model. Next, we use the forward-backward algorithm but using max-product instead of sum-product algorithm.

Text, letter

Description automatically generatedc. Baum-Welch Algorithm: Baum-Welch Algorithm is a dynamic programming algorithm which uses Expectation Maximization algorithm to tune parameters for HMM. It is used to find optimized values of A, B & 𝝅 such that the model data overlaps with the actual data.

5. Limitation:

HMM model considers only the last state. If we want to consider the last 2 states the training time grows exponentially considering that we have to build a set of cartesian multiplication of all states. Hence, our time complexity will become N2 for training. For this we can use RNN as it can use continuous states.

6. Dataset:

We will use the silver dataset from Kaggle. It has the data of daily silver price changes from January 2000 to September 2022.

Feature:

Date: Date of data instance

Open: Opening price of Silver per ounce in US Dollar

High: Highest price of Silver per ounce in US Dollar

Low: Lowest price of Silver per ounce in US Dollar

Close: Closing price of Silver per ounce in US Dollar

Volume: Ounces of Silver sold

Currency: Currency of transaction

7. Experiment

We will analyze historical silver prices using hmmlearn, downloaded from: <https://www.kaggle.com/datasets/psycon/daily-silver-price-historical-data>.

We will import the necessary libraries and also import hmmlearn. We will calculate daily change in the silver prices as it is a better way of modelling stock like data which changes daily. We will consider data after 2008, ignoring the 2008 financial crisis.

We will plot the silver price and change in silver price from 2008.

Chart, histogram

Description automatically generated

Then, we will use a Gaussian model and fit the daily silver price change with 3 hidden states. We select 3 hidden states as we expect 3 different variety of data, namely high, medium low.

Fitting will result in three unique hidden states

Text

Description automatically generated with low confidence

We see that for this our data, the model starts mostly with state 1.

Graphical user interface, text, application, email

Description automatically generated

Next we look at our Gaussian Mean:

Graphical user interface, text, application

Description automatically generated

For state 1, the mean in 0.027 for state 2 our mean is -0.013, for state 3 our mean in -0.013. State 0 & 2 have similar mean; hence our model might not be good at representing the data.

Next we will look at the Gaussian covariance:

Graphical user interface, text, application, email

Description automatically generated

As our data is one dimensional, our covariance matrix will be scalar(not vector).

For state 1 the covariance is 0.026, for state 0 the covariance is 0.127 and for state 2 the covariance 0.795. From this we can tell that for state 1 the volatility is low and for state 2 the volatility is high.

Next we will plot our models prediction :

Chart

Description automatically generated

We can tell that states 1,0,2 represent low, medium and high volatility.

From our plot we can see, the period of high volatility due to recession during 2011-2012 and the high volatility during 2020-2021 due to covid crisis.

We can also tell that the cost of silver increases during these highly volatile periods as investors purchase commodities instead of equity when the markets collapse.

8. **References:**

<https://www.kaggle.com/datasets/psycon/daily-silver-price-historical-data>

<https://scholar.harvard.edu/files/adegirmenci/files/hmm_adegirmenci_2014.pdf>

<https://medium.com/@natsunoyuki/hidden-markov-models-with-python-c026f778dfa7>

<https://medium.com/analytics-vidhya/baum-welch-algorithm-for-training-a-hidden-markov-model-part-2-of-the-hmm-series-d0e393b4fb86>



Random Forest Classifier Models Experiment Report



Diagram

Description automatically generated1. Introduction

Random Forest algorithm is a type of supervised machine learning algorithm. It has two variations: Classifier and Regressor.

Random forest Classifier uses ensemble learning i.e. many algorithms are run or one algorithm is run multiple times to create several decision trees from random subsets of the data. It uses the average from these decision tree predictions to predict the final class of the test object. This selection from several decision trees is known as voting.

As Random Forest algorithm uses several random decision trees, it is known as Random Forest algorithm.

