# Computer Vision for HCI

## Classification Intro

1

# Objective

- To which class does this image belong?

Test Image

Known Classes

2

# **Supervised** Learning

- Given: training data
  - Set of data with corresponding class labels
  - Needs to be representative of entire space of data
- Objective: build a classifier to predict output labels (classes) of data in unseen test set
  - Need to infer a function that separates the data into desirable classes
  - No single algorithm works best on all datasets
  - Need to tune algorithm parameters
  - Feature representation is important

3

# Supervised Learning Process

- Split data into training and testing sets
- Determine features to employ
- Select a classifier
- Train the classifier using the training set
- Classify the test set
- Evaluate the classification results

No “data leakage”!

4

# Training Classifiers

- Most algorithms have parameters to tune
- Want to avoid over-fitting training data
  - Fitting to noise, not generalizing
- How to tune?
  - Use "validation" data (hypothesized test data)
    - Train classifier on a **subset** of the training data
    - Evaluate the classifier on the **remaining** training data
      - Called the **validation set**
    - Tune the classifier to minimize the error on the validation set

5

# Training Classifiers (continued)

- How to tune? (continued)

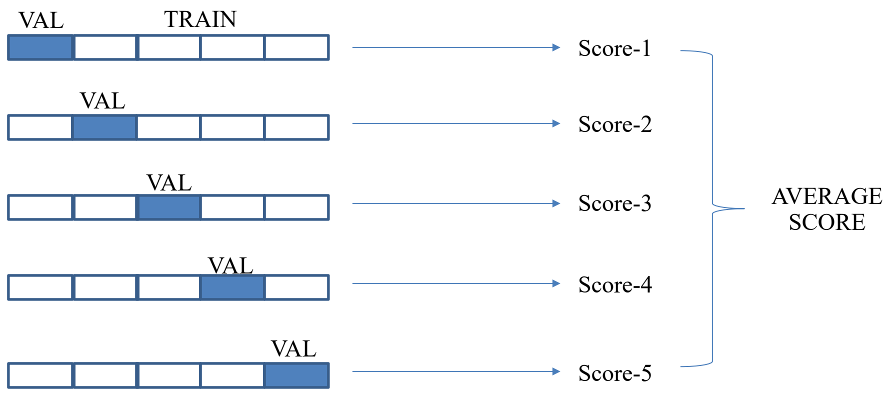
  $m$-Fold "*Cross-Validation*"
    - Pick/Set classifier options
      - e.g., parameters, model form, training time, or input features
    - Estimate **generalized** classifier performance
      - Randomly divide training set into $m$ disjoint sets of equal size
      - Train using ($m$-1) subsets and validate on the remaining subset
      - Repeat $m$ times, using different validation set each time
      - Average results
    - Repeat entire process for different classifier options and choose the options which maximize the average results

*"Cross-validation is used to estimate the skill of a model"*

*"The purpose of cross-validation is model checking, not model building"*

6

# 5-Fold Cross-Validation




7

# Evaluation

- Accuracy $= \frac{\text{Number of correct classifications}}{\text{Number of classifications}}$

- Consider the <u>binary classifier</u> situation where we are trying to detect instances of class X within a dataset containing instances of X (positive class) and Y (negative class)
  - True Positive (TP) – Correctly classifying an instance of X as X
  - False Positive (FP) – Incorrectly classifying an instance of Y as X
    - False alarm or Type I error
  - True Negative (TN) – Correctly classifying an instance of Y as Y
  - False Negative (FN) – Incorrectly classifying an instance of X as Y
    - Misdetection or Type II error

8

# Evaluation

- Precision $= \dfrac{\text{Number of correctly detected events}}{\text{Number of detected events}} = \dfrac{\text{TP}}{\text{TP}+\text{FP}}$
- Recall $= \dfrac{\text{Number of correctly detected events}}{\text{True number of events}} = \dfrac{\text{TP}}{\text{TP}+\text{FN}}$
- $F_\beta -$ Measure $= (1+\beta^2)\cdot\dfrac{\text{Precision}\cdot\text{Recall}}{\beta^2\cdot\text{Precision}+\text{Recall}}$

