Machine Learning (Theorie + Labo)

Tom Croonenborghs (P224)
Peter Karsmakers (P224)
2012-2013
http://toledo.khk.be



THOMAS MORE

1 Introduction

- Some practical information about the course
- What is "machine learning and inductive inference"?
- What is it useful for?
- Different learning tasks
- Data representation
- Brief overview of approaches
- Overview of the course
  - Theory
  - Lab



THOMAS MORE

Practical information

- Course materials:
  - Slides, exercises
  - Additional resources (links, e-books,...)
  - Optional course text:
    - “Machine Learning and Inductive Inference”
    - Hendrik Blockeel
    - Acco
    - http://www.acco.be/uitgeverij/nl/course/9789033482977/machine+learning+and+inductive+inference



THOMAS MORE

Practical information

- Examination:
  - Machine Learning
    - Oral with written preparation
    - Mix of theory and exercises
    - All questions related to lectures / exercises
    - Focus on understanding and intuitions
  - Machine Learning Labo
    - Project based on human gait analyses (using a Wiimote)
    - Some more info later



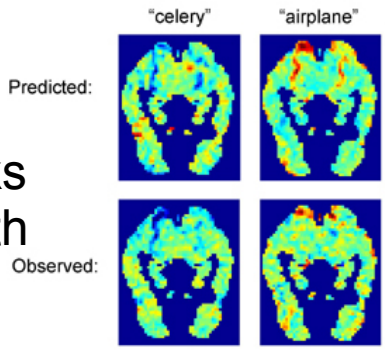
THOMAS MORE

What is Machine Learning?

- Arthur Samuel (1959): Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.



- Tom Mitchell (1998): A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E



What is machine learning?

- ***Study of how to make programs improve their performance on certain tasks from (own) experience***
  - "performance" = speed, accuracy, ...
  - "experience" = set of previously seen cases ("observations")
- For instance (simple method):
  - experience: taking action A in situation S yielded result R
  - situation S arises again
    - if R was undesirable: try something else
    - if R was desirable: try action A again

- This is a very simple example
  - only works if *precisely the same* situation is encountered
  - what if *similar* situation?
    - Need for *generalisation*
  - how about choosing another action even if a good one is already known ? (you might find a better one)
    - Need for *exploration*
- This course focuses mostly on *generalisation,* a.k.a. *inductive inference*

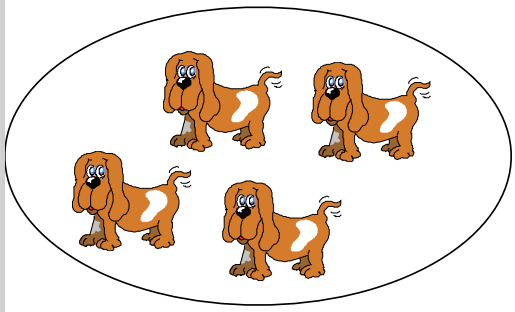


THOMAS MORE

Inductive inference

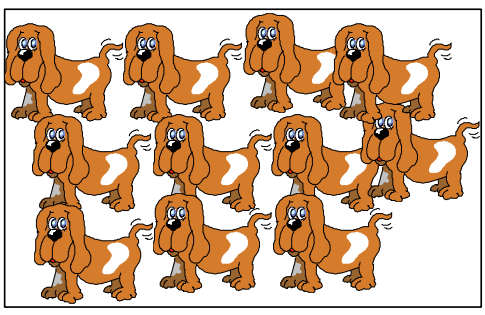
- = Reasoning from specific to general
- e.g. statistics: from sample, infer properties of population

sample



Observation: "these dogs are all brown"

population



hypothesis: "all dogs are brown"



THOMAS MORE

- Note: inductive inference is more general than statistics
  - statistics mainly consists of numerical methods for inference
    - infer mean, probability distribution, … of population
  - other approaches:
    - find symbolic definition of a concept (“concept learning”)
    - find laws with complicated structure that govern the data
    - study induction from a logical, philosophical, … point of view
    - …



THOMAS MORE

- Applications of inductive inference:
  - Machine learning
    - "sample" of observations = experience
    - generalizing to population = finding patterns in the observations that generally hold and may be *used* for future tasks
  - Knowledge discovery (Data mining)
    - "sample" = database
    - generalizing = finding patterns that hold in this database and can also be expected to hold on similar data not in the database
    - discovered knowledge = *comprehensible description* of these patterns
  - …



THOMAS MORE

What is it useful for?

- Scientifically: for understanding learning and intelligence in humans and animals
  - interesting for psychologists, philosophers, biologists, …
- More practically:
  - for building *AI systems*
    - expert systems that improve automatically with time
    - systems that help scientists make new discoveries
  - also useful *outside "typical" AI applications*
    - when we don't know how to program something ourselves
    - when a program should
      - adapt regularly to new circumstances
      - tune itself towards its user
      - try to understand better what the user wants



THOMAS MORE

Knowledge discovery

- Scientific knowledge discovery
  - Early discovery algorithms: mostly "toy" problems solved as proof of concept:
    - "Bacon" : rediscovered some laws of physics (e.g. Kepler's laws of planetary motion) from experimental data
    - "Automated Mathematician": rediscovered some mathematical theorems
  - Nowadays: more serious applications:
    - mining the human genome
    - mining the web for information on genes, proteins, …
    - drug discovery
- Also commercial applications



THOMAS MORE

Mining the genome

- Human (or other) genomes: encoded in DNA
  - long string of nucleotides (A,C,G,T)
- Methods known for determining this string (*sequencing*)
- Next step: understand the genome
  - Find out which parts perform which function
  - Machine learning / data mining techniques are used for this

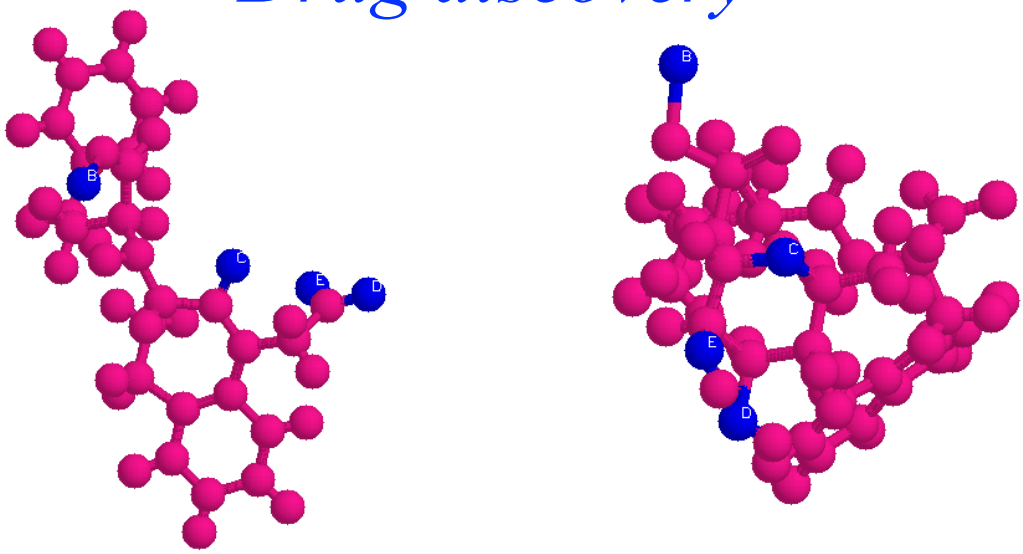
… TGAAGAGACTTAACCTATTACCGGACATCGACT…

Gene encoding for …



THOMAS MORE

Drug discovery



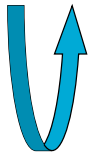
Example: given molecules that are active against some disease, find out what is common in them; this may be the reason for their activity.



THOMAS MORE

The "Robot Scientist"

- R.D. King et al., *Nature*, 2004
- Scientific research, e.g., in drug discovery, is iterative:
  - Determine what experiment to perform
  - Perform the experiment
  - Interpret the results; choose new experiments on the basis of this



- Robot Scientist automates the whole process ("closes the loop")
  - Reasons about its own learning process: which new experiments will be most informative?
  - Removing the human from the loop allows much faster progress



THOMAS MORE

Commercial applications

- Data mining in databases: looking for "interesting" patterns
- Useful for (for instance) marketing
  - based on data in DB, who should be interested in this new product? (useful for direct mailing)
  - study customer behaviour to identify typical profiles of customers
  - find out which products in store are often bought together

THOMAS MORE

Learning to perform difficult tasks

- Difficult for humans…
  - LEX system : learned how to perform symbolic integration of functions
- … or “easy” for humans, but difficult to program
  - humans can do it, but can’t explain how they do it
  - e.g.:
    - learning to play games (chess, go, …)
    - learning to fly a plane, drive a car, …
    - recognizing faces
    - …



THOMAS MORE

Intelligent monitoring in intensive care

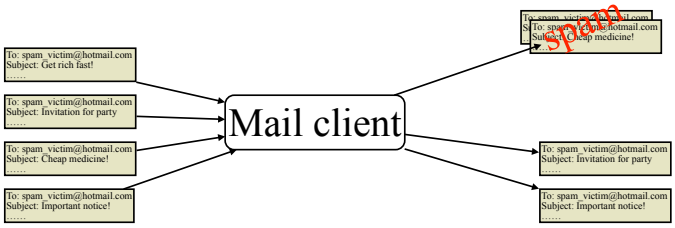
- Intensive care: continuous monitoring of patients
  - Crises may suddenly occur, need to react quickly
- Nowadays, machines continuously measure temperature, blood pressure, …
- Role of machine learning in this:
  - Try to *predict* problems / crises as early as possible, before they happen



THOMAS MORE

Anti-spam filtering

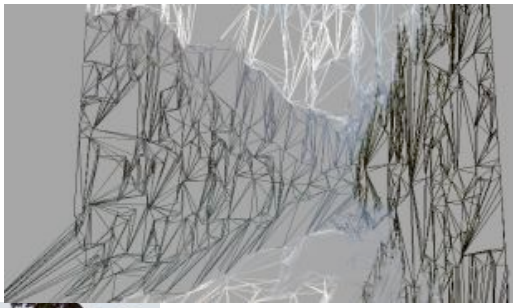
- Problem of excessive mail spam is well known
- Many spam filters have a learning component
  - Human user indicates mails that he/she considers spam
  - Filter learns the preference of the user
- Difficult problem:
  - Very low “false positive” rate needed
  - Continuous adaptation by spammers often spoils the learning efforts
- Effective anti-spam requires (combination with) other techniques



Make 3D

- Add depth to simple photos
- http://make3d.cs.cornell.edu/







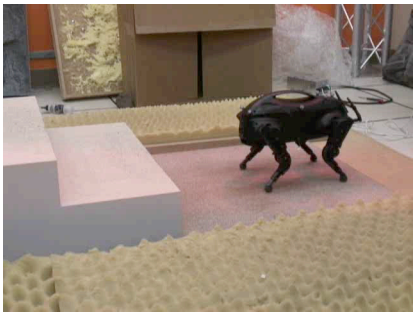
Adaptive systems

- Robots in changing environment
  - continuously need to adapt their behaviour
- Systems that adapt to the user
  - based on user modelling:
    - observe behaviour of user
    - build model describing this behaviour
    - use model to make user's life easier
  - Examples
    - adaptive web pages
    - adaptive user interfaces (e.g. intelligent Unix shell)
    - Web shops learning user's interests and offering potentially interesting info, see e.g. Google ads



THOMAS MORE

Robotics









THOMAS MORE

The Cocktail Party Problem

- Demo
  - http://research.ics.tkk.fi/ica/cocktail/cocktail_en.cgi

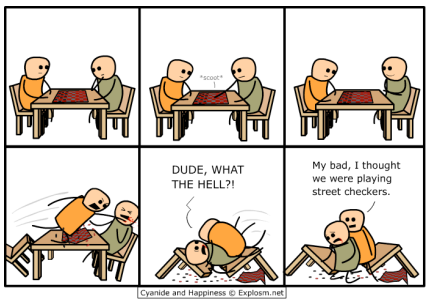


THOMAS MORE

Illustration : building a system that learns checkers

- (Based on example from Mitchell, *Machine Learning*, 1997)
- Learning = improving on task T, with respect to performance measure P, based on experience E
- Here:
  - T = playing checkers
  - P: % of games won in world tournament
  - E: games played against self
    - possible problem: is experience representative for real task?

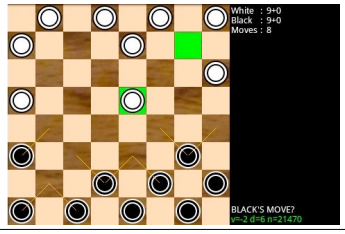


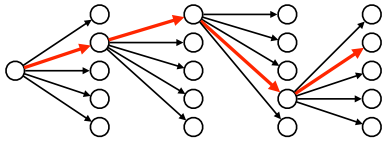




THOMAS MORE

- Need to specify more precisely:
  - What is given, what is learnt?
  - What data representations & learning algorithms to use
- What is given?
  - *direct* or *indirect* evidence ?
    - direct: e.g., which moves were good, which were bad
    - indirect: play whole game and see what outcome is
  - Advantage / disadvantage of either approach?
  - In our case: assume indirect evidence





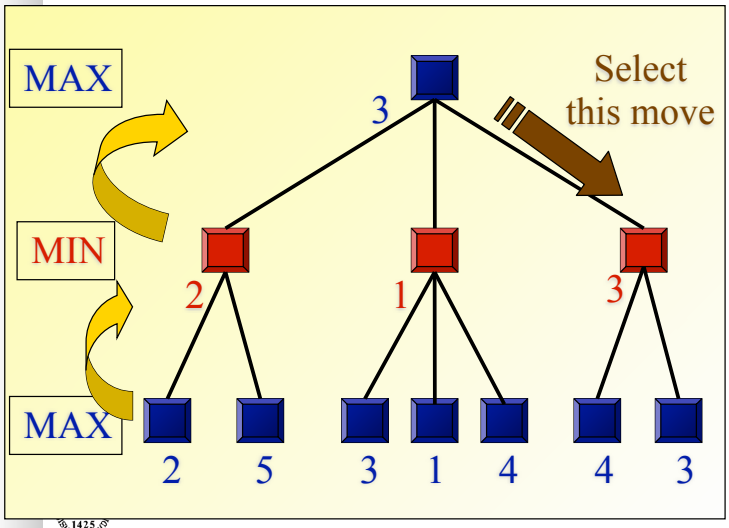
THOMAS MORE

- Want to learn: given board state, which move to make
  - Can be done in different ways…
  - Directly: *ChooseMove*: Board → Move
  - Indirectly: *V*: Board → **R**
    - Indicates how good some board state is
    - when playing, always consider all moves; choose move that leads to best state
    - "Perfect" definition for V: V(won) = 100, V(lost) = -100, V(draw) = 0, V(s) = V(e) with e best state reachable from s
      - Usually not feasible in practice! (exhaustive minimax)
      - Will need to approximate V
    - Think about connection to minimax algorithm for game playing
  - Let's choose the V function here

THOMAS MORE

Intermezzo: MINI MAX

- Restrictions:
  - 2 players: MAX (computer) and MIN (opponent)
  - ➔ deterministic, perfect information
- ⊙ Select a depth-bound (say: 2) and evaluation function



- Construct the tree up till the depth-bound
- Compute the evaluation function for the leaves
- Propagate the evaluation function upwards:
  - taking minima in MIN
  - taking maxima in MAX

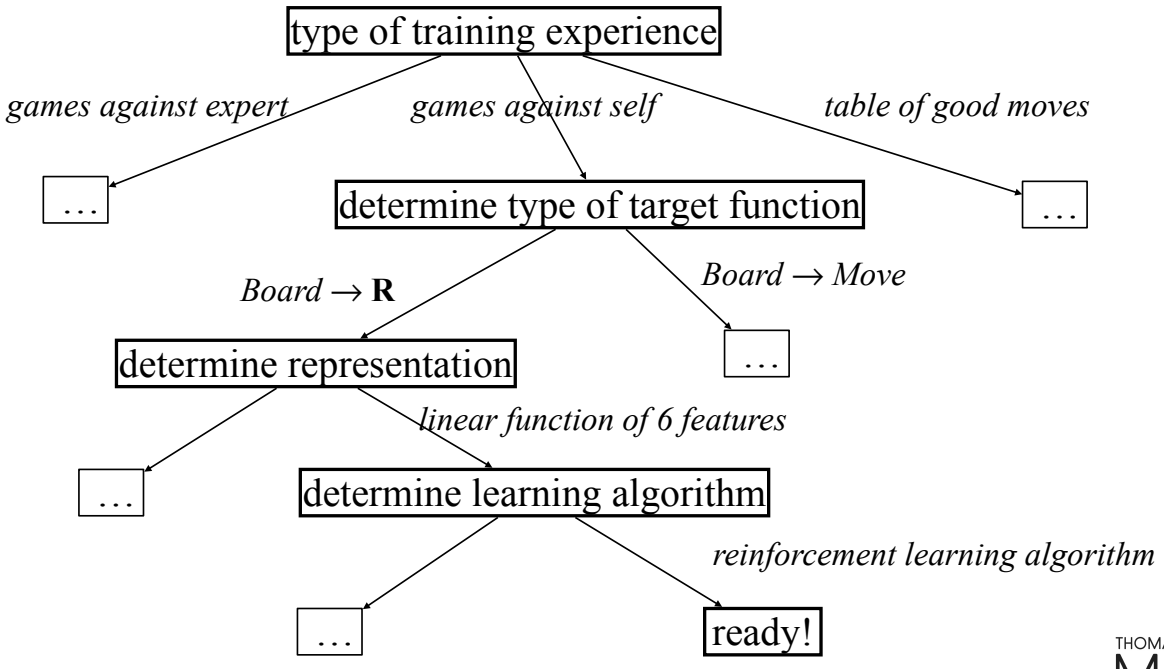
- Choose *representation* for target function V:
  - set of rules? (e.g. IF … THEN …)
  - neural network?
  - polynomial function of numerical board features?
  - …
- Let's choose: $V = w_1 bp + w_2 rp + w_3 bk + w_4 rk + w_5 bt + w_6 rt$
  - *bp, rp* : number of black / red pieces
  - *bk, rk* : number of black / red kings
  - *bt, rt* : number of black / read pieces threatened
  - w_i: *constants to be learnt from experience*
  - Note: this is where *generalisation* is going on

- How to obtain training examples?
  - we need a set of examples [*bp, rp, bk, rk, bt, rt, V*]
    - I.e., board state description + its quality
  - *bp* etc. easy to determine (just count); but how to guess *V* ?
    - We have *indirect evidence* only: no-one tells us how good a state is! (except for the end states: lost/won/drawn)
    - Getting training values for V is a bit tricky (see Reinforcement Learning)
    - Intuitively:
      - Need to learn V function and use it to generate V values at the same time
      - The trick: learn V for state s by generating V values for successor states of s
      - V gets propagated from later states to earlier states in the game
- How to learn the weights w_i?
  - In our case (linear function): least-squares linear regression



THOMAS MORE

Overview of design choices





THOMAS MORE

Issues that influence choices

- Which algorithms useful for what type of functions?
- How is learning influenced by
  - Number of training examples
  - complexity of hypothesis (function) representation
  - noise in the data
- Can we help the learner with prior knowledge?
- Could a system alter its representation itself?
- …
- For many of these, no definitive answers yet!



THOMAS MORE

Typical learning tasks

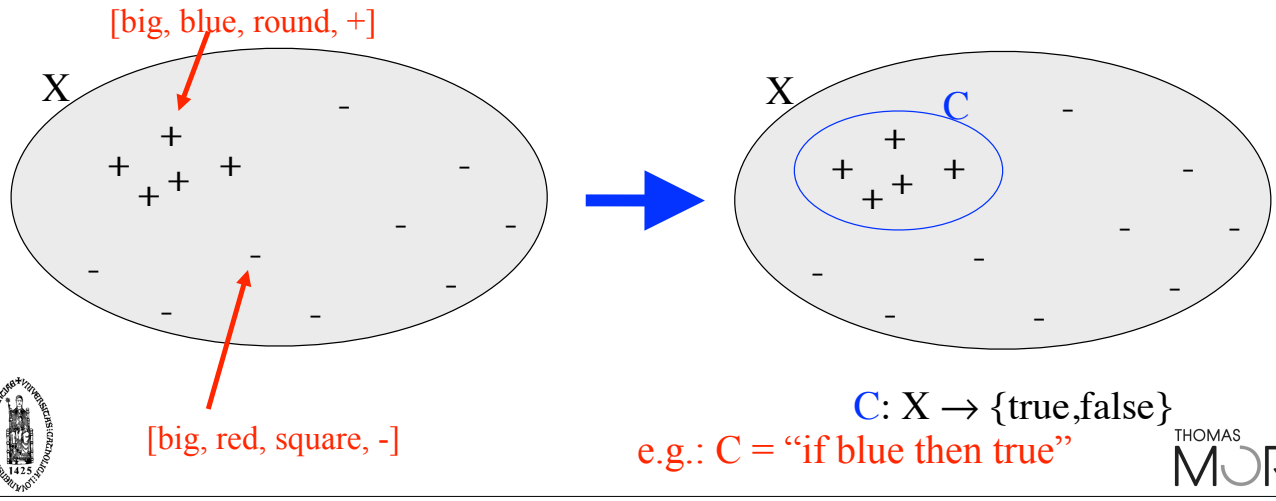
- Concept learning
  - learn a definition of a concept
  - supervised vs. unsupervised
- Function learning ("predictive modelling")
  - Discrete ("classification") or continuous ("regression")
  - Concept = function with boolean result
- Clustering
- Finding descriptive patterns



THOMAS MORE

Concept learning: supervised

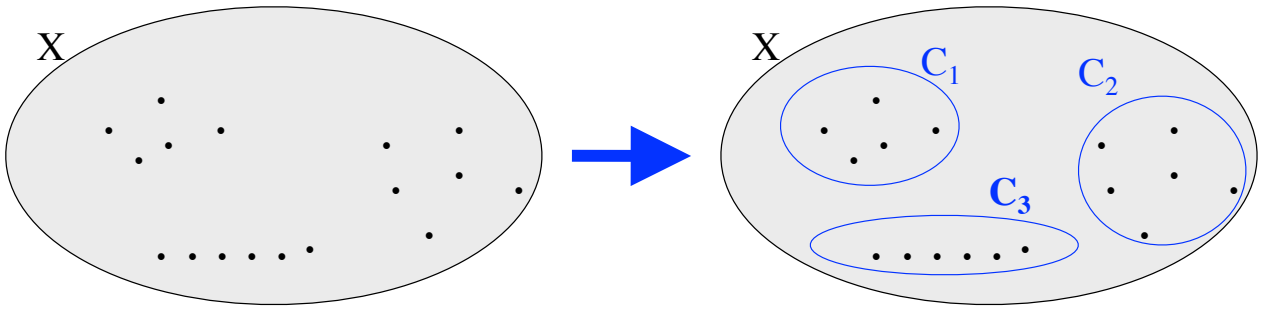
- Given positive (+) and negative (-) examples of a concept, infer properties that cause instances to be positive or negative (= concept *definition*)



THOMAS MORE

Concept learning: unsupervised

- Given examples of instances
  - *Invent* reasonable concepts (= *clustering*)
  - Find *definitions* for these concepts



- Cf. taxonomy of animals, identification of market segments, ...



THOMAS MORE

Note: Discriminating vs. Characterizing Descriptions

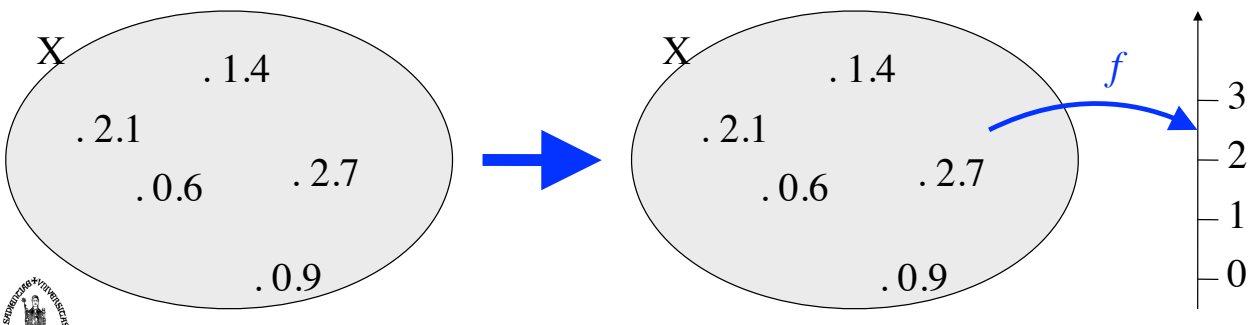
- Concept definitions can be:
  - *Discriminating*: Simple definition that *discriminates* concept from other instances
    - E.g.: if it barks, it is a dog; otherwise it isn't
  - *Characterizing*: Complex definition that *characterizes* concept as precisely as possible
    - E.g.: dogs have four legs, bark, bury bones, ...
- Some learners look for *simplest discriminating* description, others for *most complex characterizing* description



THOMAS MORE

Function learning

- Generalises over concept learning
- Learn function *f*: X→S where
  - S is finite set of values: classification
  - S is a continuous range of reals: regression

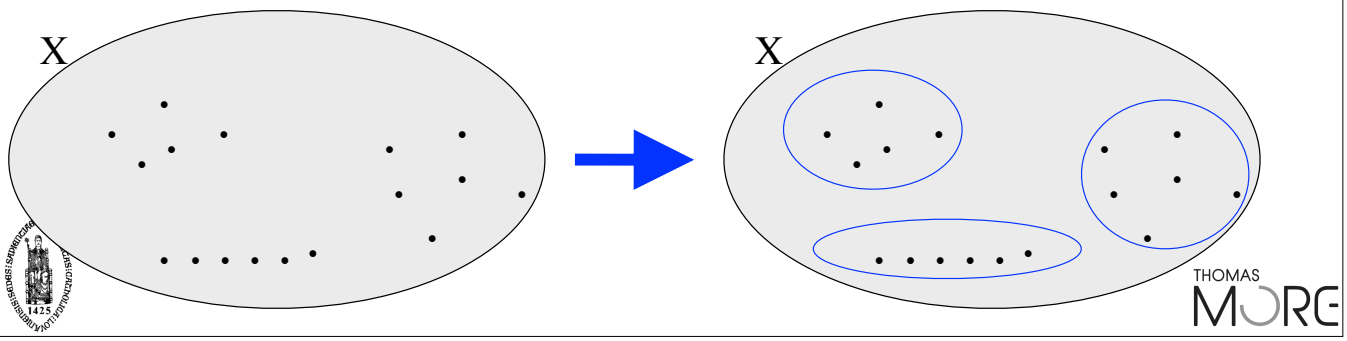




THOMAS MORE

Clustering

- Finding groups of instances that are similar
- May be a goal in itself (unsupervised classification)
- ... or used for other tasks
  - regression
  - *flexible prediction*: when it is not known in advance which properties to predict from which other properties



THOMAS MORE

Finding descriptive patterns

- Descriptive patterns = any kind of patterns, not necessarily directly useful for prediction
  - Generalises over predictive modelling (= finding predictive patterns)
- Examples of patterns:
  - "fast cars usually cost more than slower cars"
  - "people cannot be married to more than one person at the same time"

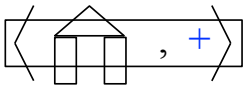
THOMAS MORE

Representation of data

- Numerical data: instances are points in $\mathbf{R}^n$

  < 1.3, 5.2, + >
  - Many techniques focus on this kind of data
- Symbolic data (true/false, black/white/red/blue, ...)
  - Can be converted to numeric data

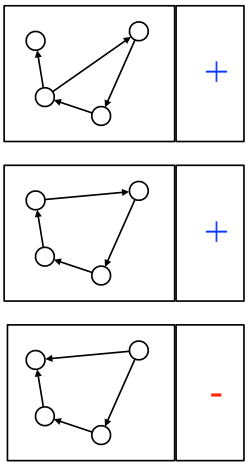
    < big, blue, square, +>
  - Some techniques work directly with symbolic data
- Structural or relational data
  - Instances have internal structure (sequences, graphs, sets, …; cf. molecules) or are related to other instances
  - Difficult/impossible to convert to simpler format
  - Not so many techniques handle these directly



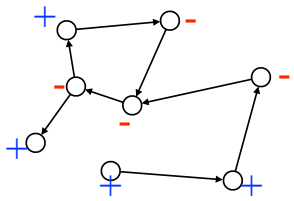


THOMAS MORE

Internal structure:
1 example = 1 graph



External structure: 1 example = 1 node
(graph connects different examples)
(1 prediction *per node,* instead of per graph)





THOMAS MORE

Categorization of learning tasks in terms of input data

- Learning from “attribute-value” data:
  - 1 example = 1 tuple (vector)
    - Tuple gives for each attribute a single value

< 1.3, 5.2, + >

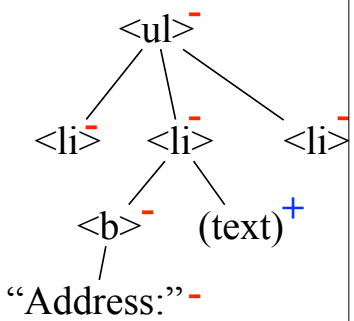
- Learning from sequences:
  - 1 prediction per sequence?
  - 1 prediction per node?
  - 1 node in sequence can be …
    - 1 number (e.g., time series)
    - 1 symbol (e.g., strings)
    - 1 tuple
    - a more complex structure

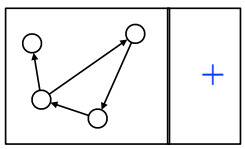
ababababb: +
aabbaabb: –



THOMAS MORE

- Trees
  - 1 prediction per tree / per node in the tree
  - Nodes can be …
    - Unlabeled
    - Labeled with symbols (e.g., HTML/XML structures)
    - …
- Graphs
  - 1 prediction per graph / per node in the graph
  - Nodes can be
    - Unlabeled
    - Labeled with numbers / symbols
    - …

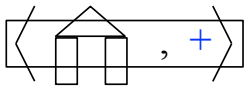






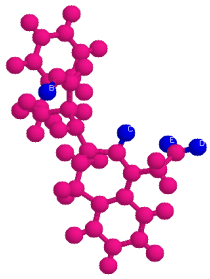
THOMAS MORE

- “Any” kind of knowledge
  - Anything expressible in some knowledge representation language



```
atm(m1,a1,o,2,3.43,-3.11,0.04).
atm(m1,a2,c,2,6.03,-1.77,0.67).
...
bond(m1,a2,a3,2).
bond(m1,a5,a6,1).
bond(m1,a6,a7,du).
...
```

```
...
hacc(M,A):- atm(M,A,o,2,_,_,_).
hacc(M,A):- atm(M,A,o,3,_,_,_).
hacc(M,A):- atm(M,A,s,2,_,_,_).
hacc(M,A):- atm(M,A,n,ar,_,_,_).
zincsite(M,A):-
    atm(M,A,du,_,_,_,_).
hdonor(M,A) :-
    atm(M,A,h,_,_,_,_),
    not(carbon_bond(M,A)), !.
...
```



- Learning task, and suitable learning algorithms, depend strongly on structure of input data!
- In this course: learning from “attribute-value” data

Brief overview of approaches

- Symbolic approaches:
  - Version Spaces, Induction of decision trees, Induction of rule sets, inductive logic programming, …
- Numeric approaches:
  - neural networks, support vector machines, …
- Probabilistic approaches (“bayesian learning”)
- Miscellaneous:
  - instance based learning, genetic algorithms, reinforcement learning



THOMAS MORE

Overview of the course

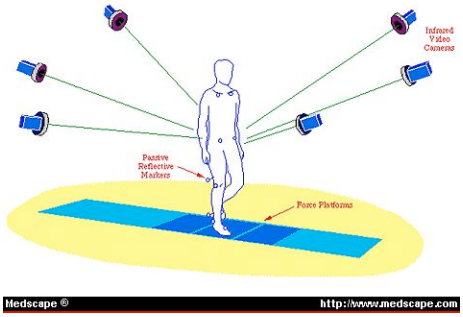
- Lecture 1: Introduction (Ch. 1-2), Versionspaces & bias (Ch. 3) (TC)
- 2: Induction of decision trees (Ch. 4) (TC)
- 3: Induction of rule sets & association rules (Ch. 5) (TC)
- 4: Association rules + exercises (TC)
- 5: Instance-based learning (Ch. 7) (PK)
- 6: Clustering (Ch. 8) (PK)
- 7: Dynamic Time Warping (PK)
- 8: Evaluating hypotheses (Ch. 6) (PK)
- 9: SVMs (Ch. 11) (PK)
- 10: ANNs (Ch. 10) (PK)
- 11: Ensembles (Ch. 13) + intro probabilistic approaches (Ch. 12) (TC)
- 12: Probabilistic approaches (Ch. 12)(TC)

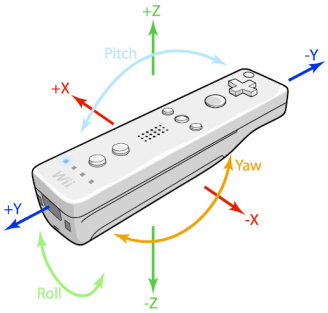


THOMAS MORE

ML Labo

- Central around project human gait analyses
- Focus on
  - Machine learning for time series
  - Feature extraction
  - Applying ML in practice





THOMAS MORE

3 Version Spaces

- Very basic idea
  - Will not be covered in detail
  - Introduce some important concepts for later use
- Difficulties with version space approaches
  - This is why we need other approaches
- Inductive bias

THOMAS MORE

A concept

- Let there be a set of objects, X.
  - X = {White Fang, Scooby Doo, Wile E, Lassie}
- A concept C is…
  - A subset of X
    - C = dogs = {Lassie, Scooby Doo}
  - A function that returns 1 only for elements in the concept
    - C(Lassie) = 1, C(Wile E) = 0



THOMAS MORE

Instance Representation

- Represent an object (or instance) as an n-tuple of attributes
- Example: Days (6-tuples)

Sky	Temp	Humid	Wind	Water	Forecast
sunny	warm	normal	strong	warm	same
sunny	warm	high	strong	warm	same
rainy	cold	high	strong	warm	change
sunny	warm	high	strong	cool	change



THOMAS MORE

Example Concept Function

- "Days on which my friend Aldo enjoys his favorite water sport"

INPUT

OUTPUT

Sky	Temp	Humid	Wind	Water	Forecast	C(x)
sunny	warm	normal	strong	warm	same	1
sunny	warm	high	strong	warm	same	1
rainy	cold	high	strong	warm	change	0
sunny	warm	high	strong	cool	change	1



THOMAS MORE

Hypothesis Spaces

- Hypothesis Space H: subset of all possible concepts
- For learning, we restrict ourselves to H
  - H may be only a small subset of all possible concepts
    (this turns out to be important – more later)



THOMAS MORE

Example Hypothesis Space

- Hypothesis h is a conjunction of constraints on attributes
- Each constraint can be:
  - A specific value : e.g. Water=Warm
  - A don't care value : e.g. Water=?
  - No value allowed: e.g. Water=Ø
- Instances x that satisfy the constraints of h have h(x) = 1, otherwise h(x) = 0
- Example hypotheses:

Sky	Temp	Humid	Wind	Water	Forecast
sunny	?	?	?	?	?
?	warm	?	?	?	same



THOMAS MORE

Basic principles Version Spaces (VS)

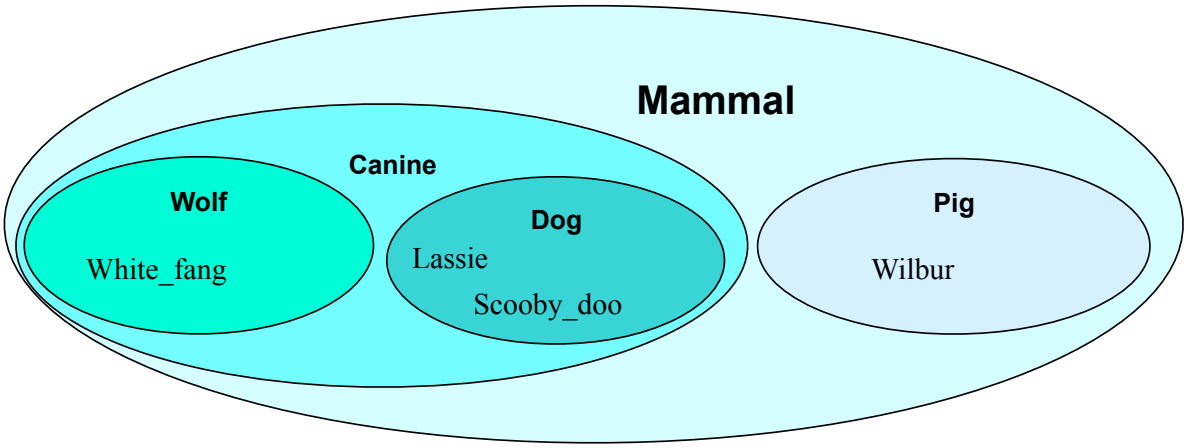
- Concept learning as search
  - **given**: hypothesis space H and data set S
    - H is a set of possible concept definitions
    - S consists of positive and negative examples of some concept
  - **find**: all h ∈ H consistent with S
    - Consistent = covers all positives and no negatives
  - this set is called the *version space*, *VS(H,S)*
- How to search in H
  - enumerate all h in H: not feasible (too many h)
  - prune search using some *generality ordering*
    - h_1 more general than $h_2 \Leftrightarrow (x \in h_2 \Rightarrow x \in h_1)$
    - Or: coverage $(h_2) \subseteq$ coverage(h_1)



THOMAS MORE

Concept Generality

- A concept P is **more general than or equal to** another concept Q iff the set of instances represented by P includes the set of instances represented by Q.