2. Algorithm

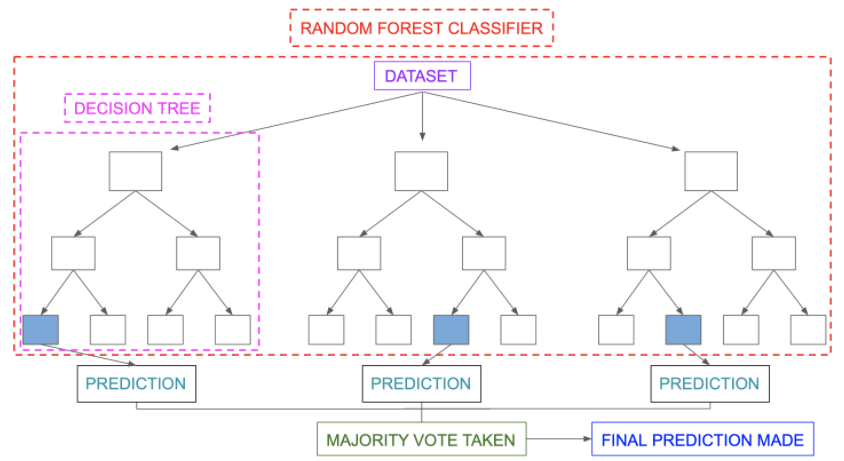
Random Forest algorithm works in the following steps:

Starts by selecting random samples from the data

It constructions decision trees for all the samples and gets prediction results from all the samples.

It performs voting for each predicted results.

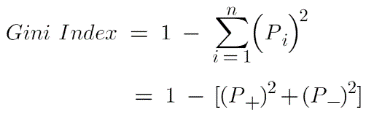
It ends by selecting the most voted prediction result.



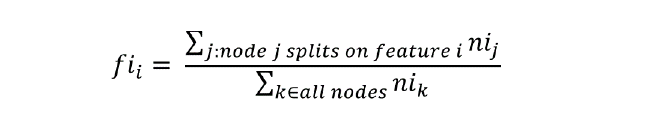
If we use a high depth decision tree we will get low bias and high variance, which means that our model will be overfitted and predictions of unknown data will not be accurate. To solve this problem we use Random forest by combining many decision trees instead of one decision tree.

Gini Index:

While splitting the data into decision trees, the algorithm uses the Gini index to get a pure split. A pure split is a split in which one node has one type of categorical data and another has a different.

 Here, P+ is the probability of a positive class and P\_ is the probability of a negative class.

We, choose the decision tree feature to split which gives us the minimum Gini index(impurity) for the root.

Feature Selection: We can perform an initial run of the model to get the important features that we should use for the main model.

3. Experiment

We will use the breast tissue dataset to classify the data into different classes using Random forest classifier. The target variable here is class.

Table

Description automatically generated

A picture containing shoji, square, silver

Description automatically generatedFor the initial analysis we will look

up for null or missing values in the

dataset. We will move on to the

analysis of the data. The data has 106

instances of electrical impedance

measurements of freshly excised

breast tissues. There are 9 features

and 1 target class data column.

There are a total of 6 classes.

From the pairplot we can analyse

That our target variable P & IO are

In a direct linear relationship.

A picture containing table

Description automatically generated

From the correlation plot we can tell that IO is highly correlated with our target variable. We can also observe that DR & DA are highly correlated and hence one of them can be removed from our analysis as removing one of them will decrease complexity without much affect to the result.

Hence, the most optimum features we can use are IO, DA, A/DA, MAX IP, DR.

Table

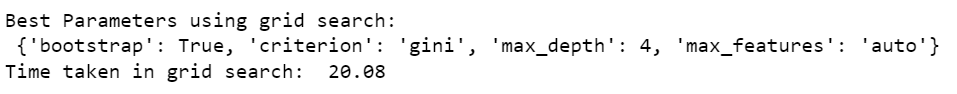
Description automatically generated

For more observations we can also use the describe function for data frame.

Then we shall remove the ‘case#’ variable split our data into test and training in the ration of 20% & 80% respectively, while also deleting the ‘class’ column, our target data, from the test data.

We will then perform Hyper-Parameter Optimization using GridSearchCV to find the best parameters to use in our model.

Chart, funnel chart

Description automatically generated

Next we will train our model with the training data and use feature\_importances\_ from sklearn to find the best features for our data and plot the importance of each feature.

We can tell that the feature ‘P’ is the most important and the feature HFS is the least. We will remove ‘HFS’ from out dataset.

Using ‘HFS’ removed we will again train our model with the best parameters given by grid search.