Common to use $\beta = 1 \rightarrow$ Harmonic mean between precision and recall

9

# Classification Approaches

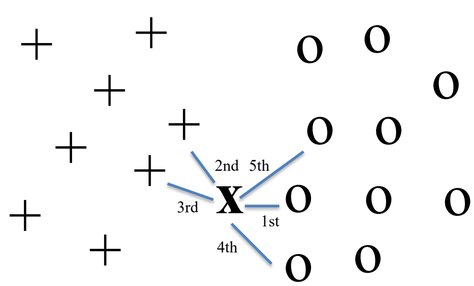
- k-Nearest Neighbors (kNN)
- Decision Trees

10

# *k*-Nearest Neighbor

- One of the <u>simplest</u> classification strategies
- Algorithm:
  - Compute distance from test sample to labeled training samples
  - Assign test sample the label most common across the first *k* nearest neighbors from the training data
    - *k* typically small and odd numbered (no ties)

11




K=1 yields X is class o

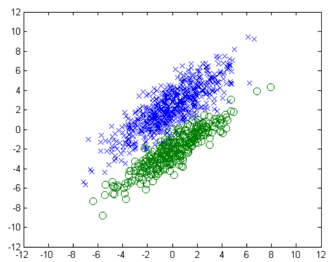
K=3 yields X is class +

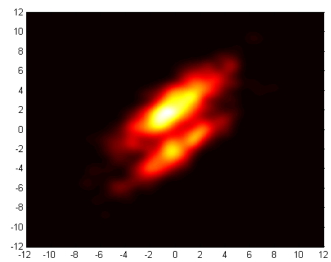
K=5 yields X is class o

12

# Example

- Training data (two Gaussians):
  - Class 1 - 667 points sampled from $N\left(\begin{bmatrix}0\\2.25\end{bmatrix},\begin{bmatrix}5&4\\4&5\end{bmatrix}\right)$
  - Class 2 - 333 points sampled from $N\left(\begin{bmatrix}0\\-2.25\end{bmatrix},\begin{bmatrix}5&4\\4&4\end{bmatrix}\right)$

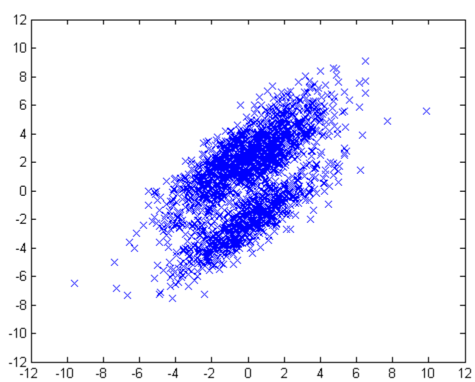



13

# Example

- Testing dataset consists of 2000 points sampled from same distributions with same priors (2/3 vs. 1/3)

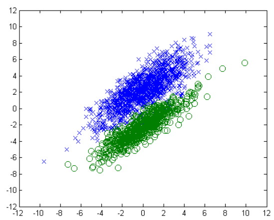



14

# Example – *k*-NN

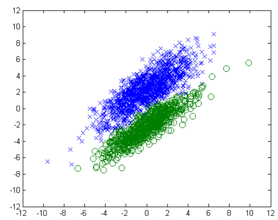
Classify each testing point using nearest neighbors from training data (known classes)

