Introduction to Machine Learning

Rafal Bogacz

Outline

- Formal issues
- What is learning?
  - Types of learning task
  - Testing learning algorithms
- Content of the lectures
- Assignments

Materials

- Syllabus, Lecture notes, Assignments:
  - http://www.cs.bris.ac.uk/Teaching/Resources/COMSM0300/
- Recommended readings:
  - Mitchell T. (1997). Machine Learning. MIT Press.
  - Witten I.H. & Frank E. (2005). Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann. 2nd Ed.

Support

- Office hours
  - MVB 3.43, Let us choose them
- Forum
  - Link in top right corner of unit website
  - Post your questions on the forum, and discuss
  - I will reply to questions during office hours
  - Do not send me e-mails
- Helpdesk
  - General computing questions

Feedback on assignments

- Final mark: 50% coursework + 50% exam
- Marking components for individual parts
- Sample answers and codes
  - If you are NOT OK with showing your work, please let me know
- Jason will be available to discuss questions
  - Detailed time will be announced after making of each assignment
- Individual feedback will be provided for written part

Aims and objectives

- Aim
  - To be able to properly use machine learning tools in practical applications
- Objectives
  - To be able to choose an appropriate learning algorithm for a given problem
  - To use machine learning algorithms in solving classification problems
  - To understand theoretical limitations of machine learning

What does it mean to learn?

- Mitchell's definition:
  - A computer program solving task T learns from Experience E with respect to performance measure P,
    if its performance in task T as measured by P improves with experience E

Examples: Classification

- Task: Spam e-mail classification
  - P: % e-mails correctly classified
  - E: database of emails:
    - <e-mail1, spam>
    - <e-mail2, ham>
    - <e-mail3, ham>
    - …
- Other examples
  - Spoken word recognition (Vue cinemas, Bristol)
  - Loan application screening (American Express)
  - http://www.aicml.cs.ualberta.ca/research/machineLearning/index.php

Example: Regression

- Task: predicting electricity load
  - P: mean prediction error
  - E: data on electricity used hour by hour:
    - <<day, time, weather, previous usage>, usage>
    - …

Examples: Reinforcement

- Task: playing backgammon
  - P: %wins
  - E: playing practice games against itself
    - <<move1, move2, …, move28>, lost>
    - <<move1, move2, …, move34>, win>
    - …
- Other games that use machine learning:
  - Board: Chess, Go, Checkers, Poker
  - Strategy: Black & White, Nero

Examples: Unsupervised

- Task: Clustering of online-customers
  - P: similarity of customers within the clusters
  - E: database of customers (NO sample clusters)
    - <titles of books bought>
    - <titles of books bought>
    - …
  - Possible application:
    - Recommendation systems

Main types of learning tasks

- Type of experience (feedback)
  - Supervised
    - Classification (e.g. spoken words)
    - Regression (e.g. share prices)
  - Reinforcement (e.g. backgammon)
  - Unsupervised (e.g. clustering)
- Expected knowledge representation
  - Symbolic (e.g. rules)
  - Subsymbolic (e.g. weights of neural network)

Content of the lectures

- Supervised: classification
  - Symbolic: focus on decision trees
  - Subsymbolic: focus on Bayesian
  - Overview of other ~1001 algorithms
- Overview of other learning tasks
- General issues
  - Why learning from examples is possible?
  - Comparing learning algorithms
- Internet applications: spam & recommend

Other specialized units

- Symbolic classification
  - Learning from structured data
- Sybsymbolic classification
  - Pattern analysis and statistical learning
- Reinforcement learning
  - Adaptation in autonomous systems
- Unsupervised learning
  - Signals patterns and Symbols
  - Pattern analysis and statistical learning

Testing learning algorithms

- Goal of learner: generalization to unseen data
- To assess a learner we need to tests it on unseen data
- Approaches
  - Split data into training and testing sets
  - Ten-fold cross-validation:
    - Divide data into 10 parts
    - Repeat 10 times, on i^{th} iteration:
      - Train learner on all parts except for i^{th}
      - Test learner on i^{th} part
    - Return mean error rate

Tools

- WEKA
  - Waikato Environment for Knowledge Analysis
  - Each learning algorithm implemented as a 'stand-alone' Java class
  - Two interfaces:
    - Graphic – easy to experiment and train the learner
    - Command line – allows to use trained learner in your own application
- Matlab – Neural networks toolbox
- Mathematica – Machine learning framework

Assignments

- Using Weka (2 parts)
- Spam filter
  - Part 1: Automatic marking on artificial data
  - Part 2: Real data, testing set not revealed
  - Prizes: highest accuracy, most interesting
- Marking
  - Only 60% specified – easy to get
  - Mark above 60% requires using one of the more advanced algorithms

Syllabus

- Detailed syllabus at:
  - http://www.cs.bris.ac.uk/Teaching/Resources/COMSM0300/
- Highlights:
  - No lectures on 2nd, 5th, 9th November
  - Instead 3 lectures in January
    - Covering case studies / not tested during exam
  - Self-review of probability theory

Why learning from examples is possible?

Lecture 2

Sample learning problem

- Examples

Sky	Temper.	Rain	Wind	Fly Balloon
Sunny	Cold	None	Weak	Yes
Cloudy	Cold	None	Weak	Yes
Cloudy	Cold	Shower	Strong	No
Sunny	Hot	None	Weak	Yes

- Are the following days OK for ballooning?

