ECE 759 Project Presentation

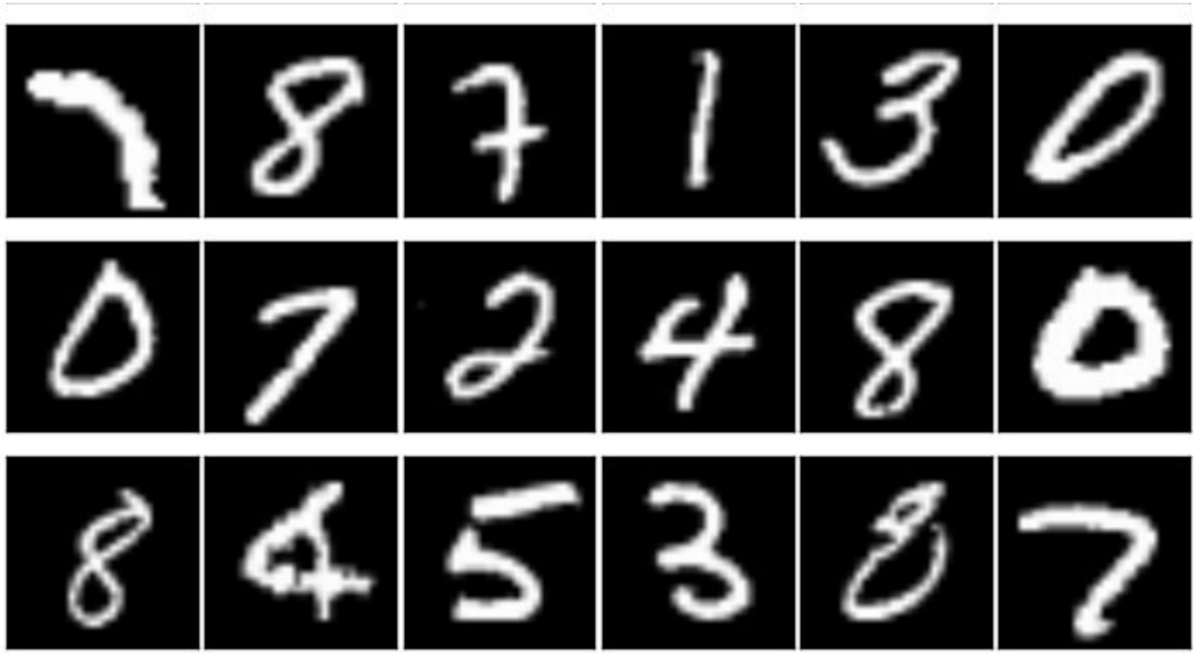
Team 17 (Matthew Conrad and Evan Williams)

Outline

- Datasets
- Feature Extraction
- Algorithm Overview
- Results
- Questions

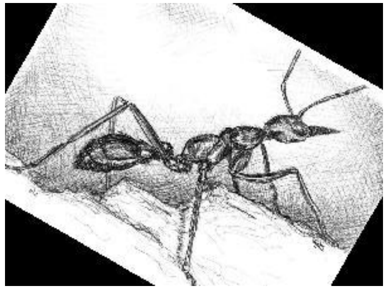
MNIST

- Handwritten digits
- 60,000 images
- 28 x 28 pixels
- Grayscale



Caltech10

- Derivative of Caltech101
- Different objects
    - Ant, bass, butterfly, camera, chair, crab, dolphin, elephant, sunflower, yin & yang
- 646 images
- Varying sizes
- RGB & Grayscale