A picture containing text

Description automatically generated

We got an accuracy score of 0.773, this does not make much sense so we will plot the results in a confusion matrix.

Table

Description automatically generatedGraphical user interface, application

Description automatically generated

Our confusion matrix looks good and acceptable, there are not many false

positive and false negatives.

From sklearn.metrics we can find the accuracy of our model as below:

4. References

1. <https://medium.com/analytics-vidhya/random-forest-classification-and-its-mathematical-implementation-1895a7bb743e>

2. <https://www.kaggle.com/code/prashant111/random-forest-classifier-tutorial>

3. <https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-random-forest-algorithm-for-beginners/>



Autoencoder Experiment Report



1. Introduction

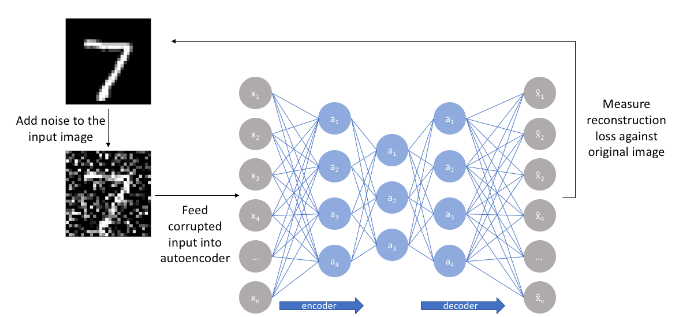
Autoencoder is a type of Artificial neural network which has the ability to learn the input representation. It can encode and compress data efficiently and then learn from the process how to reverse it and construct the input from the compressed data.

There are three main parts in an autoencoder: encoder, decoder and latent block(or bottle neck . The encoder extracts the best features, the latent space is used for storage of these features, and the decoder does the reverse of encoder and reconstructs an image from using the features.

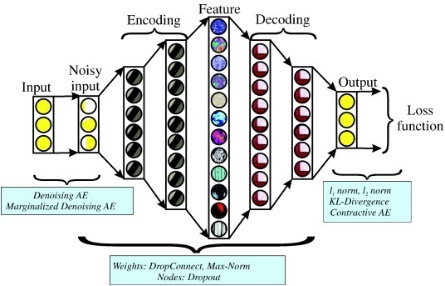
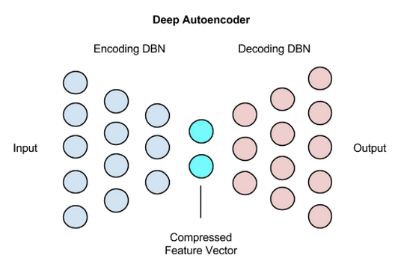
Diagram

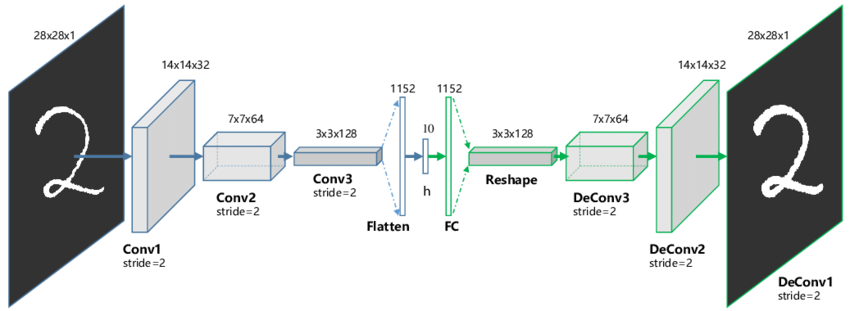
Description automatically generated

One application of Autoencoder is denoising data. Data can lose quality over transmission on the internet. Autoencoders can be used to get back the lost quality.

1. Types of Autoencoder:
2. Denoising Autoencoder: in this encoder we create a copy of the input data with some noise added. This avoids the generalization of the input and makes the models learn features from the input data and train to recover the original data.

