Tutorial 2

CPSC 340: Machine Learning and Data Mining

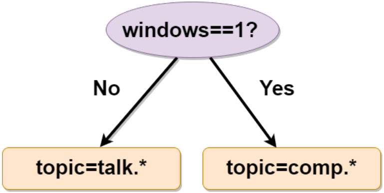
Fall 2016

Overview

1. Decision Tree
   - Decision Stump
   - Decision Tree
2. Training, Testing, and Validation Set
3. Naive Bayes Classifier

Decision Stump

- Decision stump: simple decision tree with 1 splitting rule based on 1 feature.
- Binary example:



- Assigns a label to each leaf based on the most frequent label.
- Most intuitive score: classification accuracy.

The ”newsgroups.mat” Dataset

- The newsgroups.mat Matlab file contains the following objects:
  1. X: A sparse binary matrix. Each row corresponds to a post, and each column corresponds to a word from the word list. A value of 1 means that the word occurred in the post.
  2. y: A vector with values 1 through 4, with the value corresponding to the newsgroup that the post came from.
  3. Xtest and ytest: the word lists and newsgroup labels for additional newsgroup posts.
  4. groupnames: The names of four newsgroups.
  5. wordlist: A list of words that occur in posts to these newsgroups.

Example: Binary Decision Stump for newsgroups.mat

```
function [model] = decisionStump(X,y)
% Fits a decision stump that splits on a single variable,
%   assuming that X is binary {0,1}, and y is categorical {1,2,3,...,C}.
[N,D] = size(X);
% Compute number of class lables
C = max(y);
% Address the trivial case where we do not split
count = zeros(C,1);
for n = 1:N
    count(y(n)) = count(y(n)) + 1;
end
[maxCount,maxLabel] = max(count);
minError = sum(y ~= maxLabel);
splitVariable = [];
splitLabel0 = maxLabel;
splitLabel1 = [];
% Loop over features looking for the best split
if any(y ~= y(1))
    for d = 1:D
        % Count number of class labels when the feature is 1, and when it is 0
        count1 = zeros(C,1);
        for n = find(X(:,d) == 1)'
            count1(y(n)) = count1(y(n)) + 1;
        end
        count0 = count-count1;
        % Compute majority class
        [maxCount,maxLabel1] = max(count1);
        [maxCount,maxLabel0] = max(count0);
        % Compute number of classification errors
        yhat = maxLabel0*ones(N,1);
        yhat(X(:,d) == 1) = maxLabel1;
        errors = sum(yhat~=y);
        % Compare to minimum error so far
        if errors < minError
            % This is the lowest error, store this value
            minError = errors;
            splitVariable = d;
            splitLabel1 = maxLabel1;
            splitLabel0 = maxLabel0;
        end
    end
end
model.splitVariable = splitVariable;
model.label1 = splitLabel1;
model.label0 = splitLabel0;
model.predictFunc = @predict;
end
function [y] = predict(model,X)
    [T,D] = size(X);
    if isempty(model.splitVariable)
        y = model.label0*ones(T,1);
    else
        y = zeros(T,1);
        for n = 1:T
            if X(n,model.splitVariable) == 1
                y(n,1) = model.label1;
            else
                y(n,1) = model.label0;
            end
        end
    end
end
```

Decision Tree

- Decision stumps have only 1 rule based on only 1 feature.
  - Very limited class of models: usually not very accurate for most tasks.
- Decision trees allow sequences of splits based on multiple features.
  - Very general class of models: can get very high accuracy.
  - However, it's computationally infeasible to find the best decision tree.
- Most common decision tree learning algorithm in practice:
  - Greedy recursive splitting.

Problem 1: Decision Tree for newsgroups.mat

- For a maximum depth of 2, 1) draw the learned decision tree. and 2) re-write the function as a simple program using if/else statements.

```
% Load X and y variable
load newsgroups.mat
[N,D] = size(X);
depth =2 ;
model = decisionTree(X,y, depth);
% Evaluate training error
yhat = model.predictFunc(model,X);
error = sum(yhat ~= y)/N;
```

