Supervised Classification

- Divide instances into (two or more) classes
  - Instance (feature vector): \( \mathbf{X} = \left\{ x_{1}, \ldots, x_{m} \right\} \)
  - Features may be categorical or numerical
  - Class (label): \( y \)
  - Training data: \( \mathbf{X} = \left\{ \mathbf{x}_{1}, \ldots, \mathbf{x}_{m} \right\} \)
- Classification in NLP
  - Spam filtering (spam vs. ham)
  - Spelling error detection (error vs. no error)
  - Text categorization (news, economy, culture, sport, ...)
  - Named entity classification (person, location, organization, ...)

Models for Classification

- Generative probabilistic models:
  - Model of \( P(x, y) \)
  - Naïve Bayes
- Conditional probabilistic models:
  - Model of \( P(y | x) \)
  - Logistic regression (next time)
- Discriminative model:
  - No explicit probability model
  - Decision trees, nearest neighbor classification (today)
  - Perceptron, support vector machines (next time)
Decision Trees

- Hierarchical tree structure for classification
  - Each internal node specifies a test of some feature
  - Each branch corresponds to a value for the tested feature
  - Each leaf node provides a classification for the instance
- Represents a disjunction of conjunctions of constraints
  - Each path from root to leaf specifies a conjunction of tests
  - The tree itself represents the disjunction of all paths

Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, $x_i$
    - Numeric $x_i$: Binary split: $x_i > w_m$
    - Discrete $x_i$: $n$-way split for $n$ possible values
  - Multivariate: Uses all attributes, $x$
- Leaves
  - Classification: class labels (or proportions)
  - Regression: $r$ average (or local fit)
- Learning:
  - Greedy recursive algorithm
    - Find best split $X = (X_1, ..., X_p)$, then induce tree for each $X_i$

Classification Trees (ID3, CART, C4.5)

- For node $m$, $N_m$ instances reach $m$, $N'_m$ belong to $C_i$
  \[ \hat{P}(C_i | x, m) = p^i_m = \frac{N'_m}{N_m} \]
- Node $m$ is pure if $p^i_m$ is 0 or 1
- Measure of impurity is entropy
  \[ I_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m \]
Example: Entropy

- Assume two classes \( (C_1, C_2) \) and four instances \( (x^1, x^2, x^3, x^4) \)
- Case 1:
  - \( C_1 = \{x^1, x^2, x^3, x^4\} \), \( C_2 = \{\} \)
  - \( I_m = -(1 \log 1 + 0 \log 0) = 0 \)
- Case 2:
  - \( C_1 = \{x^1, x^2, x^3\} \), \( C_2 = \{x^4\} \)
  - \( I_m = -(0.75 \log 0.75 + 0.25 \log 0.25) = 0.81 \)
- Case 3:
  - \( C_1 = \{x^1, x^2\} \), \( C_2 = \{x^3, x^4\} \)
  - \( I_m = -(0.5 \log 0.5 + 0.5 \log 0.5) = 1 \)

Best Split

- If node \( m \) is pure, generate a leaf and stop, otherwise split with test \( t \) and continue recursively
- Find the test that minimizes impurity
- Impurity after split with test \( t \):
  - \( N_{m'} \) of \( N_m \) take branch \( j \)
  - \( N_{m''} \) belong to \( C_i \)
  - \( \hat{P}(C_i | x, m, j) \) \( = \frac{N_{m''}}{N_{m'}} \)
  - \( I_m' = -\sum_{j=1}^{k} \frac{N_{m''}}{N_{m'}} \sum_{i=1}^{k} p_{m''} \log p_{m''} \)

Information Gain and Gain Ratio

- Choosing the test that minimizes impurity maximizes the information gain (IG):
  - \( IG_m = I_m - I_m' \)
  - \( V_m' = \sum_{j=1}^{k} \frac{N_{m''}}{N_{m'}} \log \frac{N_{m''}}{N_{m'}} \)
- Information gain prefers features with many values
- The normalized version is called gain ratio (GR):
  - \( GR_m = \frac{IG_m}{V_m'} \)
  - \( V_m'' = \sum_{j=1}^{k} \frac{N_{m''}}{N_m} \log \frac{N_{m''}}{N_m} \)
Pruning Trees

- Decision trees are susceptible to overfitting
- Remove subtrees for better generalization:
  - Prepruning: Early stopping (e.g., with entropy threshold)
  - Postpruning: Grow whole tree, then prune subtrees
- Prepruning is faster, postpruning is more accurate (requires a separate validation set)

Rule Extraction from Trees

C4.5 Rules (Quinlan, 1993)