THOMAS MORE

General to Specific Order

- Consider two hypotheses:
  - h_1=< Sunny,?,?,Strong,?,?>
  - h_2=< Sunny,?,?,?,?,?>

- Definition: h_j is **more general than or equal to** h_k iff:

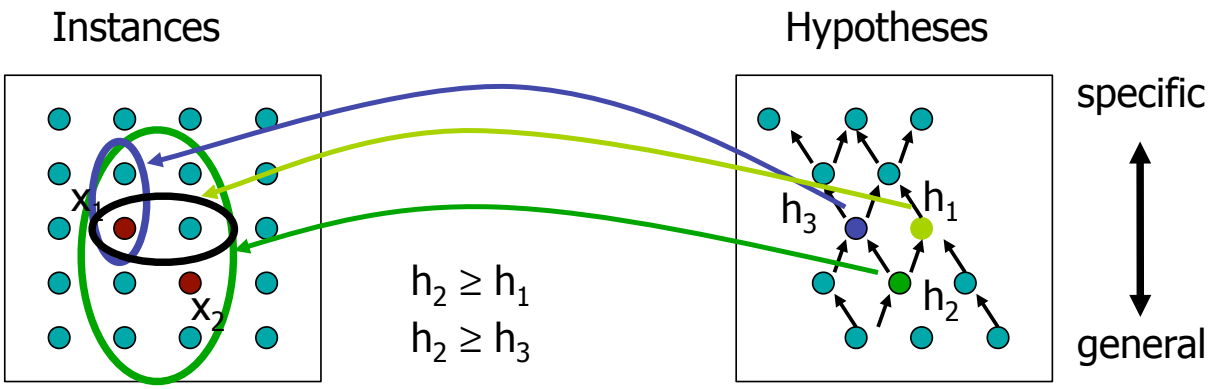
$$h_j \geq h_k \equiv \forall x \left(h_k(x) = 1 \rightarrow h_j(x) = 1 \right)$$

- This imposes a partial order on a hypothesis space



THOMAS MORE

Instance, Hypotheses and "generality"



x_1=< Sunny,Warm,High,Strong,Cool,Same>

x_2=< Sunny,Warm,High,Light,Warm,Same>

h_1=< Sunny,?,?,Strong,?,?>

h_2=< Sunny,?,?,?,?,?>

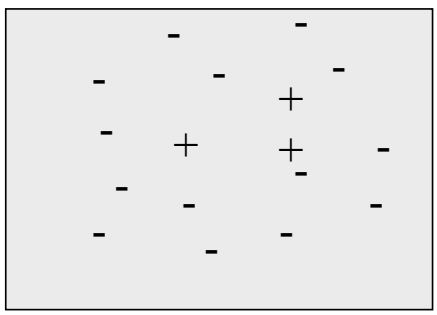
h_3=< Sunny,?,?,?,Cool,?>



THOMAS MORE

An example

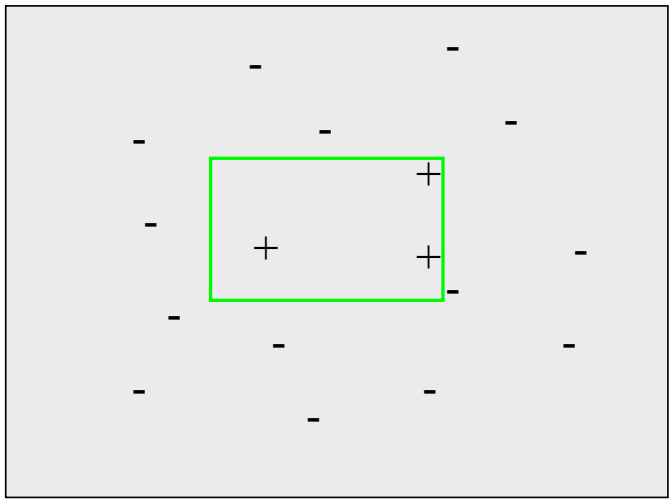
- \+ : belongs to concept; - : does not
  - S = set of these + and - examples
- Assume hypotheses are rectangles
  - I.e., H = set of all rectangles
    - Note: rectangle = *conjunctive concept*; can be represented as $<[x_l, x_u], [y_l, y_u]> = \{ (x,y) \mid x_l<x<x_u \text{ and } y_l<y<y_u\}$
  - VS(H,S) = set of all rectangles that contain all the + and none of the -





THOMAS MORE

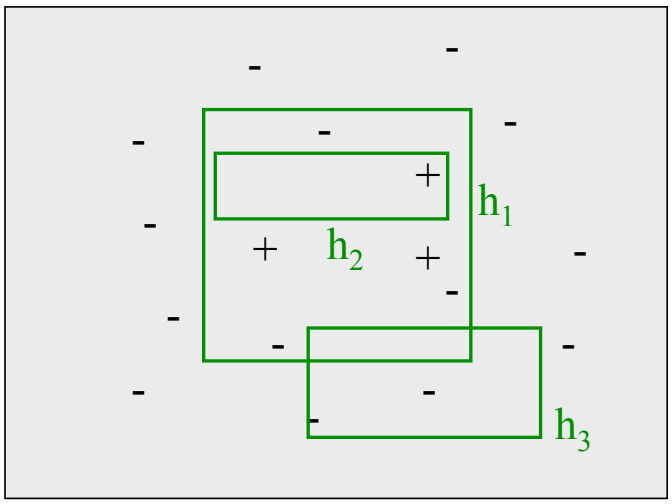
- Example of consistent hypothesis: green rectangle





THOMAS MORE

- h1 more general than h2 ⇔ h2 totally inside h1



h_1 more general than h_2
= h_2 more specific than h_1

h_3 incomparable with h_1



THOMAS MORE

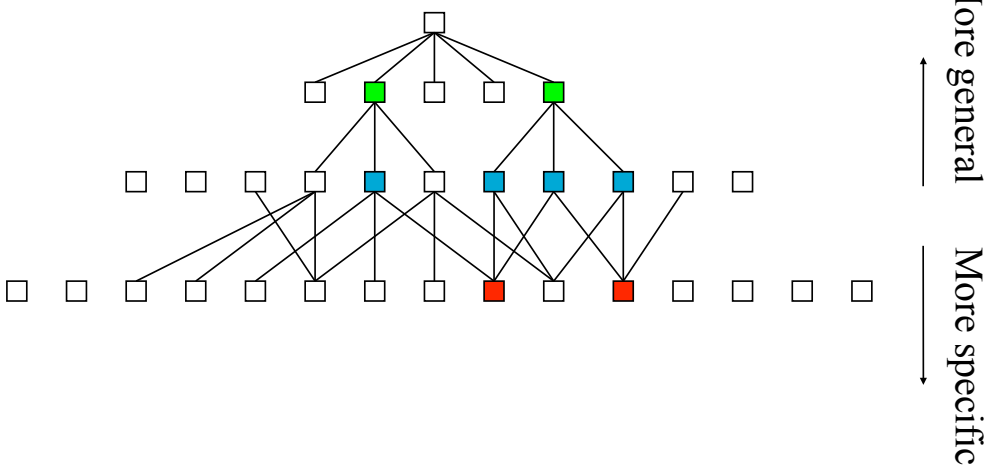
Version space boundaries

- Generality imposes lattice on versionspace
- Bound versionspace by giving its *most specific* (S) and *most general* (G) borders
  - S: rectangles that cannot become smaller without excluding a +
  - G: rectangles that cannot become larger without including a -
- Any hypothesis h consistent with the data
  - must be more general than some element in S
  - must be more specific than some element in G
- Thus, *G and S completely specify VS*



THOMAS MORE

Generality lattice

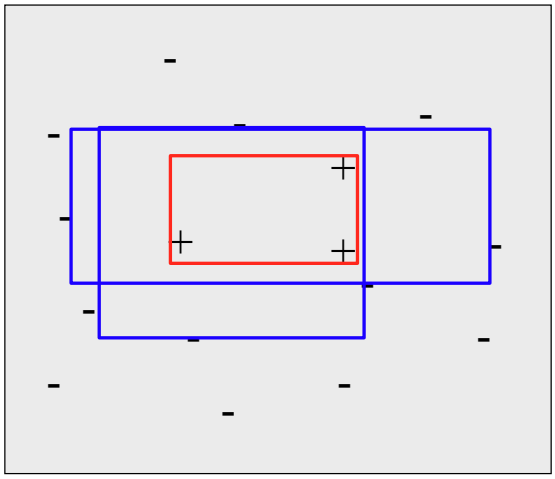




THOMAS MORE

Example, continued

- So what are S and G here?



h_1: most specific hypothesis

h_2: most general hypothesis

h_3: another most general hyp.

$S = \{h_1\}$, $G = \{h_2, h_3\}$



THOMAS MORE

Computing the version space

- Computing G and S is sufficient to know the full versionspace
- Well-known algorithms exist for computing S and G (not further discussed here, see course text more information)
  - FindS: computes just S set
  - Candidate Elimination: computes S and G



THOMAS MORE

Difficulties with version space approaches

- Idea of VS provides nice theoretical framework
- But not very useful for most practical problems
- *Difficulties* with VS approaches:
  - *Not very efficient*
    - Borders G and S may be very large (may grow exponentially)
  - *Not noise resistant*
    - VS "collapses" when no consistent hypothesis exists
    - often we would like to find the "best" hypothesis in this case
  - Basic approach handles *only conjunctive definitions*
- Many other learning methods are more practical in this respect



THOMAS MORE

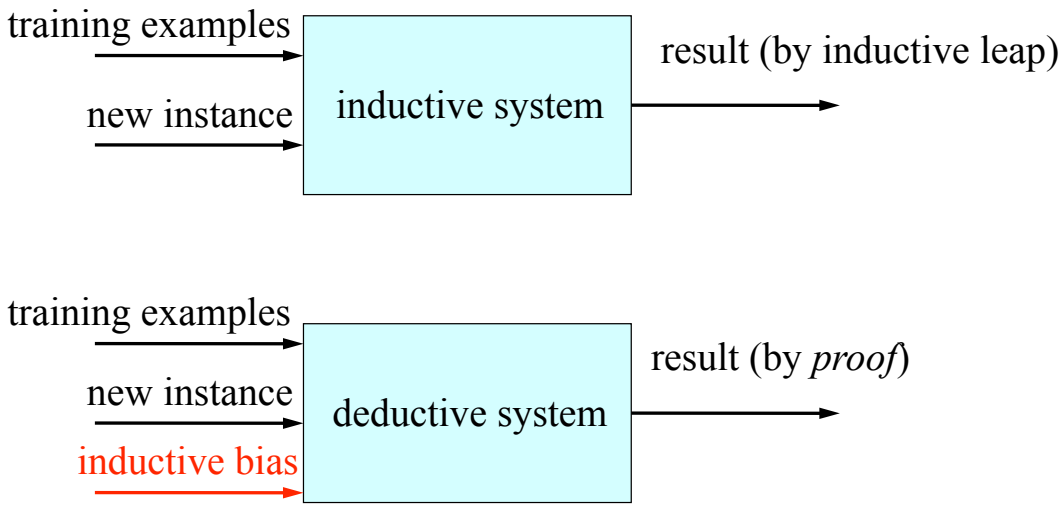
Inductive bias

- After having seen a limited number of examples, we believe we can make predictions for *unseen* cases.
- Why do we believe this ? Is there any guarantee this prediction will be correct ? *Under which conditions* can we guarantee that the predictions will be correct ?
- From seen cases to unseen cases = *inductive leap*
- *Inductive bias* : *minimal set of extra assumptions that guarantees correctness of inductive leap*



THOMAS MORE

Equivalence between inductive and deductive systems





THOMAS MORE

Definition of inductive bias

- More formal definition of inductive bias (Mitchell):
- L(x,D) denotes classification assigned to instance x by learner L after training on D
- |- denotes some proof procedure
- The *inductive bias of L* is any minimal set of assertions B such that for any target concept c and corresponding training examples D_c,

$$\forall x \in X: \; B \wedge D_c \wedge x \; |- \; L(x,D_c)$$

(i.e., minimal set of assumptions that have to be made so that you can prove that the learner will only learn models that are completely correct.)



THOMAS MORE

Effect of inductive bias

- Different learning algorithms give different results on same dataset because each may have a different bias
- Stronger bias means less learning
  - more is assumed in advance
- Is learning possible without any bias at all?
  - I.e., “pure” learning, without any assumptions in advance
  - The answer is *No*.



THOMAS MORE

Inductive bias of version spaces

- Bias of candidate elimination algorithm (which computes the version space): *target concept is in H*
- H typically consists of conjunctive concepts
- H could be extended towards disjunctive concepts
- Is it possible to use version spaces with H = set of all imaginable concepts, thereby *eliminating all bias*?



THOMAS MORE

Unbiased version spaces

- Let U be the example domain
- Unbiased: target concept C can be *any* subset of U
  - hypothesis space H = 2^U
  - any set of instances is a valid hypothesis
- Consider VS(H,D) with D a strict subset of U
- Assume you see an instance $x \in U \setminus D$
- For each $h \in VS$ that predicts $x \in C$, there must be a $h' \in VS$ that predicts $x \notin C$, and vice versa
  - **Proof**: take $h' = h \setminus \{x\}$: since $x \notin D$, h and h' are exactly the same w.r.t. D; so either both are in VS, or none of them are



THOMAS MORE

- Conclusion: version spaces without any bias do not allow generalisation
- *To be able to make an inductive leap, some bias is necessary*.
- We will see many different learning algorithms that all differ in their inductive bias.
- When choosing an algorithm to use on a practical problem, bias should be an important criterion
  - Unfortunately, for many algorithms the bias is not well understood…



THOMAS MORE

No Free Lunch

- Results on *impossibility of unbiased learning* are related to so-called *"no free lunch" theorems*
  - Essentially a different formulation of the same fact
- These state: if there exists a problem P for which learner L_1 performs better than L_2, there exists a problem P' for which L_2 is better than L_1
- *There is no universally best learning algorithm*



THOMAS MORE

To remember

- Definition of version space
- Importance of generality ordering for searching
- Definition & practical importance of inductive bias
  - how it relates inductive systems to deductive systems
  - why some bias is necessary for learning
  - "No free lunch" principle



THOMAS MORE

4 Induction of decision trees

- What are decision trees?
- How can they be induced automatically?
  - top-down induction of decision trees
  - avoiding overfitting
  - converting trees to rules
  - alternative heuristics
  - a generic TDIDT algorithm



THOMAS MORE

What are decision trees?

- Cf. guessing a person using only yes/no questions:
  - ask some question
  - depending on answer, ask a new question
  - continue until answer known



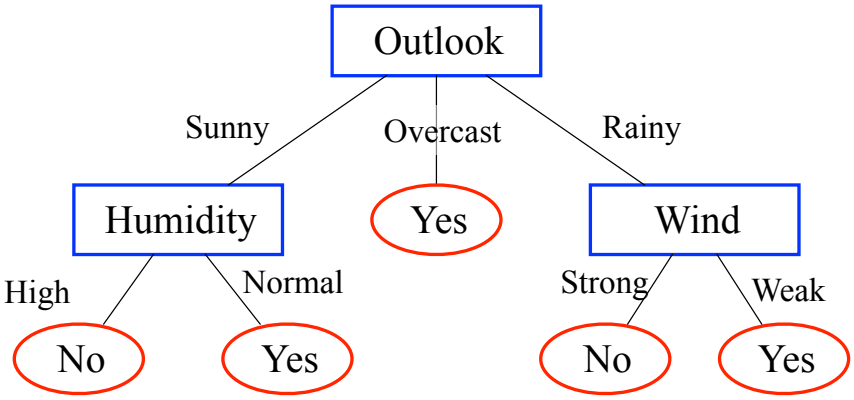
- A decision tree
  - Tells you which question to ask, depending on outcome of previous questions
  - Gives you the answer in the end
- Usually not used for guessing an individual, but for predicting some property (e.g., classification)



THOMAS MORE

Example decision tree 1

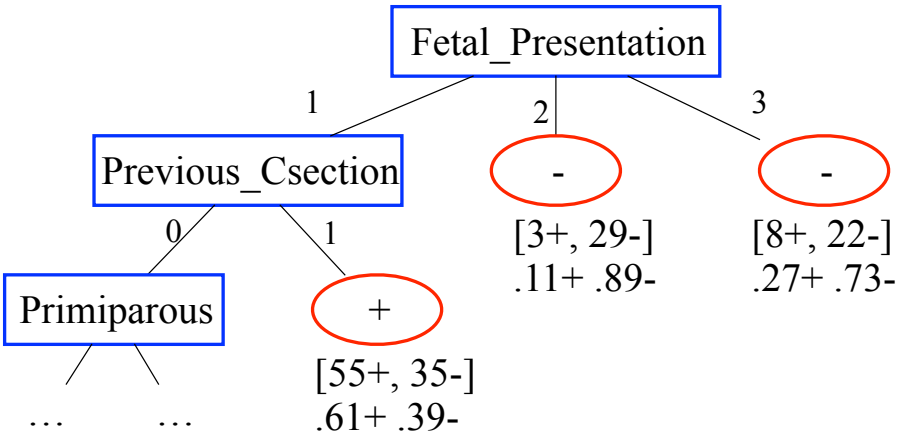
- Play tennis or not? (depending on weather conditions)



THOMAS MORE

Example decision tree 2

- Tree for predicting whether C-section necessary
- Leaves are not pure here; ratio pos/neg is given



THOMAS MORE

Representation power

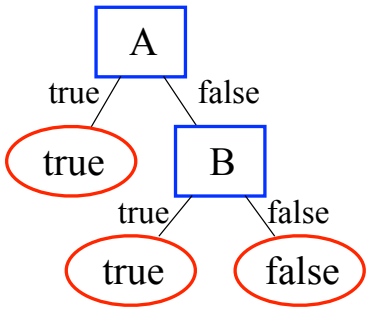
- Typically:
  - examples represented by array of attributes
    - "Attribute-value" setting
  - 1 node in tree tests value of 1 attribute
  - 1 child node for each possible outcome of test
  - Leaf nodes assign classification
- Note:
  - tree can represent any boolean function
    - i.e., also disjunctive concepts (<-> VS: conjunctive concepts)
  - tree can allow noise (non-pure leaves)



THOMAS MORE

Representing boolean formulae

- E.g., A ∨ B



- Similarly (try yourself):
  - A ∧ B, A xor B, (A ∧ B) ∨ (C ∧ ¬D ∧ E)
  - "M of N" (at least M out of N propositions are true)
  - What about complexity of tree vs. complexity of original formula?



THOMAS MORE

Classification, Regression and Clustering trees

- Classification trees represent function X -> C with C discrete (like the decision trees we just saw)
  - Hence, can be used for concept learning
- Regression trees predict numbers in leaves
  - could use a constant (e.g., mean), or linear regression model, or …
- Clustering trees just group examples in leaves
- Most (but not all) decision tree research in machine learning focuses on classification trees



THOMAS MORE

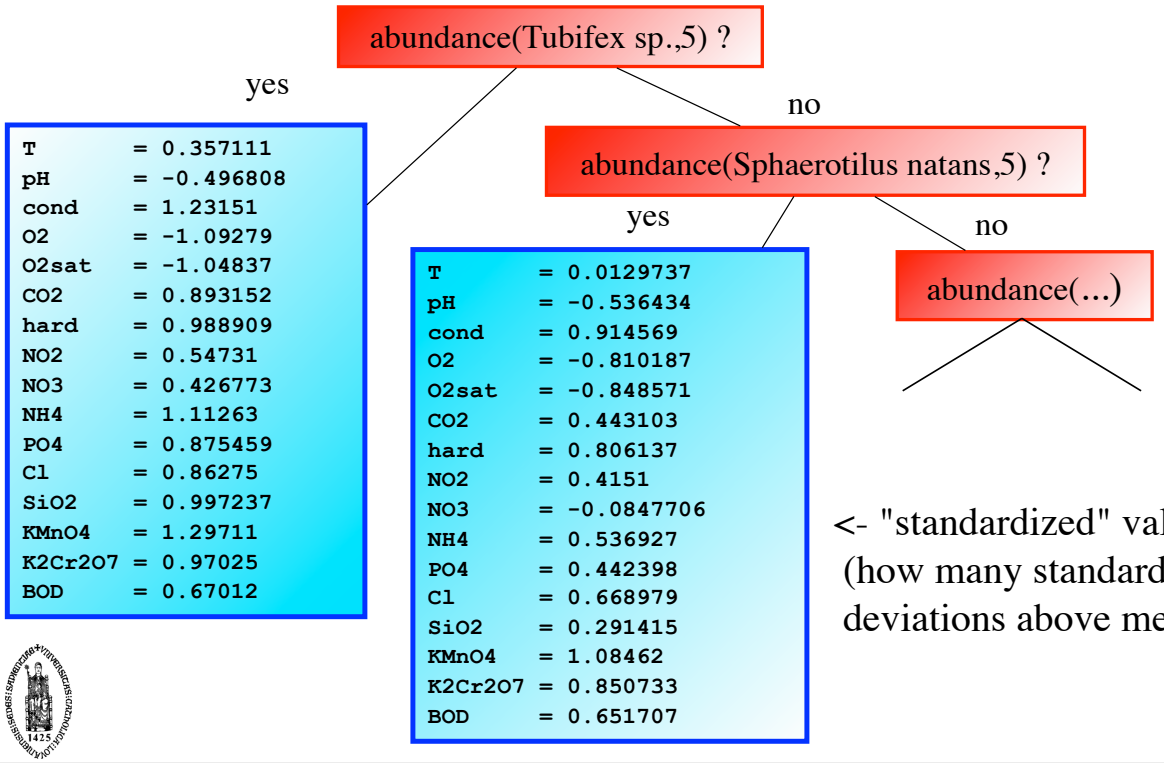
Example decision tree 3 (from study of river water quality)

- "Data mining" application
- Given: descriptions of river water samples
  - biological description: occurrence of organisms in water (“abundance”, graded 0-5)
  - chemical description: 16 variables (temperature, concentrations of chemicals (NH_4, ...))
- Question: characterize chemical properties of water using organisms that occur



THOMAS MORE

Clustering tree

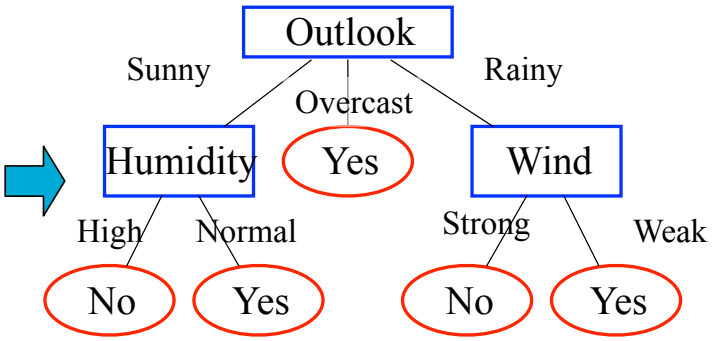


<- "standardized" values (how many standard deviations above mean)

Inducing Decision Trees...

- In general, we like decision trees that give us a result after as few questions as possible
- We can construct such a tree manually, or we can try to *obtain it automatically* (inductively) from a set of data
- The latter is what this chapter is about

Outlook	Temp.	Hum.	Wind	Play?
Sunny	85	85	False	no
Sunny	80	90	True	no
Overcast	83	86	False	yes
...	...	...	...	...



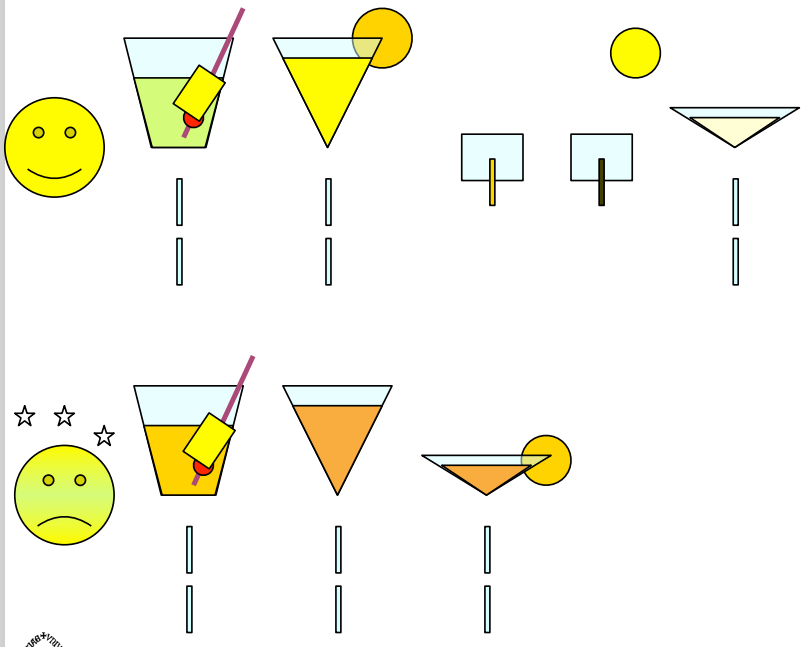
Top-Down Induction of Decision Trees

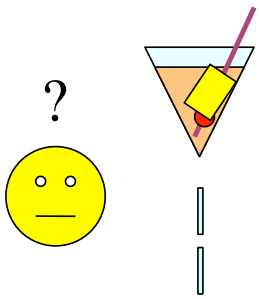
- Basic algorithm for TDIDT: *(based on ID3; later more formal version)*
  - start with full data set
  - find test that partitions examples as good as possible
    - "good" = examples with same class, or otherwise similar examples, should be put together
  - for each outcome of test, create child node
  - move examples to children according to outcome of test
  - repeat procedure for each child that is not "pure"
- Main questions:
  - how to decide which test is "best"
  - when to stop the procedure



THOMAS MORE

Example problem



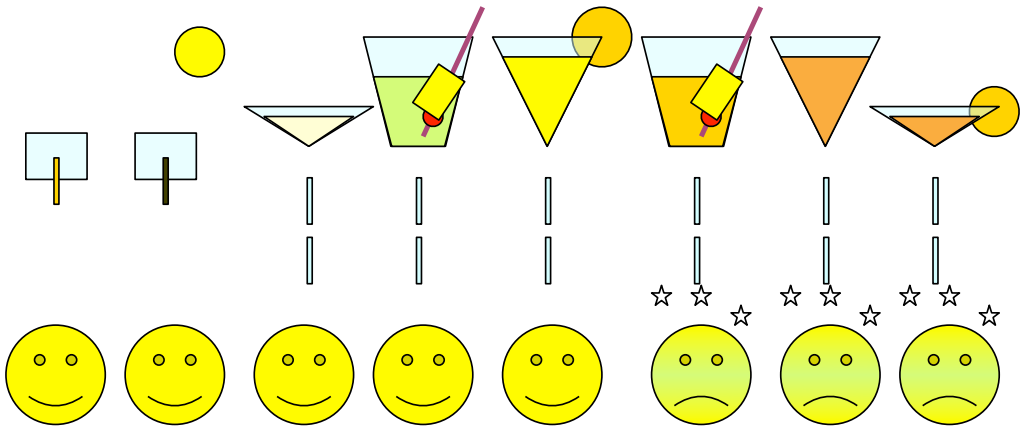


Is this drink going to make us ill, or not?

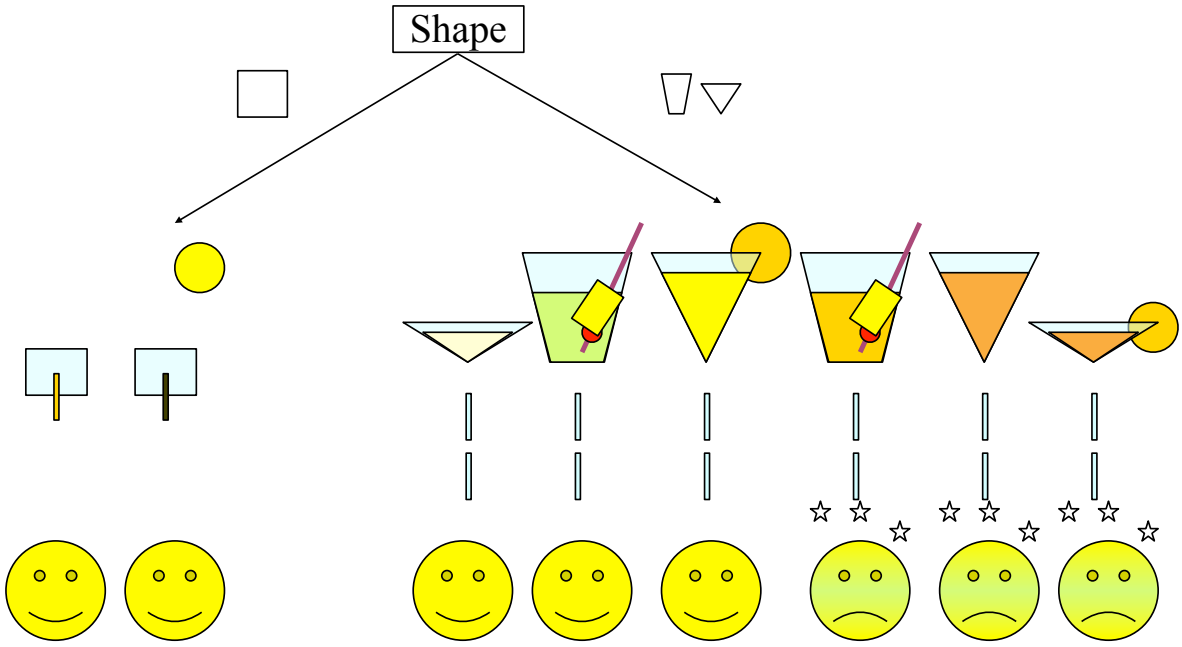


THOMAS MORE

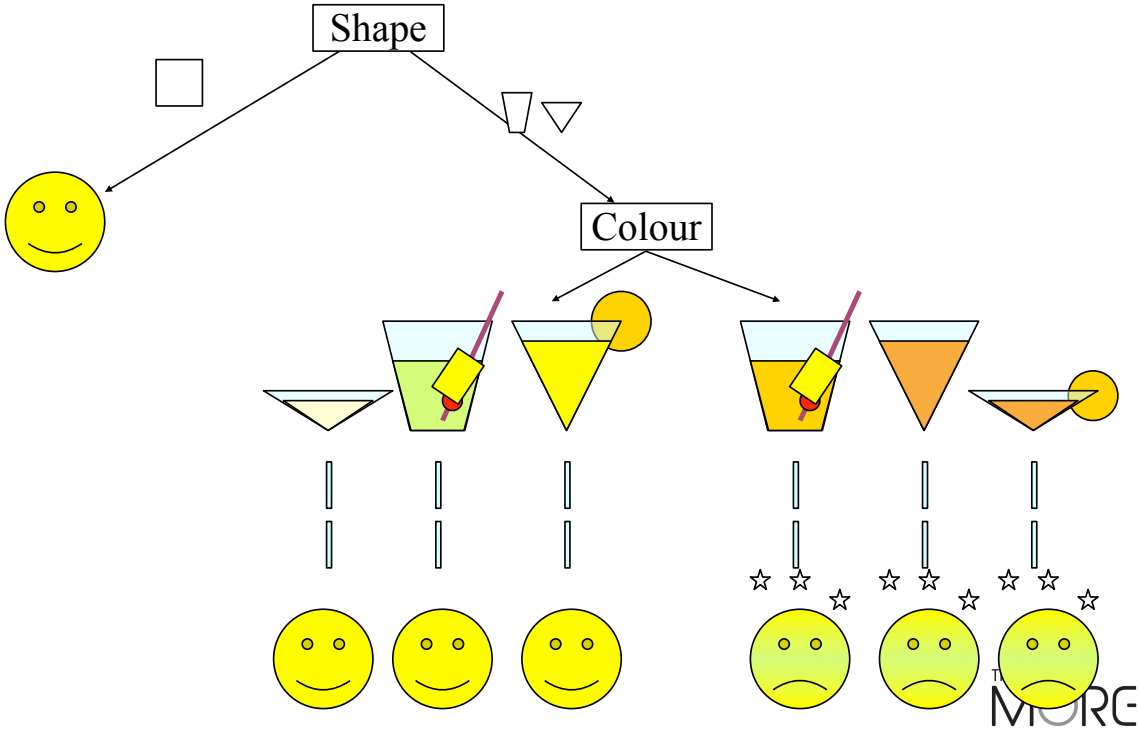
Data set: 8 classified instances



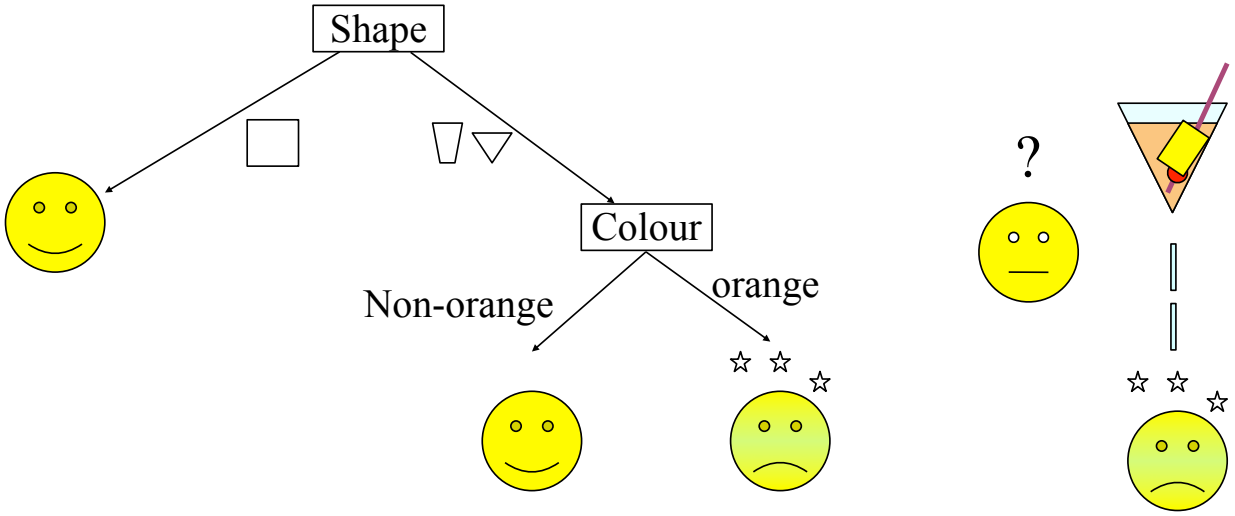
Observation 1: Shape is important



Observation 2: For some glasses, Colour is important



The decision tree



Finding the best test (for classification trees)