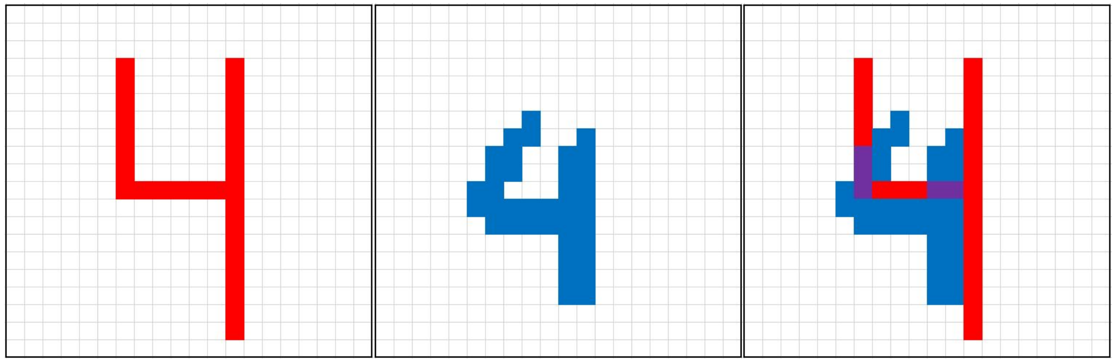



Data Preparation

- 50/50 data split of each data set into training/testing data while retaining the distribution of classes in the initial dataset
- MNIST
  - 30000 training samples
  - 30000 testing samples
- Caltech
  - 325 training samples
  - 321 testing samples