Learning Rules

- Rule induction is similar to tree induction but
  - tree induction is breadth-first
  - rule induction is depth-first (one rule at a time)
- Rule learning:
  - A rule is a conjunction of terms (cf. tree path)
  - A rule covers an example if all terms of the rule evaluate to true for the example (cf. sequence of tests)
  - Sequential covering: Generate rules one at a time until all positive examples are covered
- IREP (Furnkranz and Widmer, 1994), Ripper (Cohen, 1995)

Properties of Decision Trees

- Decision trees are appropriate for classification when:
  - Features can be both categorical and numeric
  - Disjunctive descriptions may be required
  - Training data may be noisy (missing values, incorrect labels)
  - Interpretation of learned model is important (rules)
- Inductive bias of (most) decision tree learners:
  - Prefers trees with informative attributes close to the root
  - Prefers smaller trees over bigger ones (with pruning)
  - Preference bias (incomplete search of complete space)
Nearest Neighbor Classification

- An old idea
  
  This “rule of nearest neighbor” has considerable elementary intuitive appeal and probably corresponds to practice in many situations. For example, it is possible that much medical diagnosis is influenced by the doctor’s recollection of the subsequent history of an earlier patient whose symptoms resemble in some way those of the current patient. (Fix and Hodges, 1952)

- Key components:
  
  - Storage of old instances
  - Similarity-based reasoning to new instances

$k$-Nearest Neighbor

- Learning:
  
  - Store training instances in memory

- Classification:
  
  - Given new test instance $x$, 
    - Compare it to all stored instances
    - Compute a distance between $x$ and each stored instance $x'$
    - Keep track of the $k$ closest instances (nearest neighbors)
    - Assign to $x$ the majority class of the $k$ nearest neighbors

  A geometric view of learning
  
  - Proximity in (feature) space $\rightarrow$ same class
  - The smoothness assumption

Eager and Lazy Learning

- Eager learning (e.g., decision trees)
  
  - Learning – induce an abstract model from data
  - Classification – apply model to new data

- Lazy learning (a.k.a. memory-based learning)
  
  - Learning – store data in memory
  - Classification – compare new data to data in memory

- Properties:
  
  - Retains all the information in the training set – no abstraction
  - Complex hypothesis space – suitable for natural language?
  - Main drawback – classification can be very inefficient

Dimensions of a $k$-NN Classifier

- Distance metric
  
  - How do we measure distance between instances?
  - Determines the layout of the instance space

- The $k$ parameter
  
  - How large neighborhood should we consider?
  - Determines the complexity of the hypothesis space
Distance Metric 1

- Overlap = count of mismatching features

\[ \Delta(x, z) = \sum_{i=1}^{m} \delta(x_i, z_i) \]

\[ \delta(x_i, z_i) = \begin{cases} 
  \frac{x_i - z_i}{\max_i - \min_i} & \text{if numeric, else} \\
  0 & \text{if } x_i = y_i \\
  1 & \text{if } x_i \neq y_i 
\end{cases} \]

Distance Metric 2

- MVDM = Modified Value Difference Metric

\[ \Delta(x, z) = \sum_{i=1}^{m} \delta(x_i, z_i) \]

\[ \delta(x_i, z_i) = \sum_{j=1}^{k} \left| P(C_j \mid x_i) - P(C_j \mid z_i) \right| \]

The k parameter

- Tunes the complexity of the hypothesis space
  - If \( k = 1 \), every instance has its own neighborhood
  - If \( k = N \), all the feature space is one neighborhood

A Simple Example

Training set:
1. \((a, b, a, c) \rightarrow A\)
2. \((a, b, c, a) \rightarrow B\)
3. \((b, a, c, c) \rightarrow C\)
4. \((c, a, b, c) \rightarrow A\)

New instance:
5. \((a, b, b, a)\)

Distances (overlap):
\[ \Delta(1, 5) = 2 \]
\[ \Delta(2, 5) = 1 \]
\[ \Delta(3, 5) = 4 \]
\[ \Delta(4, 5) = 3 \]

\(k\)-NN classification:
1-NN(5) = B
2-NN(5) = A/B
3-NN(5) = A
4-NN(5) = A
Further Variations on $k$-NN

- Feature weights:
  - The overlap metric gives all features equal weight
  - Features can be weighted by IG or GR
- Weighted voting:
  - The normal decision rule gives all neighbors equal weight
  - Instances can be weighted by (inverse) distance

Properties of $k$-NN

- Nearest neighbor classification is appropriate when:
  - Features can be both categorical and numeric
  - Disjunctive descriptions may be required
  - Training data may be noisy (missing values, incorrect labels)
  - Fast classification is not crucial
- Inductive bias of $k$-NN:
  - Nearby instances should have the same label (smoothness)
  - All features are equally important (without feature weights)
  - Complexity tuned by the $k$ parameter

Assignment 1

- Decision trees and nearest neighbor classification
- Software package: Weka
- Data sets:
  - German plural
  - English past tense
- Send questions to: joakim.nivre@lingfil.uu.se
- Report due November 28