UNIVERSITY of
HOUSTON
DEPARTMENT OF COMPUTER SCIENCE

Classification

Rakesh Verma

Supervised Learning vs. Unsupervised Learning

Supervised Learning undergoes supervision.

Training data is labeled with true class label values for each observation/measurement.

New data is classified using these what the model has learned from the training set.

Classification is Supervised Learning.

Unsupervised Learning doesn't.

Training data is unlabeled, and often unknown even to the experimenter.

Given a set of observations/measurements, the goal is to find latent classes or groupings within the data.

Clustering is Unsupervised Learning.

Numeric Prediction vs. Classification

Classification

- Predicts nominal class values.
- Classifies data based on a labeled training set.

Numeric Prediction

- Models continuous-valued functions and predicts their output values.

Typical Applications

- Spam Email Detection
- Facial Recognition
- Fraud Detection
- Document Categorization

The Two-Step Process for Classification

Model Training

Each instance in the training set belongs to a class (it is assigned a Class Label).

The model uses this information either through classification rules, decision trees, or other mathematical formulations.

Model Application

Quality Assurance: How good is the model?

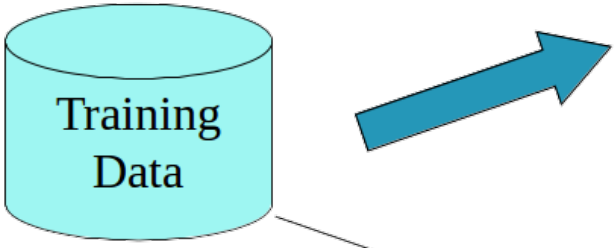
Accuracy is the percentage of correctly classified test samples.

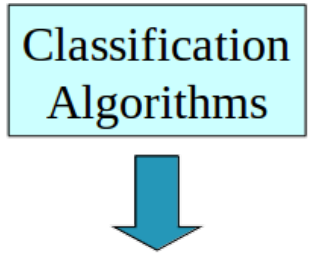
Test samples are instances for which the class label is known but the instance was not in the training set.

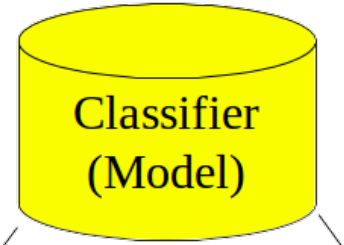
Testing on elements of the training set can lead to overfitting.

If the set of testing samples (test set) is used to select models, it's called a validation test set.

Model Training







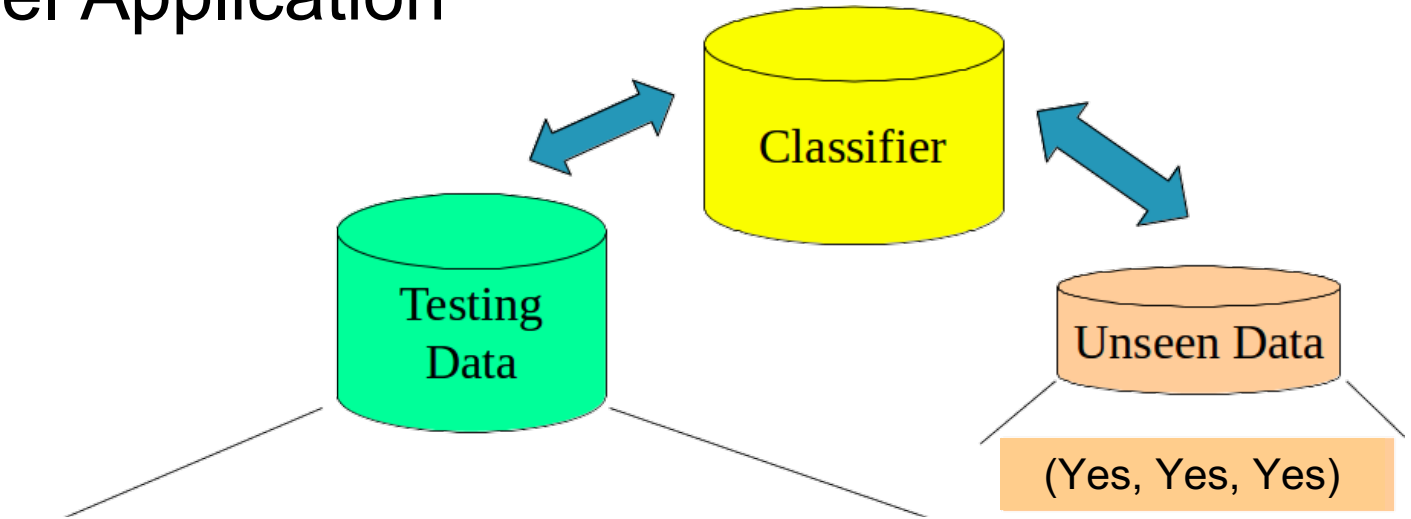
"Free money" in body	Unknown Sender	Has link	Spam
No	Yes	No	No
No	No	No	No
No	Yes	Yes	Yes
Yes	Yes	No	Yes
No	No	Yes	No