*k* = 1



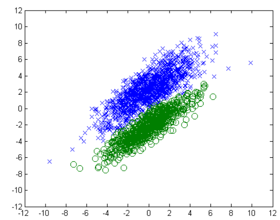
Accuracy = 0.9660

*k* = 3



Accuracy = 0.9755

*k* = 5



Accuracy = 0.9765

15

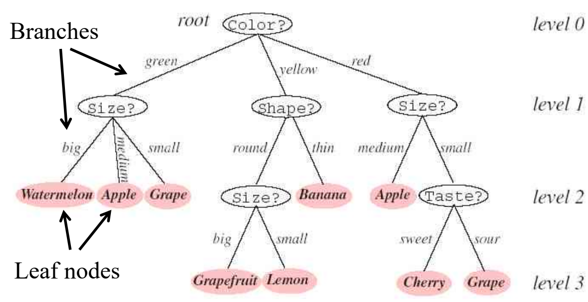
# Classification Approaches

- k-Nearest Neighbors (kNN)
- Decision Trees

16

# Decision Tree

- Classify pattern through sequence of questions
- Easy to interpret




17

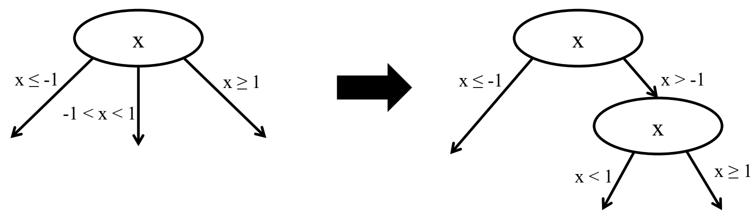
# CART

- Classification and Regression Trees (CART)
  - General framework for creating decision trees

18

# How many splits?

- Every non-binary numeric decision can be represented as combination of binary decisions




19

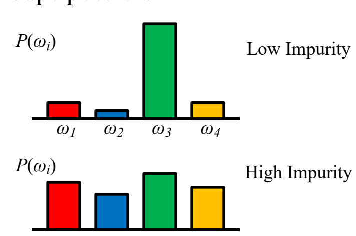
# Which property to test?

- Prefer decisions that lead to simplest tree (Occam's Razor)
  - Want property to split data into "purest" groups possible
  - Use <u>impurity</u> (randomness) measures

Node $N$

Proportion of patterns at node $N$ that belong to class $\omega_j$

Entropy Impurity $i(N) = -\sum_j P(\omega_j)\log_2 P(\omega_j)$

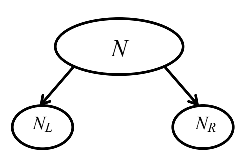



- Choose decision at node $N$ that <u>decreases</u> impurity the most

**Maximize** $\Delta i(N) = i(N) - \overbrace{\left[P_L \cdot i(N_L) + (1-P_L)\cdot i(N_R)\right]}^{\text{Want} \ll i(N)}$

Proportion of patterns that go out to node $N_L$

Impurity of patterns at node $N_L$




20

## When to stop?

- Methods to determine when to stop splitting may declare a node a leaf too early
- Alternative: grow tree out entirely (each leaf perfectly pure) and then prune
- Pruning:
  - Work bottom-up
  - Compute the increase in impurity if two child nodes linked to common parent node are eliminated
  - Merge if increase is negligible

21

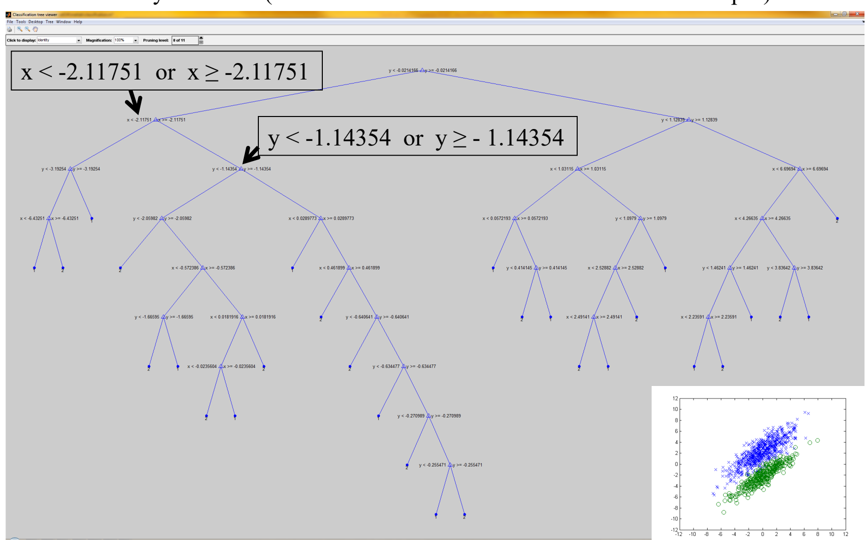
## How to assign categories to leaf nodes?