Cloudy	Hot	Shower	Strong
Cloudy	Hot	None	Weak

Outline

- Learning a concept
  - Learning as a search of hypothesis space
  - Version space
- The inductive bias = algorithm's assumptions
  - The need for the bias
  - Biases of different learning algorithms

Representing hypotheses

- Many possible representations
- Let's choose:
  - h is conjunction of constraints on attributes
  - Each constraint can be:
    - A specific value (e.g., Sky = Sunny)
    - "Don't care" (e.g., Sky = ?)
  - For example
    - Sky Temp Rain Wind
      < ? Cold ? Weak>

Concept learning task

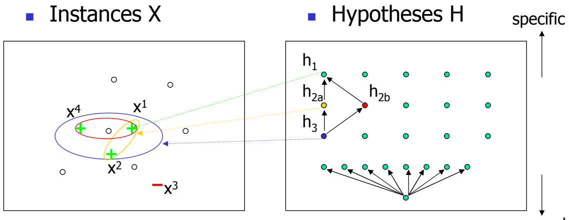
- Given:
  - Instances X: e.g. possible days described by attributes: Sky, Temperature, Rain, Wind
  - Hypotheses H: e.g. all conjunctions of literals
    - Each hypothesis h in H is a function, i.e. $h: X \rightarrow \{0,1\}$
  - Training examples D:
    $<x^1, c(x^1)>, \ldots, <x^m, c(x^m)>$
    where c – target function i.e. $C: X \rightarrow \{0,1\}$
- Determine: a hypothesis h in H such that
  - is consistent: $h(x) = c(x)$ for all x in D

Generality of hypotheses

- More general hypothesis classifies more examples as positive (covers more examples).
- For our hypothesis language:
  - More general has value replaced by "?"
  - E.g. <?,?,?,Weak> $\geq_g$ <Sunny,?,?,Weak>
- Definition:
  - Let h_1 and h_2 two hypotheses (boolean functions)
  - h_1 is more general than or equal to h_2 ($h_1 \geq_g h_2$) if and only if:
  - $\forall x \in X$: $h_2(x)=1 \Rightarrow h_1(x)=1$

Ordering of hypotheses

- Instances X
- Hypotheses H



x^1 =<Sunny,Cold,None,Weak>
x^2 =<Cloudy,Cold,None,Weak>
x^3 =<Cloudy,Cold,Shower,Strong>
x^4 =<Sunny,Hot,None,Weak>

h_1 =<Sunny,Cold,None,Weak>
h_{2a} =<?,Cold,None,Weak>
h_{2b} =<Sunny,?,None,Weak>
h_3 =<?,?,None,Weak>

Version space

- The version space with respect to hypothesis space H & training examples D is the subset of hypotheses from H consistent with all training examples in D
- Ballooning version space
  - Specific boundary: {<?,?,None,Weak>}
  - General bound: {<?,?,None,?>, <?,?,?,Weak>}

Find-S: Finding most specific hypothesis

- Initialize h to the first positive example
- For every other positive example x
  - For each training attribute constraint a_i in h
    - If a_i in h is satisfied by x
    - Then do nothing
    - Else replace a_i in h by the next more general constraint that is satisfied by x
- Output h

Problems with Find-S

- Finds the most specific hypothesis, but why?
  - Maybe hypotheses of intermediate generality would be better
- Cannot tell if the training data is inconsistent
  - Because uses only positive examples
- There may be several most specific hypotheses, but Find-S finds only one.

Assumptions we made

- By choosing the representation of hypotheses we made assumption that it is possible to represent the correct rule as conjunction
- This representation is limited,
  - e.g. No consistent hypotheses for:

Sky	Temper.	Rain	Wind	Fly Balloon
Sunny	Cold	None	Strong	Yes
Cloudy	Cold	Shower	Weak	Yes
Cloudy	Cold	Shower	Strong	No
Sunny	Hot	Shower	Strong	No

Unbiased Learner

- We can extend the hypothesis representation to allow disjunction and negation
- Sample hypotheses in this new language:
  - <?,?,None,?> $\vee$ <?,?,?,Weak>
  - $\neg$<?,?,Shower,Strong>
- Size of hypothesis space H in new language much larger than before

Futility of bias-free learning

- The most specific hypothesis
  - classifies two positive examples as positive, and all other as negative => useless
- The most general hypothesis
  - classifies two negative examples as negative, and all other as positive => useless
- A learner that makes no assumptions about the identity of the target concept has no rational basis for classifying unseen instances

13

Inductive bias

- Consider
  - Concept learning algorithm L
  - Set of possible instances X, target concept c
  - Training examples D = {<x, c(x)>}
  - Let $L(x^i,D)$ denote the classification assigned to the instance x^i by L after training on data D
- The inductive bias is any minimal set of assumptions B such that for any c, D:
  $\forall x^i \in X \quad (B \wedge D \wedge x^i) \;=>\; L(x^i,D)$

14

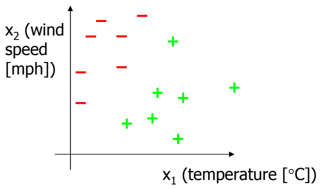
Intuition for bias

- Role of the inductive bias
  - Limits hypothesis space
  - Informs which of the consistent hypothesis chosen
- Bias of Find-S
  - All positive instances have the same values of a subset of attributes
  - All negative examples have some of these attributes different
    - E.g. this was the case in the "Ballooning for beginners" data set for attributes: Rain and Wind

15

Subsymbolic classification

- Subsymbolic classifiers draw curves (or manifolds) in the attribute space, e.g.