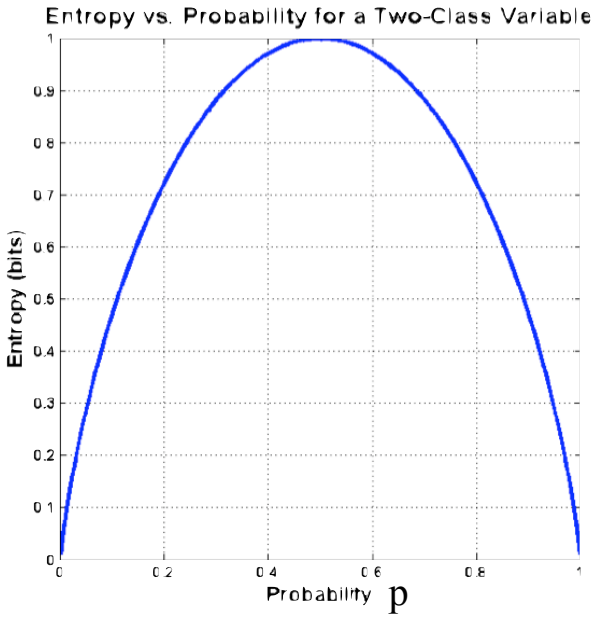
- For classification trees: find test for which children are as "pure" as possible
- Purity measure borrowed from information theory: *entropy*
  - is a measure of "*missing information*"; more precisely, number of bits needed to represent the missing information, on average, *using optimal encoding*
- Given set S with instances belonging to class i with probability p_i: $\text{Entropy}(S) = -\sum p_i \log_2 p_i$



THOMAS MORE

Entropy

- Entropy in function of p, for 2 classes:





THOMAS MORE

Information gain

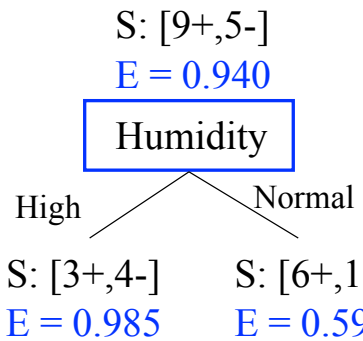
- Heuristic for choosing a test in a node:
  - choose that test that on average provides most information about the class
  - this is the test that, on average, reduces class entropy most
    - on average: class entropy reduction differs according to outcome of test
  - expected reduction of entropy = *information gain*
- $Gain(S,A) = Entropy(S) - \Sigma\ |S_v| / |S|\ Entropy(S_v)$



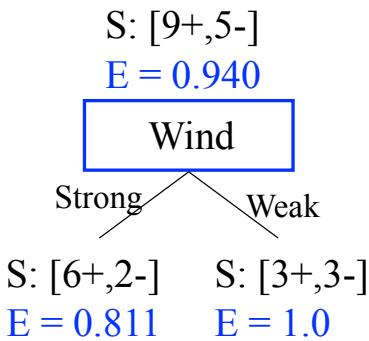
THOMAS MORE

Example

- Assume S has 9 + and 5 - examples; partition according to **Wind** or **Humidity** attribute



Gain(S, Humidity)
= .940 - (7/14).985 - (7/14).592
= 0.151

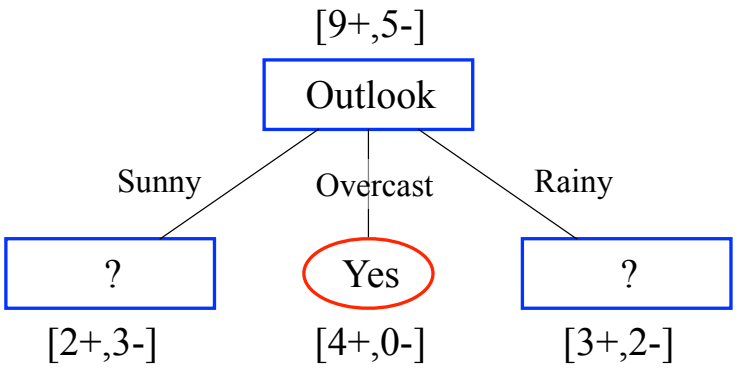


Gain(S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= 0.048



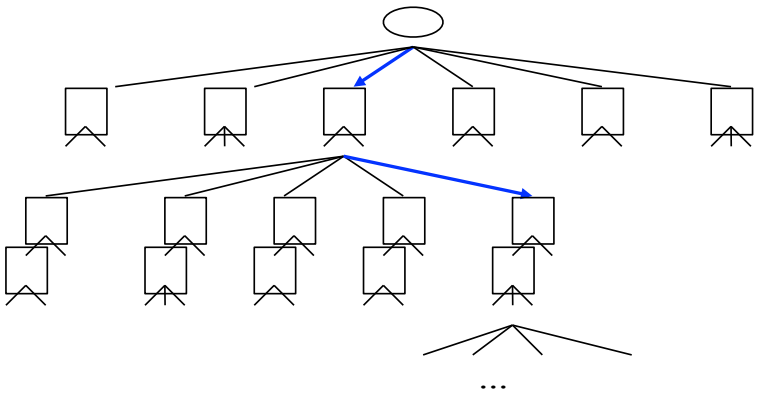
THOMAS MORE

- Assume **Outlook** was chosen: continue partitioning in child nodes



Hypothesis space search in TDIDT

- Hypothesis space H = set of all trees
- H is searched in a hill-climbing fashion, from simple to complex



Inductive bias in TDIDT

- Note: for e.g. boolean attributes, H is complete: each concept can be represented!
  - given n attributes, we can keep on adding tests until all attributes tested
- So what about inductive bias?
  - Clearly no "restriction bias" ($H \subset 2^U$) as for version spaces (conjunctive concepts)
  - *Preference bias*: some hypotheses in H are preferred over others
  - In this case: preference for *short trees* with *informative attributes at the top*



THOMAS MORE

Occam's Razor

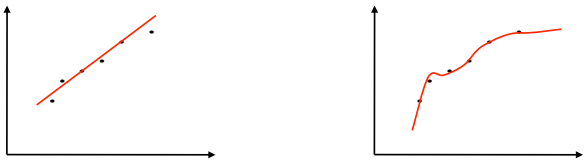
- Preference for simple models over complex models is quite generally used in machine learning
- Similar principle in science: *Occam's Razor*
  - roughly: do not make things more complicated than necessary
- Reasoning, in the case of decision trees: more complex trees have higher probability of *overfitting* the data set
  - Why? Somewhat controversial, see later



THOMAS MORE

Avoiding Overfitting

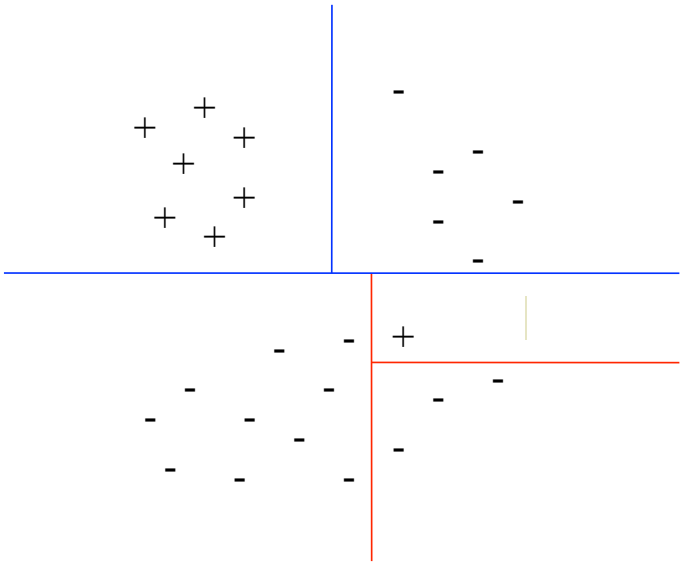
- Phenomenon of overfitting:
  - keep improving a model, making it better and better on training set by making it more complicated …
  - increases risk of modelling *noise* and *coincidences* in the data set
  - may actually harm predictive power of theory on unseen cases
- Cf. fitting a curve with too many parameters





THOMAS MORE

Overfitting: example

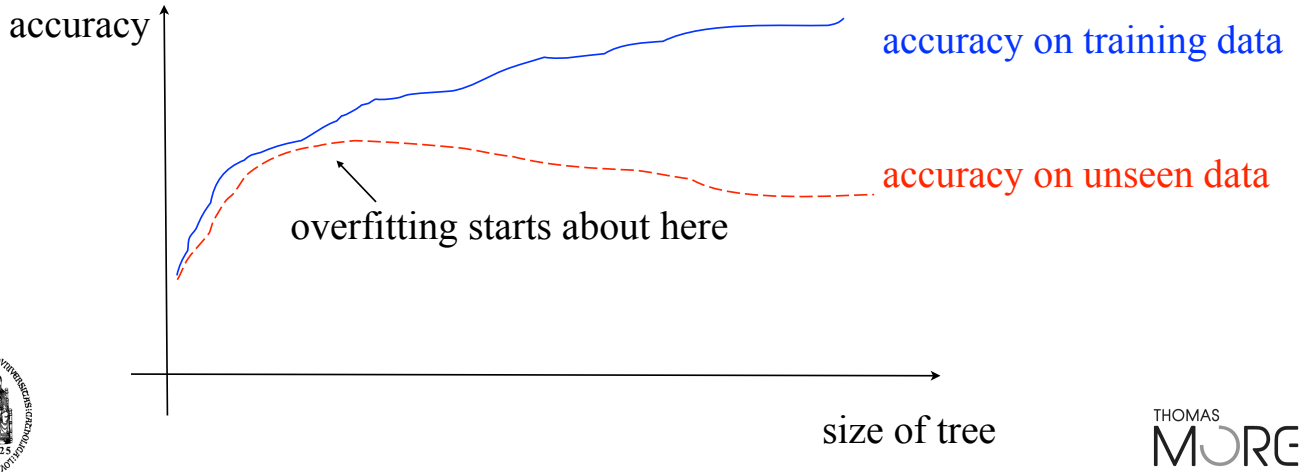




THOMAS MORE

Overfitting: effect on predictive accuracy

- Typical phenomenon when overfitting:
  - training accuracy keeps increasing
  - accuracy on unseen validation set starts decreasing





THOMAS MORE

How to avoid overfitting when building classification trees?

- Option 1:
  - stop adding nodes to tree when overfitting starts occurring
  - need *stopping criterion*
- Option 2:
  - don't bother about overfitting when growing the tree
  - after the tree has been built, start *pruning* it again



THOMAS MORE

Stopping criteria

- How do we know when overfitting starts?
  - a) use a *validation set*: data not considered for choosing the best test
    - when accuracy goes down on validation set: stop adding nodes to this branch
  - b) use some statistical test
    - significance test: is the change in class distribution significant? (χ^2-test) [in other words: does the test yield a *clearly better* situation?]
    - *MDL*: minimal description length principle
      - entirely correct theory = tree + corrections for specific misclassifications
      - minimize size(theory) = size(tree) + size(misclassifications(tree))
      - Cf. Occam's razor



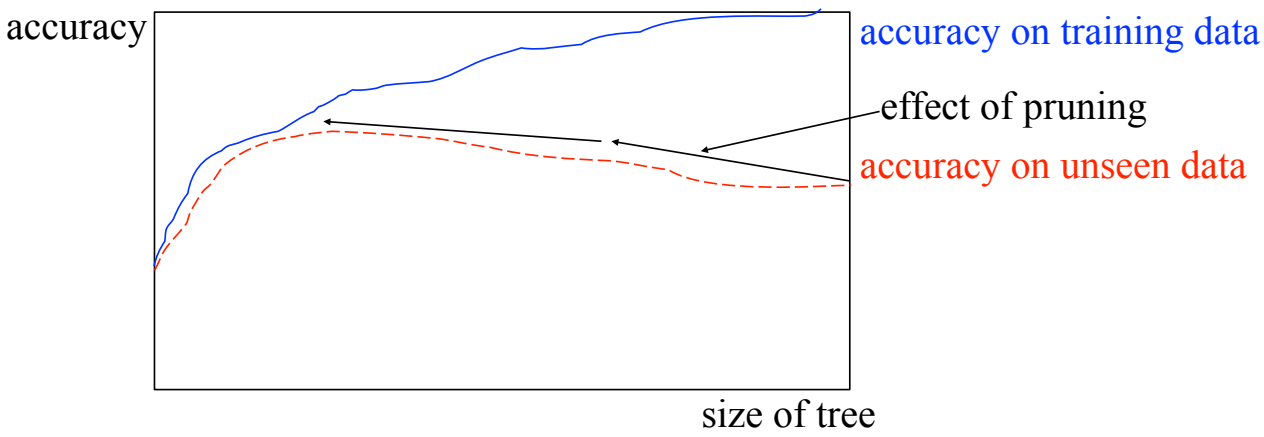
THOMAS MORE

Post-pruning trees

- After learning the tree: start pruning branches away
  - For all nodes in tree:
    - Estimate effect of pruning tree at this node on predictive accuracy
      - e.g. using accuracy on validation set
  - Prune node that gives greatest improvement
  - Continue until no improvements
- Note : this pruning constitutes a second search in the hypothesis space



THOMAS MORE





THOMAS MORE

Comparison

- Advantage of Option 1: no superfluous work
- But: tests may be misleading
  - E.g., validation accuracy may go down briefly, then go up again
- Therefore, Option 2 (post-pruning) is usually preferred (though more work)



THOMAS MORE

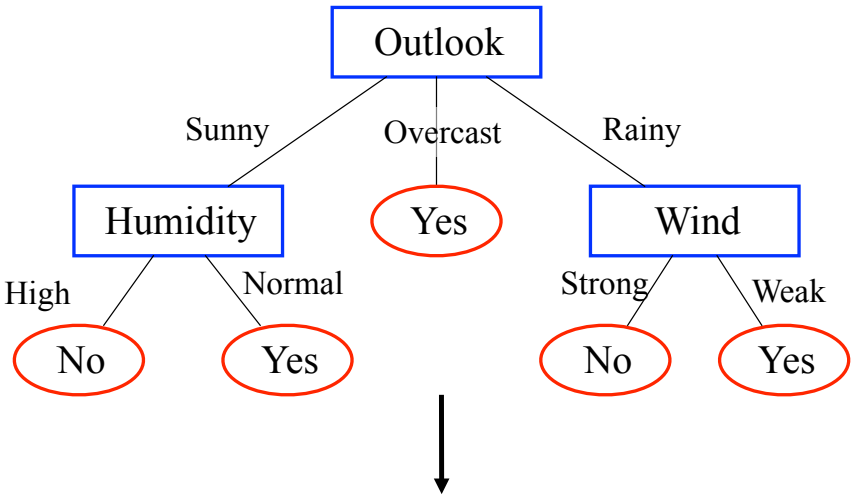
Turning trees into rules

- From a tree a rule set can be derived
  - Path from root to leaf in a tree = 1 if-then rule
- Advantage of such rule sets
  - may increase comprehensibility
    - Disjunctive concept definition
  - can be pruned more flexibly
    - in 1 rule, 1 single condition can be removed
      - vs. tree: when removing a node, the whole subtree is removed
    - 1 rule can be removed entirely



THOMAS MORE

Rules from trees: example



if Outlook = Sunny **and** Humidity = High **then** No
if Outlook = Sunny **and** Humidity = Normal **then** Yes
…



THOMAS MORE

Handling missing values

- What if result of test is unknown for example?
  - e.g. because value of attribute unknown
- Some possible solutions, when training:
  - guess value: just take most common value (among all examples, among examples in this node / class, …)
  - assign example partially to different branches
    - e.g. counts for 0.7 in yes subtree, 0.3 in no subtree
- When using tree for prediction:
  - assign example partially to different branches
  - combine predictions of different branches



THOMAS MORE

Alternative heuristics for choosing tests

- Attributes with *continuous domains* (numbers)
  - cannot different branch for each possible outcome
  - allow, e.g., binary test of the form **Temperature < 20**
  - same evaluation as before, but need to generate value (e.g. 20)
    - For instance, just try all reasonable values
- Attributes with *many discrete values*
  - unfair advantage over attributes with few values
    - cf. question with many possible answers is more informative than yes/no question
  - To compensate: divide gain by "max. potential gain" SI
  - *Gain Ratio*: $GR(S,A) = Gain(S,A) / SI(S,A)$
    - Split-information $SI(S,A) = - \sum |S_i|/|S| \log_2 |S_i|/|S|$
    - with i ranging over different results of test A



THOMAS MORE

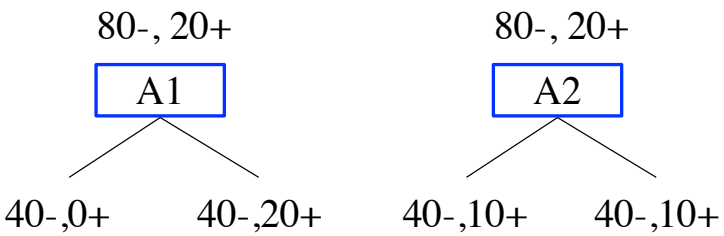
- Tests may have *different costs*
  - e.g. medical diagnosis: blood test, visual examination, … have different costs
  - try to find tree with low expected *cost*, instead of low expected *number of tests*
  - alternative heuristics, taking cost into account,have been proposed



THOMAS MORE

Properties of good heuristics

- Many alternatives exist
  - ID3 uses information gain or gain ratio
  - CART uses "Gini criterion" (not discussed here)
- Q: Why not simply use *accuracy* (% of correct predictions of tree, if leaves predict their majority class) as a criterion?

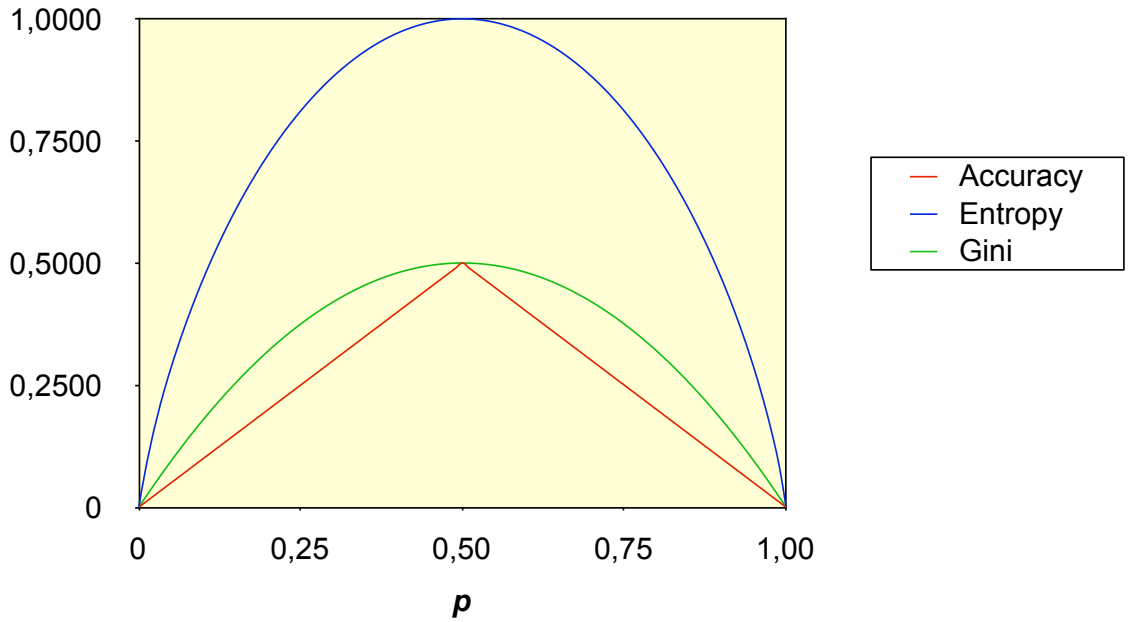


How would
- accuracy
- information gain
rate these splits?



THOMAS MORE

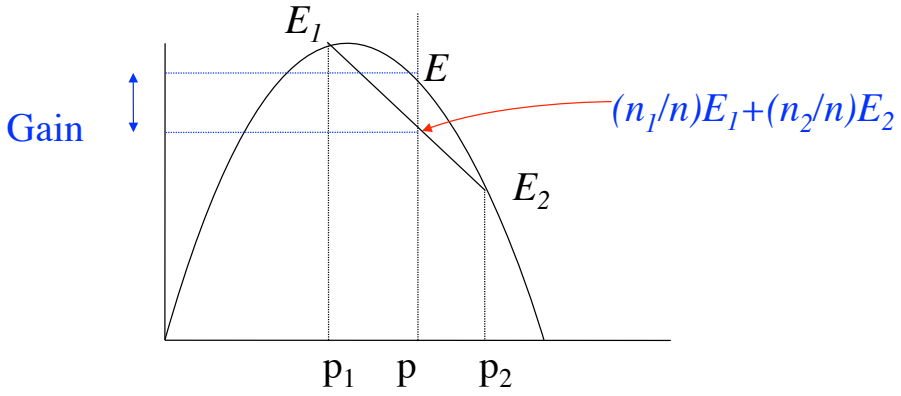
Heuristics compared



Curves show "impurity" as a function of p, when using Accuracy, Entropy, Gini

Good impurity measures are *strictly concave*

Why concave functions?



Assume node with size n, entropy E and proportion of positives p is split into 2 nodes with n_1, E_1, p_1 and n_2, E_2 p_2.

We have $p = (n_1/n)p_1 + (n_2/n)\, p_2$

and the new average entropy $E' = (n_1/n)E_1+(n_2/n)E_2$ is therefore found by linear interpolation between (p_1,E_1) and (p_2,E_2) at p.

Gain = difference in height between (p, E) and (p,E').

Generic TDIDT algorithm

- Many different algorithms for top-down induction of decision trees exist
- What do they have in common, and where do they differ?
- We look at a generic algorithm
  - General framework for TDIDT algorithms
  - Several "parameter procedures"
    - instantiating them yields a specific algorithm
- Summarizes previously discussed points and puts them into perspective



THOMAS MORE

Generic TDIDT algorithm

```
function TDIDT(E: set of examples) returns tree;
        T' := grow_tree(E);
        T := prune(T');
        return T;

function grow_tree(E: set of examples) returns tree;
        T := generate_tests(E);
        t := best_test(T, E);
        P := partition induced on E by t;
        if stop_criterion(E, P)
        then return leaf(info(E))
        else
                for all E_j in P: t_j := grow_tree(E_j);
                return node(t, {(j,t_j)});
```



THOMAS MORE

For classification...

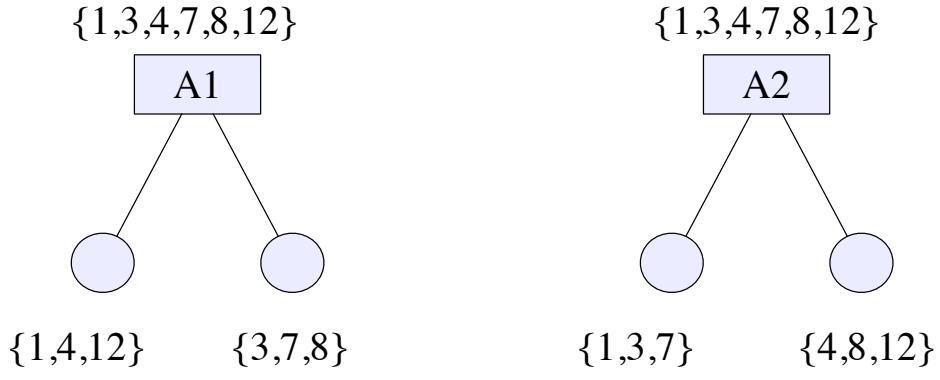
- prune: e.g. reduced-error pruning, ...
- generate_tests : Attr=val, Attr<val, ...
  - for numeric attributes: generate val
- best_test : Gain, Gainratio, ...
- stop_criterion : MDL, significance test (e.g. χ^2-test), ...
- info : most frequent class ("mode")
- Popular systems: C4.5 (Quinlan 1993), C5.0 (www.rulequest.com)



THOMAS MORE

For regression...

- change
  - best_test: e.g. minimize average variance
  - info: mean
  - stop_criterion: significance test (e.g., F-test), ...





THOMAS MORE

CART

- Classification and regression trees (Breiman et al., 1984)
- Classification: info: mode, best_test: Gini
- Regression: info: mean, best_test: variance
- prune: "error complexity" pruning
  - penalty α for each node
  - the higher α, the smaller the tree will be
  - optimal α obtained empirically (cross-validation, see later)



THOMAS MORE

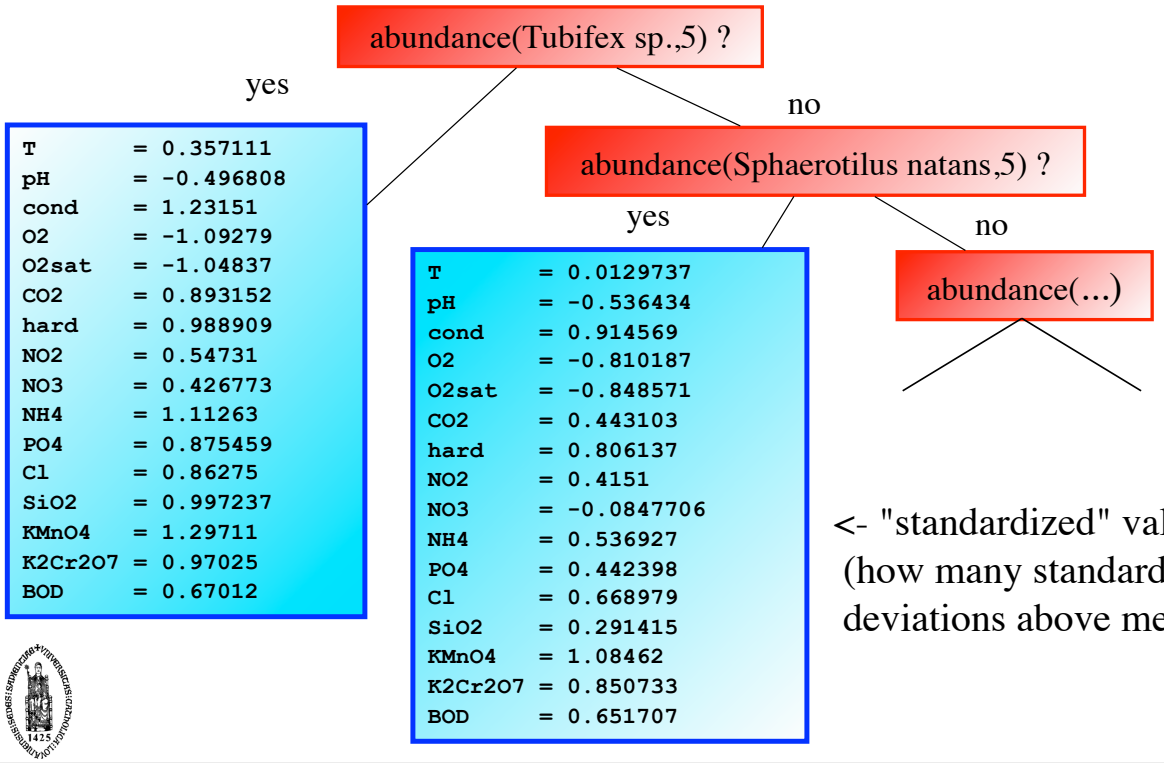
n-dimensional target spaces

- Instead of predicting 1 number, predict vector of numbers
- info: mean vector
- best_test: variance (mean squared distance) in n-dimensional space
- stop_criterion: F-test
- mixed vectors (numbers and symbols)?
  - use appropriate distance measure
- -> "clustering trees"



THOMAS MORE

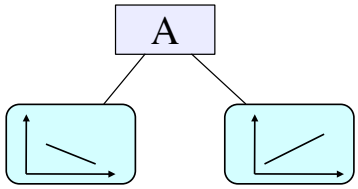
Clustering tree



<- "standardized" values (how many standard deviations above mean)

Model trees

- Make predictions using linear regression models in the leaves
- info: regression model ($y=ax_1+bx_2+c$)
- best_test: ?
  - variance: simple, not so good (M5 approach)
  - residual variance after model construction: better, computationally expensive (RETIS approach)
- stop_criterion: significant reduction of variance



Try it out…

- Weka toolbox
  - By the University of Waikato
  - Contains many machine learning algorithms
    - Including several TDIDT algorithms: ID3, J48, M5', ...
  - Contains toy data sets (incl. play-tennis example)
  - Easy-to-use interface
  - See http://www.cs.waikato.ac.nz/ml/weka



THOMAS MORE

To Remember

- Decision trees & their representational power
- Generic TDIDT algorithm and how to instantiate its parameters
- Search through hypothesis space, bias, tree to rule conversion
- For classification trees: details on heuristics, handling missing values, pruning, …
- Some general concepts: overfitting, Occam's razor



THOMAS MORE

5 Induction of Rule Sets

- Representing theories with decision rules
- Induction of predictive rules
  - Sequential covering approaches
  - Genetic algorithms
- Induction of association rules
  - The Apriori approach



THOMAS MORE

Representing Theories with Decision Rules

- Previous representations:
  - Conjunctive concepts
  - Decision trees
- Another popular representation for concept definitions: if-then-rules
  - IF <conditions> THEN belongs to concept
- How can we learn such rules ?
  - By learning trees and converting them to rules
  - Using genetic algorithms
  - With specific rule-learning methods (“sequential covering”)



THOMAS MORE

Sequential Covering Approaches

- Or: “separate-and-conquer” approach
  - Versus trees: “divide-and-conquer”
- General principle: learn a rule set one rule at a time
  - Learn one rule that has
    - *High accuracy*
      - When it predicts something, it should be correct
    - *Any coverage*
      - Does not have to make a prediction for all examples, just for some of them
  - Mark covered examples
    - These have been taken care of; now focus on the rest
  - Repeat this until all examples covered



THOMAS MORE

Sequential Covering

- General algorithm for learning rule sets
  - Based on CN2 algorithm (Clark & Niblett)

```
function LearnRuleSet(Target, Attrs, Examples, Threshold):
    LearnedRules := ø
    Rule := LearnOneRule(Target, Attrs, Examples)
    while performance(Rule,Examples) > Threshold, do
        LearnedRules := LearnedRules U {Rule}
        Examples := Examples \ {examples classified correctly by Rule}
        Rule := LearnOneRule(Target, Attrs, Examples)
    sort LearnedRules according to performance
    return LearnedRules
```



THOMAS MORE

Learning One Rule

- To learn one rule:
  - Perform greedy search
  - Could be top-down or bottom-up (in generality lattice)
    - Top-down:
      - Start with maximally general rule (has maximal coverage but low accuracy)
      - Add literals one by one
      - Gradually maximize accuracy without sacrificing coverage (using some heuristic)
    - Bottom-up:
      - Start with maximally specific rule (has minimal coverage but maximal accuracy)
      - Remove literals one by one
      - Gradually maximize coverage without sacrificing accuracy (using some heuristic)



THOMAS MORE

Learning One Rule

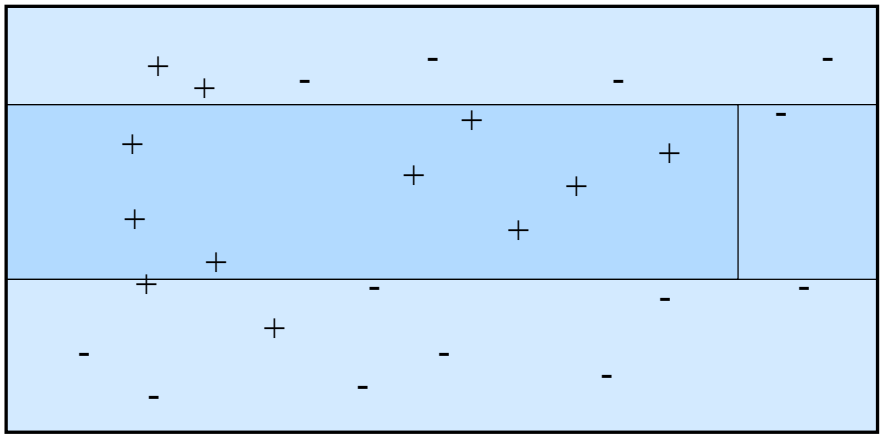
function LearnOneRule(Target, Attrs, Examples):
  NewRule := "IF true THEN pos"
  NewRuleNeg := Neg
  while NewRuleNeg not empty, **do**
    // add a new literal to the rule
      Candidates := generate candidate literals
      BestLit := $\text{argmax}_{L \in \text{Candidates}}$ performance(Specialise(NewRule,L))
      NewRule := Specialise(NewRule, BestLit)
      NewRuleNeg := {x∈Neg | x covered by NewRule}
  return NewRule

function Specialise(Rule, Lit):
  let Rule = "IF *conditions* THEN pos"
  return "IF *conditions* and Lit THEN pos"