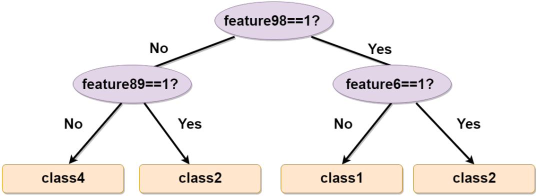
```
function [model] = decisionTree(X,y,maxDepth)
%
% Fits a decision tree that splits on a sequence of single variables,
%   assuming that X is binary {0,1}, and y is categorical {1,2,3,...,C}.

[N,D] = size(X);

% Learn a decision stump
splitModel = decisionStump(X,y);
% splitModel contains splitVariable (index of feauture used for splitting),
% label1 (the mode of the label of variables equal to 1),
%label0 (the mode of the label of variables equal to 0),
% and a predict function

if maxDepth <= 1 || isempty(splitModel.splitVariable)
    % If we have reached the maximum depth or the decision stump does
    % nothing, use the decision stump
    model = splitModel;
else
    % Fit a decision tree to each split, decreasing maximum depth by 1
    d = splitModel.splitVariable;
    model.splitModel = splitModel;

    % Find indices of examples in each split
    splitIndex1 = find(X(:,d)==1);
    splitIndex0 = find(X(:,d)==0);

    % Fit decision tree to each split
    model.subModel1 = decisionTree(X(splitIndex1,:),y(splitIndex1),maxDepth-1);
    model.subModel0 = decisionTree(X(splitIndex0,:),y(splitIndex0),maxDepth-1);

    % Assign prediction function
    model.predictFunc = @predict;
end
end
```

Solution: Decision Tree for newsgroups.mat

```
>> model.splitModel

ans =

    splitVariable: 98
           label1: 1
           label0: 4
      predictFunc: @predict

>> model.subModel1

ans =

    splitVariable: 6
           label1: 2
           label0: 1
      predictFunc: @predict

>> model.subModel0

ans =

    splitVariable: 89
           label1: 2
           label0: 4
      predictFunc: @predict
```

Solution: Decision Tree for newsgroups.mat

- Decision tree:



- If-else statement:

```
if X(i,98) ==1
    if X(i,6)==1
        return 2
    else
        return 1
    end
else
    if X(i,89)==1
        return 2
    else
        return 4
    end
end
```

Training, Testing, and Validation Set

- Given training data, we would like to learn a model to minimize error on the testing data
- How do we decide decision tree depth?
- We care about test error.
- But we can't look at test data.
- So what do we do?????
- One answer: Use part of your train data to approximate test error.
- Split training objects into training set and validation set:
  - Train model on the training data.
  - Test model on the validation data.

Cross-Validation

- Isn't it wasteful to only use part of your data?
- k-fold cross-validation:
  - Train on k-1 folds of the data, validate on the other fold.
  - Repeat this k times with different splits, and average the score.

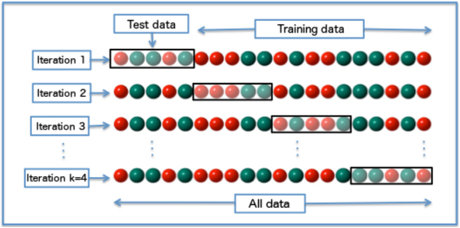


Figure 1: Adapted from Wikipedia.

- Note: if examples are ordered, split should be random.

Problem 2: 2-Fold Cross Validation for newsgroups.mat

- Modify the code below to compute the 2-fold cross-validation scores on the training data alone.
- Find the depth that would be chosen by cross-validation.

```
% Load X and y variable
load newsgroups.mat
[N,D] = size(X);
T = length(ytest);
depth = 5;
model = decisionTree(X,y,depth);
yhat = model.predictFunc(model,X);
errorTrain = sum(yhat ~= y)/N;
yhat =model.predictFunc(model,Xtest);
errorTest = sum(yhat ~= ytest)/T;
```

Solution: 2-Fold Cross Validation for newsgroups.mat

```
% Load X and y variable
load newsgroups.mat
[N,D] = size(X);
Xtest = X (floor(N/2) + 1 :N  , : );
ytest= y (floor(N/2) +1 : N) ;
X =  X ( 1:floor(N/2) , : )  ;
y = y (1: floor(N/2));
mindepth = -1 ; minError = Inf;
for depth =1 :15
    errorTrain = 0; errorTest = 0;
    for i =1:2
        [N,D] = size(X);
        T = length(ytest);
        model = decisionTree(X,y,depth);
        yhat = model.predictFunc(model,X);
        errorTrain = errorTrain +sum(yhat ~= y)/N;
        yhat = model.predictFunc(model,Xtest);
        errorTest = errorTest + sum(yhat ~= ytest)/T;
        [X, Xtest]=mySwap(Xtest, X);
        [y,ytest] = mySwap(ytest,y) ;
    end
    disp(errorTest/2 ) ;
    if errorTest/2 < minError
        minError= errorTest/2;
        mindepth = depth;
    end
end
disp(minError); disp(mindepth);
```

Naive Bayes Classifier

- Naive Bayes is a probabilistic classifier.
  - Based on Bayes' theorem.
  - Strong independence assumption between features.

Naive Bayes Classifier

- Naive Bayes is a probabilistic classifier.
  - Based on Bayes' theorem.
  - Strong independence assumption between features.
- In the rest of this tutorial,
  - We use y_i for the label of object i (element i of y).
  - We use x_i for the features of object i (row i of X).
  - We use x_{ij} for feature j of object i.
  - We use d for the number of features in object i.

