Artificial Intelligence

Intro to Machine Learning

Programming Styles

• Standard CS: Explicitly program computer to do something
• Early AI: Derive a problem description (state) and use general algorithms to solve it
  – Search: set a search state, generate successors
  – Logic: state facts as sentences, use logical inference rules to derive consequences
“Learning”

• Agent improves its performance by adjusting its own models
  – Goal of discovering “relationships” (patterns) between input and output
    • What features are best in mapping input to output?
    • Need to recognize what’s important and what is not
  – Discover properties of the environment

Various Learning Situations

• Clustering
  – Find useful groupings of data
• Classification
  – Identify hand-written digits
  – Filter mail into spam/not-spam
  – Find face in a pic
• Action
  – Robot stay balanced upright on legs
  – Autopilot fly level
  – Vehicle stay in lane
What is Clustering?

• Grouping together items that are similar
  – Similarity determined via given measurement of “closeness” or proximity

• Items from same cluster should be more similar to each other than to items from different clusters
  – Must selecting an appropriate data representation and proximity measurement

• Referred to as “Unsupervised Learning”
  – Do not have the known (ground-truth) clusters to evaluate against

Cluster/Grouping Learning

How many “groups” and who belongs to which group?
Cluster/Grouping Learning

No “Best” Clustering
No “Best” Clustering

No “Best” Clustering
No “Best” Clustering

- Need to specify objective function/evaluation criteria to compare different clusterings
  - **Internal Criteria**: quantify the quality of clustering based on data itself
  - **External Criteria**: quantify the quality of clustering based on ground-truth labels (but usually do not have these!)
K-Means Clustering

**K-Means**
(“Grouping/Clustering”)

- One of the **simplest** clustering algorithms, yet widely employed
- **Given** initial set of $K$ centroids/means (generally obtained through initialization with random data points or locations):
  - Assign each point to closest centroid (generally found by smallest Euclidean distance)
  - Re-compute centroid/mean locations based on current assignments
  - Repeat until convergence or maximum number of iterations
- Works well under constrained conditions
K-Means
with “Good” Starting Means
$K$-Means
with "Good" Starting Means

$K$-Means
with "Bad" Starting Means
**K-Means**

with “Bad” Starting Means

---

**K-Means – Comments**

- Works well when clusters are compact and well-separated
- Results dependent on initial conditions (as shown)
  - Often run multiple times and keep clustering minimizing sum of squared distances (points to centroids)
- **Need to know number of clusters \( K \) a priori
- Does not always perform well (see below)
**K-Medoids**

- Related to K-Means
- Instead of using mean to represent a cluster, use a particular example in the cluster
- Method:
  - Select initial medoids (random examples)
  - Repeat until convergence or maximum number of iterations:
    - (Re)assign each point to the cluster having the closest medoid
    - In each cluster, make the example that minimizes the sum of distances within the cluster the medoid
- More robust to noise and outliers (as compared to K-Means)

**“Supervised” Learning**
Learning

Hmmm… which objects are boxes?

![Image of objects](Courtesy NASA/JPL-Caltech)

“Supervised” Learning

Hmmm… which objects are boxes?

![Image of objects](Courtesy NASA/JPL-Caltech)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>no</td>
<td></td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

12
Supervised Learning

- Given: training data
  - Set of data with corresponding class labels
- Objective: build a classifier to predict output label (class) for unseen test example
  - Need to infer a function that separates the data into desirable classes
  - Feature representation is important
  - Learning is used to tune algorithm parameters
  - No single algorithm works best on all datasets

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 results on the test set
Evaluating Supervised Learning

• Training data must be selected so as to reflect the global data pool
• Examining performance on unseen data is crucial to prevent overfitting to the training data
  – Unintended correlations may exist between input and output
    • Recall “photos with tanks taken on sunny days” from AI video
  – Correlations specific to the set of training data may exist
    • e.g., language processing trained on Wall Street Journal may not work well for spoken conversation

Training Classifiers

• How to tune algorithm parameters and address the overfitting issues?
  – Use “Validation” data
    • Train classifier on a subset of the training data
    • Examine the classifier on the remaining training data
      – Called the “validation set”
    • Tune the classifier to minimize the error on this validation set
Training Classifiers (continued)

• How to tune? (continued)
  – **m-Fold Cross-Validation**
    • Set classifier options
      – e.g., number of parameters, model form, training time, input features, etc.
    • 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

Evaluation Metrics

• **Accuracy** = \( \frac{\text{Number of correct classifications}}{\text{Number of classifications}} \)

• **Precision** = \( \frac{\text{Number of correctly detected events}}{\text{Number of detected events}} \)

• **Recall** = \( \frac{\text{Number of correctly detected events}}{\text{True number of events}} \)

• **\( F_\beta \) – Measure** = \( (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \)

  Common to use \( \beta = 1 \) → Harmonic mean between precision and recall
$k$-Nearest Neighbor Classifier

$k$-Nearest Neighbor ($k$-NN)

- One of the simplest supervised classification strategies
- Algorithm:
  - Compute distance from test example to all labeled samples in the training set
  - Assign test sample the label most common across the first “$k$ nearest neighbors” from the training data
    - $k$ is typically small and odd numbered (so no ties in voting)
K=1 yields X is class o
K=3 yields X is class +
K=5 yields X is class o

Decision Tree Classifier
Decision Trees

- **Input:** Features per example
- **Output:** Classification label
- Learns by subdividing the data into classes with same properties
- Good at determining which features are good discriminators

![Decision Tree Diagram]

Decision Tree

- Classify pattern through sequence of questions
- Easy to interpret
How many splits?

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

How Determine the Split?

- Prefer decisions that lead to simplest tree (Occam’s Razor)
  - Want property to split data into “purest” groups possible
  - Use impurity measure

- Choose decision at node $N$ that decreases impurity the most
  - Maximize $\Delta (N) = \hat{h}(N) - P_L h(N_L) + (1 - P_L) h(N_R)$
When to stop?

- 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 in impurity is negligible

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
Two (Gaussian) Groupings of Training Data

Example – Decision Tree (Matlab)
Mostly Full Tree (nodes must have at least 10 observations to be split)

\[
x < -2.11751 \text{ or } x \geq -2.11751
\]

\[
y < -1.14354 \text{ or } y \geq -1.14354
\]
Example – Decision Tree

Slightly Pruned Tree

Example – Decision Tree

Moderately Pruned Tree
Example – Decision Tree

**Heavily Pruned Tree**

Example – Using **Test** Data
(taken from same distribution as training 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
Neural Networks

• A Neural Network is another example of a supervised learning classifier
  – More (much more) coming soon!

Summary

• Unsupervised learning
  – K-Means, K-Medoids
• Supervised learning
  – Training and evaluation
  – $k$-NN
    • Majority vote of nearest neighbors in training data
  – Decision Trees