THOMAS MORE

Illustration

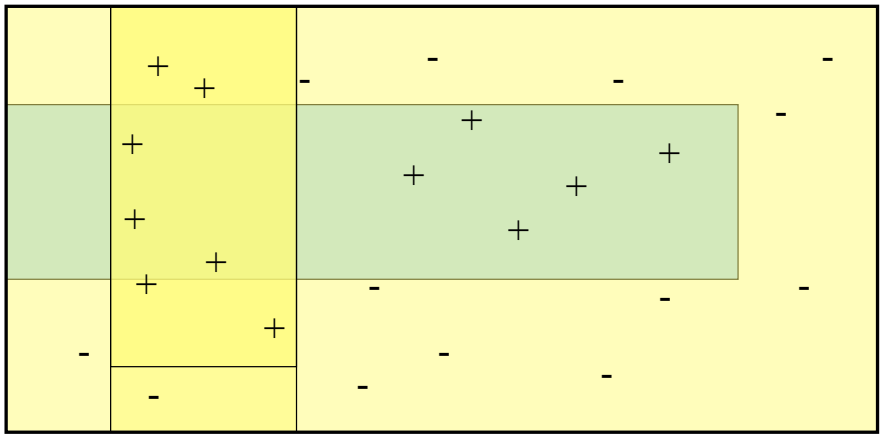


IF A & B THEN pos



THOMAS MORE

Illustration



IF A & B THEN pos

IF C & D THEN pos



THOMAS MORE

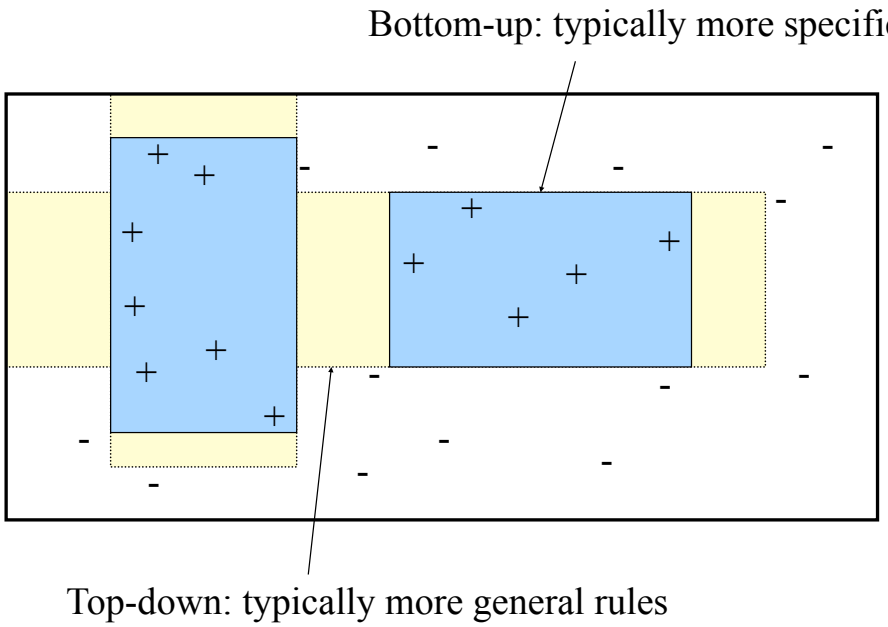
Some Options

- Options for learning 1 rule:
  - Top-down or Bottom-up? (see further)
  - Example-driven? (see further)
  - Hill-climbing, beam search, ... ?
- Learn rules for 1 class at a time, or for multiple classes?
  - E.g., first learn ruleset for pos, then one for neg; vs. learning 1 set with pos and neg rules
- Learn ordered or unordered set of rules?
  - Ordered: first rule that applies will be used
  - Allows for easy incorporation of “exceptions”
    - Exceptional rules are listed first
    - Consecutive rules can ignore these exceptions



THOMAS MORE

Illustration: Bottom-up vs. Top-down

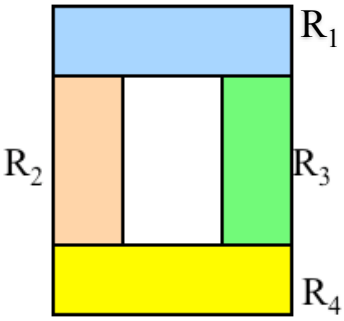




THOMAS MORE

Illustration: Rules with Exceptions

Concept = gray area



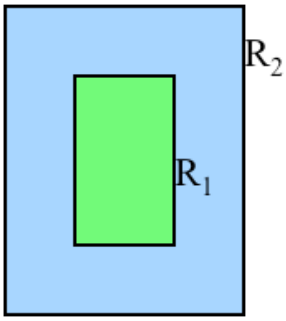
Unordered rules

R_1: if .... then pos

R_2: if .... then pos

R_3: if .... then pos

R_4: if .... then pos



Ordered rules:

R_1: if .... then neg

R_2: if .... then pos



THOMAS MORE

Heuristics

- Heuristics
  - When is a rule “good”?
    - High accuracy
    - Somewhat less important: high coverage
  - Possible evaluation functions:
    - Accuracy: *p / (p+n)* (p=#positives, n=#negatives)
    - A variant of accuracy: m-estimate: *(p+mq) / (p+n+m)*
      - Weighted mean between accuracy on covered set of examples and a priori estimate of true accuracy q (m is weight)
    - Entropy: more symmetry between pos and neg
- Post-pruning of rules
  - Cf. what was done for decision trees



THOMAS MORE

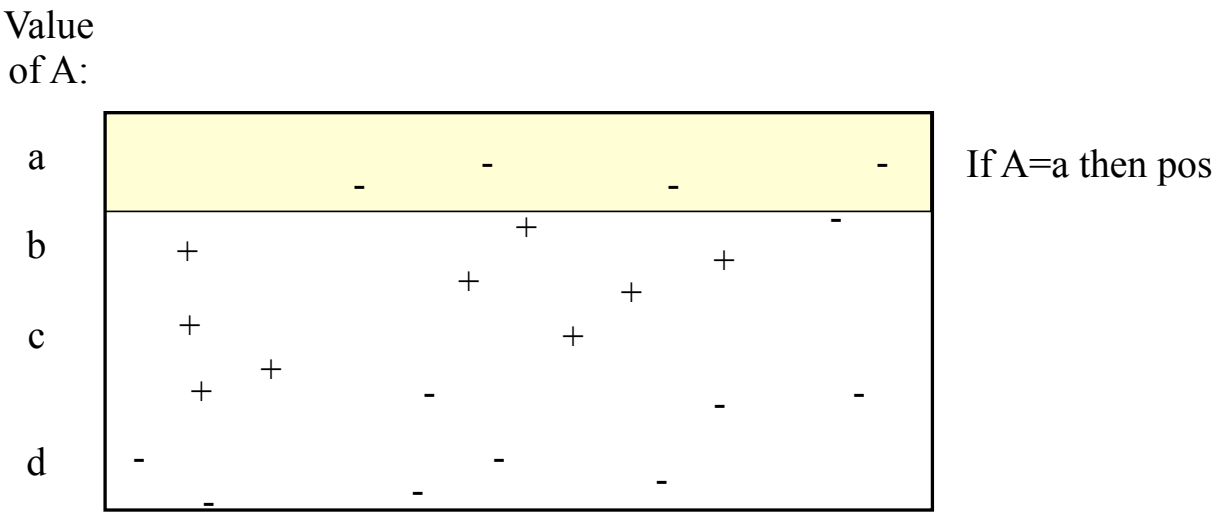
Example-driven Top-down Rule Induction

- Example: AQ algorithms (Michalski et al.)
- for a given class C:
  - as long as there are uncovered examples for C
    - pick one such example e
    - consider H_e = {rules that cover this example}
    - search top-down in H_e to find best rule
- Much more efficient search
  - Hypothesis spaces H_e much smaller than H (set of all rules)
- Less robust with respect to noise
  - what if noisy example picked?
  - some restarts may be necessary



THOMAS MORE

Illustration: not example-driven

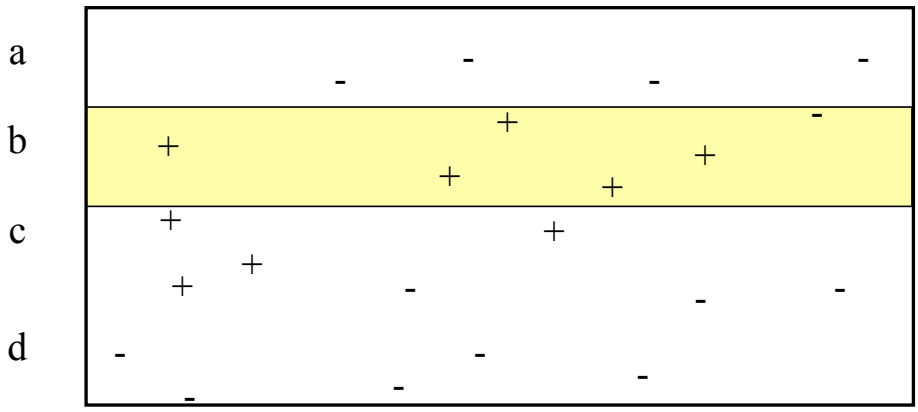


Looking for a good rule in the format "IF A=... THEN pos"



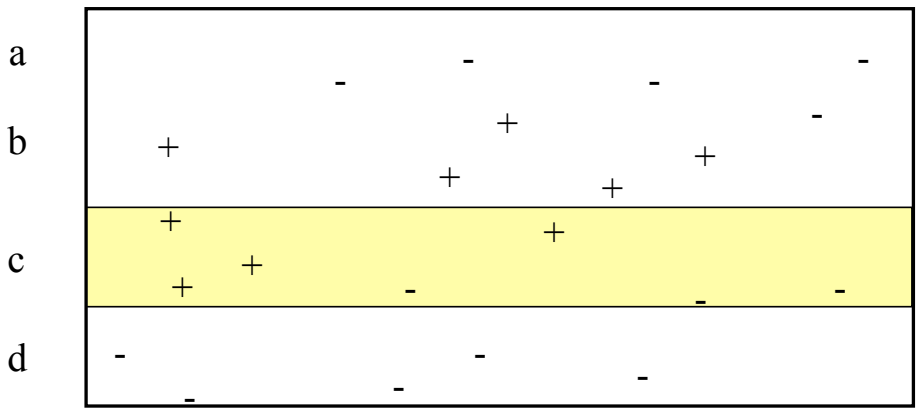
THOMAS MORE

Illustration: not example-driven



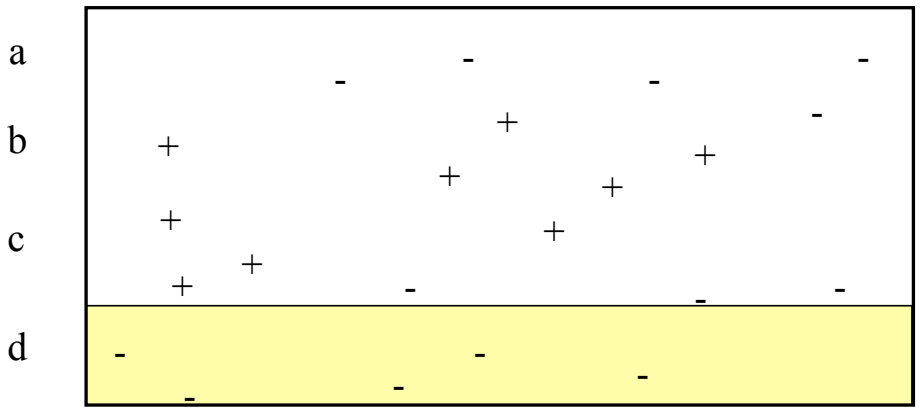
If A=b then pos

Illustration: not example-driven



If A=c then pos

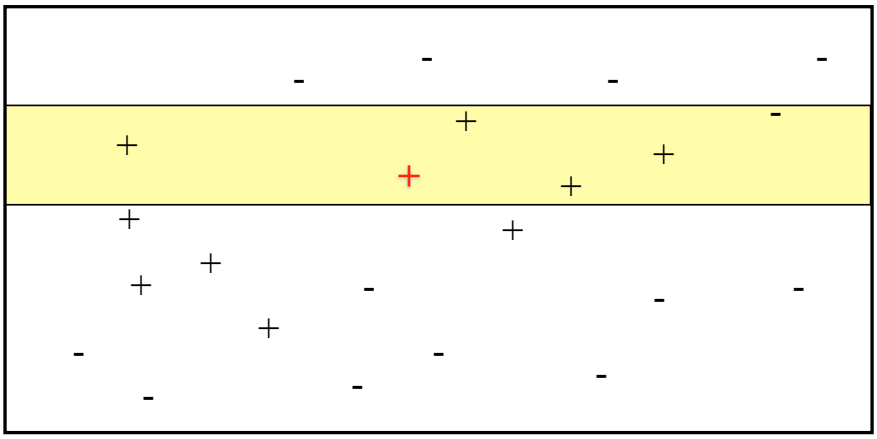
Illustration: not example-driven



If A=d then pos

THOMAS MORE

Illustration: example-driven



If A=b then pos

Try only rules that cover the seed "+" which has A=b.
Hence, A=b is a reasonable test, A=a is not.
We do not try all 4 alternatives in this case! Just one.

THOMAS MORE

Genetic Algorithms

- Not covered in detail here
- Here we discuss briefly
  - Basic principles
  - How to apply genetic algorithms to machine learning
    - Rule learning



THOMAS MORE

Evolutionary Computation

- Based on evolution as it occurs in nature
  - Lamarck, Darwin, Wallace: evolution of species, survival of the fittest
  - Mendel: genetics provides inheritance mechanism
    - Hence “genetic algorithms”
- Essentially a massively parallel search procedure
  - Start with random population of individuals
  - Gradually move to better individuals



THOMAS MORE

Basic Genetic Algorithm

GA(Fitness, Fitness_threshold, p, r, m):

Initialize: P := p random individuals ("hypotheses")

Evaluate: for each h∈P, compute Fitness(h)

while $\max_{h \in P}$ Fitness(h) < Fitness_threshold

1. *Select*: select (1-r) h's from P (each h∈P has a certain probability of being chosen), and put them in P'
2. *Crossover*:
   repeat $rp/2$ times:
   select h_1, h_2 at random from P
   (h_3,h_4) := crossover(h_1,h_2)
   add h_3 and h_4 to P'
3. *Mutation*: apply mutation operator to mp random members of P'
4. P := P'
5. *Evaluate*: for each h∈P, compute Fitness(h)

return $\text{argmax}_{h \in P}$ Fitness(h)



THOMAS MORE

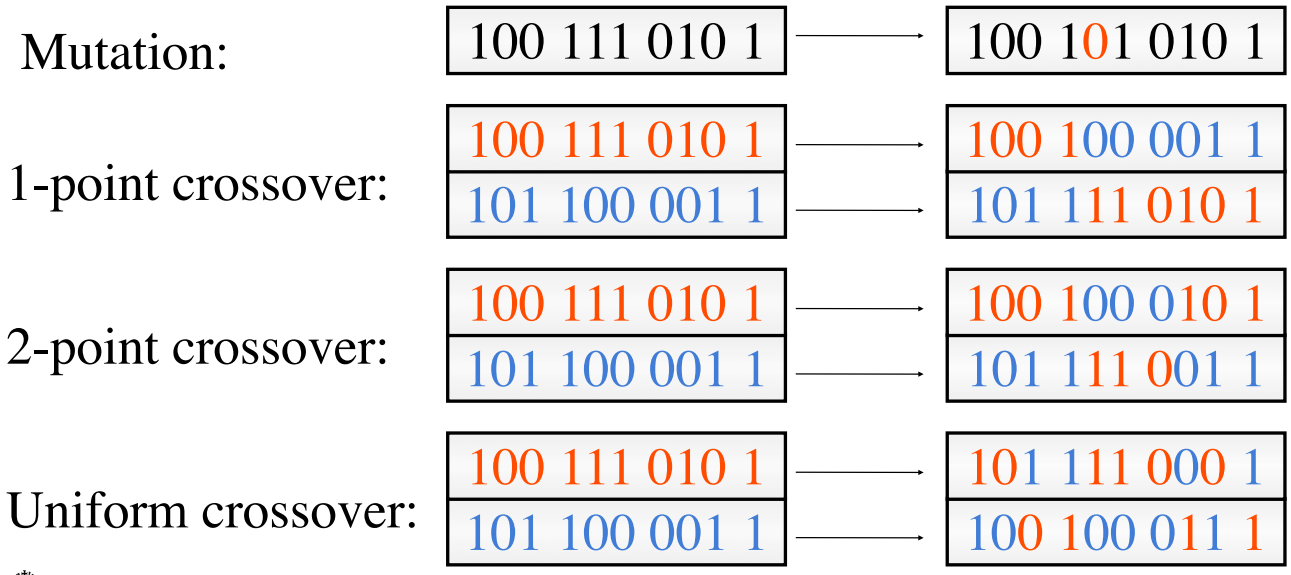
Options

- Options in basic GA:
  - How to represent individuals
    - Typically bitstrings (cf. our DNA)
    - Mapping bitstring-hypothesis?
  - How to select individuals
    - What is the probability of a hypothesis h being selected?
    - What is the fitness of an individual?
  - How to apply crossover, mutation, ...
    - Exactly what effect do they have?



THOMAS MORE

Examples of Operators



Selection Criteria

- Fitness-proportionate :

$$P(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

- Can lead to *crowding*
  - 1 individual (+ variants) dominates population
  - Not good if it happens too early
    - Convergence to not-so-good solution
- Alternatives:
  - *Tournament selection*
    - Choose h1 and h2 at random with uniform probability
    - With probability p, select the fittest of the two
  - *Rank selection*
    - Probability of selection proportional to *rank* of individual (1st, 2nd, 3rd, ...) w.r.t. fitness criterion

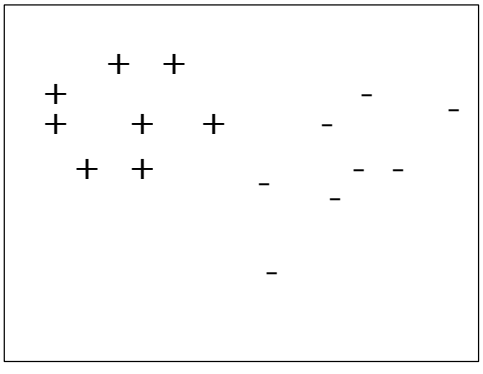
Discovery of Association Rules

- An example of descriptive induction as opposed to predictive induction
- Predictive: learn a function that predicts for new instances the value for a certain attribute (e.g., class)
- Descriptive: learn patterns in the data
  - e.g., find groups of similar instances: clustering
  - e.g., find associations between attributes

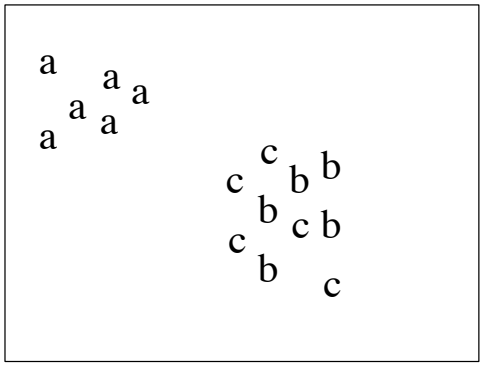


THOMAS MORE

Predictive Induction vs. Descriptive Induction



Find association between
1 specific property (+/-) and
any other properties



Find associations between
any properties
- e.g. clusters (assoc. X-Y)
- e.g. a in top left (assoc XY-a)



THOMAS MORE

- Difference not clear-cut
- Many views on relationship between predictive and descriptive induction
  - For instance: discriminatory induction
    - Predictive induction = learn to discriminate +/-
    - Can be done by performing descriptive induction on separate classes
      - Descr: find patterns that generally hold in whole set
      - Pred: find patterns that hold for + and not for -



THOMAS MORE

Association rules

- Association rules:
  - similar to decision rules: IF ... THEN ...
  - describe relationships between sets of boolean attributes
  - e.g., market basket analysis : learn which products often bought together

Client	cheese	bread	butter	wine	jam	ham
1	yes	yes	yes	yes	no	yes
2	yes	no	yes	no	no	no
3	no	yes	yes	no	no	yes
...	...	...	...	...	...	...

```
IF bread & butter THEN cheese
        confidence: 50%
          support: 5%
```



THOMAS MORE

Some Characteristics

- Association rule: IF a_1, ..., a_n THEN a_{n+1}, ..., a_{n+m}
- Is characterised by a
  - *Support*: how many % of all clients actually buy $a_1 ... a_{n+m}$?
    - if too low: rule not very important
  - *Confidence*: how many % of buyers of $a_1 ... a_n$ also buy $a_{n+1} ... a_{n+m}$?
    - need not be close to 100% percent
    - even small increase w.r.t. normal level may be interesting



THOMAS MORE

Searching for association rules

- Often very large databases to be analysed
- Efficient algorithm needed
  - Repeatedly running normal (adapted) rule induction algorithms is ***not*** efficient
  - Moreover, typical rule algorithms give a *minimal* set of rules that is sufficient to define a concept
    - But we want *all* rules satisfying criteria, not a minimal set
- The *APRIORI* algorithm (Agrawal et al., 1993)
  - Parameters: min. support, min. confidence
  - Finds all association rules with sufficient support/confidence
  - Note: available in Weka toolbox



THOMAS MORE

A Key Observation

- Observation:
  - let freq(S) = number of examples containing S
  - consider IF $a_1...a_n$ THEN $a_{n+1}...a_{n+m}$
  - support = freq($\{a_1, ..., a_{n+m}\}$)/freq({})
  - confidence = freq($\{a_1...a_{n+m}\}$)/freq($\{a_1...a_n\}$)
  - => all association rules with sufficient confidence and support can be derived from list of "frequent sets" + their frequencies
    - S is "frequent set" iff freq(S)>min_support*freq({})



THOMAS MORE

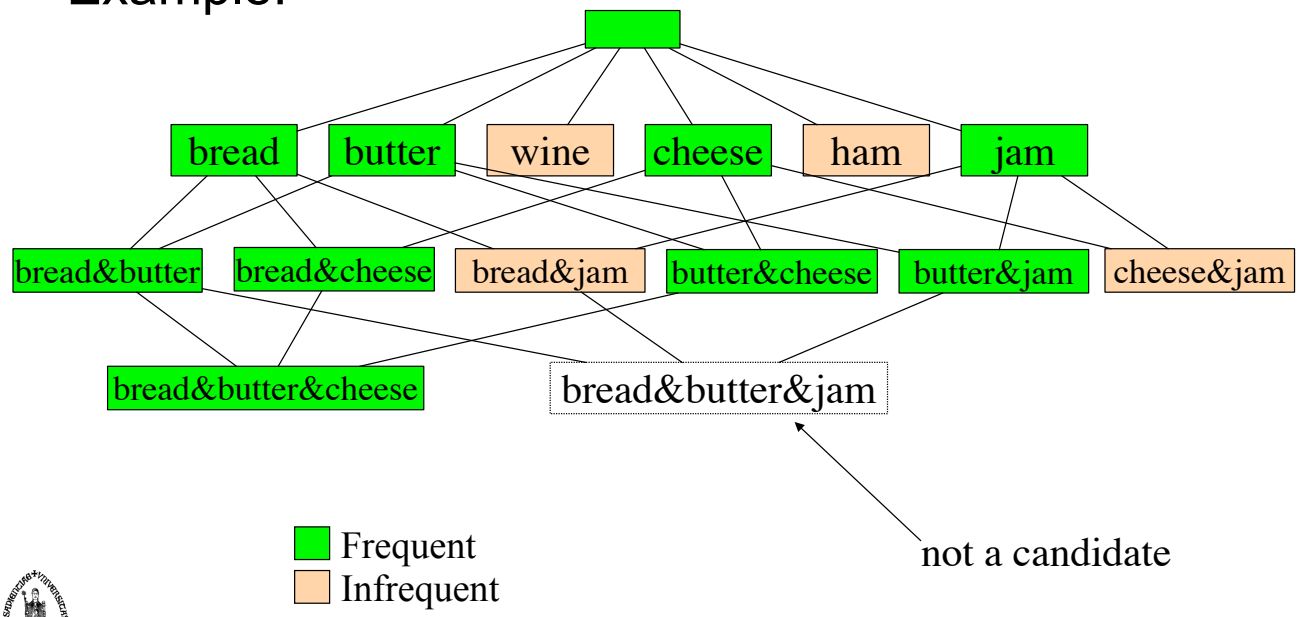
Finding Frequent Sets

- Step 1: find frequent sets
  - Observation: if $a_1...a_i$ is not frequent, then $a_1...a_{i+1}$ is not frequent
  - -> breadth-first, general-to-specific search
    - find all frequent sets of cardinality 1
    - find all frequent sets of cardinality 2
      - set $\{a_1,a_2\}$ can be frequent only if $\{a_1\}$ and $\{a_2\}$ both frequent
      - many pairs pruned before actually computing their frequency by looking at data
      - others: "candidates" -> need to check frequencies
    - find all frequent sets of cardinality 3
      - $\{a_1,a_2,a_3\}$ frequent only if $\{a_1,a_2\}$, $\{a_2,a_3\}$ and $\{a_1,a_3\}$ frequent...



THOMAS MORE

- Example:





THOMAS MORE

- Algorithm: finding frequent sets

```
min_freq := min_support * freq(∅);
d := 0;
Q0 = {∅ };  /* Qi = candidates for level i */
F := ∅ ;      /* F = frequent sets */
while Qd ≠ ∅ do
        for all S in Qd: find freq(S);  /* data access */
        delete those S in Qd with freq(S)<min_freq;
        F := F ∪ Qd;
        compute Qd+1;
        d := d+1
return F
```



THOMAS MORE

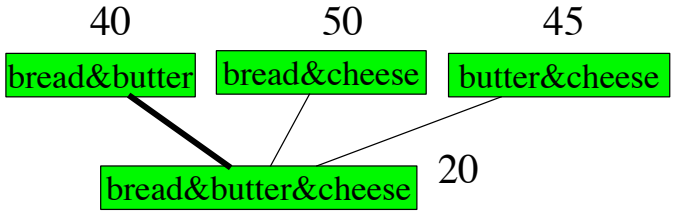
- Offline computation of new candidates:
  - "offline" = without having to look at examples

compute Q_{d+1} from Q_d and F:
  $Q_{d+1} := \varnothing$;
  for each S in Q_d:
    for each item x not in S:
      S' := S ∪ {x};
      if each subset of S' obtained by removing 1 element of S' is in F
      then add S' to Q_{d+1};



THOMAS MORE

- Step 2: infer association rules from frequent sets
  - if S ∪ {a} in F and freq(S ∪ {a}) / freq(S) > min_confidence
  - then return rule "IF S THEN a"




```
IF bread and butter THEN cheese
     confidence: 50% (=20/40)
        support: 5%
```



THOMAS MORE

Post-Processing of Rules

- Often many association rules found
- How to process them, as a human?
  - sort according to some criterion
    - e.g. support or confidence
    - sometimes used : statistical significance of deviation
      - p1 = P(cheese) = 0.6
      - p2 = P(cheese | bread&butter) = 0.7
      - rule more interesting if deviation of p2 from p1 is statistically more significant
      - combines both support and confidence
  - Other post-processing methods exist
    - Might even use a query language to query rule base
      - Find all rules for which conditions ... hold
      - Sort rules according to ...



THOMAS MORE

To Remember

- Sequential covering approaches
  - Different variants of basic algorithm
- Genetic algorithms
  - GA for rule learning (incl. details on GABIL approach)
- Association rules
  - What they represent, difference with predictive rules



THOMAS MORE

1 Machine Learning Oefeningen
Leren van concepten, version spaces en beslissingsbomen

1.1 Algemeenheid (Generality)

Sorteer de volgende concepten van booleaanse attributen A en B volgens hun algemeenheid ($\geq_g$): $A \wedge B$, $A \vee B$, A xor B, T (true), F (false).

1.2 Version Spaces

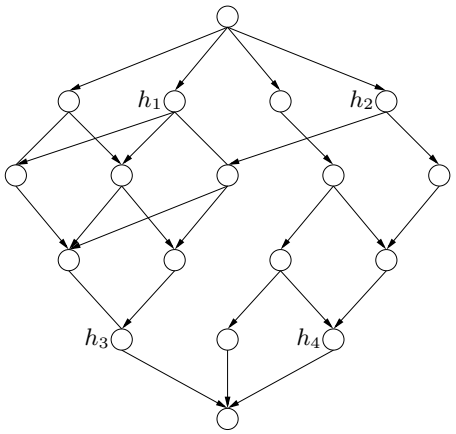
Beschouw de hypotheseruimte in Figuur 1 (geordend in een rooster volgens $\geq_g$). Een Version Space VS is gedefinieerd volgens zijn meest algemene grens $G = \{h_1, h_2\}$ en zijn meest specifieke grens $S = \{h_3, h_4\}$.

a. Duidt alle hypothesen aan die behoren tot VS (op Figuur 1).

b. Welke instanties worden positief geclassificeerd door VS?

c. Welke instanties worden negatief geclassificeerd door VS?

d. Welke instanties worden als "don't know" geclassificeerd door VS?



Figuur 1: Hypotheseruimte voor oefening 1.2.

1.3 Logische concepten uitdrukken met beslissingsbomen

Stel de volgende concepten voor dmv een beslissingsboom:

a. $A \wedge \neg B$

b. $\neg A \vee B$

c. $A \vee (B \wedge C)$

d. $(\neg A \wedge B) \vee (A \wedge \neg B)$

e. $(A \vee B) \wedge (C \vee D \vee \neg E)$

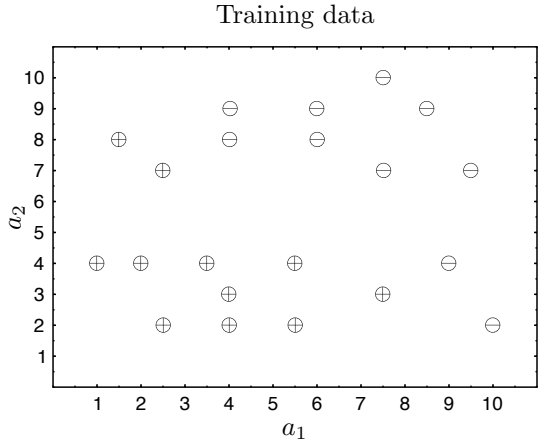
f. $(A \vee B \vee C) \wedge (D \vee E \vee F)$

1.4 Beslissingsoppervlak bij beslissingsbomen

Beschouw een dataset met twee *numerieke* attributen a_1 en a_2 en een nominaal doelattribuut c met twee mogelijke waarden: $\oplus$ en $\ominus$. Figuur 2 toont de leervoorbeelden.

a. Teken een beslissingsboom die alle leervoorbeelden correct classificeert.

b. Teken het beslissingsoppervlak van deze boom op Figuur 2.



Figuur 2: Leervoorbeelden voor oefening 1.4

1.5 Entropie en Information Gain

Beschouw de volgende tabel met leervoorbeelden:

Voorbeeld	Classificatie	a_1	a_2
1	$\oplus$	T	T
2	$\oplus$	T	T
3	$\ominus$	T	F
4	$\oplus$	F	F
5	$\ominus$	F	T
6	$\ominus$	F	T

a. Wat is de entropie van deze verzameling leervoorbeelden met betrekking tot het "classificatie" attribuut? (Hint: los op zonder rekenmachine)

b. Wat is de information gain van a_2 met betrekking tot deze leervoorbeelden? (Hint: los op zonder rekenmachine)

c. Wat is de information gain van a_2 met betrekking tot deze leervoorbeelden? (Hier zal je wel een rekenmachine voor nodig hebben)

1.6 Het ID3 algoritme

a. Toon een beslissingsboom die zou kunnen geleerd zijn met ID3, gegeven de volgende leervoorbeelden:

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	sunny	warm	normal	strong	warm	same	yes
2	sunny	warm	high	strong	warm	same	yes
3	rainy	cold	high	strong	warm	change	no
4	sunny	warm	high	strong	cool	change	yes

b. Voeg nu het volgende voorbeeld toe:

5 sunny warm normal weak warm same no

en toon hoe ID3 een beslissingsboom zou induceren, gegeven deze 5 voorbeelden.

2 Machine Learning Oefeningen
Extra oefening op leren van regelsets

Beschouw de dataset in Tabel 1.

a. Toon de kandidaattermen voor de eerste iteratie van:

- het top-down *LearnOneRule* algoritme (Slide 6 in Les3.pdf).
- het top-down example-driven *LearnOneRule* algoritme (kies eerst e_1).

b. Geef een set van alle regels die alle positieve voorbeelden van Tabel 1 omvatten (*coveren*) door gebruik te maken van het top-down example-driven *LearnRuleSet* algoritme (start altijd met het eerste positieve voorbeeld dat nog niet omvat is). Gebruik 'accuracy' als heuristiek om het zoeken te leiden.

c. Vind een regel die minstens e_1 omvat door gebruik te maken van de bottom-up versie van het example-driven *LearnOneRule* algoritme. Bottom-up wil zeggen dat je dient te beginnen met de meest specifieke regel (die e_1 omvat) en deze steeds meer algemeen moet maken zonder een negatief voorbeeld te omvatten. Gebruik opnieuw accuracy als heuristiek.

ex	temp	hum	wind	play
e_1	mild	norm	strong	pos
e_2	mild	norm	weak	pos
e_3	cool	norm	weak	pos
e_4	mild	high	strong	neg
e_5	cool	norm	strong	neg

Tabel 1: Voorbeeld dataset voor het induceren van regels.

7 Instance based learning

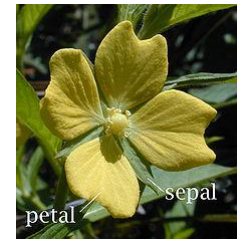
- General idea
- *k*-nearest neighbour
- Radial basis functions
- Lazy vs. eager learning

Instance Based Learning

- Key idea:
  - Store all training examples
  - When seeing a new instance:
    - Look at most similar stored instances
    - Make prediction based on those instances
- E.g. *k-nearest-neighbour*:
  - Find *k* most similar instances
  - Use most frequent class (classification) or mean target value (regression) as prediction
  - “Nearest neighbour” = 1-nearest-neighbour

knn example

- Iris data: 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample, they are the length and the width of sepal and petal, in centimeters.
- MATLAB example

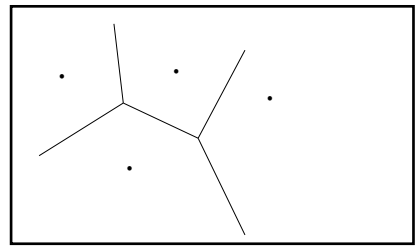


knn example 2

- Ripley data: artificial data set
- MATLAB example

Voronoi Diagrams

- Indicate areas in which prediction influenced by same set of examples
- E.g., 1 nearest neighbour:

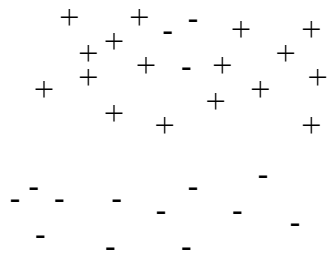


Properties

- Advantages:
  - Learning is very fast
  - No information is lost
- Disadvantages:
  - Slow at query time
  - Easily fooled by irrelevant attributes
  - Good similarity measure necessary
    - Easiest in numeric space ($\mathbf{R}^n$)

Keeping All Information

- Advantage: no details lost
- Disadvantage: "details" may be noise



The "Curse of Dimensionality"

- The curse of dimensionality: high-dimensional instance space (many attributes) has bad effect on learnability
- Especially bad for IBL:
  - Assume there are 20 attributes, of which only 2 relevant
  - Similarity w.r.t. 18 non-relevant attributes dominates similarity w.r.t. 2 relevant ones!
- (Less bad for, e.g., decision trees / rules : why?)

Rescaling Attributes

- 2 reasons for rescaling attributes
  - Attributes with different original scale should have equal importance
    - E.g. A_1: [0, 1]; A_2: [-10, +10]
    - Distance w.r.t. A1 always small, hence A1 has small influence
  - Give higher weight to more relevant attributes
    - Partially solving irrelevance problem

Selecting attribute weights

- Several options:
  - Experimentally find out which weights work well (cross-validation)
  - Other solution: (e.g. Langley,1996)
    - Normalise attributes (to scale 0-1)
      - Removes "different scales" problem
    - Then select weights according to "average distances *within* class"
      - Removes "irrelevance" problem

$$w_i = 1 - 1/n \sum_{k=1}^{c} \sum_{j=1}^{n_k} | p_{ki} - x_{ji} |$$

p_k = average of class k

Distance-weighted kNN

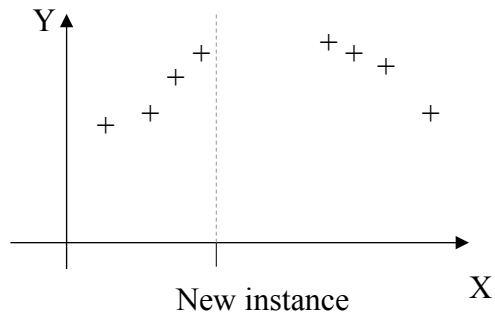
- Idea: give higher weight to closer instances
- Can now use all training instances instead of only *k* ("Shepard's method")

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i} \quad \text{with } w_i = \frac{1}{d(x_q, x_i)^2}$$

! In high-dimensional spaces, a function of d that "goes to zero fast enough" is needed. (Again "curse of dimensionality", see course notes.)