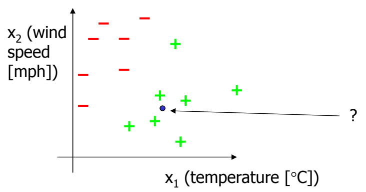


- What to draw: line vs. curve? Where?

16

Bias of subsymbolic classifiers

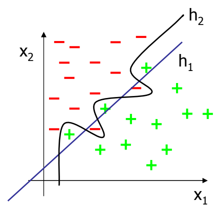
- The subsymbolic classifiers (like you) assume that "the world is smooth".



17

The importance of correct bias

- Consider data

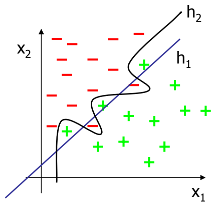


- Biases of hypotheses
- h_1 assumes data are noisy
- h_2 assumes data are precise
- If data are precise, h_2 will classify unseen examples better than h_1
- If data are noisy, h_1 will classify unseen examples better than h_2

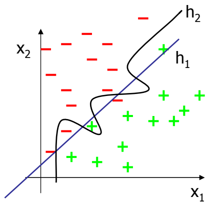
18

The importance of correct bias

- Consider data



- Likely testing set



19

Remember from this unit!!!

- The choice of learning algorithm will influence the accuracy of classification of unseen data
- The best classification of unseen data will be achieved by the algorithm whose bias best matches the statistics of the data
- There is no single best classifier
- Hence before you choose the algorithm
  - Learn as much as possible about your data
  - Visualize data
- The more background knowledge you add to the algorithm, the better it will work

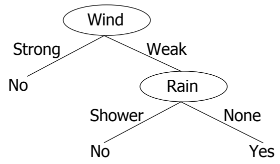
20

Decision trees

Lectures 4-5

Example

- Representation:
  - Internal node = attribute
  - Branch = attribute value
  - Leaf = classification



Outline

- Induction of Decision Trees (ID3)
- Bias of decision trees
- Avoiding over-fitting
- Converting tree into set of rules
- Rule learning

Induction of Decision Trees

- Recursive algorithm:
  - ID3 (examples)
    - If all examples belong to the same class
    - THEN root = leaf: class of examples
    - ELSE
      - Choose A <- "best" decision attribute
      - Assign A as decision attribute for root
      - For each value v of attribute A
        - Let $examples_v$ = subset of examples for which A = v
        - Create new $node_v$ = ID3($examples_v$)
    - Return root
- Question: what is the "best" attribute?

Entropy(1)

- Denote
  - S – sample of examples
  - p_i – proportion of examples from class c_i
- Entropy (S) =
  - expected number of bits needed to encode class of randomly drawn example under optimal code

Entropy(2)

- Example: 4 classes
  - Case1: $p_1=p_2=p_3=p_4=25\%$
    - Optimal codes: 00, 01, 10, 11
    - Entropy = 2
  - Case2: $p_1=50\%$, $p_2=25\%$, $p_3=p_4=12.5\%$
    - Optimal codes: 0, 10, 110, 111
    - Entropy = ½*1 + ¼*2 + 2*1/8*3 = 1.75
- Optimal code assigns $-\log_2 p_i$ bits to message with probability p_i
- So expected number of bits to encode the class is:

$$Entropy(S) = \sum_i -p_i \log_2 p_i$$

Information gain

- Gain (examples, A) =
  - expected reduction in entropy due to sorting on A

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

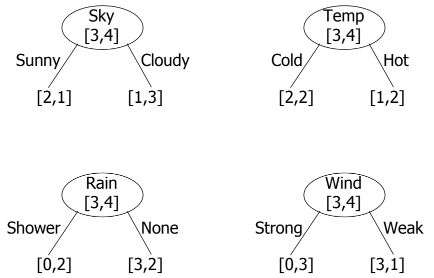
- Where
  - S_v – subset of examples with attribute A equal to v

Example

- Ballooning for beginners

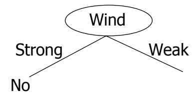
Sky	Temper.	Rain	Wind	Fly Balloon
Sunny	Cold	None	Weak	Yes
Cloudy	Cold	None	Weak	Yes
Cloudy	Cold	Shower	Strong	No
Sunny	Hot	None	Weak	Yes
Sunny	Hot	None	Strong	No
Cloudy	Hot	None	Strong	No
Cloudy	Cold	Shower	Weak	No

Choice of attribute



Building of the tree

- We choose "Wind" for root
  - All examples with "Strong" are "No"
  - We have to choose feature for examples with "Weak"

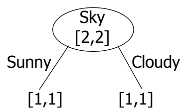


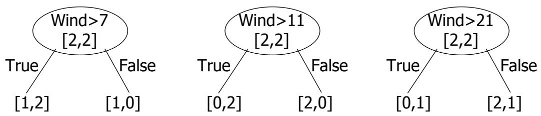
Sky	Temper.	Rain	Wind	Fly Balloon
Sunny	Cold	None	Weak	Yes
Cloudy	Cold	None	Weak	Yes
Sunny	Hot	None	Weak	Yes
Cloudy	Cold	Shower	Weak	No

Continuous attributes

- Try all possible values of threshold, e.g.