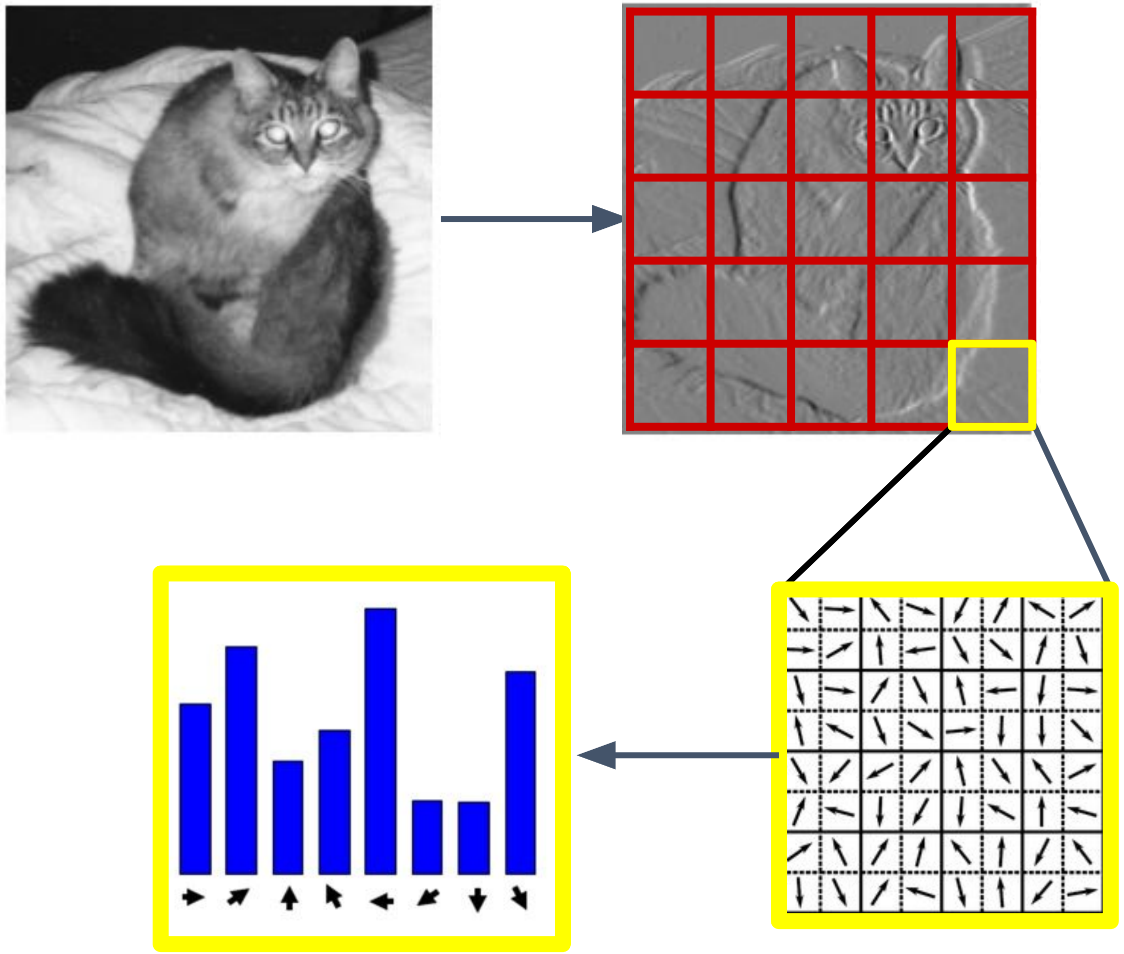
MNIST Feature Extraction

- Cannot use raw pixel values
- No color or texture
- Must rely on shape feature descriptors



Histogram of Oriented Gradients (HOG)

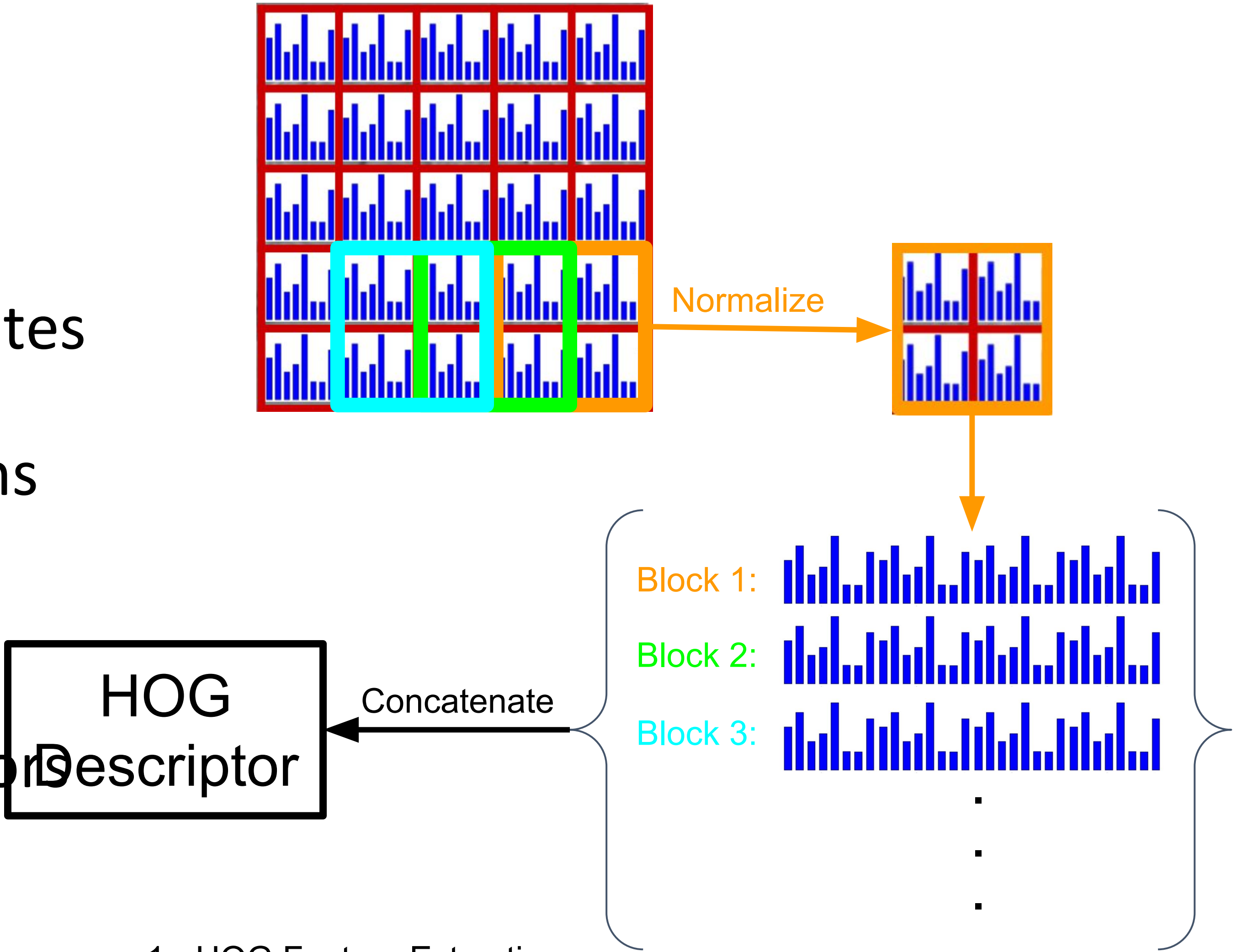
- Find gradient of image
- Create cells
- For each cell, construct histogram from weighted votes