IF Unknown_Sender = 'Yes'

AND Has_Link = 'Yes'

THEN Spam = 'Yes'

Model Application



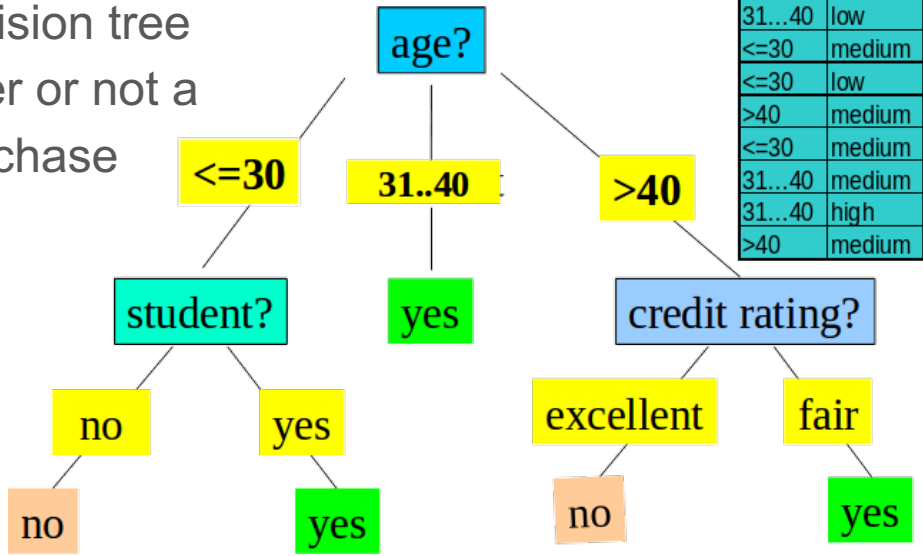
"Free money" in body	Unknown Sender	Has link	Spam
No	Yes	No	No
No	Yes	Yes	Yes
Yes	Yes	No	Yes
No	No	Yes	No

Decision Trees

Decision Trees use a series of decisions to arrive at a reasonable classification decision.

Here, we see a decision tree predicting whether or not a customer will purchase a computer.

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no



Algorithm for Building a Decision Tree

- Basic Algorithm (Divide-and-conquer)
  - At the beginning, no instances are separated.
  - Select a feature that hasn't been branched on at any ancestor node.
    - Partition instances at the current node based on that instances value for that feature.
    - Instances whose feature values are in the same category go down the same branch.
      - This is usually determined through inequalities or binning of values.
  - At the next level repeat the partitioning further and further down the tree.
  - The choice of features should be done using some knowledge or heuristic to efficiently build the tree.
- Stopping Conditions (per tree path)
  - All samples in the current node are in the same class.
  - All possible feature branches have been used.
    - In this case, some measure should be used to determine what the class of this leaf is.
    - Majority voting, usually.

Entropy

Entropy (specifically Shannon Entropy) is the expected amount of information within a random variable.

This can be considered to be a measure of how uncertain you can expect to be of the random variable taking any one value.

Let X be a discrete random variable. The Shannon Entropy is H(X), where H(X) is defined as:

H(X) = - $\sum_{i=1}^{n} P(x_i) \log P(x_i)$, where the base of the log is 2 usually

The higher the entropy, the higher the uncertainty (less likely to guess state).

The lower the entropy, the lower the uncertainty (more likely to guess state).

Conditional Entropy

The uncertainty of X given that you measure the value of A.

$$H(X|A) = \sum_{a \in \mathcal{A}} p_a H(X|A = a)$$

$$H(X|A) = \sum_{a \in \mathcal{A}, x \in \mathcal{X}} p(a, x) \log \frac{p(a)}{p(a, x)}$$

Attribute Selection – Information Gain

Let D be the dataset, a collection of instances.

Let C be a random variable that takes the class of an instance x ϵ D

The information of the dataset D, Info(D), is the Shannon entropy of C

So Info(D) = H(C) = $-\sum_c P(c) \log P(c)$, where P(c) is the probability of correctly guessing class c.

Attribute Selection - Information Gain

Information gain is the change in uncertainty after observing A.

Let A be an attribute of D.

$$Info_A(D) = H(C|A)$$

$$Gain(A) = Info(D) - Info_A(D)$$

The attribute with the highest information gain is the most important.

Example of Information Gain

Class: **buys_computer**

Using log with base 2 for entropy

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Info(D) = 0.940$$
$$Info_{age}(D) = 0.694$$
$$Gain(age) = 0.940 - 0.694 = 0.246$$

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Information Gain Ratio

A flaw of Information Gain is that more diverse attributes have higher gain.

It's harder to guess the value of an attribute when there's more possible values.

Therefore, measuring the value of the attribute will give us a lot of information.