- Simplest approach is to take majority vote of class labels at leaf node
  - Ideally there will be one dominant class
- Potential options when tie occurs:
  - Random assignment
  - Take into account priors
  - Take into account classification risks
    - Cost of misdetections or false alarms of categories

22

# Example – Decision Tree (Matlab)

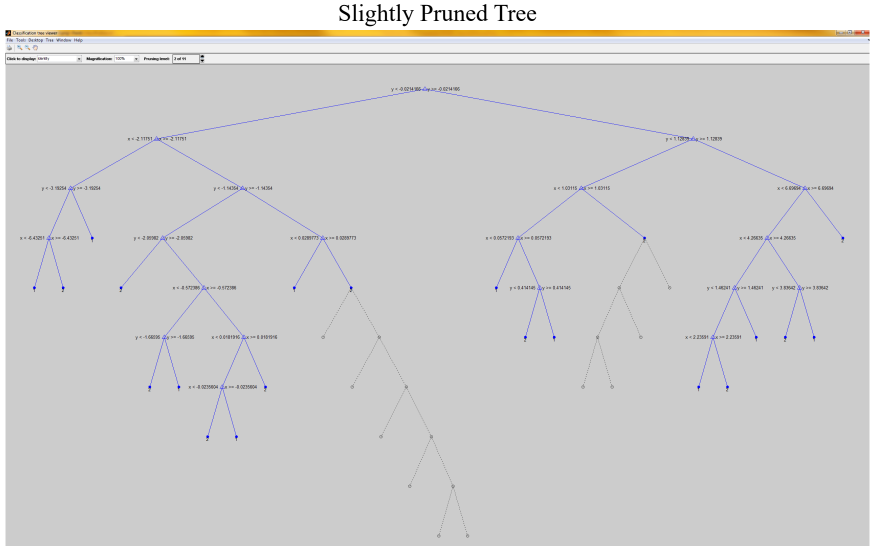
Mostly Full Tree (nodes must have at least 10 observations to be split)