1 - HOG Feature Extraction

Histogram of Oriented Gradients (HOG)

- Find gradient of image
- Create cells
- For each cell, construct histogram from weighted votes
- Create overlapping blocks
- Normalize the cell histograms in each block
- Vectorize the block
- Create HOG descriptor by concatenating all block vectors



1 - HOG Feature Extraction

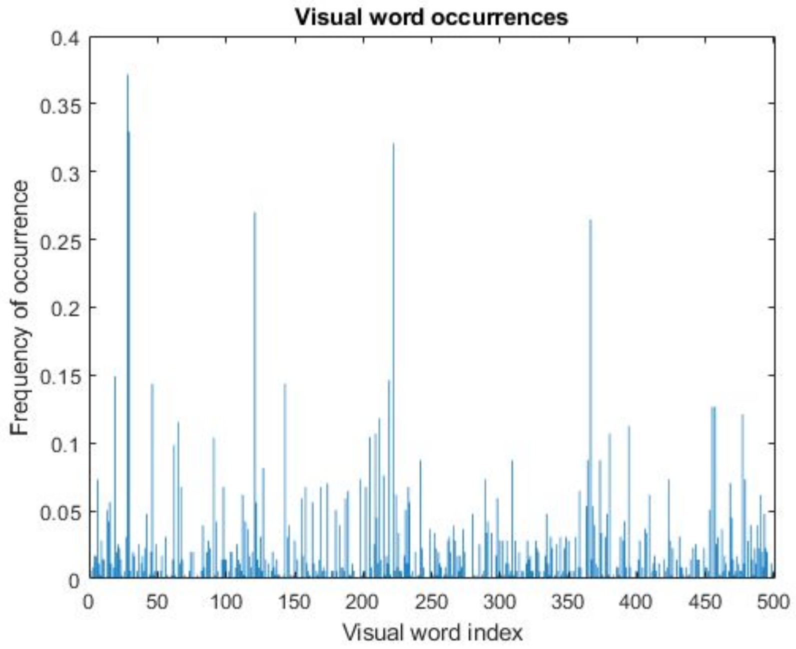
Caltech Feature Extraction

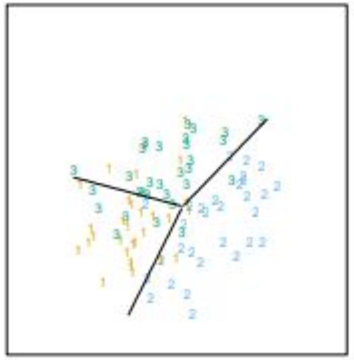
- Conversion from RGB to Grayscale
- Use Bag-of-Features (BoF) instead of HOG
- BoF comes from Bag-of-Words (BoW) which is illustrated as follows:
  - “John likes to watch movies.” “Mary likes movies too.”