A solution is to scale information gained by the uncertainty of that attribute.

$$IGR_A(D) = \frac{Info_A(D)}{H(A)}$$

As an attribute's uncertainty increases, the ratio increases, but at a fair rate compared with the other attributes.

Overfitting and Tree Pruning

- Overfitting describes a model that is incapable of generalizing
  - Very accurate predictions within the training set.
  - Poor accuracy on unseen samples.
- For Decision Trees, having too many branches constitutes overfitting
- By pruning, we increase the ability of a Decision Tree to generalize.
  - Pre-pruning
    - During construction, halt construction of a tree path if gets too long
      - What is too long?
  - Post-pruning
    - After construction, remove branches from the tree
      - When do you stop?
      - Which branches do you remove?
        - Information Gain and Gain Ratio Measures can be used.

Decision Trees for Unbalanced Data

Classification against Big Data

Scalability

We want results eventually, but "next week" is unacceptable.

As the database increases in size, the time it takes to get results should stay reasonable.

Benefits of Decision Trees against Big Data

They're fast to train.

They're fast to apply.

Decision rules are easily convertible to SQL queries

This makes them a great online algorithm over an ever-changing database.

Their accuracy is comparable to other existing methods.

Bayesian Classification

Bayes' Theorem is at the heart of Bayesian Classification

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

When we want to know the probability of an event A given that we have measured B, we can use the probability of an event B given that we measured A.

In practice, the relationship between two events is known in one direction.

We don't know how likely you are to get cancer if we know you're 65 years old.

But we do know how many cancer patients were diagnosed at 65 years old.

Predicting with Bayes - Hack

X is an instance that describes a potential hacked webserver

P(Hacked|X) is the probability of X being hacked.

- This is the Posterior

P(X|Hacked) is the probability of webserver with the same description as X being compromised.

- For example, if X was IIS, version 6.0, and running on Windows Server 2013, this description could describe others.
- This is the Likelihood

P(X) is the probability of webserver having the same description as X

- This is the Evidence

P(Hacked) is the probability of any webserver being hacked.

- This is the Prior

Naive Bayes

- Applying Bayes directly is hard
  - Evidence and Likelihood involve complex relationships between attributes.
  - The overhead in tracking the dependence between attributes is sometimes intractable
    - Existing data doesn't necessarily encapsulate their true relationship either.
- The solution is to pretend that all attributes are independent
  - Whatever that complex relationship is? Ignore it.
  - This is called **strong** or **naive** independence assumption.
- This allows us to use properties of independent random variables.
  - The joint probability of independent random variables is their product.
  - P(age, gender, smoking_status) = P(age) P(gender) P(smoking_status)

Naive Bayes Example - Buy Computer

Class

P(comp = “yes”) = 9/14 = 0.643

P(comp = “no”) = 5/14= 0.357

Age

P(age = “<=30” | comp = “yes”) = 2/9 = 0.222

P(age = “<= 30” | comp = “no”) = 3/5 = 0.6

Income

P(income = “medium” | comp = “yes”) = 4/9 = 0.444

P(income = “medium” | comp = “no”) = 2/5 = 0.4

Student Status

P(student = “yes” | buys_comp = “yes) = 6/9 = 0.667

P(student = “yes” | buys_comp = “no”) = 1/5 = 0.2

Credit Rating

P(credit_rating = “fair” | comp = “yes”) = 6/9 = 0.667

P(credit_rating = “fair” | comp = “no”) = 2/5 = 0.4

P(X|comp) = 0.222 * 0.444 * 0.667 * 0.667 = 0.044
P(X|no comp) = 0.6 * 0.4 * 0.2 * 0.4 = 0.019

P(X|comp) P(comp) = 0.044 * 0.643 = **0.028**
P(X|no comp) P(comp) = 0.019 * 0.357 = 0.007

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naive Bayes - A shortcut

Why didn't we have to calculate P(X)?

Answer: P(X) was a common term between all class calculations.

P(X|C) P(C) > P(X|Not C) P(Not C)

This relation is preserved regardless of whether or not we include the division by P(X).

Naive Bayes Classification can therefore be simplified.

Select the class for which P'(C|X) is largest

$$P'(C|X) = P(X|C)P(C)$$

Zero-Probability

- What happens when the likelihood is zero for some X?
  - The entire probability of predicting the class C for that X is 0.
  - If we've never seen it, we'll never expect it.
    - A failure to generalize.
- Laplacian correction
  - Given an attribute for which some values are unrepresented
    - Calculate probabilities as if frequency for each value were one larger.
      - Order relation is preserved.
      - There are no longer 0 probabilities.
      - Actual probabilities are close enough to the augmented probabilities.

Naive Bayes Summary

- Pros
  - Easy to implement
  - "Good enough" results
- Cons
  - Centered around strong independence assumption.
    - Assumption is often wrong in practice.
    - Accuracy loss is a likely direct result of this assumption
- Improvements
  - Bayesian Belief Networks do not make use of strong independence
    - Therefore, they can encapsulate the dependence relationship between attributes