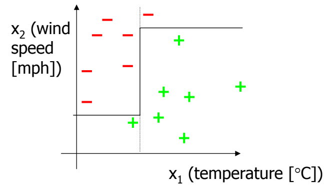
Sky	Wind	Balloon
Sunny	5	Yes
Cloudy	9	Yes
Sunny	13	No
Cloudy	30	No





Inductive bias of trees

- Divides feature space into "boxes"



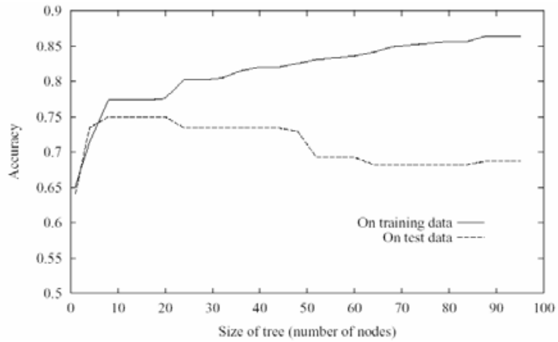
- Prefers shorter trees
  - High information gain attributes are near the root

Occam's Razor

- "If two theories explain the facts equally well, then the simpler theory is to be preferred"
- Rationale
  - There is fewer short hypotheses than long hypotheses
  - A short hypothesis that fits data is unlikely to be a coincidence
  - A long hypothesis that fits data may be a coincidence
- Formal treatment: Lecture 6.5

William of Occam
(1285-1349)

Overfitting on noisy data



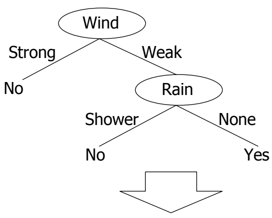
Avoiding overfitting

- How can we avoid overfitting?
  - Stop growing when data split not statistically significant
  - Grow full tree, then post-prune
- How to select the "best" tree?
  - The one that minimizes the error on test set
  - Minimum Description Length
    - Minimize: ER + w * size (tree)

Reduced-Error Pruning

- Split data into training and testing set
- Induce a tree from the training set
- Repeat until further pruning is harmful
  - Evaluate impact on testing set of pruning each possible node (and those below it)
  - Greedy remove the one that most improves error on testing set

Converting a tree to rules



- IF Wind=Strong THEN Balloon=No
- IF Wind=Weak $\wedge$ Rain=Shower THEN Balloon=No
- IF Wind=Weak $\wedge$ Rain=None THEN Balloon=Yes

Rule learning

- Algorithms which learn rules (below) directly from the data, without deriving the tree
  - IF Wind=Weak $\wedge$ Rain=None THEN Balloon=Yes
- Two types
  - Sequential covering
    - Learn rule
    - Remove data it covers
    - Repeat until accuracy criterion satisfied
  - Unordered rules
    - On test, different rules may give different answer
    - Resolution method required

Learning single rule

- For each class c
  - Start with empty rule: IF THEN class = c
  - Repeat
    - For each possible condition, compute:
      - t – total number of examples the condition covers
      - p – number of examples covered belonging to class c
    - Add condition which maximizes p/t
      - Break ties by choosing condition with largest p
  - Until accuracy criterion satisfied

Example

- Consider class Balloon=Yes

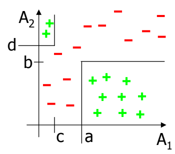
Condition	p	t
Sky=Sunny	2	3
Sky=Cloudy	1	4
Temper.=Cold	2	4
Temper.=Hot	1	3
Rain=None	3	5
Rain=Shower	0	2
Wind=Weak	3	4
Wind=Strong	0	3

Rule learners vs. Trees

- Difference in choice of condition
  - Rule learner:
    - Maximize accuracy of classification between class c and all other classes (disregarding what happens to other classes)
  - Decision Trees:
    - Maximize information gain (take all classes into account trying to maximize purity of the split)
- Differences in "separation lines"
  - See next two slides

Example: Rule learner

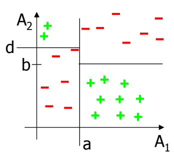
- Training set



- Rules:
  - IF A_1>a $\wedge$ A_2<b THEN +
  - IF A_1<c $\wedge$ A_2>d THEN +
  - IF true THEN –
- Note:
  - Rules must be executed in order
  - Sets of instances they cover overlap

Compare: Trees

- Training set



- Rules:
  - IF A_1>a $\wedge$ A_2<b THEN +
  - IF A_1>a $\wedge$ A_2>b THEN –
  - IF A_1<a $\wedge$ A_2>d THEN +
  - IF A_1<a $\wedge$ A_2<d THEN –
- Note:
  - Rules may be executed in any order
  - Sets of instances they cover do not overlap

Summary

- Decision trees – provide very easy to understand explanations for choices made
- Divide feature space into "boxes"
- Overfitting
  - When the number of nodes too large, error on testing set will increase
- Avoiding overfitting
  - Criterion for stopping growing the tree
  - Post-pruning