Locally weighted regression

- Obvious problem with following kind of data:
  - Given new x, what value for y would you predict?
  - What will k-NN (e.g., 3-NN) predict?



Building local models

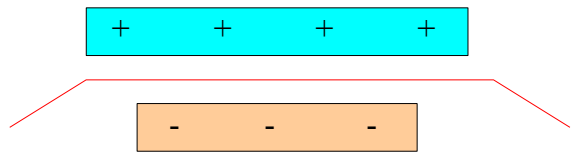
- Build local model in region around x
  - e.g. linear or quadratic model, ...
  - Minimizing
    - Squared error for k neighbours
    - Distance-weighted squared error for all neighbours
  - ... $E(x_q) = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$

Using Prototypes

- Some disadvantages of IBL...
  - Rough decision surfaces
  - Slow at prediction time
- ... can be solved by using prototypes
  - Representative for a whole group of instances



- Prototype can be:
  - Single instance, replacing a group
  - Other structure (e.g., rectangle, rule, ...)
    - In this case: define distance
- In book: radial basis functions



Radial basis function networks

- Build global approximation as linear combination of local approximations
  - Influence of 1 local approximation u goes down quickly with distance

$$f(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x))$$

$$\text{with for instance } K_u(d(x_u, x)) = e^{-\frac{1}{2\sigma_u^2} d^2(x_u, x)}$$



Learning RBFNs

- Questions:
  - How to choose K_u, x_u
    - E.g., x_u scattered uniformly, or according to example distribution
  - How to train weights w_u
    - Linear regression

Lazy versus eager learning

- Lazy learning:
  - Wait for a question to be asked
  - Then start thinking
  - Examples: kNN, case based reasoning
- Eager learning:
  - Generalize before seeing question
  - Examples: TDIDT, Naive Bayes, ...

Differences

- Differences?
  - Eager learner creates global approximation
    - 1 theory has to work for all predictions
  - Lazy learner creates local approximations on demand
    - In a sense, many different theories used
  - With same hypothesis space H, lazy learner is in fact more expressive

To Remember

- Generalities of instance based learning
  - Basic idea, (dis)advantages, use of Voronoi diagrams, lazy vs. eager learning
- Various instantiations
  - kNN, distance-weighted methods, ...
  - Rescaling attributes
  - Use of prototypes, RBFs

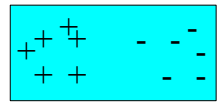
8 Clustering

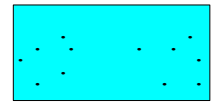
- What is clustering?
- Flat clustering
  - K means clustering, EM algorithm
- Hierarchical clustering
  - Distance-based approaches
- Conceptual clustering
  - The Cobweb algorithm
- Using clustering for prediction

What is clustering?

- Find groups (clusters) of instances so that
  - Instances in the same group are similar
  - Instances in different groups are different
- Compared to classification:
  - Similarity: assign classes to examples
  - Difference: classes not known in advance
    - Hence also called "unsupervised learning"
    - Classes (even taxonomies) are "invented"

Classification problem





Clustering problem

Examples

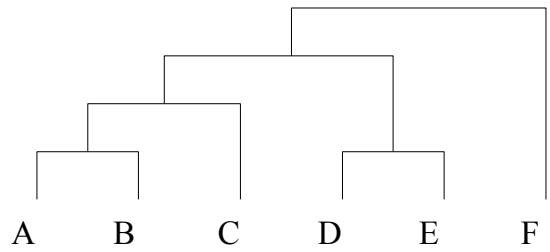
- Typical example: construct taxonomy of, e.g., animals
  - Example of "hierarchical clustering"
- In machine learning / data mining:
  - Applicable, for instance, in marketing:
    - identify typical customers, e.g., car drivers
    - Produce products (e.g. cars) that aim at one specific group
  - Auxiliary method for other techniques
    - e.g., constructing local models (cf. instance based methods), ...

Similarity Measures

- How to measure similarity between instances?
  - Similar problem as for instance based methods
- Possible options:
  - Distance metric
    - Euclidean distance
    - Other...
  - More general forms of similarity
    - Do not necessarily satisfy triangle inequality,symmetry, ...

Flat vs. Hierarchical Clustering

- *Flat* clustering:
  - Given data set, return partition
- *Hierarchical* clustering:
  - Combine clusters into larger clusters, etc. until 1 cluster = full data set
  - Gives rise to *cluster hierarchy* or *taxonomy*



Extensional vs. Conceptual Clustering

- *Extensional* clustering:
  - Clusters defined as sets of examples
  - < statistics
- *Conceptual* clustering:
  - Clusters *described* in some language
  - Typical criteria for good conceptual clustering:
    - High intra-cluster similarity
    - Simple conceptual description of clusters

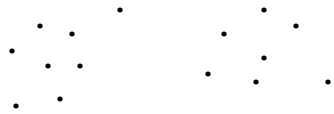
Flat Extensional Clustering

- Flat clustering:
  - Given: set of unlabeled data
  - Find: clusters of similar instances
    - "similar" = close to each other in some space
    - Number of clusters may be given
    - Other quality criteria for clusters may be given
  - Example algorithms: K-means, EM

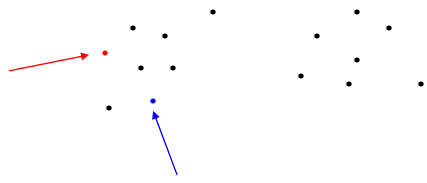
K-means clustering

- Input : data, number of clusters K
- Algorithm:
  - Start with K random seeds for clusters
  - Repeat until no changes:
    - For all instances e:
      - Add e to cluster of closest seed
    - Compute centres of all clusters
      - E.g., for numeric data: compute average
    - These centres are the seeds for the next iteration

Illustration

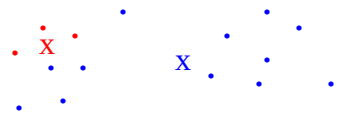


Illustration



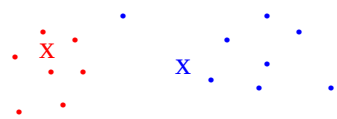
2 “seeds” chosen at random

Illustration



- Other points assigned to closest seed:
  gives “red” and “blue” cluster
- Centers of these clusters (x) serve as new seeds

Illustration



Reassign points to closest seed

Illustration



Recompute seeds (centers of current clusters)

Illustration



Reassign points, recompute seeds, reassign points.
When no points are re-assigned to another cluster, stop.

The EM algorithm (1)

- "Expectation maximisation"
- Similar to K-means, but:
  - Now we assume e.g. normal distributions
  - Examples not assigned to 1 cluster, but partially to different clusters (proportionally to distribution)
    - E.g., example x belongs to Cluster A for 70%, B for 25%, C for 5%
- EM actually much more general than just "clustering"
  - Find a number of distributions generating the data
    - = building "mixture models"

The EM algorithm (2)

function CLUSTEREM(data set S, integer k) **returns** set of Gaussians:
  Initialize μ_i and σ_i randomly
  while stop criterion not reached:
    for $j = 1$ **to** N:
      for $i = 1$ **to** k:

$$Pr(x_j \in C_i) := \frac{exp(-(x_j-\mu_i)^2/2\sigma_i)}{\sum_{i'=1}^{k} exp(-(x-\mu_{i'})^2/2\sigma_{i'})}$$

    for $i = 1$ **to** k:

$$\mu_i := \frac{\sum_j x_j Pr(x_j \in C_i)}{\sum_j Pr(x_j \in C_i)}$$

$$\sigma_i^2 := \frac{\sum_j (x_j-\mu_i)^2 Pr(x_j \in C_i)}{\sum_j Pr(x_j \in C_i)}$$

  return $\{(\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots, (\mu_k, \sigma_k)\}$

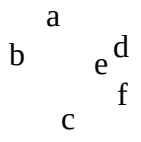
Hierarchical Extensional Clustering

- Top-down (“divisive”) methods
  - Start with 1 cluster (whole data set)
  - Divide it into subsets
  - Subdivide subsets further, etc.
- Bottom-up (“agglomerative”) methods
  - Start with singleton clusters
  - Join closest clusters together
  - Repeat until 1 cluster

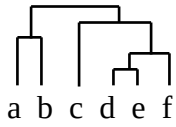
Example Agglomerative Methods

- A number of well-known agglomerative methods exist
  - Variants according to definition of “closest clusters”
    - Distance criterion for examples assumed known
    - How does it generalise to distance between clusters?
  - Options:
    - *Single linkage*: distance between clusters = distance between closest points
    - *Complete linkage*: cluster distance = distance between furthest points
    - Average distance, ...

Illustration



Single linkage:
d-e closest -> join first
e-f next
...

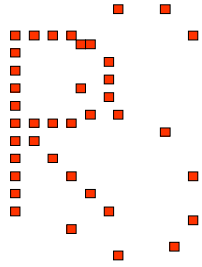


Conceptual clustering

- Previous methods just form groups of examples
- Conceptual: return description of groups in some language L
  - Quality of clusters depends on properties of cluster description as well
    - Typically, simple cluster description in L preferred
    - Language defines *context* in which quality of clustering is evaluated
  - Whether 2 elements are in same cluster, may not only depend on themselves, but also on other data

Example

- How would you cluster these points?



"Intuitive clusters" cannot be defined by looking only at distances

TDIDT for Clustering

- Decision tree = conceptual hierarchical clustering:
  - Each node = 1 cluster
  - Tests from root to node = *conjunctive description* of cluster
  - Hence, language L of cluster descriptions = conjunctions of attribute tests

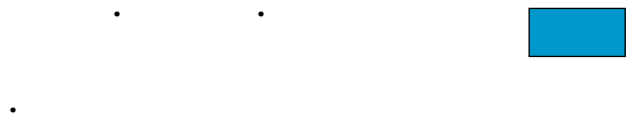
Cobweb

- Well-known clustering algorithm (Fisher, 1987)
- Finds conceptual hierarchical clustering
  - *Probabilistic description* : probability distribution of attribute values for each cluster
  - Heuristic:
    - Maximize *predictiveness* and *predictability* of attribute values
    - Predictability : given cluster, how well can you predict attribute values?
    - Predictiveness : given attribute values, how well can you predict cluster?
    - Maximize a combination of both

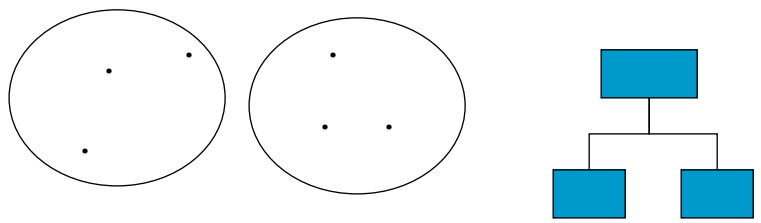
Cobweb Algorithm

- *Incremental* algorithm
- For each example:
  - For each level (top to bottom) of current taxonomy:
    - Change current level using one of several *operators*
      - Add example to cluster
      - Create new cluster (with 1 example)
      - Merge clusters
      - Split clusters
    - Move down to relevant subcluster
  - Evaluation of clustering: try to maximize a combination of predictiveness $P(C_k|A_i=v_{ij})$ and predictability $P(A_i=v_{ij}|C_k)$
    (C_k=cluster, A_i=attribute, v_{ij} = value)

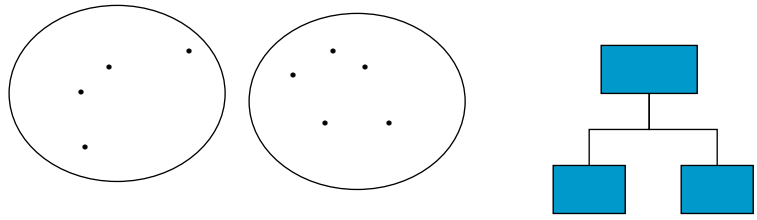
Illustration



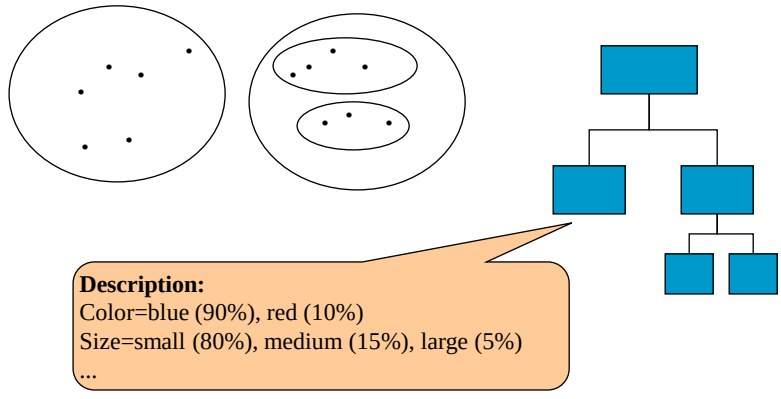
Illustration



Illustration



Illustration



Using clustering for prediction

- Clustering can be used for prediction of any property
- Once clusters found, prediction is made in a 2 step process
  - Given known attributes of instance, predict cluster (OK if high predictiveness)
  - Given cluster, predict unknown attributes (OK if high predictability)
- = "flexible prediction": not known in advance what will be given and what will need to be predicted

- If something is known about what will be predicted, clustering process can be tuned
  - Maximize predictiveness of attributes that will be given
  - Maximize predictability of attributes that will need to be predicted
- Many learning approaches can be described in this "predictive clustering" framework
  - Try, e.g., decision trees, instance-based learning

To Remember

- Important concepts:
  - Similarity measures, distances
  - Flat & hierarchical clustering
  - Extensional vs. Conceptual clustering
- Clustering algorithms:
  - K-means, EM
  - Single/complete linkage
  - Cobweb
- Use of clustering for prediction

Dynamic Time Warping

- Dynamic programming
- Distance measures
- Applications in spell corrector and speech recognition

Dynamic Programming in every day life

- Transliterations in E-mails, SMS, shortcuts
  - /thx/ = /thanks/
  - /cy/ = /see_you/
- Spelling Mistakes & Corrections
  - /last_yaer/ = /last_year/

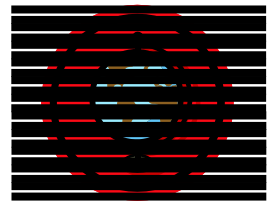
We use

- Substitutions
- Insertions & deletions

to make

context dependent modifications !!

- We are masters in dealing with imperfect or incomplete data
- Image processing



- Some Example Sentences of not so proper English (1)

describtion of prononciation is necessairy	*description of pronunciation is necessary …*
lqst yqer	*last year*
bcum a mmbr 4frE !	*bum a member 4frE !*

- Correction was done by an off-the-shelf commercial “spell checker”
  - found and corrected most (typographical errors)
  - found some but miscorrected them (SMS language)
  - didn’t detect the more difficult ones or couldn’t come up with any alternative suggestions (SMS language, phonetics)

WHY can we recognize from imperfect data ?

- Intrinsic redundancy in Natural Language
  - Of all possible letter combinations only a small fraction make up existing words
    - aa,ab,ac,ad,ae,af,ag,ah,ai,aj,ak,al,am,an,ao,ap,aq,ar,as,at,au, … (23%)
    - ace,act,add,ado,aft,age,ago,aha,aid,ail,aim,air,alb,ale,all,alp,amp,.. (5%)
  - Of all possible word combinations only a small fraction make up existing sentences

- By using Contextual information
  - Topic
  - Visual cues
  - Knowledge about how things happen around us

HOW can we recognize from imperfect data ?

➔ By finding the best hypothesis given an imperfect input

What's involved in this recognition process ?

- imperfect input (test)
- knowledge about the acceptable hypotheses
  - lexicon (words)
  - grammar
  - context, topic
- a distance metric
  - that measures the distance between the input and a hypothesis
- a search algorithm
  - a method to generate (all possible) hypotheses and find the best one
  - this should be done in a computational efficient way as the number of potential hypotheses may be very large and often infinite

A Metric for Word Similarity?

- How similar are following word pairs ?
  - *describtion vs. description*
  - *home vs. mother*
  - *input vs. peanut*
  - *words vs. sword*
  - *words vs. swords*
  - *lqst vs. last*
  - *mohter vs. mother*
  - *emit vs. time*
- Intuition gives us hints, but is insufficient as a formal MEASURE is required
  - is lqst more similar to last or to list ?
  - is describion more similar to description or to describing ?

A "bag-of-letters" Distance Measure

- $d_B(w1,w2)$ = ?
  - The bag-of-letter distance measure
  - is given by the sum of the difference between the letter histograms of both strings
- Examples:
  - d_B(mohter, mother) = 0
  - d_B(describion, description) = 2
  - d_B(swords, words) = 1
  - d_B(words, sword) = 0
  - d_B(time, emit) = 0

A "bag-of-letters" Distance Measure

- Observations:
  - not consistent with intuition
  - certain "mistakes" don't influence the score at all
  - total disregard for sequential nature of strings of letters

A Letter Substitution Distance Measure

- $d_S(w1,w2)$ = ?
  - The letter substitution measure counts
  - the sum of the required substitutions to turn one string into the other; extra white space is added at the end of words when required
- Examples
  - d_S(mohter, mother) = 2
  - d_S(describion, description) = 5
  - d_S(word, words) = 1
  - d_S(word, sword) = 5
  - d_S(time, emit) = 4

A Letter Substitution Distance Measure

- Observations:
  - minimal “mistakes” (single letter insertion or deletion) can lead to very different distances dependent on where in the word these happen
  - not consistent with intuition

Levenshtein Distance or Minimum Edit Distance

- The Minimum Edit Distance (Levenshtein distance) between two strings is
  - the minimum number of basic operations required to transform one string into another
  - with as basic operations: substitutions(Sub), insertions(Ins), deletions(Del)
- Counting Sub, Ins and Del's
  - d_L(describtion, description) = 1Sub
  - d_L(describion, description) = 1Sub+1Del
  - d_L(sword, works) = 1Del + 1Sub + 1Ins
  - d_L(lqst, last) = 1Sub
  - d_L(time, emit) = 4Sub
  - d_L(mohter, mother) = 2Sub
- Observations:
  - reasonably consistent with intuition
  - having insertions and deletions allows to find good matches between sequences that are not perfectly aligned and not of equal length
  - equal weight assumption for all operations seems to be a weak assumption
- Extension
  - Often letter reversal(R) is also added as a basic operation (Dammerau-Levenshtein)

Minimum Edit Distance for Spelling Correction examples

- Problem:
  - references { `describe, describing, description, descriptions, …` }
  - input `describtion`
- Solution:
  - Find the minimum edit distance for each word in the dictionary
  - Select the one with the smallest distance as the correct word

description	**descriptions**	**describ-i-ng**	**describe---**
describtion	**describtion-**	**describtion-**	**describtion**
p→b	**p→b -s**	**+t +o -g**	**e→t +i +o +n**
1Sub	**1Sub+1Del**	**2Ins+1Del**	**1Sub+3Ins**
D=1	**D=2**	**D=3**	**D=4**

- Issues:
  - ambiguity: many hypotheses (words) will lead to the same distance
  - computing the minimum edit distance for long sequences is not trivial

Computing the Minimum Edit Distance

- Intuitive:
  - STEP 1: Align the two strings
  - STEP 2: Add up all costs of edit operations for the given alignment
  - Issue:
    - Finding ‘the’ alignment is not trivial for long strings and many edits

Computing the Minimum Edit Distance

- Better:
  - Enumerate all possible alignments
  - Compute the total edit cost for each hypothesized alignment
  - Find the lowest total edit cost among all hypothesis
  - This yields implicitly the optimal alignment
  - Issue:
    - the number of possible alignments may be very large ($O(N^3)$, N=string length)
- Optimal:

  ➔DYNAMIC PROGRAMMING (DP)

Aligning Strings and Minimum Edit Distance

- Example
  - reference: sword
  - input: works

  - alignment 1
    - (s,w) (w,o) (o,r) (r,k) (d,s)
    - Edit Distance: 5xSub = 5

w	o	r	k	s
|	|	|	|	|
s	w	o	r	d
Sub	Sub	Sub	Sub	Sub

  - alignment 2
    - (s,-) (w,w) (o,o) (r,-) (-,k) (d,d) (-,s)
    - Edit Distance: 2xDel + 2xIns = 4

-	w	o	r	-	k	s
|	|	|	|	|	|	|
s	w	o	r	d	-	-
Del	☺	☺	☺	Del	Ins	Ins

  - alignment 3
    - (s,-) (w,w) (o,o) (r,k) (d,d) (-,s)
    - Edit Distance: 1xDel + 1xSub + 1xIns = 3

-	w	o	r	k	s
|	|	|	|	|	|
s	w	o	r	d	-
Del	☺	☺	☺	Sub	Ins

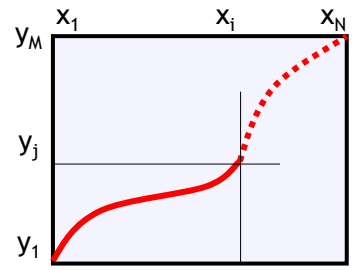
- The Minimum Edit Distance implicitly finds the optimal alignment and vice versa

Dynamic Programming Concept

- Avoid enumeration and computation of sub-problems that can not possibly be optimal
- Applied to string matching: abandon computation on alignments that can not possibly be the optimal ones

Dynamic Programming Concept

- Assume you know that (x_i,y_j) is on the optimal path, then:
  - the path can be decomposed into 2 partial paths with (x_i,y_j) as anchor points
  - each of the partial paths must be optimal for the each of the partial input/output combinations otherwise another and better global alignment path could be constructed by joining the better partial paths together $D(X_{1:N},Y_{1:M}) = D(X_{1:i},Y_{1:j}) + D(X_{i+1:N},Y_{j+1:M})$



- In practice: build up the paths recursively

Trellis

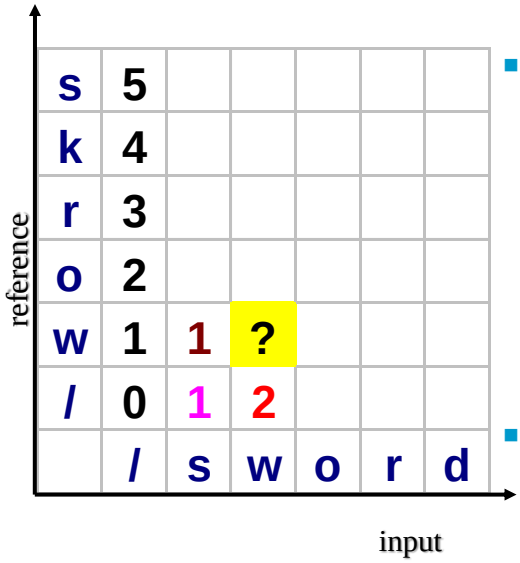
reference \ input	/	b	c	u	m
e	6	5	4	4	3
m	5	4	3	3	2
o	4	3	2	2	2
c	3	2	1	2	3
e	2	1	1	2	3
b	1	0	1	2	3
/	0	1	2	3	4

D(N,M) gives the distance between the full strings

- A trellis organizes input vs. reference in a kind of matrix, representing not only full reference and input strings but also all partial strings
- Notations and Conventions
  - $X_{1:i} = \{x_1 \ldots x_i\}$ and $Y_{1:j} = \{y_1 \ldots y_j\}$;
  - D(i,j) = distance = "the number of edits" used to match the partial strings $X_{1:i}$ and $Y_{1:j}$ (*)
  - e.g. D("bcu","be") = 2 = 1SUB+1INS
- A dummy start symbol '/' may be added to allow for insertions/deletions at the start of the strings
  - e.g. D("/","bec")=3= 3DEL

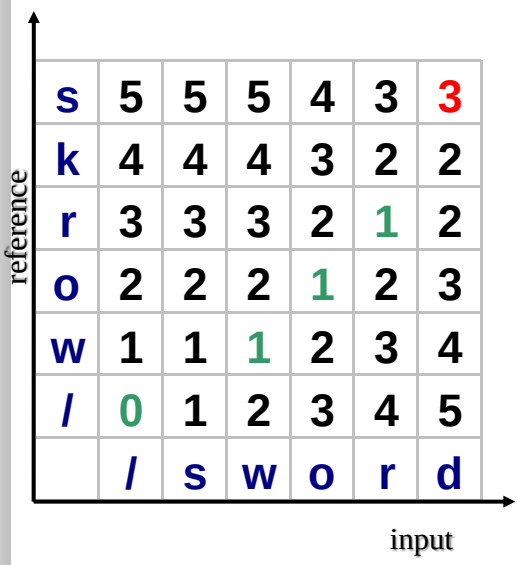
NOTE(*): we may not use traditional matrix conventions for the numbering of rows and columns in a trellis

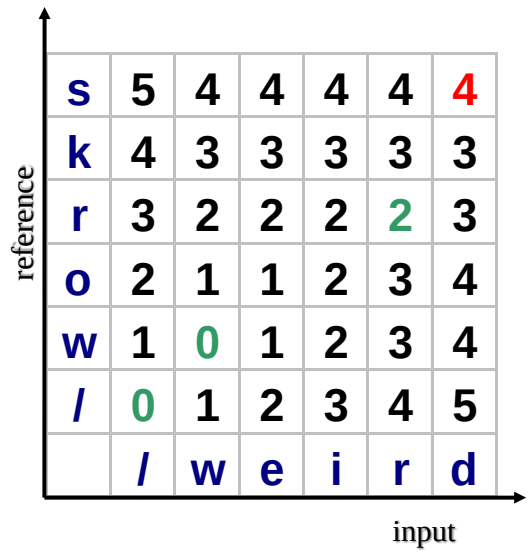
Trellis and Dynamic Programming



- How to compute a cell ?
  - principle: find the best combination of a smaller path and one additional basic operation [Del, Ins, Sub/Match]
  - example: D(“sw”,”w”) = min of
    - D(“sw”,”/”) + Del(‘w’) = 2+1
    - D(“s”,”/”) + Match(‘w’,’w’)= 1+0
    - D(“s”,”w”) + Ins(‘w’) = 1+1
- Fill the trellis in the proper order so that all required sub-problems are solved first
  - column-by-column
  - bottom-to-top

Trellis - Examples





Weighted Edit Distance

- Levenshtein distance is often suboptimal
  - e.g. in SMS language we omit characters much more often, than we insert new ones
- Weighted Edit Distance
  - assign different weights (costs) to the different basic operations
  - e.g. Sub=2, Ins=5, Del=1
  - REMARK: This distance is asymmetric !! [$d_W(s_1,s_2) \neq d_W(s_2,s_1)$]
  - Our standard usage will be
    - d_W(Input,Reference) = the sum of the costs of the operations to transform a Reference to an Input
- Further Extensions
  - apply letter specific weights
  - apply position specific weights

Weighted Edit Distance for Spelling Correction example

- Problem:
  - references { bum, become, … }
  - input bcum
- Weighted edit distance with weighted uniform costs
  - Sub=2 Ins=3 Del=1
- Weighted Edit Distance with vowel/consonant specific costs
  - Sub_v=2 Ins_v=7 Del_v=1
  - Sub_c=3 Ins_c=5 Del_c=2

b-um	become
bcum	b-cum-
+c	+e u→o +e
1Ins	1Sub+2Del
D=3	D=4

b-um	become
bcum	b-cum-
+c	-e o→u -e
$1Ins_c$	$1Sub_v+2Del_v$
D=5	D=4

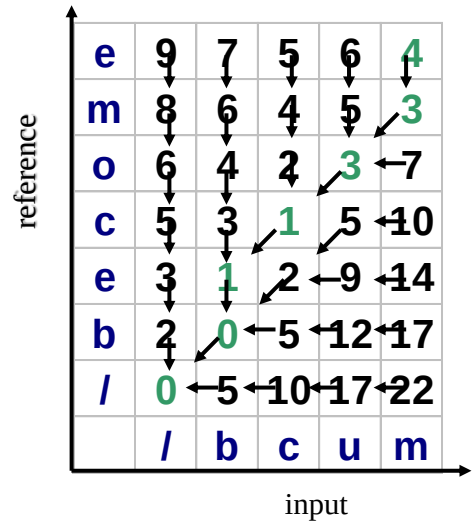
Weighted Levenshtein Distance and DP

Local Distance Matrix: d(i,j)
= single edit operation costs

e	1	2	2	2	2
m	2	3	3	3	0
o	1	2	2	2	2
c	2	3	0	3	3
e	1	2	2	2	2
b	2	0	3	3	3
-	0	5	5	7	5
	-	b	c	u	m

deletions insertions substitutions/matches

Global Distance Matrix: D(i,j)
= string distances

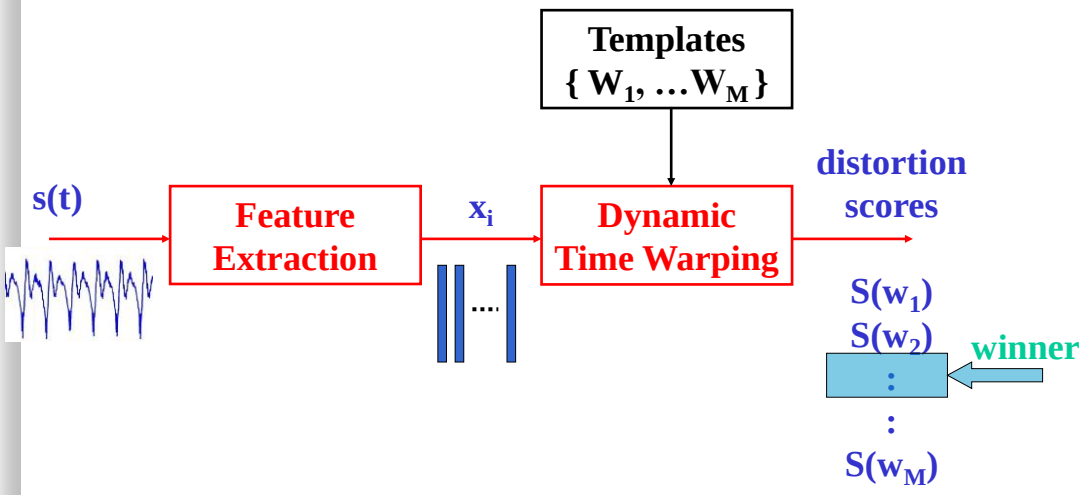


Backtracking & Alignment

- The trellis computes progressively the string edit scores but does not yield (in a direct manner) the alignment between the two strings
- The alignment is easily obtained by backtracking
  1. Compute the accumulated distance matrix in a forward pass, making a note of following extra information for each cell:
     - a pointer to the matching strings without the last edit operation (=backpointers)
     - the last edit operation yielding this optimal partial match
  2. In a backward pass
     - starting from the cell showing the match of the full strings
     - trace back to the beginning following the backpointers stored in the cells
     - do this up to the beginning of the matched strings
- Example: cfr. green cells in previous example

DTW for speech recognition

Word Recognition by Template Matching



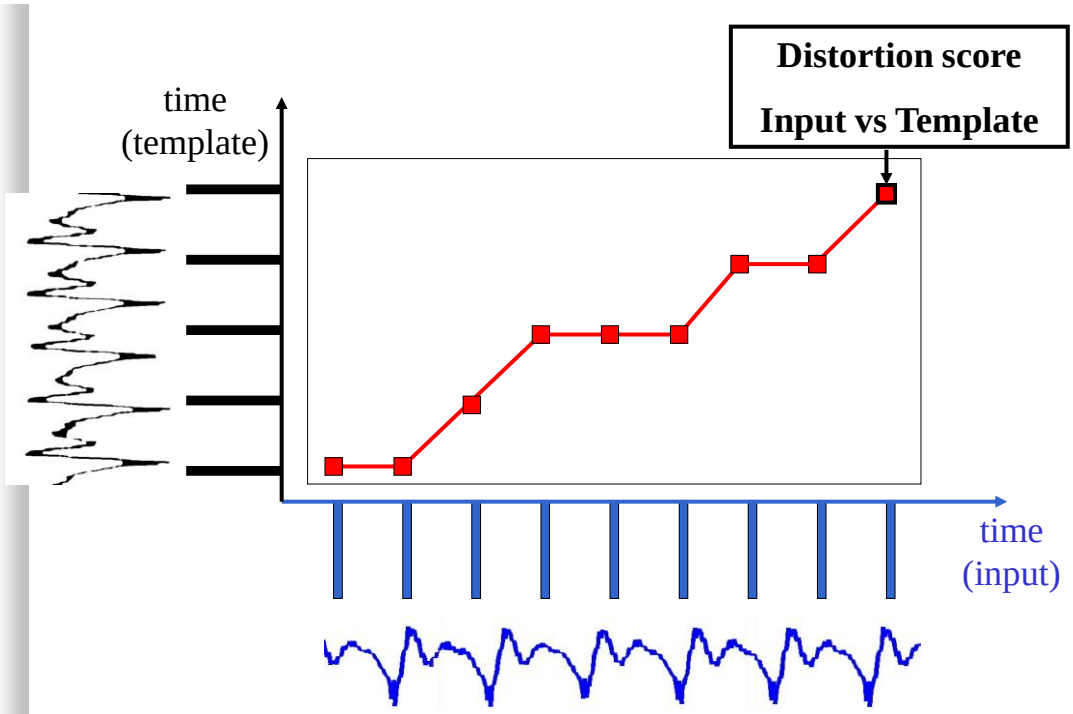
Word Recognition by Template Matching

- Represent words by a sequence of feature vectors
  - feature stream: 100 feature vectors/sec
  - feature vector: 12-dimensional mel cepstrum (representation of spectral envelope)

- "Local" Distance Metric
  - compute similarity between feature vectors as Euclidean distance between mel cepstrum vectors

$$\left\|\vec{c}_1 - \vec{c}_2\right\|^2 = \sum_{i=1}^{L}\left[c_1(i) - c_2(i)\right]^2$$

- "Global" Distance Metric
  - compute similarity between words as a sum of frame based local distances
  - use dynamic programming to find the smallest distance and corresponding best alignment between test and reference utterances



Dynamic Time Warping (DTW) from frame level to utterance level

- PROBLEM: Utterances have different lengths, hence there is no trivial way of computing a global distortion between two utterances by accumulating local (frame level) distances

- SOLUTION: DYNAMIC TIME WARPING

  Distance along a predefined path between 2 utterances

  = SUM of FRAME DISTANCEs along that alignment path

  Distance between 2 utterances

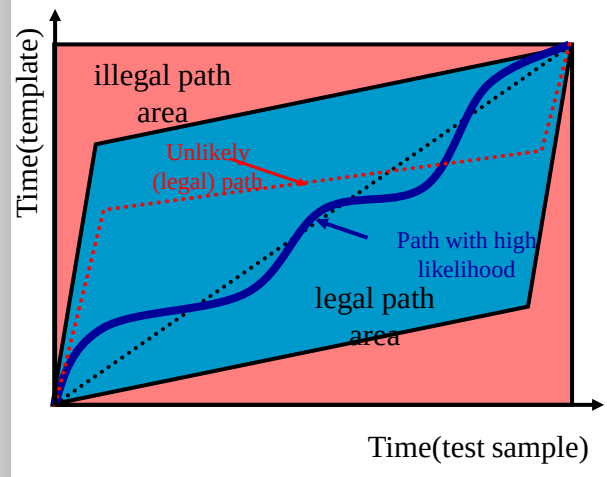
  = SMALLEST of all possible distances along any path