Vocabulary	John	Mary	likes	watch	movies	to	too
Vector 1:	1	0	1	1	1	1	0
Vector 2:	0	1	1	0	1	0	1

Caltech Feature Extraction

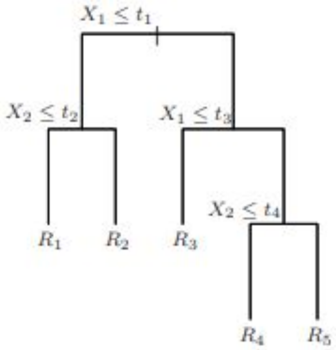
- How "Vocabulary" is created
    - Create SURF features from ALL images
        - SURF = like HOG but local and more robust
    - Put all features in the same space
    - Cluster similar vectors into "words"
        - # Clusters = # Words
- How to create feature vector for an image
    - Run SURF on image to get features
    - Map each feature to a "word"
    - Create histogram of how many of each "word" is in the image
    - Histograms = feature vectors for classification





Example of LDA classifier
3 - Hastie et. al

Algorithm Overview



Example of Decision Tree classifier
3 - Hastie et. al

LDA

- Linear Classifier
- Assumes the data is distributed in a multivariate gaussian distribution and the classes have a common covariance matrix
- Does not have any hyperparameters
- Trains linear discriminant functions to classify data

Decision Tree

- Nonlinear Classifier
- Objective function to build the tree is to maximize the information gain at each split
- Hyperparameters include MaxDepth, MinimumNodeSize, MaxSplits, and Stopping Criteria
- Builds a tree structure to classify data

Algorithm Pseudocode

Algorithm 2: Linear Discriminant Analysis for multi-class classification

<u>function LDA</u> ($TrainFeatures, TrainLabels, TestFeatures$);

Input : Feature Matrices ($TrainFeatures$) and ($TestFeatures$) as well as Class Labels for Training Data ($TrainLabels$)

Output: Predicted Class Labels of Testing Data ($TestPredictions$)

First, determine the number of unique classes (K) using the Training Labels.

for *each class k to class K* **do**

  Calculate the class mean $\hat{\mu}_k$, pooled covariance $\hat{\Sigma}$, and class prior $\hat{\pi}_k$ using equations 6-8

end

Now that the parameters have been estimated for each class we can go through and calculate the discriminant functions for each class. To clarify we note that the pooled covariance matrix is initially a zero matrix of size which is iteratively updated within the first for loop.

for *each class k up to class K* **do**

  Calculate the discriminant function for each class k using equations 3 and 4.

end

Next, evaluate the testing data using the learned model.

for *testing point i in M2 testing points* **do**

  for *each class k up to class K* **do**

    Calculate the likelihood of the datapoint i belonging to class k as seen in equation 5

  end

  Predict each testing point i as the class with the argmax of the likelihood scoring

end

Finally, return the vector of M2 predictions of the testing data as your output.

Algorithm 3: Decision Tree for multi-class classification

<u>function GreedyDecisionTree</u> ($TrainFeatures, TrainLabels, TestFeatures, MaxSplits, StoppingCriteria, MaxDepth$);

Input : Feature Matrices ($TrainFeatures$) and ($TestFeatures$) as well as Class Labels for Training Data ($TrainLabels$). ($MaxSplits$,$StoppingCriteria$,$MaxDepth$) are hyperparameters that denote the maximum number of splits before the tree stops growing, the error threshold at which the tree stops growing, and the maximum depth that a tree can grow respectively.

Output: Predicted Class Labels of Testing Data ($TestPredictions$)

First, determine the number of unique classes (K) using the Training Labels and initialize a Tree Data Structure with a RootNode.

while *SplitCount < MaxSplits and BestGiniCost ≥ StoppingCriteria* **do**

  for *TerminalNodes of depth < MaxDepth* **do**

    Find the best split (j^*,s^*) for each TerminalNode using equations 9-11

    Calculate the difference between Node Impurity of the Parent Node and the average of the impurities of the children node evaluated using the best split (j^*,s^*).

  end

  Split the TerminalNode with the largest difference in GiniIndex between ParentNode and ChildNodes.