Discrete Bayesian classifiers

Lectures 6-7

Outline

- Bayes theorem
- Maximum likelihood classification
- "Brute force" Bayesian learning
- Naïve Bayes

Bayes theorem

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

- P(c) – prior probability of class c
  - Expected proportion of data from class c on test
- P(x) – prior probability of instance x
  - Probability of instance with attribute vector x to occur
- P(c|x) – probability of an instance being of class c given it is described by vector of attributes x
- P(x|c) – probability of an instance of having attributes described by x given it comes from class c

Maximum likelihood

- From training data estimate for all x and c_i
  - $P(c_i)$
  - $P(x|c_i)$
- During classification:
  - Choose class c_i with maximal $P(c_i|x)$
- $P(c_i|x)$ is also called likelihood

'Brute force' Bayesian learning

- Instance x described by attributes $<a_1,\ldots,a_n>$
- Most probable class:

$$c(x) = \arg\max_{c_i} P(c_i \mid a_1,\ldots,a_n) =$$

$$= \arg\max_{c_i} \frac{P(a_1,\ldots,a_n \mid c_i)P(c_i)}{P(a_1,\ldots,a_n)} =$$

$$= \arg\max_{c_i} P(a_1,\ldots,a_n \mid c_i)P(c_i)$$

Example

- Consider Data

Wind	Rain	Balloon
Strong	Shower	No
Strong	Shower	No
Strong	None	No
Weak	None	Yes
Weak	Shower	No
Weak	None	Yes
Weak	None	No
Weak	None	Yes

- We estimate:
  - P(No) = 62%
  - P(Yes) = 38%
  - P(<Weak,None>|No) = 20%
  - P(<Weak,None>|Yes)=100%
  - P(<Weak,Show.>|No) = 20%
  - …
- Classification of <Weak,None>
  - L(No) ~ 0.62 x 0.2= 0.12
  - L(Yes) ~ 0.38 x 1= 0.38
  - Answer: Yes

Problem with 'Brute force'

- It cannot generalize to unseen examples x^{new}, because it does not have estimates $P(c_i|x^{new})$
- It is useless
- Brute force does not have any bias
- So in order to make learning possible we have to introduce a bias

Naïve Bayes

- Brute force: $c(x) = \arg\max_{c_i} P(a_1,\ldots,a_n \mid c_i)P(c_i)$
- Naïve Bayes assumes that attributes are independent for instances from a given class:

$$P(a_1,\ldots,a_n \mid c_i) = \prod_j P(a_j \mid c_i)$$

- Which gives: $\boxed{c(x) = \arg\max_{c_i} P(c_i)\prod_j P(a_j \mid c_i)}$
  - Assumption of independence is often violated by Naïve Bayes works surprisingly well anyway

Example

- Recall 'advanced ballooning' set:

Sky	Temper.	Rain	Wind	Fly Balloon
Sunny	Cold	None	Strong	Yes
Cloudy	Cold	Shower	Weak	Yes
Cloudy	Cold	Shower	Strong	No
Sunny	Hot	Shower	Strong	No

- Classify: x=<Cloudy, Hot, Shower, Strong>
  - P(Y|x) ~ P(Y) P(Cl|Y) P(H|Y) P(Sh|Y) P (St|Y)
    = 0.5 x 0.5 x 0 x 0.5 x 0.5 = 0
  - P(N|x) ~ 0.5 x 0.5 x 0.5 x 1 x 1 = 0.125

Missing estimates

- What if none of training instances of class c_i have attribute value a_j? Then:
  - $P(a_j|c_i) = 0$, and
  - $P(a_1,\ldots,a_n \mid c_i) = \prod_j P(a_j \mid c_i) = 0$
  - no matter what are the values of other attributes
- For example:
  - x = <Sunny, Hot, None, Weak>
  - P(Hot|Yes) = 0, hence
  - P(Yes|x) = 0

Solution

- Let m denote the number of possible values of attribute a_j
- For each class let us consider adding m "virtual examples" with different values of a_j
- Bayesian estimate for $P(a_j|c_i)$ becomes:

$$P(a_j \mid c_i) = \frac{n_{ciaj} + 1}{n_{ci} + m}$$

- Where:
  - n_{ci} – number of training examples with class c_i
  - n_{ciaj} – number of training examples with class c_i and attribute a_j

Learning to classify text

- For example: is an e-mail a spam?
- Represent each document by a set of words
  - Independence assumptions:
    - Order of words does not matter
    - Co-occurrences of words do not matter
- Learning: estimate from training documents:
  - For every class c_i estimate $P(c_i)$
  - For every word w and class c_i estimate $P(w|c_i)$
- Classification: maximum likelihood

Learning in detail

- Vocabulary = all distinct words in training text
- For each class c_i
  - $P(c_i) = \dfrac{\textit{Number of documents of class } c_i}{\textit{Total number of documents}}$
  - $Text_{ci}$ = concatenated documents of class c_i
  - n_{ci} = total # words in $Text_{ci}$ (count duplicates multiple times)
  - For each word w_j in Vocabulary
    - n_{ciwj} = number of times word w_j occurred in text $Text_{ci}$
    - $P(w_j \mid c_i) = \dfrac{n_{ciwj} + 1}{n_{ci} + |Vocabulary|}$

Example

- Training set:
  - <"buy rolex", spam>, <"machine learning", ham>, <"buy machine", spam>
- Learning:
  - P(spam) = , P(ham)=
  - n_{spam} = , n_{ham} =
  - |Vocabulary| =
  - P(machine|spam) =
  - P(machine|ham) =