  = SUM of FRAME DISTANCEs along the optimal path

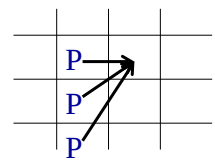
DTW - Issues

- Computational efficiency
  - the number of possible paths growths exponentially with the utterance length
  - enumerating all possible paths is too elaborate
  - efficient computation is possible by dynamic programming principle
- Beyond DP efficiency
  - not all mathematically possible alignments make sense: speech sounds can be made shorter or longer, but not arbitrarily so
  - Additional constraints to the possible paths can be added
    - a path should be more or less diagonal
    - slopes of alignments are expected to be within 1:2 and 2:1
  - Details are provided by the 'path constraints"
    - say with which other cells (i-k,j-l) can be connected to cell(i,j)
    - say if there is an extra cost associated with certain connections
    - extra 'path costs' can be either fixed constants or in function of local distance

DTW - Path Constraints



- Implementing a 2:1 constraints with local constraints (Itakura constraints)
- D(i,j) = min{ D(i-1,j-p)+d(j-p,j) }
  - p=0,1,2
  - d(j-p,j) = a(p)*[d(i,j)+b(p)]
  - eg.
    - a(1)=1; b(1)=0
    - a(0)=a(2)=1.2; b()=0
- possible "predecessors" from perspective on newly formed cell



Dynamic Programming - Example

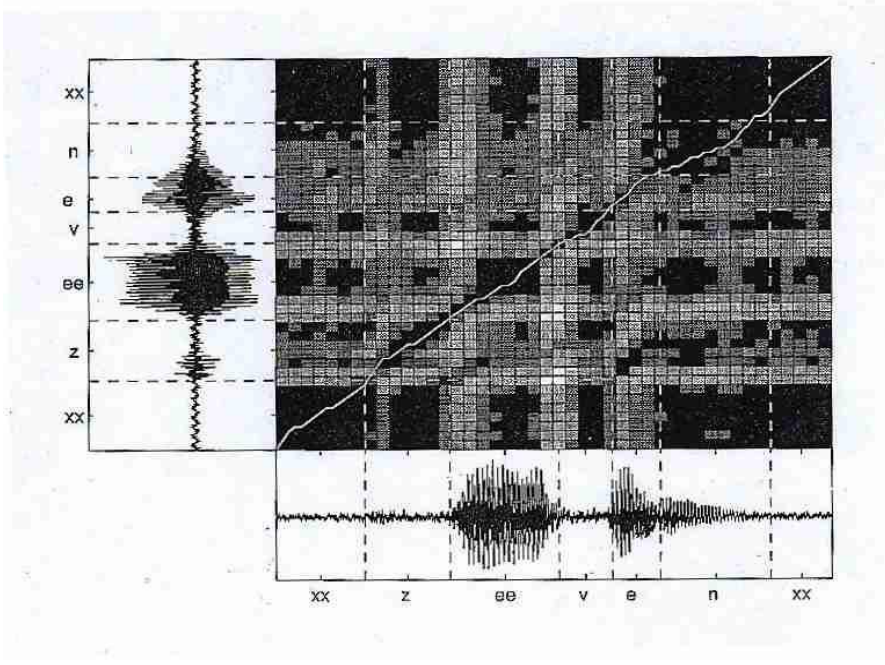
- Local Distance Matrix:
  - $d(x_i,y_j)$

y6	2	3	6	3	2
y5	7	2	1	2	3
y4	8	3	0	3	4
y3	7	2	1	2	3
y2	5	0	3	0	1
y1	1	4	7	4	3
	x1	x2	x3	x4	x5

- Accumulated Distance Matrix
  - $D(X_{1:i},Y_{1:j})$
- no path constraints applied
- Alignment Path via 'backtracking'

6	30	11	9	6	6
5	28	8	3	4	7
4	21	6	2	5	8
3	13	3	2	4	7
2	6	1	4	4	5
1	1	5	12	16	19
	1	2	3	4	5

Example: Local Distances + Optimal Path



Word Recognition System by Template Matching

1. TRAINING PHASE:
   - Select the vocabulary for the task
   - Record an example utterance of each utterance in the vocabulary
   - Compute feature vectors for each of the recordings and store the feature vectors as templates
2. RECOGNITION PHASE:
   - Select the ‘active’ templates
   - Receive an incoming voice command
   - Signal process the data to create a stream of feature vectors
   - For each of the active words in the vocabulary
     - find the best possible time alignment between the utterance and the template
     - obtain a distortion measure between template and incoming utterance by accumulating local distortion scores along the optimal alignment path
   - Select the utterance with minimal distortion as ‘recognized’

Template Matching PROS and CONS

- PRO
  - EASY and ACCURATE
  - implicit modeling of intra-word co-articulation
  - language independent
- CON
  - typically based on one or a few examples only
  - difficult to extend to continuous speech
  - difficult to extend to speaker independence
  - difficult to extend to phoneme based recognition
  - for limited vocab’s only
  - explicit training required

Applications for Speaker Dependent Isolated Word Recognition

- Name dialing
  - hands-free operation of cell phone
  - speed dialing in PABX or as central office service
  - voice organizers

- Basic command and control

6 Evaluation of hypotheses

- How good are results of learning process ?
- Some statistics needed
- Evaluating a single hypothesis
- Comparing two hypotheses
- Comparing two learning algorithms
- ROC analysis

A Bit of Statistics

- Evaluation of predictive model often based on *predictive accuracy* : probability that hypothesis makes correct prediction for random instance from population
  - *acc(h) = P(h(X) = c(X)) = 1 - error(h)*
- Estimating this accuracy = standard statistics
- Recall:
  - binomial & normal distribution
  - confidence intervals & hypothesis tests

Binomial distribution

- An experiment that succeeds with probability p is repeated n times; what is the probability of having x successes?

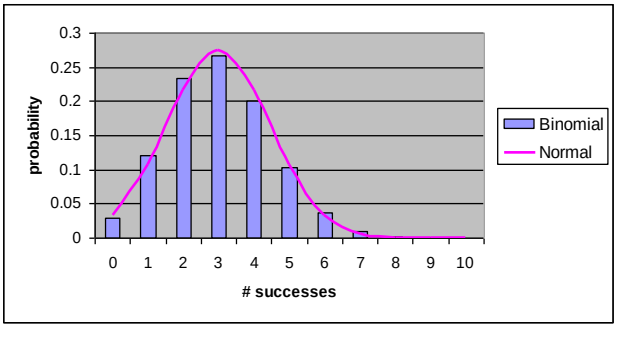
$$P(x) = \binom{n}{x} p^x (1-p)^{n-x} \qquad \binom{n}{x} = \frac{n!}{x!(n-x)!}$$

  - assumptions: constant p, independent experiments
  - Given h with accuracy *p*, *n* instances in some test set, gives probability of making *x* correct predictions in this test set

Normal distribution

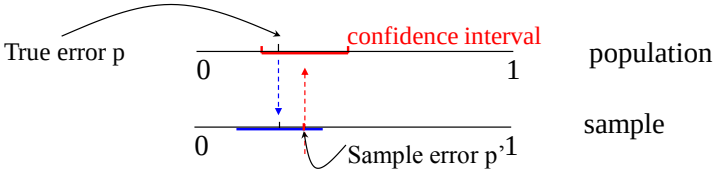
- Binomial distribution can be approximated by *normal distribution*
- Formulae used in practice are derived from this approximation

Example:
n=10, p=0.3



Confidence intervals

- From p and n, we can compute an interval in which x (or x/n) lies with probability close to 1
  - With binomial / normal distribution
- In practice we want to do the opposite
  - given p' = x/n, give an interval for p that contains p' with probability close to 1
  - interval that contains p with probability c% is called a *c% confidence interval*



Hypothesis tests

- Principle of a hypothesis test:
  - given a certain claim (hypothesis) H_0, test it by looking at a sample
  - if sample gives a result that is very unlikely if H_0 were true, reject H_0
- E.g.:
  - claim: h predicts correctly in 90% of cases (H_0: p=0.9)
  - test on data set: p' = 0.8
  - If this is "abnormally" low, reject the claim
    - "abnormal": confidence interval around p' does not contain p

Evaluating a single hypothesis

- To estimate true accuracy of hypothesis h:
  - compute accuracy of h on sample of unseen data
  - compute e.g. 95% confidence interval
- Formula for 100α% confidence interval:
  - derived from normal distribution
  - p' is accuracy on sample, n is size of sample

$$p' \pm z_\alpha \sqrt{\frac{p'(1-p')}{n}}$$

α	z_α
0.90	1.64
0.95	1.96
0.99	2.58

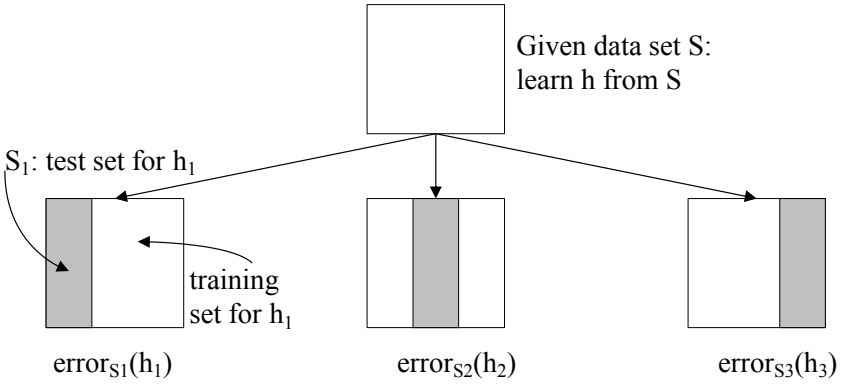
The importance of test sets

- Important: theory assumes *random*, *independent* sample
- Hypothesis h is *not* independent from the training set used to learn it!
  - if we denote
    - $error_{Tr}(h)$: error of h on training set
    - error(h): true error of h on population
    - $error_{Te}(h)$: error of h on sample different from training set
  - then typically: (E() denotes expected value)
    - $E(error_{Tr}(h)) < error(h)$ (cf. overfitting)
    - $E(error_{Tr}(h))-error(h)$ = *bias* of estimator $error_{Tr}$
    - $E(error_{Te}(h)) = error(h)$ (unbiased estimator)

Creating test sets

- How to obtain an independent test set?
  - Simple method: use e.g. 2/3 of available data for training, 1/3 for testing
  - Problem if not much data available
    - smaller training set makes it more difficult to learn
    - smaller test set gives less accurate estimates
  - Popular solution: *cross-validation*
    - learn h from full set S
    - partition S in n subsets; learn n hypotheses h_i, each time leaving out a different subset; use average of test set accuracies of h_i as estimate of accuracy of h

Example: 3-fold cross-validation



$$error_S(h) = \sum_{i=1}^{f} \frac{|S_i|}{|S|} error_{S_i}(h_i)$$

- Crossvalidation is not entirely unbiased - usually small *pessimistic* bias
  - S contains more elements than $S-S_i$
  - different folds not entirely independent
    - Distribution of training set consistently complementary to that of test set
    - E.g., try estimating accuracy of “majority class” classifier using 10-fold crossvalidation with data set consisting of 5+ and 5-
- Still preferable over using training set accuracy

Comparison of hypotheses

- Given two hypotheses, which one has lower true error?
- Statistical hypothesis test:
  - claim that both are equally good
  - if claim rejected, accept that 1 is better
- 2 cases:
  - compare 2 hypotheses on possibly different test sets
  - compare 2 hypotheses on same test set

Comparing 2 hypotheses

- To compare h_1 and h_2, estimate p_1-p_2 from samples S_1 (p'_1) and S_2 (p'_2)
  - if "very likely" p_1-$p_2 > 0$ (i.e., confidence interval is entirely to the right of 0): h_1 is better
  - similarly, < 0 : h_2 is better
  - otherwise, no difference demonstrated
- Formula for confidence interval of difference:

$$p_1' - p_2' \pm z_\alpha \sqrt{\frac{p_1'(1-p_1')}{n_1} + \frac{p_2'(1-p_2')}{n_2}}$$

Comparing 2 hypotheses on the same data set

- When comparing hypotheses on the same data set, *more powerful* procedure possible
  - uses more information from test
  - possible influence of easy/difficult examples removed
- More informative method:
  - for each single example, compare h_1 and h_2
  - how often was h_1 correct and h_2 wrong on the same example, vs. the other way around?
  - use *McNemar's* test

McNemar's test

- Consider table:

	h1 correct	h1 wrong
h2 correct	A	B
h2 wrong	C	D

- If h_1 is equally good as h_2:
  - for each instance where h_1 and h_2 differ, probability 0.5 that either is correct
  - hence we expect $B \approx C \approx (B+C)/2$
  - B and C follow binomial (+/- normal) distribution
- reject equality if B deviates too much from (B+C)/2

Example comparison

- Consider table below
- Method with independent test sets:
  - 55-45 in favour of h_2 (out of 100)
  - not very convincing
- Method with same test set:
  - much more convincing: 10-0 in favour of h_2

	h1 correct	h1 wrong
h2 correct	45	10
h2 wrong	0	45

- h2 clearly better than h1
- might not be discovered using "conservative" comparison

Comparing learning algorithms

- Compare these two questions:
  - Q1: given hypotheses h_1 and h_2, which one has better predictive accuracy?
  - Q2: given learners L_1 and L_2 and data set S, which learner can be expected to build best hypothesis from S?
    - note that hypotheses themselves may vary
    - more difficult to answer than Q1

- One possible method:
  - For several data sets S_i similar to S:
    - split S_i into training set S_{tr} and test set S_{te}
    - learn h_1 and h_2 from S_{tr} using L_1 resp. L_2
    - compute $\delta_i = error_{S_{te}}(h_1) - error_{S_{te}}(h_2)$
- Hypothesis test / confidence interval for mean of δ
- What if limited set of data is available?

- If limited data available:
  - repeated runs within 1 data set?
    - e.g. cross-validation: n splits into S_{tr} and S_{te}
    - e.g. 30 random splits into S_{tr} and S_{te}
  - problem: dependencies between data sets
    - may make it *very easy* to draw incorrect conclusions!
      - high probability of "type 1" error: concluding one learner is better than other when this is not the case
    - to be avoided
  - reasonable approach: 5 2-fold cross-validations
    - details: T. Dietterich, Neural computation 10(7), 1998
  - only really good solution: collect more data!

ROC analysis

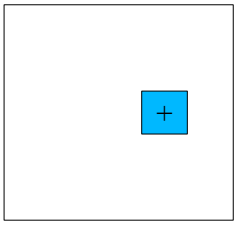
- Evaluation based on accuracy is not always appropriate
- Shortcomings:
  - can sometimes be misleading
  - unstable when class distribution may change
  - assumes symmetric misclassification costs
- Alternatives:
  - correlation
  - ROC analysis

1: Accuracy can be misleading

- E.g., "99% correct prediction": is this good?
  - Yes, if 50% "+" and 50% "-"
  - No, if 1% "+" and 99% "-"
    - always predicting "neg" gives 99% accuracy
- Accuracy is a *relative* measure
  - Should be compared with "base accuracy" of always predicting the majority class
  - base accuracy in table = max{a+c,b+d} / T
- Even then, it may be misleading...

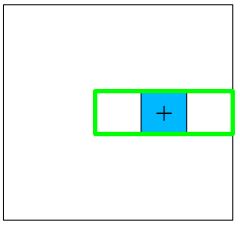
- Assume all examples -, except blue region (+)
- Which of these classifiers is best?

Classifier 1



IF false THEN pos
96% correct

Classifier 2



IF green area THEN pos:
92% correct

- An alternative measure: correlation
  - e.g., correlation ϕ = (ad-bc) / $\sqrt{T_{pos} T_{neg} T_{+} T_{-}}$
    - close to 1: high correlation predictions - classes
    - close to 0: no correlation
    - (close to -1: predicting the opposite)
  - Avoids the unintuitive results just mentioned

actual value

prediction	+	-	Sum
Pos	a	b	T_{pos}
Neg	c	d	T_{neg}
Sum	T_{+}	T_{-}	T

note:
+/- are actual values
pos/neg are predictions

2: Accuracy ignores different misclassification costs

- Accuracy ignores possibility of *different misclassification costs*
  - sometimes, incorrectly predicting "pos" costs more/less than incorrectly predicting "neg"
  - e.g.:
    - not treating an ill patient vs. treating a healthy patient
    - refusing credit to client who would have paid back vs. assigning credit to client who won't pay back
- Need to distinguish probability of making different types of errors

- Solution: distinguish “predictive accuracy” for different classes
  - Acc: probability that *some* instance is classified correctly
  - Decomposed into
    - *TP*: “true positive” rate, (estimated) probability that a *positive* instance is classified correctly
    - *TN*: “true negative” rate, (estimated) probability that a *negative* instance is classified correctly
  - We also define
    - *FP = 1-TN*: “false positive rate”: estimated probability that a negative is classified as positive
    - analogously *FN = 1-TP*

- Consider costs C_{FP} and C_{FN}
  - cost of false positive resp. false negative
- Expected cost of a single prediction:
  - $C = C_{FP}\ P(pos|-)\ P(-) + C_{FN}\ P(neg|+)\ P(+)$
  - estimated by $C = C_{FP}\ FP\ T_-/T + C_{FN}\ FN\ T_+/T$
- Note :
  - Acc is weighted average of TP and TN
    - $Acc = TP\ T_+/T + TN\ T_-/T$
  - C is not computable from Acc alone

3: Accuracy is sensitive to changing class distributions

- If class distribution in test set differs from that in training set, accuracy will also differ
- E.g.:
  - Suppose a classifier has TP = 0.8, TN = 0.6
  - Tested on test set with $T_+/T = 0.5$, $T_-/T = 0.5$:
    - Acc = 0.7
  - Employed in environment with $T_+/T = 0.3$, $T_-/T = 0.7$:
    - Acc = 0.66

ROC diagrams

- ROC = "Receiver operating characteristic"
- Allows to see
  - how well a classifier will perform given certain misclassification costs and class distribution
  - in which environments one classifier is better than another
- Explicitly aims at solving problems 2 and 3 mentioned before
- Ref: Provost & Fawcett (2001), Robust classification under imprecise environments, *Machine Learning* 42(3):203-231.

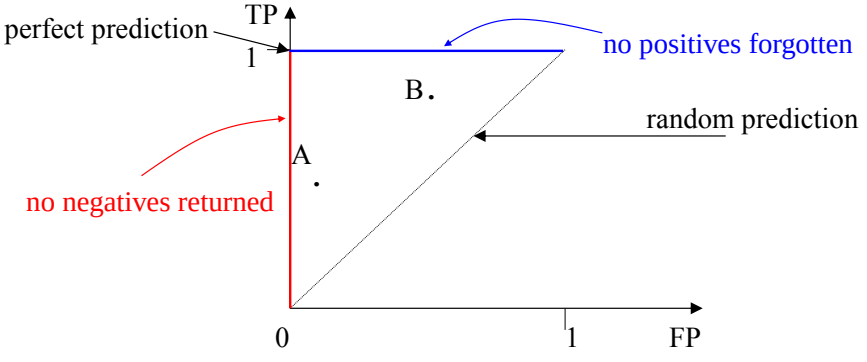
- ROC diagram plots *TP versus FP*
- From confusion matrix:
  - *TP = a/(a+c) = a/T_+*
  - *FP = b/(b+d) = b/T_-*

actual value

prediction	+	-	Sum
Pos	a	b	T_{pos}
Neg	c	d	T_{neg}
Sum	T_+	T_-	T

Classifier in ROC diagram

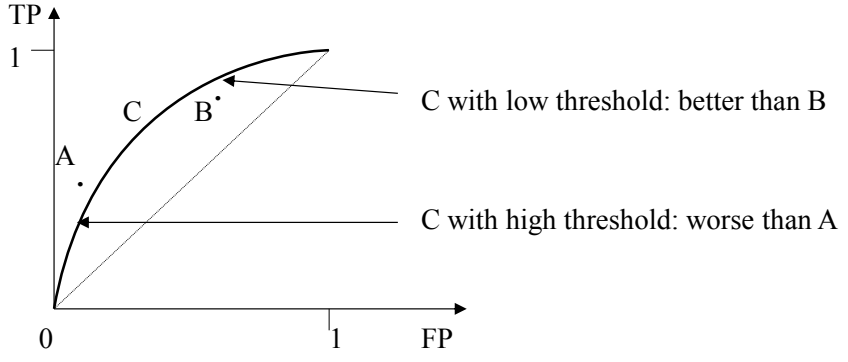
- 1 classifier = 1 point on ROC diagram
- The closer to the upper left, the better



Rank classifiers

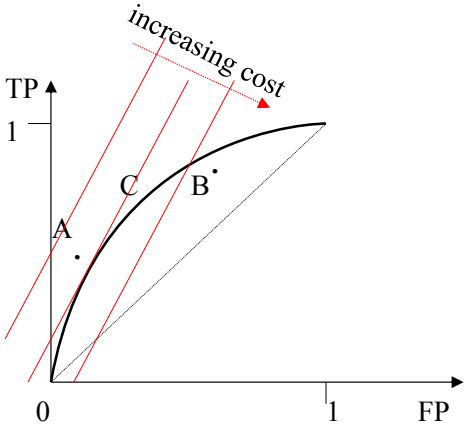
- Rank classifiers assign a rank to their predictions
  - some predictions are more certain than others -> higher rank
- e.g.: neural nets:
  - criterion: <0.5 = neg, >=0.5 = pos
  - but 0.9 is more certainly positive than 0.51
  - raise/lower threshold of 0.5 : effect?
    - Higher threshold means fewer positives: TP and FP go down
- e.g.: decision trees:
  - use purity of leaf used for prediction to rank it
  - E.g. leaf with 90% positives is ranked higher than leaf with 80% positives

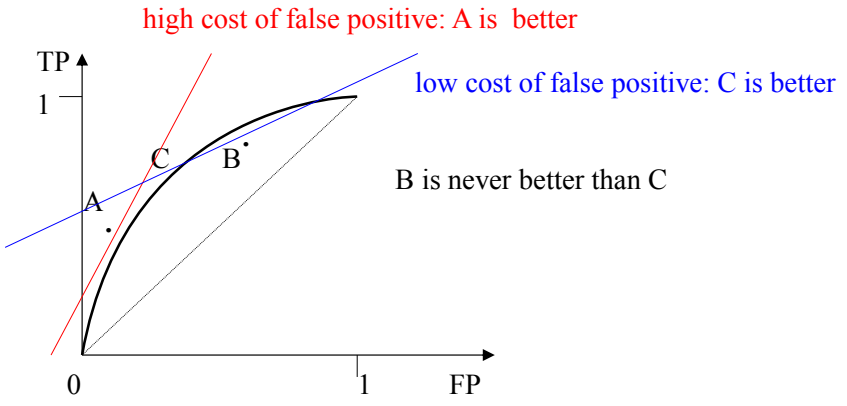
- Rank classifiers yield an *ROC curve*
- With a specific threshold: 1 point on that curve



Costs in ROC diagram

- Given misclassification costs:
  - c_{FP}: cost of a false positive
  - c_{FN}: cost of a false negative (undetected "+")
- Average cost is
  - $c = c_{FP} * FP * T_-/T + c_{FN} * (1-TP) * T_+/T$
  - *Lines of equal cost* can be drawn in ROC diagram (straight lines)
    - Slope of such a line : $(c_{FP} * T_-/T) / (c_{FN} * T_+/T)$



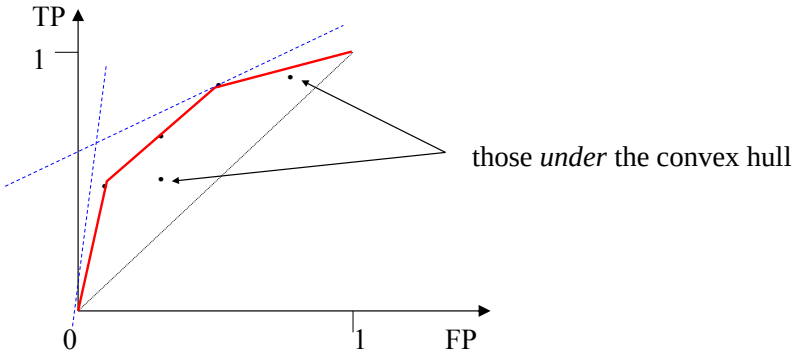


Sets of classifiers

- Different classifiers may be good in different environments
  - Different misclassification costs
  - Different class distributions
- Given a set of classifiers, ROC analysis allows to
  - decide in which cases a classifier is optimal
  - remove classifiers that are never optimal
- Classifiers that may be optimal are always on *convex hull* of set of points

Example: convex hull

- Which classifiers are never optimal?



Evaluation of regression models

- Predicting numbers: no "right or wrong" approach
- Possible measures:
  - Sum of squared errors SSE
    - Is an absolute measure
  - Relative error RE: measures improvement over trivial model
    - RE = SSE(hypothesis) / SSE(trivial hypothesis)
    - Trivial hypothesis: e.g. always predict mean
    - RE normally between 0 and 1
  - Spearman correlation r
    - measures how well predictions and actual values correlate
    - less sensitive to actual errors

To remember

- Accuracy based evaluation:
  - methods for evaluating a single hypothesis, comparing hypotheses, comparing algorithms
  - limitations of accuracy as evaluation criterion
- Other evaluation criteria
  - criteria for regression
  - correlation as alternative for accuracy
  - ROC analysis: plotting classifiers and rank classifiers, iso-cost lines, convex hull

Numerical Approaches

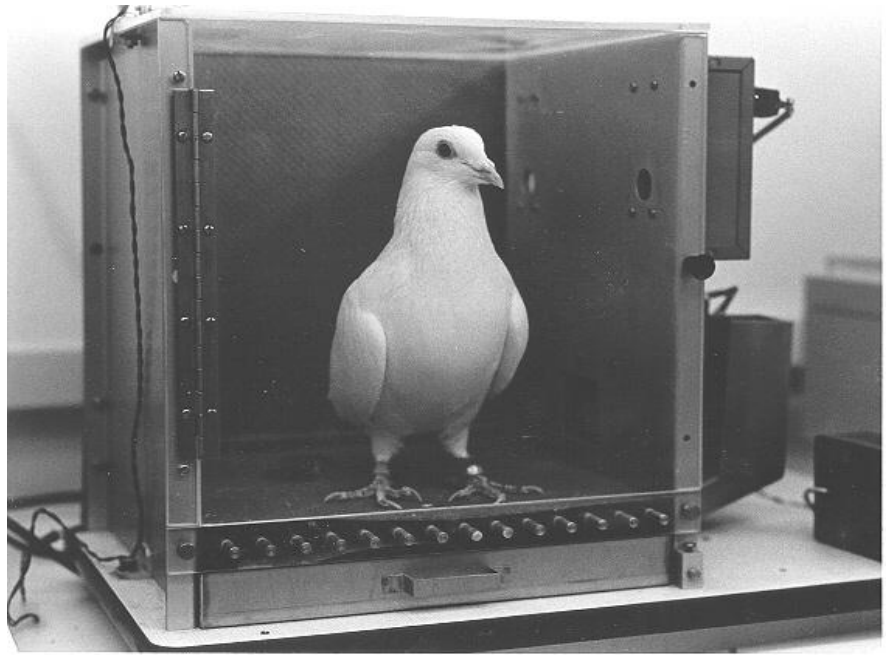
- Artificial Neural Networks
- Support Vector Machines

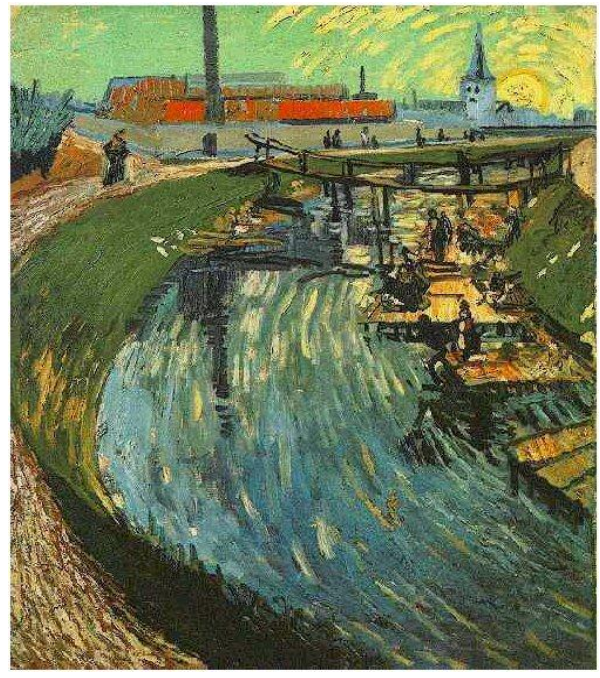
Using numerical approaches

- Numerical approaches can be used after encoding data as numbers
- Advantages of this approach:
  - Powerful modelling methods available
- Disadvantages
  - Results (predictive or descriptive model) may be more difficult to interpret (translating model to symbolic form is not always possible)

10. Artificial neural networks

- Modelled after biological neural systems
- Biological Neural Nets
  - Pigeons as art experts (Watanabe *et al.* 1995)
  - Experiment:
    - Pigeon in a box
    - Present paintings of two different artists (e.g. Chagall / Van Gogh)
    - Reward for pecking when presented a particular artist (e.g. Van Gogh)





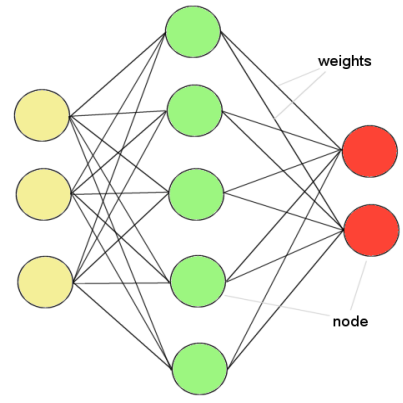


- Pigeons were able to discriminate between Van Gogh and Chagall with 95% accuracy (when presented with pictures they had been trained on)
- Discrimination still 85% successful for previously unseen paintings of the artists
- Pigeons do not simply memorise the pictures
- They can extract and recognise patterns (the 'style')
- They generalise from the already seen to make predictions
- This is what neural networks (biological and artificial) are good at (unlike conventional computer)

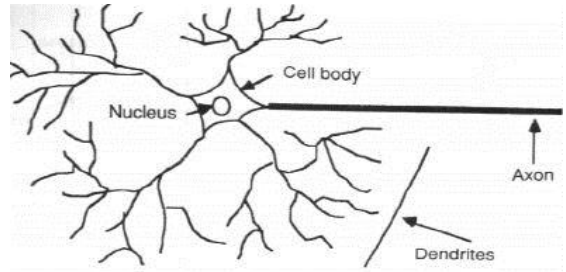
ANNs – The basics

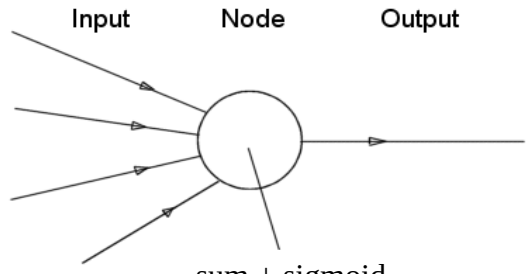
- ANNs incorporate the two fundamental components of biological neural nets:

1. Neurones (nodes)
2. Synapses (weights)



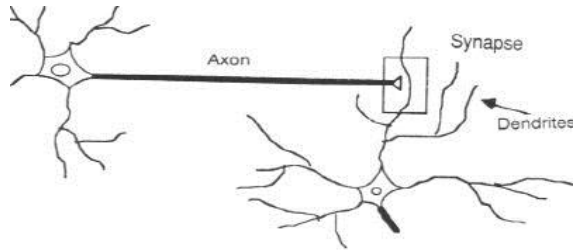
- Neuron vs. Node

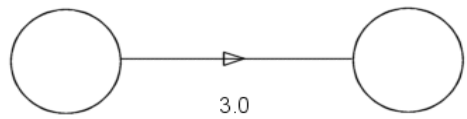




sum + sigmoid

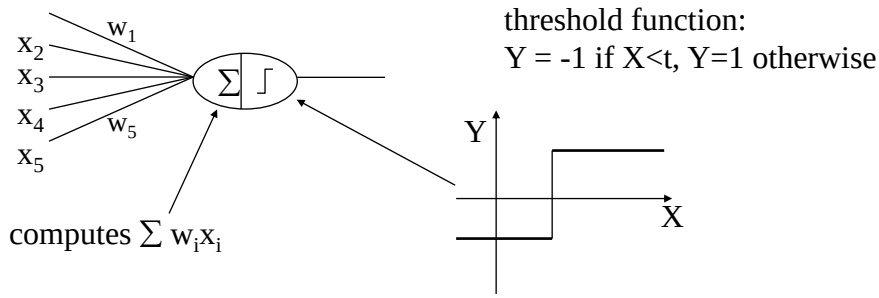
- Synapse vs. weight





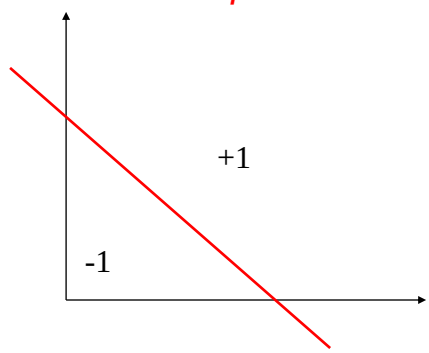
Perceptrons

- Simplest type of neural network
  - Perceptron simulates 1 neuron
  - Fires if sum of (inputs * weights) > some threshold
- Schematically:



2-input perceptron

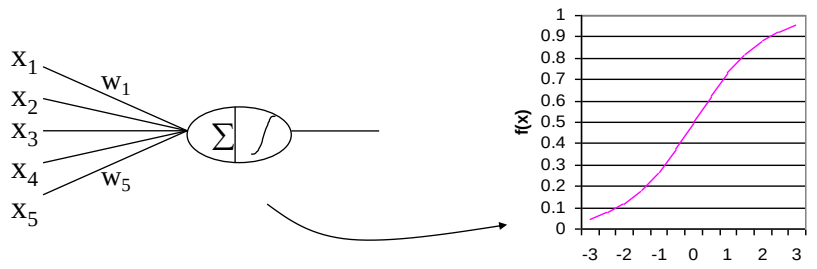
- represent inputs in 2-D space
- perceptron learns a function of following form:
  - if aX + bY > c then +1, else -1
  - i.e., creates *linear separation* between classes + and -



n-input perceptrons

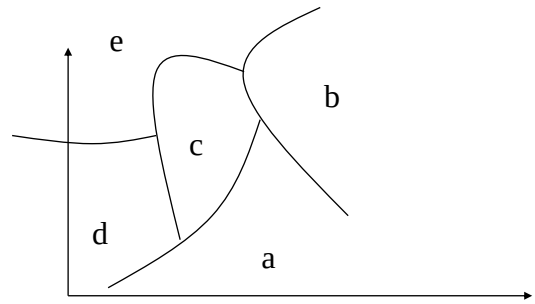
- In general, perceptrons construct a hyperplane in an n-dimensional space
  - one side of hyperplane = +, other side = -
- Hence, classes must be *linearly separable,* otherwise perceptron cannot learn them
- E.g.: learning boolean functions
  - encode true/false as +1, -1
  - is there a perceptron that encodes 1. A and B? 2. A or B? 3. A xor B?