Naive Bayes Classifier

- Bayes' rule

Posterior probability   Likelihood   Prior probability

$$p(y_i|x_i) = \frac{p(x_i|y_i)p(y_i)}{p(x_i)}$$

Evidence

Naive Bayes Classifier

- Bayes' rule

$$\underbrace{p(y_i|x_i)}_{\text{Posterior probability}} = \frac{\overbrace{p(x_i|y_i)}^{\text{Likelihood}}\overbrace{p(y_i)}^{\text{Prior probability}}}{\underbrace{p(x_i)}_{\text{Evidence}}}$$

- Since the denominator does not depend on y_i, we are only interested in the numerator:

$$p(y_i|x_i) \propto p(x_i|y_i)p(y_i)$$

Naive Bayes Classifier

- The numerator is equal to the joint probability:

$$p(x_i|y_i)p(y_i) = p(x_i, y_i) = p(x_{i1}, ..., x_{id}, y_i)$$

Naive Bayes Classifier

- The numerator is equal to the joint probability:

$$p(x_i|y_i)p(y_i) = p(x_i, y_i) = p(x_{i1}, ..., x_{id}, y_i)$$

- Chain rule:

$$p(x_{i1}, ..., x_{id}, y_i) = p(x_{i1}|x_{i2}, ..., x_{id}, y_i)p(x_{i2}, ..., x_{id}, y_i)$$

$$= ...$$

$$= p(x_{i1}|x_{i2}, ..., x_{id}, y_i)p(x_{i2}|x_{i3}, ..., x_{id}, y_i) \text{ ... } p(x_{id}|y_i)p(y_i)$$

Naive Bayes Classifier

- The numerator is equal to the joint probability:

$$p(x_i|y_i)p(y_i) = p(x_i, y_i) = p(x_{i1}, ..., x_{id}, y_i)$$

- Chain rule:

$$p(x_{i1}, ..., x_{id}, y_i) = p(x_{i1}|x_{i2}, ..., x_{id}, y_i)p(x_{i2}, ..., x_{id}, y_i)$$

$$= ...$$

$$= p(x_{i1}|x_{i2}, ..., x_{id}, y_i)p(x_{i2}|x_{i3}, ..., x_{id}, y_i) \textbf{ ... } p(x_{id}|y_i)p(y_i)$$

- Each feature in x_i is independent of the others given y_i:

$$p(x_{ij}|x_{ij+1}, ..., x_{id}, y_i) = p(x_{ij}|y_i)$$

Naive Bayes Classifier

- The numerator is equal to the joint probability:

$$p(x_i|y_i)p(y_i) = p(x_i, y_i) = p(x_{i1}, ..., x_{id}, y_i)$$

- Chain rule:

$$p(x_{i1}, ..., x_{id}, y_i) = p(x_{i1}|x_{i2}, ..., x_{id}, y_i)p(x_{i2}, ..., x_{id}, y_i)$$

$$= ...$$

$$= p(x_{i1}|x_{i2}, ..., x_{id}, y_i)p(x_{i2}|x_{i3}, ..., x_{id}, y_i) \text{ ... } p(x_{id}|y_i)p(y_i)$$

- Each feature in x_i is independent of the others given y_i:

$$p(x_{ij}|x_{ij+1}, ..., x_{id}, y_i) = p(x_{ij}|y_i)$$

- Therefore:

$$p(y_i, x_i) \propto p(y_i) \prod_{j=1}^{d} p(x_{ij}|y_i)$$

Problem 4: Naive Bayes Classifier

headache	runny nose	fever	flu
N	Y	Y	N
Y	N	N	N
N	N	N	N
Y	Y	Y	Y
Y	Y	N	Y
N	N	Y	Y

Problem 4: Naive Bayes Classifier

headache	runny nose	fever	flu
N	Y	Y	N
Y	N	N	N
N	N	N	N
Y	Y	Y	Y
Y	Y	N	Y
N	N	Y	Y

headache	runny nose	fever	flu
Y	N	Y	?