Classification in detail

- Index all words in document to classify by j
  - i.e. denote j^{th} word in the document by w_j
- Classify: $c(document) = \arg\max_{c_i} P(c_i) \prod_j P(w_j \mid c_i)$
- In practice $P(w_j|c_i)$ are small so their product is very close to 0; it is better to use:

$$c(document) = \arg\max_{c_i} \log\left[P(c_i)\prod_j P(w_j \mid c_i)\right] =$$

$$= \arg\max_{c_i}\left[\log P(c_i) + \sum_j \log P(w_j \mid c_i)\right]$$

Pre-processing

- Allows adding background knowledge
- May dramatically increase accuracy
- Sample techniques:
  - Lemmatisation - converts words to basic form
  - Stop-list - removes 100 most frequent words

Understanding Naïve Bayes

- Although Naïve Bayes is considered to be subsymbolic, the estimated probabilities may give insight on the classification process
- For example in spam filtering
  - Words with maximum $P(w_j|spam)$ are the words whose presence most predicts en e-mail to be a spam e-mail

Summary

- Inductive bias of Naïve Bayes:
  - Attributes are independent.
- Although this assumption is often violated, it provides a very efficient tool often used
  - E.g. For spam filtering.
- Applicable to data:
  - with many attributes (possibly missing),
  - which take discrete values (e.g. words).

Bayesian belief networks

Lecture 8

Outline

- Conditional independence
- Bayesian Belief Networks

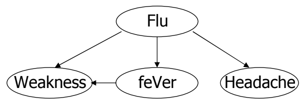
Conditional independence

- X is conditionally independent of Y given Z if
  - $\forall$x,y,z: P(X=x | Y=y, Z=z) = P(X=x | Z=z)
  - Usually written: P(X|Y,Z) = P(X|Z)
  - Example: P(Thunder|Rain,Ligthining)=P(Thunder|Lightning)
- Used by Naïve Bayes:
  - $P(A_1,A_2|C) = P(A_1|A_2,C)\ P(A_2|C) = P(A_1|C)P(A_2|C)$

Always true

Only true if A_1 and A_2 conditionally independent

Bayesian Belief Network

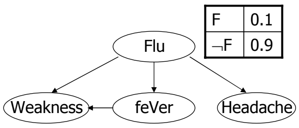


- Connections describe dependence & causality
  - Each node is conditionally independent of its nondescendants, given its immediate predecessors
  - Examples:
    - feVer and Headache are independent given flu
    - feVer and weakness are not independent given flu

Learning Bayesian Network

- Probabilities of attribute values given parents can be estimated from the training set

F	0.1
¬F	0.9



	FV	¬FV	F¬V	¬F ¬V
W	0.9	0.8	0.5	0.05
¬W	0.1	0.2	0.5	0.95

	F	¬F
V	0.8	0.1
¬V	0.2	0.9

	F	¬F
H	0.6	0.1
¬H	0.4	0.9

Inference

- During Bayesian classification we compute:

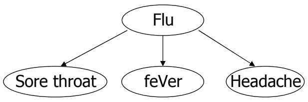
$$c(x) = \arg\max_{c_i} P(a_1,\ldots,a_n \mid c_i)P(c_i) = \arg\max_{c_i} P(a_1,\ldots,a_n,c_i)$$

- In general in Bayesian network with nodes Y_i:

$$P(y_1,\ldots,y_n) = \prod_{i=1}^{n} P(y_i \mid Parents(Y_i))$$

  - Thus $P(a_1,\ldots,a_n,c) = \prod_{i=1}^{n} P(a_i \mid Parents(A_i))P(c)$
- Example: Classify patient: W,V,¬H
  - P(W,V,¬H,F) = P(W|VF) P(V|F) P(¬H|F) P(F)

Naïve Bayes network



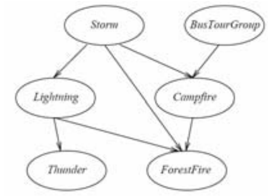
- In case of this network:

$$P(a_1,\ldots,a_n,c)=\prod_{i=1}^{n}P(a_i \mid Parents(A_i))P(c)$$

$$=\prod_{i=1}^{n}P(a_i \mid c)P(c)$$

Extensions to Bayesian nets

- Network with hidden states, e.g.



- Learning structure of the network from data

Summary

- Bayesian belief networks
  - Assume that only subset of attributes independent
  - Allow prior knowledge about dependencies

Comparing accuracy of classifiers

Lectures 11-12

Example

- We did five-fold cross validation
- Can we tell that learning algorithm A is better than B on a given dataset based on:

Test set	Error of A	Error of B
Data 1	5%	6%
Data 2	10%	6%
Data 3	8%	4%
Data 4	20%	15%
Data 5	12%	11%

- How confident can we be of our judgement?

Outline

- Review of statistics
  - Estimators
- Comparing accuracy using paired t-test
- T-tests in general
- Considering different types of errors

Mean vs. average

- When we know probability distribution:
  - Mean

$$E(X) = \sum_x x P(x)$$

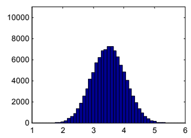
- When we observe sample $x_1, \ldots, x_N$:
  - Average is an estimator of mean

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

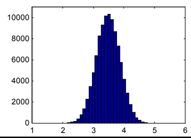
- When $N \to \infty$, then $\bar{x} \to E(X)$
- How different $\bar{x}$ and E(X) may be for finite N?