- "Sigmoid" function instead of crisp threshold
  - changes continuously instead of in 1 step
  - has advantages for training multi-layer networks



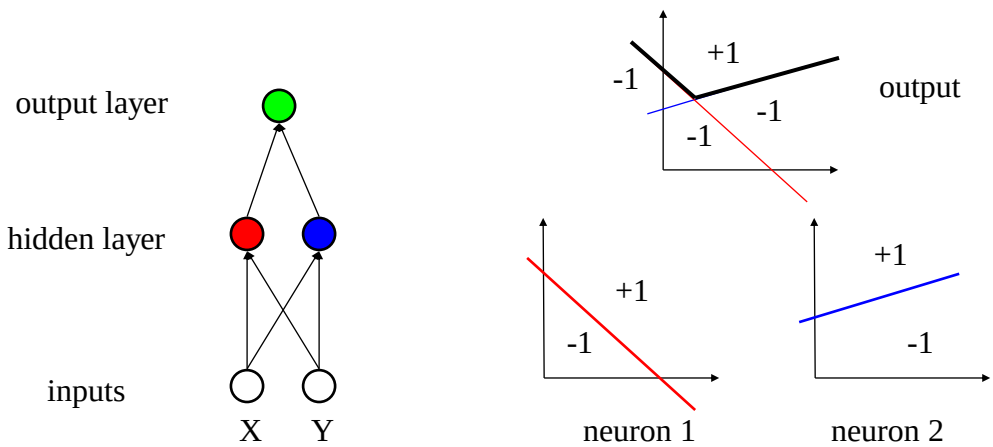
$$f(x) = \frac{1}{1+e^{-x}}$$

- Non-linear sigmoid function causes non-linear decision surfaces
  - e.g., 5 areas for 5 classes a,b,c,d,e
- Very powerful representation



Multi-layer networks

- Increase representation power by combining neurons in a network



- Note : previous network had 2 *layers* of neurons
- Layered feedforward neural networks:
  - each layer has output from previous layer as input
    - neurons in successive layers fully interconnected
  - successive layers = *different representations* of input
    - Remember the importance of representations
- 2-layer feedforward networks very popular…

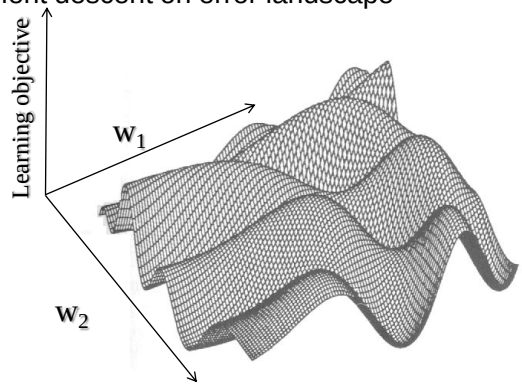
- Weight settings determine the behaviour of a network

  → How can we find the right weights?

Training the Network - Learning

- Backpropagation
  - Requires training set (input / output pairs)
  - Starts with small random weights
  - Error is used to adjust weights (supervised learning)

  → Gradient descent on error landscape



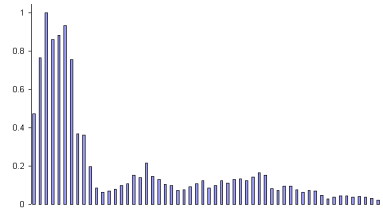
- **Advantages**
  - It works!
  - Relatively fast

- **Downsides**
  - Requires a training set
  - Can be slow
  - Probably not biologically realistic

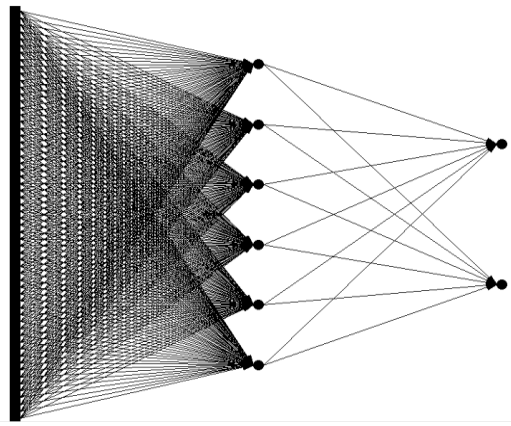
Example: Voice Recognition

- Task: Learn to discriminate between two different voices saying “Hello”

- Data
  - Sources
    - Steve Simpson
    - David Raubenheimer
  - Format
    - Frequency distribution (60 bins)
    - Analogy: cochlea

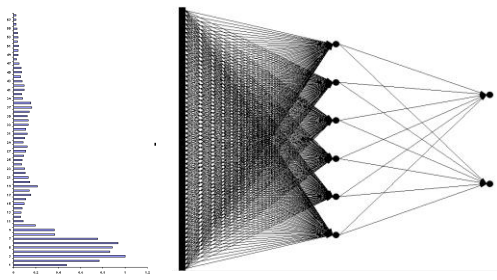


- Network architecture
  - Feed forward network
    - 60 input (one for each frequency bin)
    - 6 hidden
    - 2 output (0-1 for “Steve”, 1-0 for “David”)

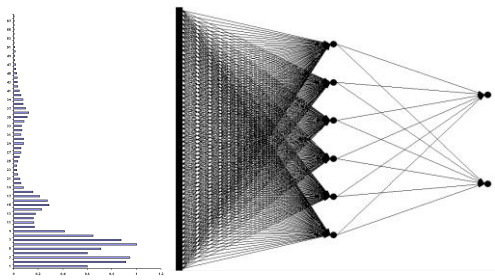


- Presenting the data

Steve

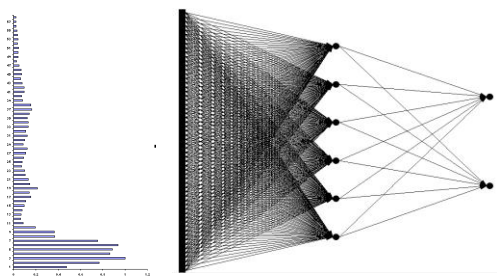


David

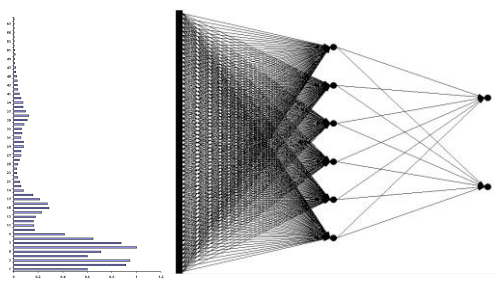


- Presenting the data (untrained network)

Steve

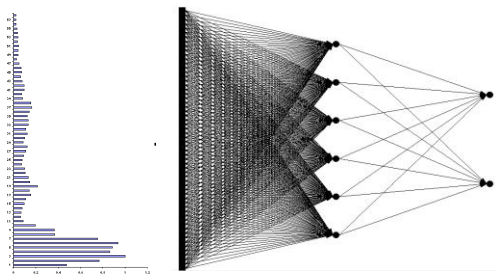


David



- Calculate error

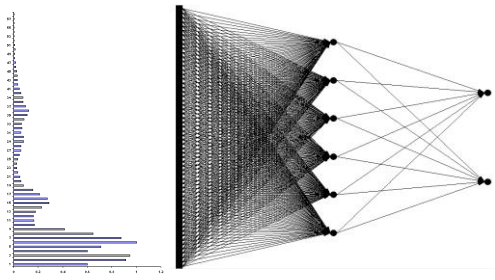
Steve



0.43 – 0 = 0.43

0.26 –1 = 0.74

David

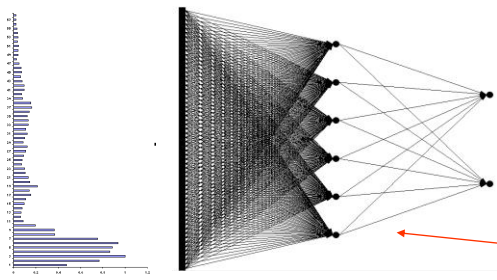


0.73 – 1 = 0.27

0.55 – 0 = 0.55

- Backprop error and adjust weights

Steve

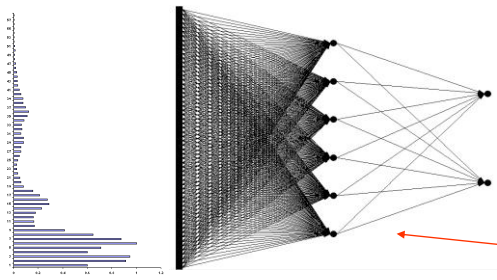


| 0.43 – 0 | = 0.43

| 0.26 – 1 | = 0.74

1.17

David



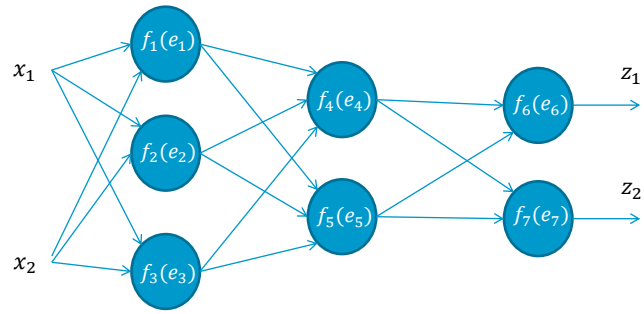
| 0.73 – 1 | = 0.27

| 0.55 – 0 | = 0.55

0.82

- Repeat process (sweep) for all training pairs
  - Present data
  - Calculate error
  - Backpropagate error
  - Adjust weights

- Notations



- $\underline{x}_i \rightarrow$ input vector for node i ($\underline{x}_{ij}$ is j'th element of $\underline{x}_i$)
- $\underline{w}_i \rightarrow$ weight vector for node i ($\underline{w}_{ij}$ is j'th element of $\underline{w}_i$)
- $e_i = \underline{w}_i^T \underline{x}_i \rightarrow$ weighted sum of inputs for node i
- $f_i(e_i) \rightarrow$ output of node i
- $z_i \rightarrow$ target for node i
- $\mathfrak{S}_i \rightarrow$ set of nodes whose immediate inputs include the output of node i
- $\mathfrak{O} \rightarrow$ set of output nodes

- Note that only one example is used to update the weights in the network. So we now assume a training sample consisting out of a single example
- The error of the network is then expressed as

$$J = \frac{1}{2}\sum_{i \in \mathfrak{O}} \left(f_i(e_i) - z_i\right)^2$$

- The incremental stochastic gradient descent based update formula for the weights in iteration k is defined as

$$\underline{w}_{ij}^{(k)} = \underline{w}_{ij}^{(k-1)} - \eta \frac{\partial J}{\partial \underline{w}_{ij}}$$

- The gradient can split-up as follows

$$\frac{\partial J}{\partial \underline{w}_{ij}} = \frac{\partial J}{\partial e_i}\frac{\partial e_i}{\partial \underline{w}_{ij}}$$

- The second factor gives

$$\frac{\partial e_i}{\partial \underline{w}_{ij}} = \underline{x}_{ij}$$

- The first factor is independent of the input j for a given node i so it is denoted by δ_i.

- Furthermore, each individual $\underline{w}_{ij}$ corresponding to an output node ($i \in \mathfrak{O}$) is only depending on a single output node. Therefore the sum in J can be dropped. This gives

$$\delta_i = \frac{\partial J}{\partial e_i} = \frac{1}{2}\frac{\partial \left(f_i(e_i) - z_i\right)^2}{\partial e_i}$$
$$= -\left(f_i(e_i) - z_i\right)\frac{\partial f_i(e_i)}{\partial e_i}$$

- Note that in case $f(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function , $\frac{\partial f(x)}{\partial x} = \left(1 - f(x)\right)f(x)$
- For weights associated to a hidden node i further calculation is required

$$\frac{\partial J}{\partial \underline{w}_{ij}} = \sum_{k \in \mathfrak{S}_i} \frac{\partial J}{\partial e_k}\frac{\partial e_k}{\partial f_i(e_i)}\frac{\partial f_i(e_i)}{\partial e_i}\frac{\partial e_i}{\partial \underline{w}_{ij}}$$

- Using $\frac{\partial J}{\partial e_k} = \delta_k, \frac{\partial e_k}{\partial f_i(e_i)} = \underline{w}_{ij}$ one can write

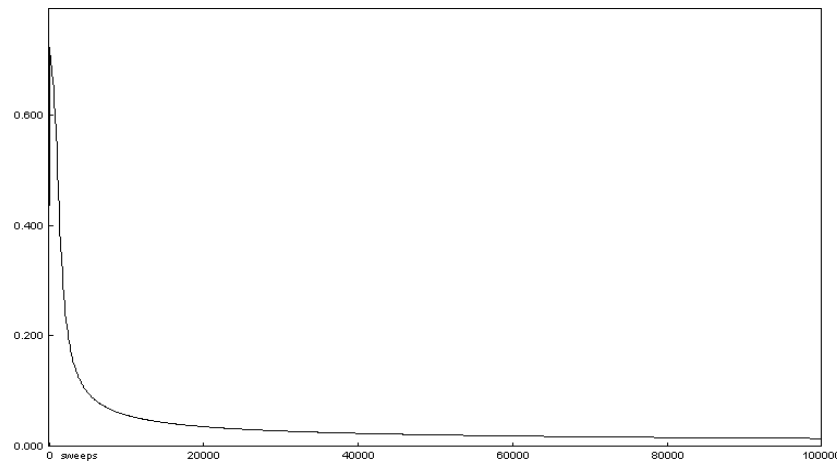
$$\frac{\partial J}{\partial \underline{w}_{ij}} = \frac{\partial f_i(e_i)}{\partial e_i}\sum_{k \in \mathfrak{S}_i} \delta_k \underline{w}_{ij}\underline{x}_{ij}$$

- This finally results in the following backpropagation algorithm:

Assume a feed forward network with an input, hidden and output layer. Initialize the weights to small random values (e.g. between -0.05 and 0.05).

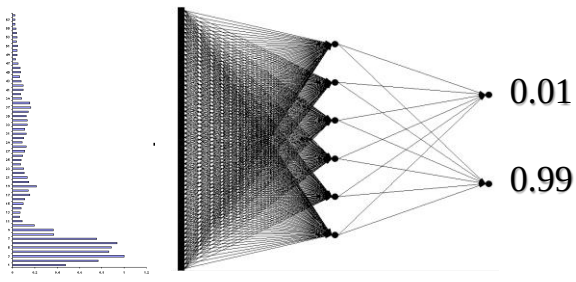
Until stopping criterion is met do
  For each training example $x \in \mathbb{R}^D$, D the number of inputs, with corresponding label z do
    1. Compute $f_i(e_i)$ with $i \in \mathfrak{O}$.
    2. Compute δ_i with $i \in \mathfrak{O}$.
    3. For all other nodes $i \notin \mathfrak{O}$ compute $\delta_i = \frac{\partial f_i(e_i)}{\partial e_i} \sum_{k \in \mathfrak{G}_i} w_{ki} \delta_k$.
    4. Update each weight $\underline{w}_{ij}^{(k)} = \underline{w}_{ij}^{(k-1)} + \eta \delta_i \underline{x}_{ij}$.
  end
end

- Repeat process multiple times

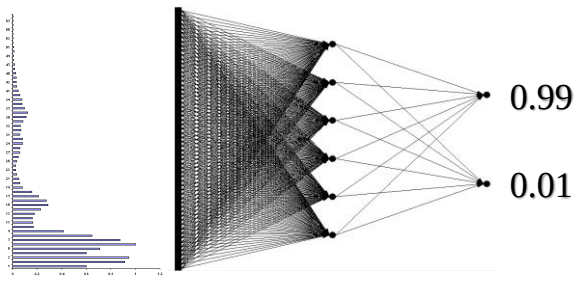


- Presenting the data (trained network)

Steve



David

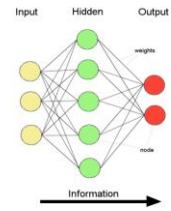


- Results – Voice Recognition
  - Performance of trained network
    - Discrimination accuracy between known “Hello”s
      - 100%
    - Discrimination accuracy between new “Hello”’s
      - 100%

- Results – Voice Recognition (ctnd.)
  - Network has learnt to generalise from original data
  - Networks with different weight settings can have same functionality
  - Trained networks ‘concentrate’ on lower frequencies
  - Network is robust against non-functioning nodes

Recurrent Networks

- Feed forward networks:
  - Information only flows one way
  - One input pattern produces one output
  - No sense of time (or memory of previous state)

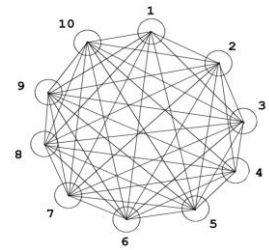


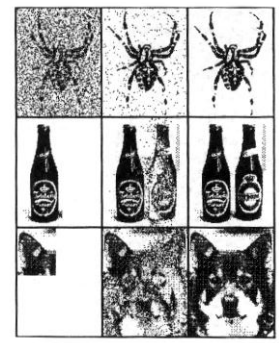
- Recurrency
  - Nodes connect back to other nodes or themselves
  - Information flow is multidirectional
  - Sense of time and memory of previous state(s)
- Biological nervous systems show high levels of recurrency (but feed-forward structures exists too)

Hopfield Networks

- Sub-type of recurrent neural nets
  - Fully recurrent
  - Weights are symmetric ($w_{ij} = w_{ji}$)
  - Nodes can only be *on* or *off*
  - Random updating
- Learning: **Hebb rule** (cells that fire together wire together)
  - Biological equivalent to LTP and LTD
- Can recall a memory, if presented with a corrupt or incomplete version

→ **auto-associative** or **content-addressable memory**





Task: store images with resolution of 20x20 pixels
→ Hopfield net with 400 nodes

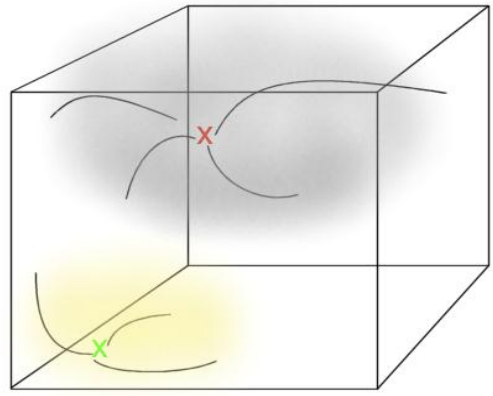
Memorise:

1. Present image
2. Apply Hebb rule (*cells that fire together, wire together*)
   - Increase weight between two nodes if both have same activity, otherwise decrease
3. Go to 1

Recall:

1. Present incomplete pattern
2. Pick random node, update
3. Go to 2 until settled

- Memories are attractors in state space



Properties of neural networks

- Useful for modelling complex, non-linear functions of numerical inputs & outputs
  - symbolic inputs/outputs representable using some encoding, cf. true/false = 1/-1
  - 2 or 3 layer networks can approximate a huge class of functions (if enough neurons in hidden layers)
- Robust to noise
  - but: risk of overfitting! (because of high expressiveness)
    - may happen when training for too long
  - usually handled using e.g. validation sets

- All inputs have some effect
  - cf. decision trees: selection of most important attributes
  - ANN “selects” attributes by giving them higher/lower weights
- Explanatory power of ANNs is limited
  - model represented as weights in network
  - no simple explanation why network makes a certain prediction
    - contrast with e.g. trees: can give a “rule” that was used
  - Networks can not easily be translated into symbolic model (tree, ruleset)

- Hence, ANNs are good when
  - high-dimensional input and output (numeric or symbolic)
  - interpretability of model unimportant
- Examples:
  - typical: image recognition, speech recognition, …
    - e.g. images: one input per pixel
    - see http://www.cs.cmu.edu/~tom/faces.html for illustration
  - less typical: symbolic problems
    - cases where e.g. trees would work too
    - performance of networks and trees then often comparable

11. Support vector machines

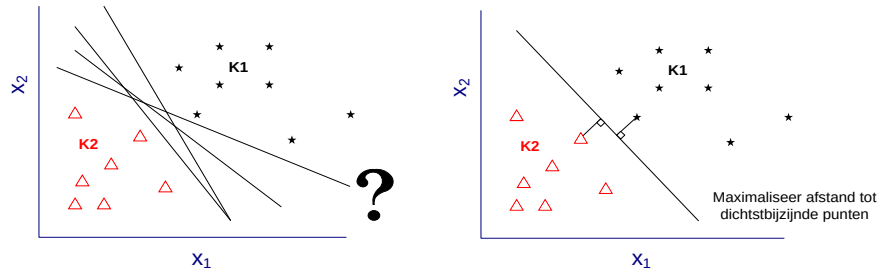
- Linear support vector machines
- Non-linear support vector machines
- Least-squares support vector machines
- Multi-class extension

Support vector machines

- Relatively new approach
- Lot of interest recently:
  - many successes, e.g., for text classification
- Usable in similar situations as neural networks
- Important concepts:
  - Finding a "maximal margin" separation
  - Transformation into high dimensional space
  - Structural risk minimization
- Good generalization even in high dimensional spaces
- Reference: [Vapnik, 1995]

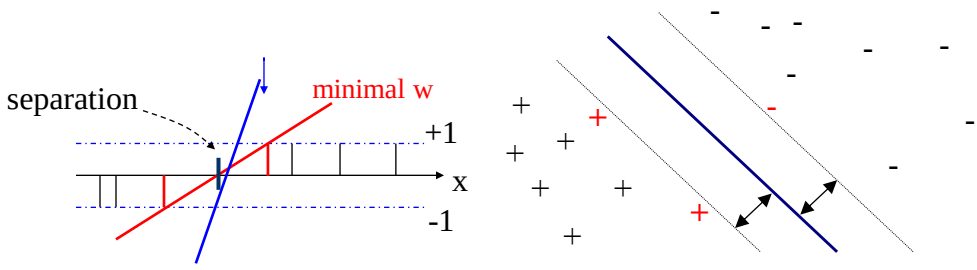
Linear SVMs

- Find hyperplane that discriminates + from –
- Which hyperplane?

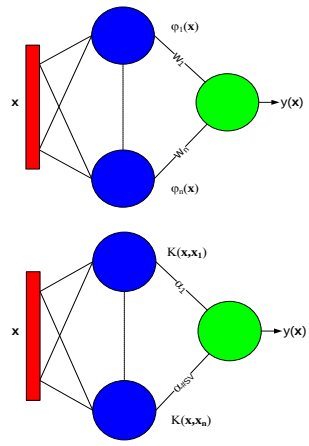


Linear SVMs

- Idea:
  - "margin" should be maximal
    - Margin = distance of hyperplane to closest points
    - Solution is unique, and determined by just a few points ("support vectors")



Primal-dual formulation



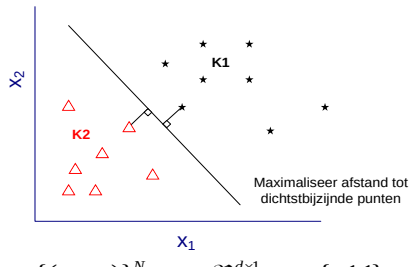
Primal problem

$$y(x) = sign(w^T \varphi(x) + b)$$

Dual problem

$$y(x) = sign[\sum_{k=1}^{\#sv} \alpha_k y_k K(x, x_k) + b]$$

Primal problem: separable classes



$$\{(x_i, y_i)\}_{i=1}^{N}, x_i \in \Re^{d \times 1}, y_i \in \{-1,1\}$$

- **Points closest to hyperplane** $w^T x + b = 0$

$$\min_i \frac{|w^T x_i + b|}{\| w \|_2} \Leftrightarrow \min_i \frac{y_i (w^T x_i + b)}{\| w \|_2}$$

Minimize signed distance

- **Maximize the margin**

$$\max_{w,b} \min_i \frac{y_i (w^T x_i + b)}{\| w \|_2}$$

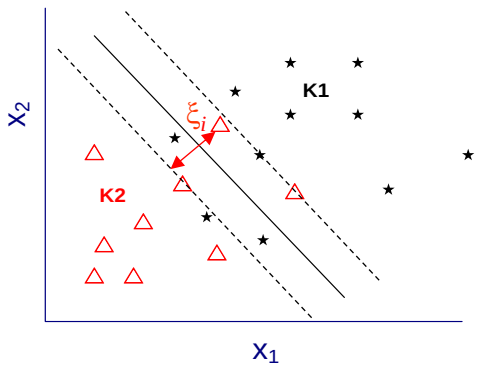
$$\begin{cases} \max_{\rho,w,b} \rho \\ s.t. \forall i, \frac{y_i (w^T x_i + b)}{\| w \|_2} > \rho \Leftrightarrow y_i (w^T x_i + b) > \rho \| w \|_2 \end{cases}$$

Scale to 1, $\rho = \frac{1}{\|w\|_2}$

- **Primal learning problem**

$$\begin{cases} \max_{w,b} \frac{1}{\| w \|_2} \Leftrightarrow \min_{w,b} \| w \|_2 \Leftrightarrow \min_{w,b} \| w \|_2^2 \\ s.t. \forall i, y_i (w^T x_i + b) \geq 1 \end{cases}$$

Primal problem:non-separable classes



- **Introduce slack variables ξ_i in primal learning problem**

$$\begin{cases} \min\limits_{w,b,\xi} \| w \|_2^2 + C\sum\limits_{i=1}^{N} \xi_i \\ s.t. \\ \forall i, y_i(w^T x_i + b) \geq 1 - \xi_i \\ \forall i, \xi_i \geq 0 \end{cases}$$

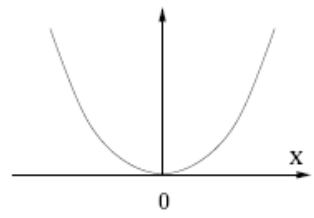
- **Trade-off between maximum margin maximization and tolerating misclassifications**

Intermezzo: constrained optimization

- Example of unconstrained optimization problem

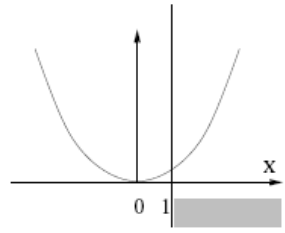
$$\min_x x^2$$

solution x=0



- Example of constrained optimization problem

$$\min_x x^2 \text{s.t.} x \geq 1$$



Possible way to solve this

- Consider Lagrangian

$$\Im(x;\lambda) = x^2 - \lambda(x-1), \lambda \geq 0$$

- For this convex cost function with linear inequality constraint the optimal solution is characterized by the saddle point

$$\max_\lambda \min_x \Im(x;\lambda)$$

- First compute

$$\frac{\partial \Im}{\partial x} = 2x - \lambda = 0 \Rightarrow x = \frac{\lambda}{2}$$

- Substitute this expression back into the Lagrangian and solve the following dual problem (in the Lagrange multiplier)

$$\max_{\lambda}(\frac{\lambda}{2})^2-\lambda(\frac{\lambda}{2}-1)$$

which gives solution

$$\lambda=2\Rightarrow x=1$$

SVM dual problem

- **Obtain solution via Lagrangian**

$$\zeta(w,b,\xi;\alpha,\nu)=\tfrac{1}{2}\|w\|_2^2+C\sum_{i=1}^{N}\xi_i-\sum_{i=1}^{N}\alpha_i(y_i[w^Tx_i+b]-1+\xi_i)-\sum_{i=1}^{N}\nu_i\xi_i \quad \text{with } \alpha_i\geq 0,\nu_i\geq 0$$

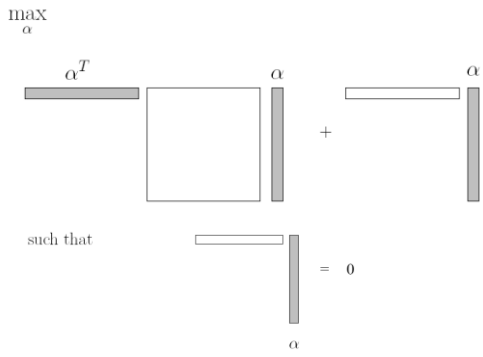
- **Conditions for optimality**

$$\max_{\alpha,\nu}\min_{w,b,\xi}\zeta(w,b,\xi;\alpha,\nu)\quad\begin{cases}\frac{\partial\zeta}{\partial w}=0\longrightarrow w=\sum_{i=1}^{N}\alpha_i y_i x_i\\ \frac{\partial\zeta}{\partial b}=0\longrightarrow\sum_{i=1}^{N}\alpha_i y_i=0\\ \frac{\partial\zeta}{\partial\xi_i}=0\longrightarrow\forall i,0\leq\alpha_i\leq C\end{cases}$$

- **Write solution in terms of α_k, this leads to a convex Quadratic Programming (QP) problem**

$$\begin{cases}\max_{\alpha_k}-\frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}y_k y_l x_k^T x_l\alpha_k\alpha_l+\sum_{k=1}^{N}\alpha_k\\ s.t.\\ \sum_{k=1}^{N}\alpha_k y_k=0\\ 0\leq\alpha_k\leq C,k=1,\ldots,N\end{cases}$$

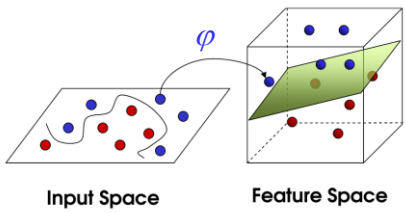
Intermezzo: standard QP problem



Note that:
• the size of the matrix grows with the number of training data points (if N = 10^6 then the size of the matrix is $10^6 \times 10^6$!)
• the size of the solution vector α is only determined by the number of training data N and not by the dimension of the input space

Kernels

- Idea: To have non-linear decision in input space first transform to other space where classes are lineary separable and apply previous techniques
- Kernel defines inner product between the images of the data items
- Can be interpreted as a similarity measure



$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$ ("Kernel trick")

Where $\varphi : \Re^d \rightarrow \Re^{d_\varphi}$ denotes the feature map induced by a positive definite kernel

Example transformations

- E.g.: learning quadratic decision surfaces in 2D:
  - map x,y to space with dimensions x, y, xy, x^2, y^2
  - learn "hyperplane" $ax+by+cxy+dx^2+ey^2+f=0$
    - $(a,b,c,d,e)\ (x,y,xy,x^2,y^2)^T + f = 0$
  - in original (x,y) space this is a quadratic form
- Some examples of kernels:
  - $K(\mathbf{x}_1,\mathbf{x}_2) = (\mathbf{x}_1\mathbf{x}_2+1)^p$ polynomial decision surfaces
  - $K(\mathbf{x}_1,\mathbf{x}_2) = \exp(-||\mathbf{x}_1-\mathbf{x}_2||^2/2\sigma^2)$ RBFs
  - …
- Some nice SVM demo's on the Web
  - http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml

Non-linear SVM

- **Primal problem:**

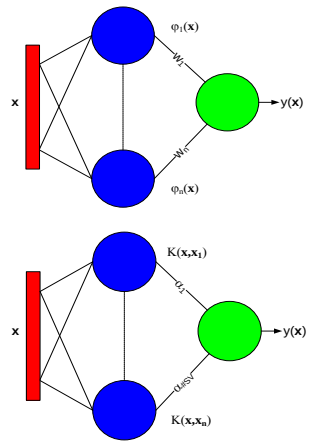
$$\begin{cases} \min\limits_{w,b,\xi} \| w \|_2^2 + C\sum\limits_{i=1}^{N}\xi_i \\ s.t. \\ \forall i, y_i(w^T\varphi(x_i)+b) \geq 1-\xi_i \\ \forall i, \xi_i \geq 0 \end{cases}$$

$K(x_i, x_j) = \varphi(x_i)^T\varphi(x_j)$ ("Kernel trick")

- **Dual problem:**

$$\begin{cases} \max\limits_{\alpha_k} -\frac{1}{2}\sum\limits_{k,l=1}^{N} y_k y_l K(x_k, x_l)\alpha_k\alpha_l + \sum\limits_{k=1}^{N}\alpha_k \\ s.t. \\ \sum\limits_{k=1}^{N}\alpha_k y_k = 0 \\ 0 \leq \alpha_k \leq C, k = 1,\ldots, N \end{cases}$$

Primal-dual formulation



Primal problem

$$y(x) = sign(w^T \varphi(x) + b)$$

$\Leftrightarrow$

Dual problem

$$y(x) = sign[\sum_{k=1}^{\#sv} \alpha_k y_k K(x, x_k) + b]$$

Least squares support vector machine (LS-SVM)

- Preserve SVM methodology, but simplify via least squares and equality constraints [Suykens, 1999]
  - Primal problem

$$\begin{cases} \min_{w,b,e'} \zeta(w,b,e') = \frac{1}{2} \| w \|_2^2 + \gamma \frac{1}{2} \sum_{i=1}^{N} e'^2_i \\ s.t. \\ \forall i, y_i [w^T \varphi(x_i) + b] = 1 - e'_i \end{cases} \quad \leftrightarrow \quad \begin{cases} \min_{w,b,e} \zeta(w,b,e) = \frac{1}{2} \| w \|_2^2 + \gamma \frac{1}{2} \sum_{i=1}^{N} e_i^2 \\ s.t. \\ \forall i, w^T \varphi(x_i) + b = y_i - e_i \end{cases}$$

  - Dual Problem

$$\begin{bmatrix} 0 & 1_N^T \\ 1_N & \Omega + I/\gamma \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \qquad \forall i, j; \Omega_{i,j} = \varphi(x_i)^T \varphi(x_j)$$

$$y = [y_1; \ldots; y_N]$$

- Proof: the Lagrangian of the primal

$$\zeta(w,b,e_i;\alpha_i) = \tfrac{1}{2}\| w\|_2^2 + \gamma \tfrac{1}{2}\sum_{i=1}^{N} e_i^2 - \sum_{i=1}^{N}\alpha_i(w^T\varphi(x_i)+b-y_i+e_i)$$

where the α_i values are the Lagrange multipliers ($\alpha_i \in \mathbb{R}$).

The conditions for optimality yield

$$\max_{\alpha_i}\min_{w,b,e_i}\zeta(w,b,e_i;\alpha_i) \rightarrow \begin{cases} \frac{\partial\zeta}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{N}\alpha_i\varphi(x_i) \\ \frac{\partial\zeta}{\partial e_i} = 0 \rightarrow \alpha_i = \gamma e_i, i = 1,\ldots,N \\ \frac{\partial\zeta}{\partial b} = 0 \rightarrow \sum_{i=1}^{N}\alpha_i = 0 \\ \frac{\partial\zeta}{\partial \alpha_i} = 0 \rightarrow w^T\varphi(x_i)+b-y_i+e_i = 0, i = 1,\ldots,N \end{cases}$$

Eliminating w and e_i leads to

$$\begin{cases} \frac{\alpha_i}{\gamma} + \sum_{j=1}^{N}\alpha_j\varphi(x_j)^T\varphi(x_i) + b = y_i, i = 1,\ldots N \\ \sum_{i=1}^{N}\alpha_i = 0 \end{cases}$$

Using $K(x,x') = \varphi(x)^T\varphi(x')$ and writing the above equations in matrix form gives

$$\begin{bmatrix} 0 & 1_N^T \\ 1_N & \Omega + I/\gamma \end{bmatrix}\begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \qquad \begin{array}{l} \forall i,j; \Omega_{i,j} = \varphi(x_i)^T\varphi(x_j) \\ y = [y_1;\ldots;y_N] \end{array}$$

- The classifier model can be evaluated in the dual representation as:

$$\hat{f}(x) = sign\left(\sum_{i=1}^{N}\alpha_i K(x,x_i) + b\right)$$

Solve Primal LS-SVM

- See LS-SVM primal derivation on Toledo
- Learning objective:

$$\min_{w'} \zeta(w'),$$

with $\zeta(w') = \frac{1}{2\gamma} w'^T w' + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{D}(w'_j x_{i(j)} - y_i)^2$.