Solution: Naive Bayes Classifier

- We need

$p(headache=Y \mid flu=N)$	1/3
$p(headache=Y \mid flu=Y)$	2/3
$p(runny\ nose=N \mid flu=N)$	2/3
$p(runny\ nose=N \mid flu=Y)$	1/3
$p(fever=Y \mid flu=N)$	1/3
$p(fever=Y \mid flu=Y)$	2/3
$p(flu=N)$	1/2
$p(flu=Y)$	1/2

Solution: Naive Bayes Classifier

- We need

p(headache=Y|flu=N)	1/3
p(headache=Y|flu=Y)	2/3
p(runny nose=N|flu=N)	2/3
p(runny nose=N|flu=Y)	1/3
p(fever=Y|flu=N)	1/3
p(fever=Y|flu=Y)	2/3
p(flu=N)	1/2
p(flu=Y)	1/2

- $p(\text{flu} = N|\text{headache} = Y, \text{runny nose} = N, \text{fever} = Y) \propto p(\text{headache} = Y|\text{flu} = N)p(\text{runny nose} = N|\text{flu} = N)p(\text{fever} = Y|\text{flu} = N)p(\text{flu} = N) = \frac{1}{3} * \frac{2}{3} * \frac{1}{3} * \frac{1}{2} = 0.0370$

Solution: Naive Bayes Classifier

- We need

p(headache=Y|flu=N)	1/3
p(headache=Y|flu=Y)	2/3
p(runny nose=N|flu=N)	2/3
p(runny nose=N|flu=Y)	1/3
p(fever=Y|flu=N)	1/3
p(fever=Y|flu=Y)	2/3
p(flu=N)	1/2
p(flu=Y)	1/2

- $p(\text{flu} = N|\text{headache} = Y, \text{runny nose} = N, \text{fever} = Y) \propto p(\text{headache} = Y|\text{flu} = N)p(\text{runny nose} = N|\text{flu} = N)p(\text{fever} = Y|\text{flu} = N)p(\text{flu} = N) = \frac{1}{3} * \frac{2}{3} * \frac{1}{3} * \frac{1}{2} = 0.0370$
- $p(\text{flu} = Y|\text{headache} = Y, \text{runny nose} = N, \text{fever} = Y) \propto p(\text{headache} = Y|\text{flu} = Y)p(\text{runny nose} = N|\text{flu} = Y)p(\text{fever} = Y|\text{flu} = Y)p(\text{flu} = Y) = \frac{2}{3} * \frac{1}{3} * \frac{2}{3} * \frac{1}{2} = 0.0741$

Solution: Naive Bayes Classifier

- We need

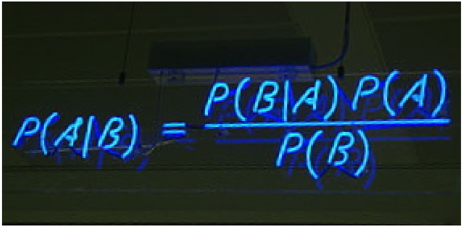
p(headache=Y|flu=N)	1/3
p(headache=Y|flu=Y)	2/3
p(runny nose=N|flu=N)	2/3
p(runny nose=N|flu=Y)	1/3
p(fever=Y|flu=N)	1/3
p(fever=Y|flu=Y)	2/3
p(flu=N)	1/2
p(flu=Y)	1/2

- $p(\text{flu} = N|\text{headache} = Y, \text{runny nose} = N, \text{fever} = Y) \propto p(\text{headache} = Y|\text{flu} = N)p(\text{runny nose} = N|\text{flu} = N)p(\text{fever} = Y|\text{flu} = N)p(\text{flu} = N) = \frac{1}{3} * \frac{2}{3} * \frac{1}{3} * \frac{1}{2} = 0.0370$
- $p(\text{flu} = Y|\text{headache} = Y, \text{runny nose} = N, \text{fever} = Y) \propto p(\text{headache} = Y|\text{flu} = Y)p(\text{runny nose} = N|\text{flu} = Y)p(\text{fever} = Y|\text{flu} = Y)p(\text{flu} = Y) = \frac{2}{3} * \frac{1}{3} * \frac{2}{3} * \frac{1}{2} = 0.0741$

headache	runny nose	fever	flu
Y	N	Y	Y

Bayes' Theorem

- Bayes' Theorem enables us to reverse probabilities:



Problem 3: Prosecutor's fallacy

- A crime has been committed in a large city and footprints are found at the scene of the crime. The guilty person matches the footprints, $p(F|G) = 1$. Out of the innocent people, 1% match the footprints by chance, $p(F| \sim G) = 0.01$. A person is interviewed at random and his/her footprints are found to match those at the crime scene. Determine the probability that the person is guilty, or explain why this is not possible, $p(G|F) = ?$
  - Let F be the event that the footprints match.
  - Let G be the event that the person is guilty
  - $\sim G$ be the event that the person is innocent.

Solution: Prosecutor's fallacy

$$p(G|F) = \frac{p(F|G)p(G)}{p(F)} = \frac{p(F|G)p(G)}{p(F|G)p(G) + p(F|\sim G)p(\sim G)}$$

Solution: Prosecutor’s fallacy

$$p(G|F) = \frac{p(F|G)p(G)}{p(F)} = \frac{p(F|G)p(G)}{p(F|G)p(G) + p(F|\sim G)p(\sim G)}$$

- $p(G) = ? \rightarrow$ Impossible!