Simulation

- Repeat 100,000 times
  - Generate $x_1, \ldots, x_N$ from "simulated die"
  - Compute average $\bar{x}$
- Histogram of $\bar{x}$ for N=10



- Histogram of $\bar{x}$ for N=20



Distribution of the average

- Let X has mean μ and standard deviation σ
- According to the central limit theorem
  - the sum of $x_1, \ldots, x_N$ has approximately normal distribution with mean $N\mu$ and standard dev. $\sqrt{N}\sigma$

$$\sum_{i=1}^{N} x_i \sim Norm\left(N\mu, \sqrt{N}\sigma\right)$$

- The average has also normal distribution:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \sim Norm\left(\mu, \frac{\sigma}{\sqrt{N}}\right)$$

Example

- One week before the election in country C two parties A and B have exactly equal support of 50%
- What is the probability that opinion poll of 100 voters will show support for B ≤ 45% ?
- If X=voting for B, then recall (from Lecture 3)
  - E(X)=½, std(X)=½
- Let $\bar{x}$ the average of 100 respondents
  - E($\bar{x}$)= , std($\bar{x}$)=

7

Comparing paired data

- Consider the results:

Test set	Error of A	Error of B	Difference
Data 1	5%	6%	-1%
Data 2	10%	6%	4%
Data 3	8%	4%	4%
Data 4	20%	15%	5%
Data 5	12%	11%	1%

- Comparing A vs. B is equivalent to asking whether the difference between their accuracies has positive/negative mean

8

Statistical testing (1)

- Denote the differences between accuracies of classifiers by $x_1, \ldots, x_N$
- Assume $x_1, \ldots, x_N$ were sampled from X
- When can we say with confidence if the mean of X is positive or negative?
- We first ask if we are sure that mean of X≠0
  - Will be explained on the following slides…
- If yes, we compute average $\bar{x}$, and
  - If $\bar{x}$>0, we say mean of X significantly >0
  - If $\bar{x}$<0, we say mean of X significantly <0

9

Statistical testing (2)

- When can we say with confidence if the mean of X is different from 0?
- We will proceed as follows:
  - We will assume that the mean of X is equal to 0
  - We will see how unlikely $x_1, \ldots, x_N$ are given this assumption
  - If $x_1, \ldots, x_N$ are very unlikely, we will say that the mean is significantly different from 0

10

Statistical testing (3)

- For sample $x_1, \ldots, x_N$, we estimate $\bar{x}$=2.6 and S=2.51
- We assume X has mean 0 and std=S
- We ask how unlikely it is for the average of N samples from X to be equal to $\bar{x}$ or more
- Let $x'_1, \ldots, x'_N$ be a new set of samples from X
- Significance level (usually denoted by p) = the probability that the average of $x'_1, \ldots, x'_N$ is further from 0 than $\bar{x}$?

11

T-test for computer scientists

```
function p = ttest4CS (x)
ITER = 100000;
N = length (x);
av = mean (x);
S = std (x);
p = 0;
for i = 1:ITER
    xprime = randn(1,N)*S;
    if abs (mean (xprime)) > abs (av)
        p = p+1;
    end
end
p = p / ITER;
```

12

Correction for small sample

- The function on the previous slide will return close approximation of p only for large N
- For small N, a correction is required for incorrect estimate of S
- This correction is done by the t-test or Student's test

13

A Student of Statistics

William Sealey Gosset
(1876-1937)

14

Statistical testing (4)

- When $p<0.05$, test is significant
  - Typically denoted by "*" next to the comparison
- Example
  - Differences in error [%]: -1, 4, 4, 5, 1
  - T-test returns $p=0.08$
  - Hence we cannot say that classifier B is significantly more accurate than A on the data set

15

Independence of accuracies

- t-test assumes that $x_1, \ldots, x_N$ are independent
- Are the accuracies in cross validation independent?
  - No, because training sets overlap
- Weka uses corrected t-test (Nadeau & Bengio, 1999)

16

T-tests in general

- Use t-test whenever you compare performance of algorithms
- Examples of problems from MSc theses:
  - Compare compression rate of video on 10 different movies
  - Compare average fitness of two evolutionary algorithms solving the same optimization problem
    - Run the algorithm 10 times with different initial populations

17

Types of t-test (1)

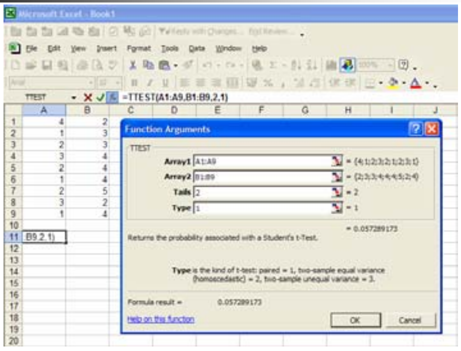
- Paired t-test
  - Apply when two algorithms tested on the same data sets
  - Described on previous slides
- Unpaired t-test
  - Apply when two algorithms tested on different data sets
  - E.g. two drugs tested on different patients
  - Excel distinguishes two subtypes:
    - Populations with equal vs. unequal variance