23

# Example – Decision Tree

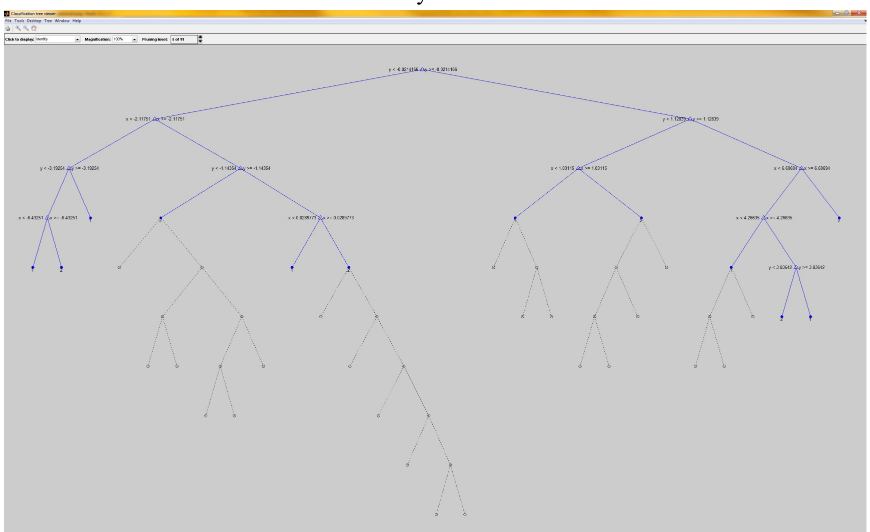
Slightly Pruned Tree



24

# Example – Decision Tree

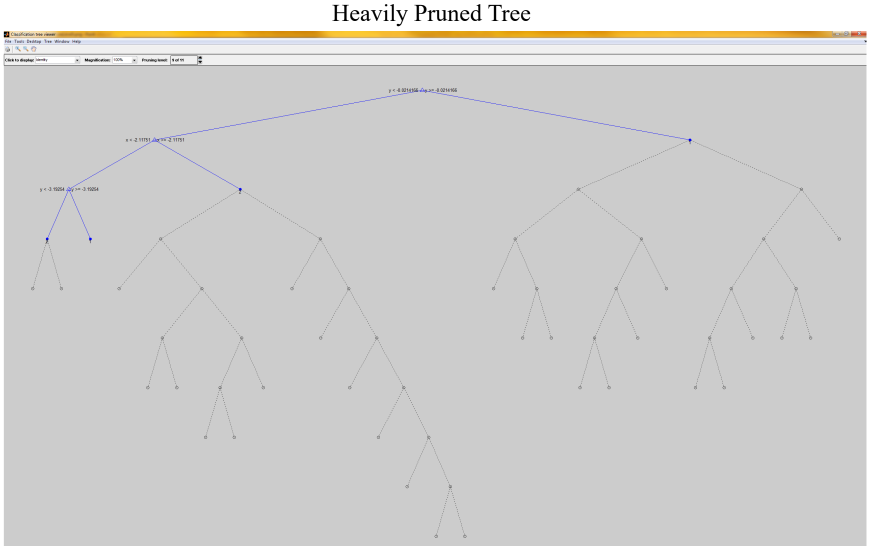
Moderately Pruned Tree



25

# Example – Decision Tree

Heavily Pruned Tree

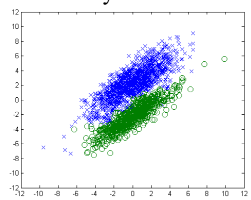


26

# Example – Decision Tree

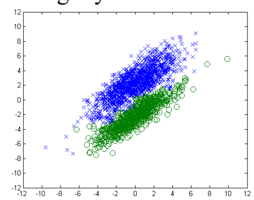
Results from test data

Mostly Full Tree



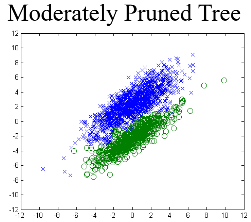
Accuracy = 0.9635

Slightly Pruned Tree



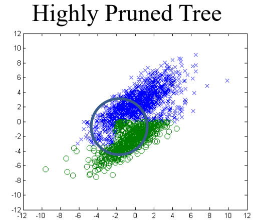
Accuracy = 0.9655

Moderately Pruned Tree



**Accuracy = 0.9670**

Highly Pruned Tree



Accuracy = 0.9165

27

# **Extra: <u>UN</u>supervised** Learning

- Given: ~~training~~ data
  - Set of data WITHOUT corresponding class labels
  - Still needs to be representative of possible space of data
- Primary objective: Group the data into “clusters”
  - Need to measure of closeness/proximity to determine which examples are closer than others
  - Feature representation is still important

– Example: K-means clustering

28

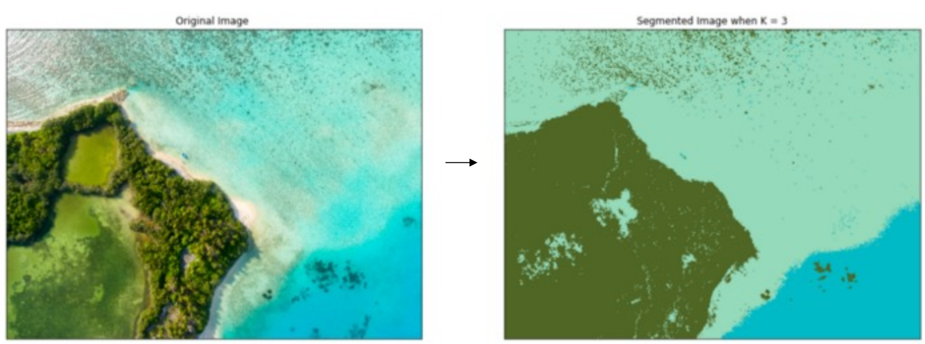
# K-means Clustering

- Each "point" is a 3-D vector of color (RGB)
- Initialization:
  – Choose *k* cluster center points (how pick *k*?)
- Repeat:
  – **Assignment step:**
    - For every point, find its closest center point
  – **Update step:**
    - Update every center point as the mean of its assigned points
- Until:
  – The maximum number of iterations is reached, or
  – No changes during the assignment step, or
  – The average distortion per point drops very little

29

29

# Previous Segmentation Example




30

30

# Summary

- Supervised training concepts
  - Train/Val/Test
  - Cross Validation
- *k*NN
  - Majority vote of nearest neighbors in training data
- Decision Trees
  - CART

31