It minimizes the loss function between the output and the noisy input. It is easy to setup. The only drawback is that his model does not memorize the training as the input and output keeps changing

1. Sparse Autoencoder: in this encoder, we have sparsity penalty in the training criterion. There are two ways to add sparsity: L1 regularization and KL-divergence. Adding sparsity helps to activate only fewer but insightful nodes, which helps the model learn from latent representations instead of redundant nodes. This also prevents overfitting.
2. Deep Autoencoder: It works by creating two identical deep belief networks: one for encoding and one for decoding. Unsupervised layer by layer pretraining is used for this model. Each is a Restricted Boltzmann Machine which is the base of deep belief network, a binary transform is performed after each layer. Encoding is fast using Deep autoencoder and they are used in topic modeling across a set of documents.
3. Contractive Autoencoder: It is just like sparse autoencoder, but here the regularizer used is the calculated by the Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input. This model is robust, and less sensitive to variations.
4. Undercomplete Autoencoder: They have a small number of hidden layers which helpt to capture the most important features in the data. They do not need any regularization as they maximize probability.



1. Convolutional Autoencoder: Convolutional Autoencoder uses a mix sum of signals to reconstruct the input data rather than taking them one at a time like other models. It uses convolution neural network, which is better at retaining connected information between the data.
2. Experiment

We will use the ‘Labeled Faces in the Wild’ dataset. It contains the photographs of faces used for face recognition. We will use this dataset to build a model for image resolution enhancement.

First, we will download the data set using wget and extract it.

A picture containing table

Description automatically generated

Text

Description automatically generated

We will reduce the size of our images as per the limitations of our device and to reduce the training time. I have set it at 80x80.

Then we will split the date into training and test. We will keep aside the test dataset and used it for model evaluation.

We then reduce the resolution of the images to 40% so as to add noise in the training data, which we will later enhance using our model. We will use this low resolution image dataset as training data for our model.

We will then define our encoding model.

Text

Description automatically generated with medium confidence

Our encoder consists of 2 Convolution neural networks followed by a max pool layer and then again a Convolution neural network. We have used the activation function as relu for all layers.

We will then define our decoding model.

Text

Description automatically generated

Our decoder consists of 1 Convolution neural network layer followed by an up-sampling layer and then 2 Convolution neural networks. We have used the activation function as relu for all layers.

Our model summary is:

Table

Description automatically generated

I have set the model parameters as epochs = 5 and batch size = 256 depending on the limitations of my machine.

Text

Description automatically generated with medium confidence

Now we can use the trained model on our test dataset:

A collage of a person

Description automatically generated with medium confidence

4. References

1. <https://www.analyticsvidhya.com/blog/2020/02/what-is-autoencoder-enhance-image-resolution/>

2. <https://medium.com/ai%C2%B3-theory-practice-business/understanding-autoencoders-part-i-116ed2272d35>

3. <https://blog.francium.tech/enhance-images-with-autoencoders-58afa4fd638f>

4. <https://medium.com/@syoya/what-happens-in-sparse-autencoder-b9a5a69da5c6>

5. <http://vis-www.cs.umass.edu/lfw/lfw.tgz>



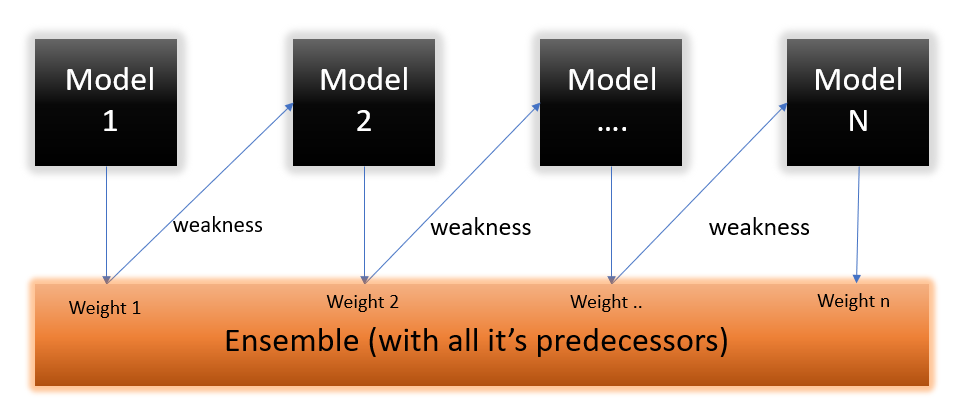
AdaBoost Experiment Report



1. Introduction

Adaptive Boosting is a machine learning algorithm used in ensemble method. It iteratively combines and learns from the mistakes of weak classifiers and promote them into strong ones. It can use any classifier but the most commonly used is Decision tree with one 1. This is called a decision stump.