18

Types of t-test (2)

- Two tailed
  - Most frequently used
  - Described on previous slides
- One tailed
  - Modification which gives p value two times lower than two-tailed
  - Application:
    - Two-tailed test was not significant
    - But you still want to publish your paper ;-)

T-test in Excel



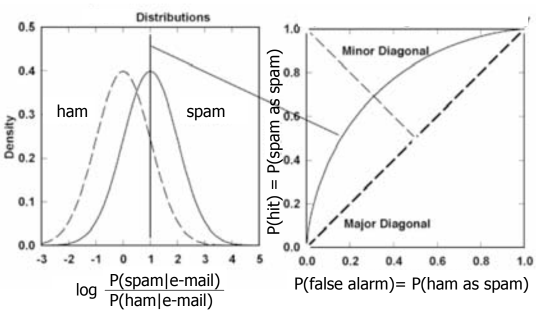
T-tests in Matlab

- Defining observations
  - A = [1,3,2,4,1,3,2,1,3];
  - B = [3,4,2,4,5,2,4,1,3];
- Paired t-test:
  - [h,p]=ttest(A-B);
- Unpaired t-test:
  - [h,p]=ttest2(A,B);

Different types of error

- Medical diagnoses: 2 types of errors:
  - False positive – classifying healthy patient as ill
  - False negative – classifying ill patient as healthy
- One type of error may be worse than another
- For some domains, it must be considered while comparing classifiers
- ROC analysis
  - Comparison method considering 2 error types
  - Use it to get extra points in SPAM assignment
  - More info: website of Prof. Peter Flach

ROC analysis



Summary

- Average accuracy (e.g. 10-x validation) alone cannot tell if one classifier is better than another for a given data set
- t-test checks how unlikely it is that the difference between accuracies of classifier has mean 0
  - If this is more than 95% unlikely than the test reports a significant difference.

What is the best classifier?

Lecture 13

Outline

- Conservation of generalization
- Quantifying inductive bias
- Predicting the error on testing set

2

Conservation of generalization

- Recall from lecture 2:
  - There is no single best classifier
  - A classifier performing well on one data set will be bad on other
- Shocking law:
  - When averaged over all learning problems all classifiers achieve equal average accuracy on testing set
  - Schaffer, C. (1994) A conservation law for generalization performance. ICML 11: 259-265.

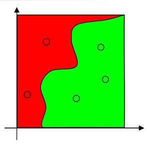
3

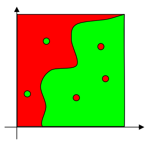
More formally

- For a given instance space X, e.g.
  - Binary vectors of length N
  - [0,1] x [0,1], etc.
- Learning problems are defined by
  - Target concept function c: X → {0,1}
  - Training set (subset of X)
  - Testing set (subset of X non-overlapping with tr.)
- Theorem:
  - Accuracy of any classifier on testing set averaged over all learning problems is 50%

4

Sketch of the proof





- Consider a learning problem A with a testing set
- There exist a "symmetric" weird learning problem A' identical to A except of classification of instances from the testing set
- For any learning algorithm:
  - Accuracy(A) = 1 - Accuracy(A')
  - Thus average accuracy on A and A' = 50%

5

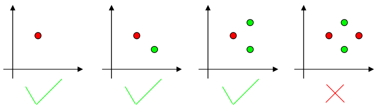
Smoothness of the world

- But most of real world problems are smooth
  - Two points in the instance space very close to each other are more likely to belong to the same class than two distant points
- Hence on average in real world problems:
  - k-nearest neighbour is more accurate than
  - random guesser
- The real practical question is:
  - What is the most appropriate inductive bias for a given learning problem?

6

Measure of inductive bias

- The complexity of the separating curve of a classifier can be measured by VC-dimension
  - Def: the largest number of points that can be separated in all possible way using given classifier
- Example: perceptron (linear separation)



  - VC-dimension (perceptron) = 3

VC-dimension

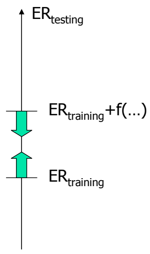
- Invented by Vapnik & Chervonenkis (1971)
- Vladimir Vapnik worked in Institute of Control Sciences, Moscow.
- Then moved to: AT&T Labs, NEC Labs, Princeton University
- Now: Royal Holloway, London

Bound on testing error

- Let $ER_{testing}$ – expected error rate on testing set
- $ER_{testing} \leq ER_{training} + f\ (VC_{classifier}\ /\ N_{training})$
  - Details given in Mendelson & Smola, p.45
- To remember:
  - The larger training set, the smaller expected difference $ER_{testing}$ - $ER_{training}$
  - The higher VC dimension of a classifier (the more complex curve or smaller inductive bias), the larger expected difference $ER_{testing}$ - $ER_{training}$
  - You can compute an estimate for your problem

William of Occam would love it!!!

Two bounds



- For non-crazy classifiers:
- $ER_{testing} \geq ER_{training}$

Summary

- When averaged over all learning problems all classifiers achieve equal average accuracy on testing set
- VC-dimension ~ complexity of boundary
- $ER_{testing}$ - $ER_{training}$ is
  - Decreased by high number of training examples
  - Increased by complexity of boundary (VC-dim.)
- When choosing classifier, consider:
  - The number of training examples you have