  The ChildNodes are new Terminal Nodes and the ParentNode is removed from the Terminal Nodes list.

  Increment the SplitCount variable by 1.

end

The decision tree should now be fully generated according to the hyperparameters.

Next, evaluate the testing data using the learned model.

for *testing point i in M2 testing points* **do**

  First, set the CurrentNode to be the RootNode.

  while *CurrentNode is not a Terminal Node* **do**

    if x_i*(feature j)* ≥ *CurrentNode.threshold* **then**

      CurrentNode = child node to the right

    else

      CurrentNode = child node to the left

    end

  end

  Now that we have iteratively traversed the tree to a terminal node we predict each testing point i as the class most associated with the terminal node.

end

Finally, return the vector of M2 predictions of the testing data as your output.

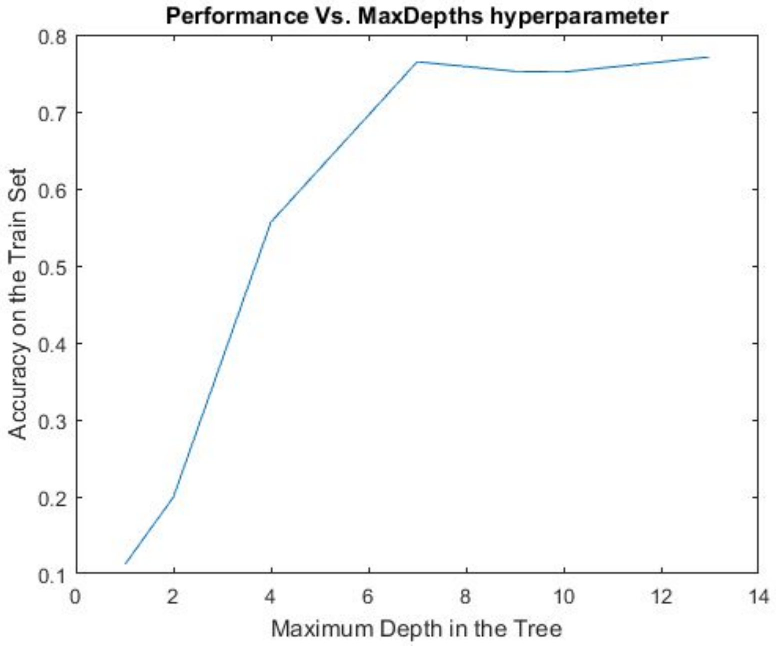
MNIST Results

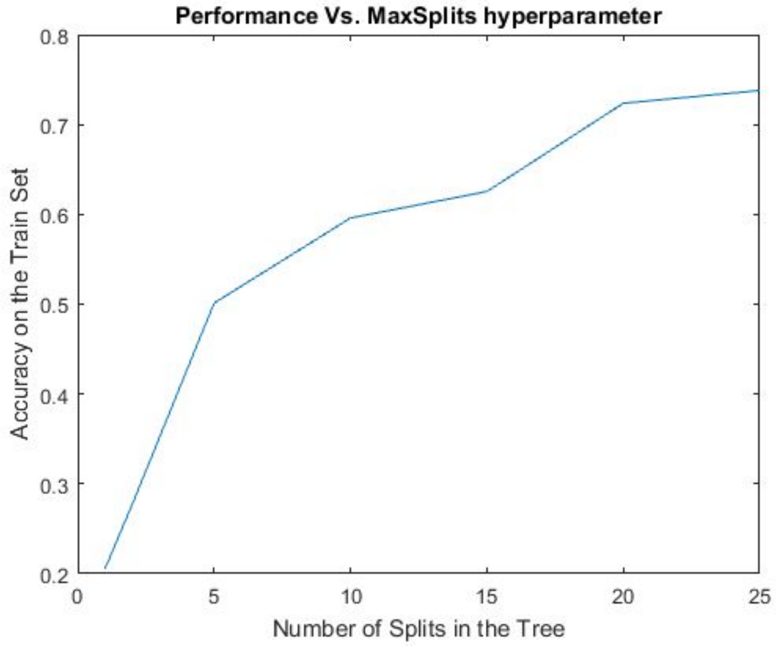
- Optimal Decision Tree hyperparameters found during cross validation with a generalized validation accuracy of 75.85%:
  - Max Splits = 25
  - Stopping Criteria = 0.01
  - Max Depth = 13
  - Minimum Leaf Size = 750