Icon

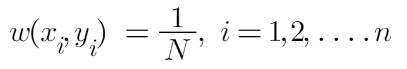
Description automatically generated with medium confidence

1. Algorithm

Steps for the algorithm:

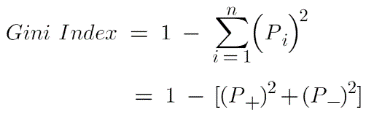
1. A decision stump is created which is a weak classifier on the training data based on the weights and are given equal weights for the initial stump.

Equation for assigning weights:



N: number of datapoints

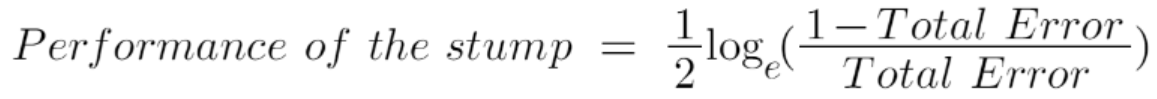
1. Decision stumps are created for each variable. Each decision stump is analyzed to see how well it classifies the data to the target class. This analysis is done by Gini index of each stump.

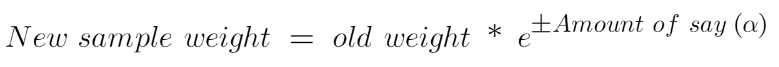


Here, P+ is the probability of a positive class and

P\_ is the probability of a negative class.

1. We increase the weights of the stumps which classified incorrectly, this in turns make them classify correctly in the next decision stump. More accurate accuracy stumps are given more weights as well.





1. We iterate step 2 and 3 until all the data has been correctly classified.

Text, letter

Description automatically generated

ADA Boosting steps with mathematics:

1. Experiment

We are using asteroseismology data which has the distinct features of oscillations of 6008 Kepler stars and using it to predict whether it is a Red Giant or a Helium burning star using ADA bosting.

Feature:

**Table

Description automatically generatedPOP**: Target Variable, if 0: Red giant star & 1: Helium burning star

**Dnu**: Mean large frequency separation of modes with the same degree and consecutive order

**Numax**: Frequency of maximum oscillation power

**Epsilon**: Location of the l=0 mode

Table

Description automatically generatedWe will read the dataset from csv and store it into a data frame. Then we will perform our initial analysis of the data. We can see that all the features are integer or float so will not require any labeling or one hot encoding them.

Then we will look for null values. We can tell that the data set does not have any null values and will not require imputation.

A picture containing building

Description automatically generated

We will perform bivariate analysis using pair plot. Pairplot helps us plot pairwise bivariate distributions in a dataset which helps us summaries the data visually. The x & y axis have all the features plotting against the target variable POP and the diagonal has the distribution of each element against the target variable POP.

A picture containing logo

Description automatically generated

Then we will check for the count of each classification in the target variable POP to check if the data is not highly imbalanced. Our data is not highly imbalanced.

A screenshot of a computer

Description automatically generated with low confidenceWe will then plot the correlation for each variable in the dataset. This helps us find highly correlated features which we can remove from our model training as they will not affect our training by much.

We will then split our dataset into test(30%) and training(70%)

We train the ADA boost model from our training set with n\_estimators = 100

Chart, treemap chart, square

Description automatically generatedText

Description automatically generated

Now we will use our model to predict our test data and evaluate it using a confusion matrix:

And check our accuracy scores:  
Table

Description automatically generated

Shape, square

Description automatically generatedGraphical user interface, text

Description automatically generated

Our ROC/AUC curve fits as:

We can see that we have got a good accuracy score although it is a little overfit and our AUC is close to one so we can say that our model is good for use

1. References
2. <https://blog.paperspace.com/adaboost-optimizer/>
3. <https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/>
4. <https://towardsdatascience.com/adaboost-for-dummies-breaking-down-the-math-and-its-equations-into-simple-terms-87f439757dcf>
5. <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>