	Training Set Accuracy	Testing Set Accuracy
LDA	98.22%	97.55%
Decision Tree	77.11%	75.96%

MNIST Results (cont.)

- Performance versus various hyperparameters is shown in these two plots





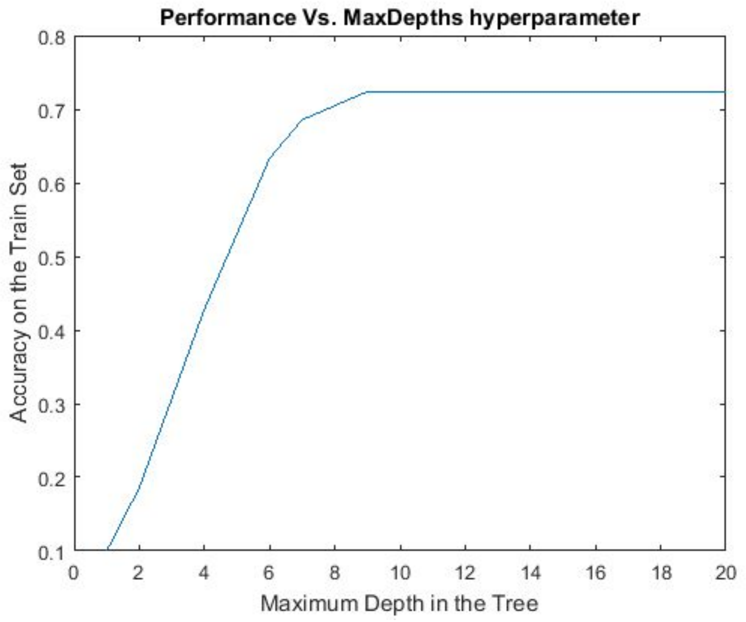
Caltech Results

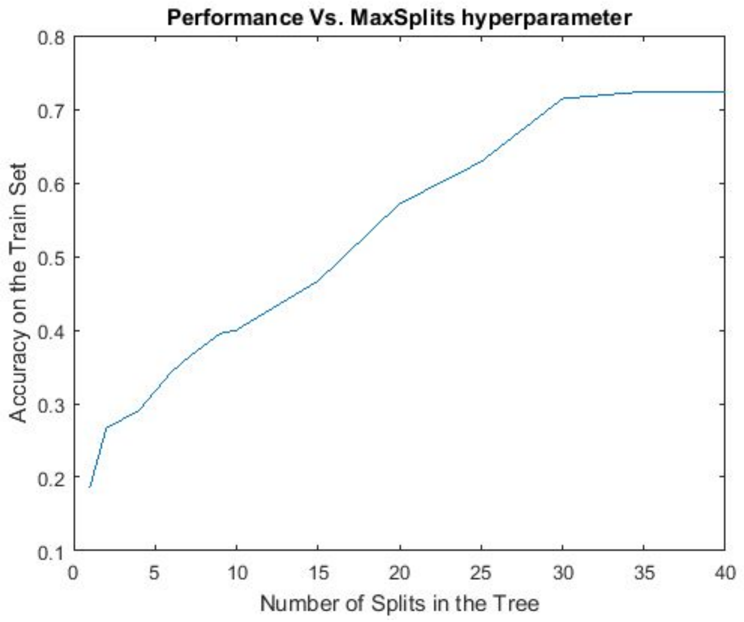
- Optimal Decision Tree hyperparameters found during cross validation with a generalized validation accuracy of 26.19%:
  - Max Splits = 20
  - Stopping Criteria = 0.01
  - Max Depth = 5
  - Minimum Leaf Size = 10

	Training Set Accuracy	Testing Set Accuracy
LDA	71.90%	39.04%
Decision Tree	53.80%	24.76%

Caltech Results (cont.)

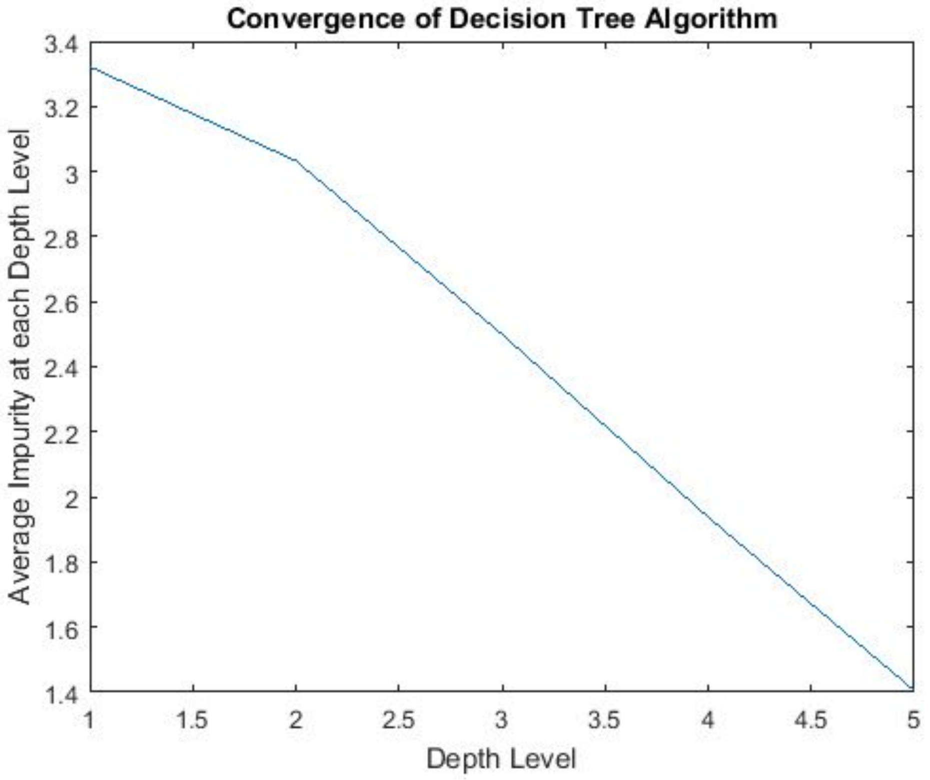
- Performance vs various Hyperparameters is shown in these two plots





Convergence of Decision Tree

- This plot shows that the average entropy at each depth level is perpetually decreasing showing how as the decision tree grows it will better fit the data.



Plot of convergence of the decision tree

References

1 - Extract histogram of oriented gradients (HOG) features - MATLAB extractHOGFeatures. (n.d.). Retrieved April 23, 2018, from https://www.mathworks.com/help/vision/ref/extracthogfeatures.html

2 - Image Category Classification Using Bag of Features - MATLAB & Simulink. (n.d.). Retrieved April 23, 2018, from https://www.mathworks.com/help/vision/examples/image-category-classification-using-bag-of-features.html

3 - T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2 edition, 2009.

4 - S. Theodoridis and K. Koutroumbas. Pattern Recognition. Associated Press, 4 edition, 2009.

Questions