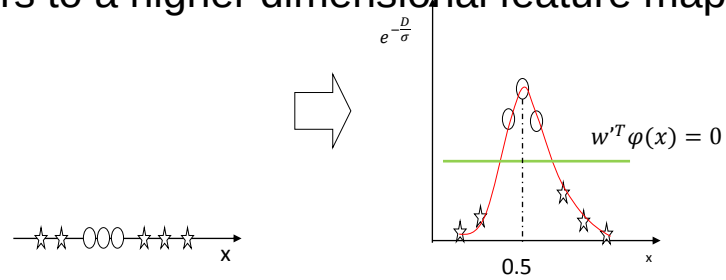
- Solution for w' can be found by solving

$$w' = \left(X^T X + \frac{1}{\gamma} I\right)^{-1} X^T y,$$

```
MATLAB:
w'=(1/gamma+X'X)\X'y
```

Add φ manually

- non-linear decision function by mapping of input vectors to a higher dimensional feature map.



- transformed to 2-D using manual map $\varphi(x) = (x, e^{-\frac{D}{\sigma}})$, where $D = (x - 0.5)^2$ → enables a linear classification in transformed space

Multi-Class SVM Classification

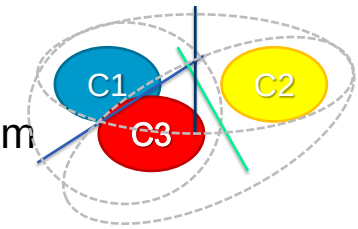
- 1-vs-rest
- 1-vs-1
  - MaxWin
  - DB2
  - Error Correcting Output Coding
- K-class SVM

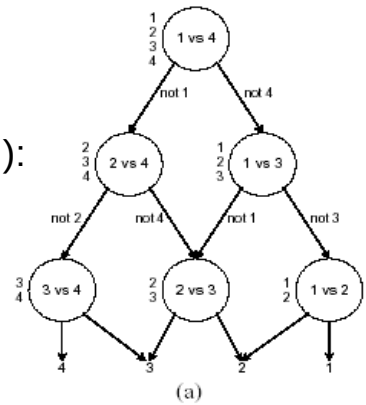
1-vs-rest

- For any class C, train a binary classifier to distinguish C from C.
- For an unseen sample, find the binary classifier with highest confidence score for the final decision.

1-vs-1

- Train 0.5*C*(C-1) classifiers, which distinguish one class from another one.
  - Pairwise:
    - MaxWin (0.5*C*(C-1) tests)
    - Error-correcting output code
  - Directed Acyclic Graph (DAG):
    - Pachinko-machine (C-1 tests)





Error Correcting Output Coding

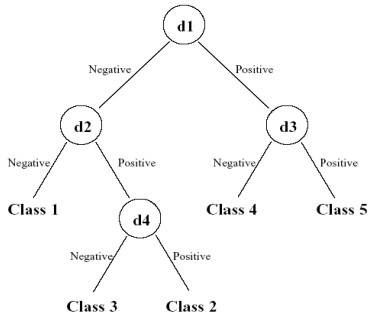
- Code Matrix (M^{NxK})
  - N classes, K classifiers
- Hamming Distance
- Class C_i with Minimum Error wins

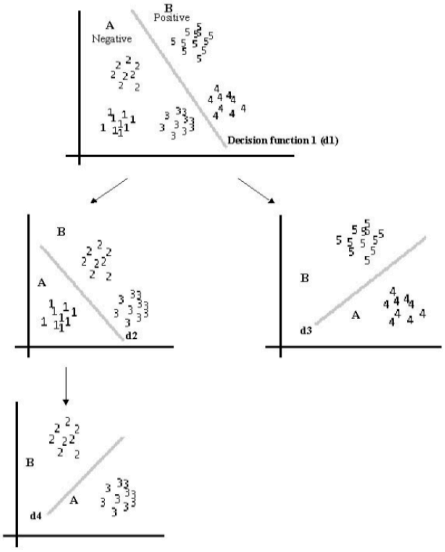
M	1-2	1-3	1-4	2-3	2-4	3-4
1	1	1	1	0	0	0
2	-1	0	0	1	1	0
3	0	-1	0	-1	0	1
4	0	0	-1	0	-1	-1

M	1,2	1,3	1,4	2,3	2,4	3,4
1	1	1	1	-1	-1	-1
2	1	-1	-1	1	1	-1
3	-1	1	-1	1	-1	1
4	-1	-1	1	-1	1	1

Divided-by-2 (DB2)

- Hierarchically divide the data into two subsets until every subset consists of only one class.





Divided-by-2 (DB2)

- Data partitioning criterion:
  - group the classes such that the resulting subsets have the largest margin.
- Trade-off: use clustering methods
  - k-means: use the mean of each class
  - Balanced subsets: minimal difference in sample number.

K-class SVM

- Change the loss function and constraints

$$\begin{cases} \min\limits_{w,\xi} \dfrac{1}{2} \| w \|_2^2 + C \sum\limits_{i=1}^{N} \sum\limits_{m \neq y_i} \xi_i^m \\ s.t. w_{y_i}^T x_i + b_{y_i} \geq w_m^T x_i + b_m + 2 - \xi_i^m \\ \xi_i^m \geq 0; i = 1,..., N; m \in \{1,..., k\} \setminus y_i \end{cases}$$

Compare...

- Interesting to compare SVM with
  - instance based learning:
    - IBL: stores all examples; classification based on distance to stored examples
    - SVM: stores examples that identify boundary between classes; classificiation based on which side of the boundary new example is
  - multi-layered neural networks:
    - Perceptron: linear separation, not with maximal margin
    - ANN obtains better expressiveness by changing representation throughout its layers
    - SVM obtains better expressiveness through non-linear transformation

To Remember

- Support vector machines:
  - Maximum margin classifiers
  - Support vectors
  - Linear SVMs
  - Construction of non-linear SVMs
  - Simplyfied algorithm: LS-SVM
- Perceptrons, neural networks:
  - Inspiration, what they are, how they work
  - representation power
  - explanatory power
- Comparison SVM, ANN and IBL

11 Combining classifiers

- Rationale behind combining classifiers (or, “ensemble methods”)
- Some methods:
  - Voting
  - Bagging
  - Boosting
  - Stacking



Combining hypotheses

- Idea:
  - Assume one learner learns h_1, another learns h_2
    - Different learners have different inductive bias
    - Hence corresponding hypotheses each make their own typical mistakes
  - Can we combine them into a hypothesis $h_{1,2}$ that works better than each of them separately?
    - A so-called *ensemble* of hypotheses
  - Maybe devise a learning method that immediately exploits this idea?
- A few approaches: voting, bagging, boosting



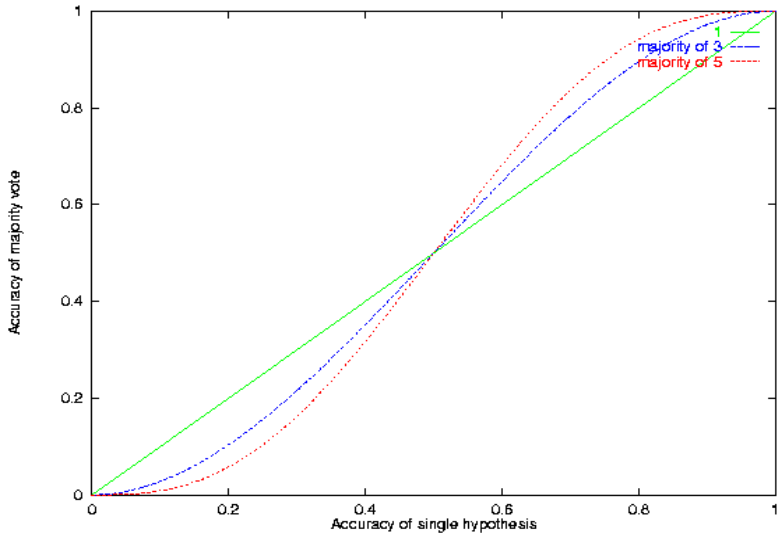
Voting

- Given: classifiers h_1, h_2, ..., h_n
- Define classifier h as follows:
  - Given a new example e,
  - Compute $h_1(e)$, $h_2(e)$, ..., $h_n(e)$
  - Output the most popular prediction (average or majority class)
- Will this work better?
  - Relatively easy to analyse *under simplifying conditions* (e.g. all hypotheses independent)
    - When all h_i have same accuracy p and make *independent* predictions: P(majority correct) computed using binomial distribution



Combining Independent Predictions

- Compare majority vote of 1, 3, 5 independent hypotheses, each having accuracy x





Voting in Practice

- But what if predictions are dependent?
  - It's reasonable to assume they will be
  - Not easy to analyse the effect in this case
  - In general, we can still hope for improvement
- What if some hypotheses are rather bad?
  - We might give higher weights to better ones...
- More complex approaches:
  - e.g., identify "regions" where certain hypotheses do well, and adapt the hypothesis weights according to these regions



Bagging

- So using majority votes can help... but how do we get different hypotheses?
- 1 trick: (bootstrap aggregating, “bagging”)
  - Apply "bootstrapping":
    - from training set T of size n, draw sample T_i of size $n' \leq n$ (same element can be drawn multiple times)
    - Do this m times -> m different training sets from T
  - Now learn one h_i from each different data set T_i
  - Use voting among h_i for new predictions
- Gives good results for "unstable" problems
  - = when slightly different data yield different results



Boosting

- Similar to bagging, but:
  - Instead of resampling data set, give weights to examples
  - Loop: (until some stopping criterion fulfilled)
    - Learn a hypothesis h_i from weighted data set
      - No problem if this is a relatively bad hypothesis
    - Give *higher weights* to examples predicted *incorrectly* by h_i (so future hypotheses will focus on these "hard cases")
    - i ⟵ i+1
  - Final prediction based on weighted votes of h_i
- Very good results achieved in many cases



THOMAS MORE

AdaBoost

- Popular boosting algorithm (Freund & Shapire, 1996)
- Repeatedly learns hypotheses using a “weak learner” (just a bit better than random guessing)
- We here give only one (simple) version

for t=1 to T **do** *// T is a parameter: number of rounds*
learn h_t, using weak learner L, from data distribution D_t
$\varepsilon_t = \Sigma_i D_t(x_i)$ for all i where $h_t(x_i) \neq c(x_i)$
if $\varepsilon_t > 1/2$ **then** stop
$\beta_t = \varepsilon_t / (1 - \varepsilon_t)$
$D_{t+1}(i) = D_t(i) * \beta_t$ if $h_t(x_i)=c(x_i)$,
$D_t(i)$ otherwise
normalize D_{t+1} (so that $\Sigma_i D_{t+1}(x_i) = 1$)
Output $h_{final}(x) = \text{argmax}_c \Sigma_{t:\, h_t(x)=c} \log(1/\beta_t)$



THOMAS MORE

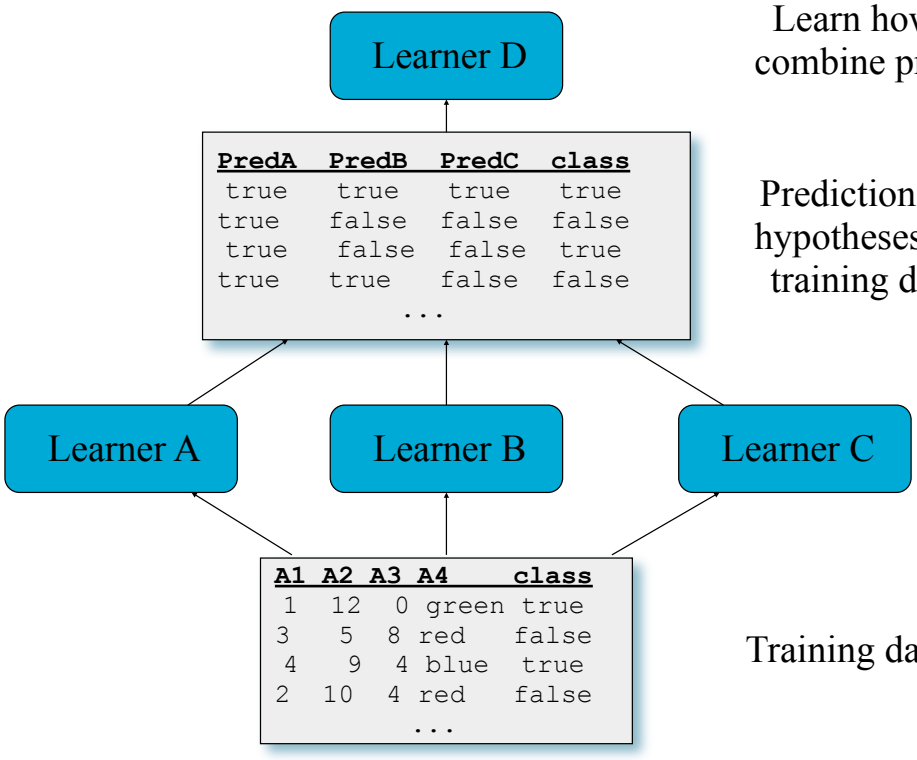
Stacking

- Bagging & boosting:
  - *Same* algorithm is run on *variations of data set*
- Voting can also combine predictions of hypotheses found by *different algorithms*
- Basic idea of stacking:
  - Have different hypotheses vote
  - But do not just use majority vote for prediction
  - Instead, *learn* how to combine their votes
    - I.e. one learner is *stacked* on top of the other learners
    - Can also view this as a *data transformation* based on learning algorithms
      - Prediction by hypothesis is just 1 extra attribute



THOMAS MORE

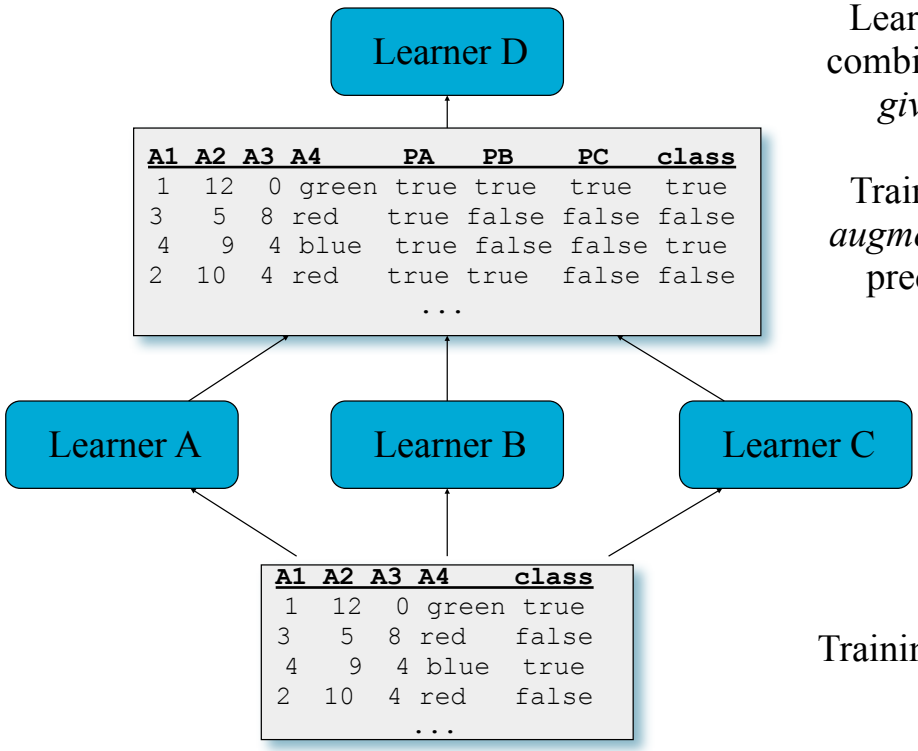
Stacking: Illustration





THOMAS MORE

Data Transformation Viewpoint



Learn how to best combine predictions, *given the data*

Training data *augmented* with predictions

Training data

To Remember

- Idea of combining classifiers
- Principles of voting, bagging, boosting, stacking, data transformation

12 Bayesian learning

- Introduction: probabilistic (Bayesian) methods
- For describing (results of) learning processes
  - MAP and ML hypotheses
  - Minimum description length principle
  - Bayes optimal classifier
- Developing practical learning algorithms
  - Naïve Bayes learner
    - application: learning to classify texts
  - Learning Bayesian belief networks



THOMAS MORE

Probability Theory

- Appendix A
  - Some basics
- Experiments, outcomes, events, probabilities
- Conditional probabilities
- Bayes' rule
- Rule of total probability
- Continuous variables
- Joint probability distribution



THOMAS MORE

Probabilistic Experiment

- A *Probabilistic Experiment* is a situation in which
  - More than one thing can happen
  - The outcome is potentially uncertain
- The *Sample Space* Ω of a probabilistic experiment E is the set of all possible outcomes of E.



THOMAS MORE

The Sample Space

Examples:

E_1 = Toss a coin, observe whether it is a Head (H) or a Tail (T)

Ω_1 = {H, T}

E_2 = Toss a fair die, observe the outcome.

Ω_2 = {1, 2, 3, 4, 5, 6}



THOMAS MORE

Probability of an event

- Event
  - = a subset of possible outcomes
  - an event occurs if
    - experiment yields outcome that is member of that event
- Probability of event E Pr(E)
  - = limit of the proportion n_E/n with $n \longrightarrow \infty$
    - n=number of experiments
    - n_E =number of times that event E occurs
- Probability of outcome e
  - Pr(e)=Pr(E) with E={e}



THOMAS MORE

Rolling 2 fair dice

- Ω_{d1} = {1, 2, 3, 4, 5, 6}; Ω_{d2} = {1, 2, 3, 4, 5, 6}
- $\Omega_{d1,d2}$ = {(1,1),(1,2),...,(1,6),(2,1), ...,(6,6)}
- each outcome is equally likely: $Pr(e \in \Omega_{d1,d2}) = \frac{1}{36}$
- X=outcome d1; Y=outcome d2; Z=X+Y
- $E_{X=3}$={(3,1),(3,2),(3,3),(3,4),(3,5),(3,6)}
- $E_{z=5}$={(1,4),(2,3),(3,2),(4,1)}



THOMAS MORE

Basic operations

- A and B are events
- $A \cup B$ contains all outcomes that are in A or B
  - interpreted as “A or B”
- $A \cap B$ contains all elements that are in both A and B
  - interpreted as “A and B”
- $\bar{A}$ contains all outcomes not in A
  - interpreted as “not A”
- Pr(A={a1,a2,...,an})=Pr({a1})+Pr({a2})+...+Pr({an})
- $Pr(A \cup B) = Pr(A) + Pr(B) - Pr(A \cap B)$



THOMAS MORE

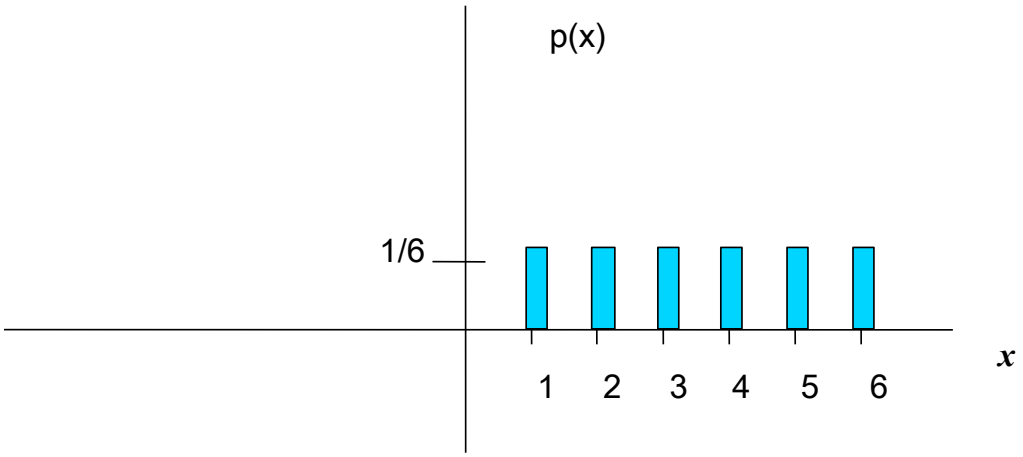
Probability functions

- A probability function maps the possible values of x against their respective probabilities of occurrence, p(x)
- p(x) is a number from 0 to 1.0.
- The area under a probability function is always 1.



THOMAS MORE

p(x) for roll of a die



$$\sum_{\text{all } x} P(x) = 1$$

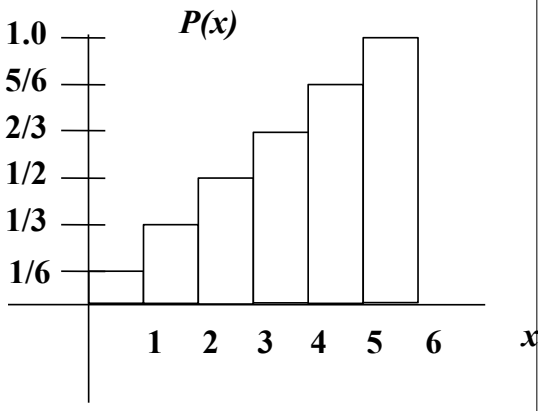
probability mass function

x	p(x)
1	p(x=1)=1/6
2	p(x=2)=1/6
3	p(x=3)=1/6
4	p(x=4)=1/6
5	p(x=5)=1/6
6	p(x=6)=1/6

THOMAS MORE

Cumulative distribution function (CDF)

x	P(x≤A)
1	P(x≤1)=1/6
2	P(x≤2)=2/6
3	P(x≤3)=3/6
4	P(x≤4)=4/6
5	P(x≤5)=5/6
6	P(x≤6)=6/6





THOMAS MORE

Conditional probabilities

- Conditional probability of an event A given an event B=Pr(A|B)
  - = limit of the proportion $n_{A,B}/n_B$ with $n \rightarrow \infty$
    - n_B =number of times that B occurs
    - $n_{A,B}$ =number of times that both A and B occur
  - =Pr(A ∩ B)/Pr(B)

- Pr(A ∩ B)=Pr(A|B) Pr(B)



THOMAS MORE

Some more rules

- *Bayes' rule*
  - $P(A|B) = P(B|A)\ P(A) / P(B)$
- Rule of total probability
  - $P(A) = \sum_i P(A|B_i)\ P(B_i)$
    - with B_i exhaustive and mutually exclusive



THOMAS MORE

Continuous case

- Continuous variables
  - have infinite continuum of possible outcomes
  - probability 1 specific outcome=0
  - only consider intervals
- Cumulative probability distribution
  - $Pr((-infty,x]))=Pr(X<x)=F_X$
- Probability density function (pdf) f_X
  - derivative of F_X

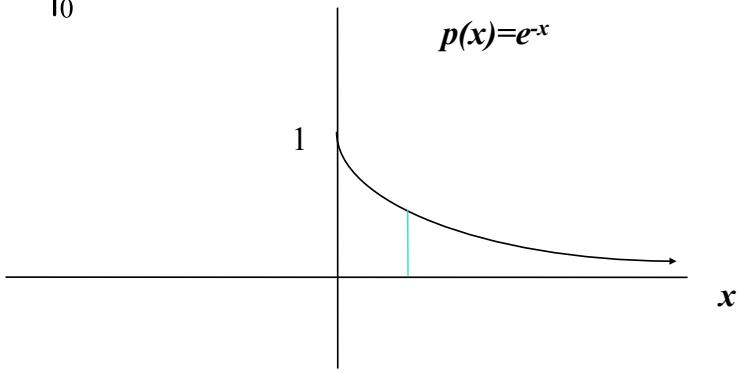


THOMAS MORE

Example 1: Exponential distribution

- negative exponential function $f(x) = e^{-x}$
  - function integrates to 1

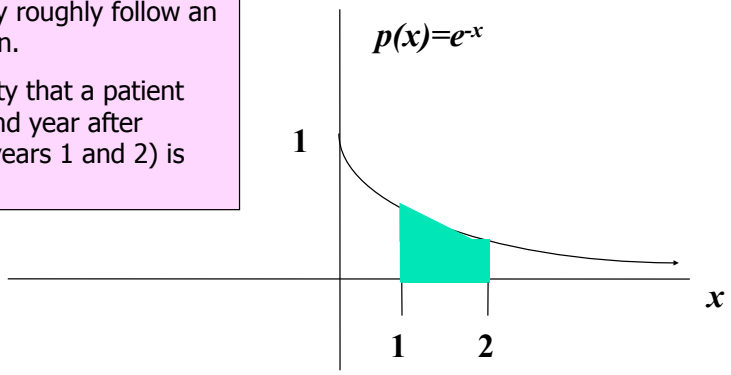
$$\int_0^{+\infty} e^{-x} = -e^{-x} \Big|_0^{+\infty} = 0 + 1 = 1$$



For example, the probability of x falling within 1 to 2:

Clinical example: Survival times after lung transplant may roughly follow an exponential function.

Then, the probability that a patient will die in the second year after surgery (between years 1 and 2) is 23%.

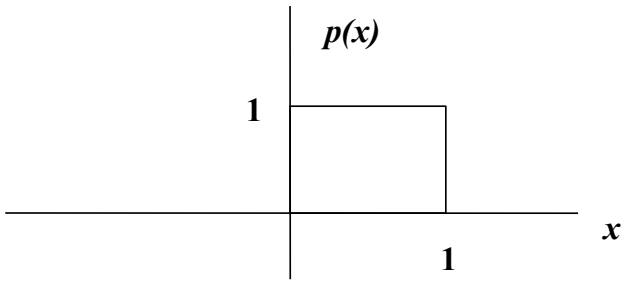


$$P(1 \le x \le 2) = \int_1^2 e^{-x} = -e^{-x} \Big|_1^2 = -e^{-2} - -e^{-1} = -.135 + .368 = .23$$

Example 2: Uniform distribution

The uniform distribution: all values are equally likely.

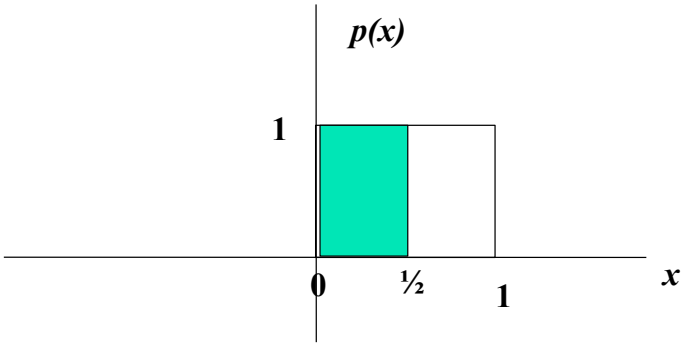
$f(x) = 1$, for $1 \geq x \geq 0$



We can see it's a probability distribution because it integrates to 1 (the area under the curve is 1):

$$\int_0^1 1 = x \Big|_0^1 = 1 - 0 = 1$$

What's the probability that x is between 0 and ½?



Clinical Research Example: When randomizing patients in an RCT, we often use a random number generator on the computer. These programs work by randomly generating a number between 0 and 1 (with equal probability of every number in between). Then a subject who gets X<.5 is control and a subject who gets X>.5 is treatment.

P(½ ≥x≥ 0)= ½

Expected value and variance

- All probability distributions are characterized by an expected value (mean) and a variance (standard deviation squared).
- Expected value is just the average or mean (μ) of random variable x.
  - It's sometimes called a "weighted average" because more frequent values of X are weighted more highly in the average.
  - It's also how we expect X to behave on-average over the long run



THOMAS MORE

Expected value, formally

Discrete case:

$$E(X) = \sum_{\text{all x}} x_i \, p(x_i)$$

Continuous case:

$$E(X) = \int_{\text{all x}} x_i \, p(x_i) dx$$



THOMAS MORE

Variance/standard deviation

σ^2=Var(x) =E(x-μ)^2

"The expected (or average) squared distance (or deviation) from the mean"

$$\sigma^2 = Var(x) = E[(x-\mu)^2] = \sum_{\text{all x}} (x_i - \mu)^2 \, p(x_i)$$



THOMAS MORE

Variance, continuous

Discrete case:

$$Var(X) = \sum_{\text{all x}} (x_i - \mu)^2 \, p(x_i)$$

Continuous case:

$$Var(X) = \int_{\text{all x}} (x_i - \mu)^2 \, p(x_i) dx$$



THOMAS MORE

Binomial Probability Distribution

- Important discrete probability distribution
- A fixed number of observations (trials), n
  - e.g., 15 tosses of a coin; 20 patients; 1000 people surveyed
- A binary outcome
  - e.g., head or tail in each toss of a coin; disease or no disease
  - Generally called "success" and "failure"
  - Probability of success is p, probability of failure is 1 – p
- Constant probability for each observation
  - e.g., Probability of getting a tail is the same each time we toss the coin



THOMAS MORE

Binomial distribution

Take the example of 5 coin tosses. What's the probability that you flip exactly 3 heads in 5 coin tosses?



THOMAS MORE

Binomial distribution

Solution:
One way to get exactly 3 heads: HHHTT

What's the probability of this exact arrangement?
P(heads)xP(heads) xP(heads)xP(tails)xP(tails) $=(1/2)^3 \; x \; (1/2)^2$

Another way to get exactly 3 heads: THHHT
Probability of this exact outcome = $(1/2)^1 \; x \; (1/2)^3 \; x \; (1/2)^1 = (1/2)^3 \; x \; (1/2)^2$



THOMAS MORE

Binomial distribution

In fact, $(1/2)^3 \; x \; (1/2)^2$ is the probability of each unique outcome that has exactly 3 heads and 2 tails.

So, the overall probability of 3 heads and 2 tails is:
$(1/2)^3 \; x \; (1/2)^2 + (1/2)^3 \; x \; (1/2)^2 + (1/2)^3 \; x \; (1/2)^2 + \ldots..$
for as many unique arrangements as there are—but how many are there??



THOMAS MORE

$\binom{5}{3}$ ways to arrange 3 heads in 5 trials

Outcome	Probability
THHHT	$(1/2)^3 \times (1/2)^2$
HHHTT	$(1/2)^3 \times (1/2)^2$
TTHHH	$(1/2)^3 \times (1/2)^2$
HTTHH	$(1/2)^3 \times (1/2)^2$
HHTTH	$(1/2)^3 \times (1/2)^2$
HTHHT	$(1/2)^3 \times (1/2)^2$
THTHH	$(1/2)^3 \times (1/2)^2$
HTHTH	$(1/2)^3 \times (1/2)^2$
HHTHT	$(1/2)^3 \times (1/2)^2$
THHTH	$(1/2)^3 \times (1/2)^2$
10 arrangements $\times (1/2)^3 \times (1/2)^2$	

The probability of each unique outcome (note: they are all equal)

$${}_5C_3 = 5!/3!2! = 10$$

$$\therefore \mathbf{P(3\ heads\ and\ 2\ tails)} = \binom{5}{3} \boldsymbol{x\ P(heads)^3\ x\ P(tails)^2} =$$

$$\boldsymbol{10\ x\ (½)^5 = 31.25\%}$$

THOMAS MORE

Binomial distribution in general

Note the general pattern emerging → if you have only two possible outcomes (call them 1/0 or yes/no or success/failure) in n independent trials, then the probability of exactly X "successes"=

$$\binom{n}{X} p^X (1-p)^{n-X}$$

n = number of trials

$1\text{-}p$ = probability of failure

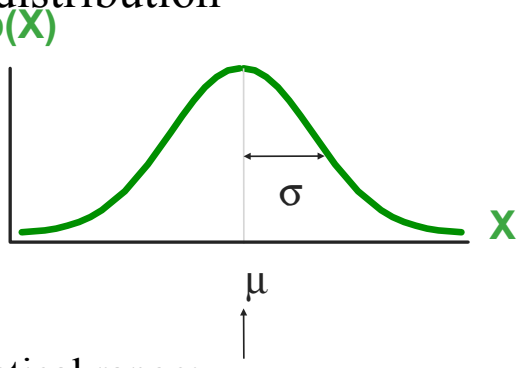
X = # successes out of n trials

p = probability of success

THOMAS MORE

Normal distribution

- Important continuous probability distribution
- 'Bell Shaped'
- Symmetrical
- Mean, Median and Mode are Equal



μ=mean

σ = standard deviation

The random variable has an infinite theoretical range:

$+\infty$ to $-\infty$

Mean = Median = Mode

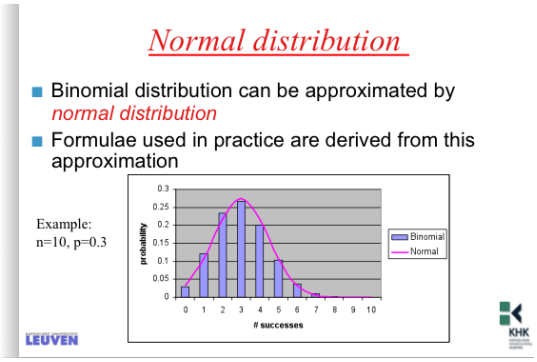
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \times e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



THOMAS MORE

Approximation of Binomial

- Remember slide from lecture 5



- Many distributions can be approximated by a normal distribution for suitably large values of relevant parameters



THOMAS MORE

Bayes' Theorem

- $P(A|B) = P(B|A)\,P(A) / P(B)$
- Mainly 2 ways of using Bayes' theorem:
  - Applied to learning a hypothesis h from data D:
    - $P(h|D) = P(D|h)\,P(h) / P(D) \sim P(D|h)P(h)$
      - P(h): a priori probability that h is correct
      - P(h|D): a posteriori probability that h is correct
      - P(D): probability of obtaining data D
      - P(D|h): probability of obtaining data D if h is correct
  - Applied to classification of a single example e:
    - $P(class|e) = P(e|class)P(class)/P(e)$



THOMAS MORE

Bayes' theorem: Example

- Example:
  - assume some lab test for a disease has 98% chance of giving positive result if disease is present, and 97% chance of giving negative result if disease is absent
  - assume furthermore 0.8% of population has this disease
  - given a positive result, what is the probability that the disease is present?
    - *P(Dis|Pos) = P(Pos|Dis)P(Dis) / P(Pos) = 0.98*0.008 / (0.98*0.008 + 0.03*0.992)=0,00784/0,0376=0,2085*



THOMAS MORE

MAP and ML hypotheses

- Question: *given the current data D and some hypothesis space H, return the hypothesis h in H that is most likely to be correct*.
- Note: this h is *optimal* in a certain sense
  - no method can exist that finds with higher probability the correct h



THOMAS MORE

MAP hypothesis

- Given some data D and a hypothesis space H, find the hypothesis $h \in H$ that has the highest probability of being correct; i.e., P(h|D) is maximal
  - This hypothesis is called the *maximal a posteriori hypothesis* h_{MAP} : $\boxed{h_{MAP} = argmax_{h \in H} P(h|D)}$
    - $= argmax_{h \in H} P(D|h)P(h)/P(D) = argmax_{h \in H} P(D|h)P(h)$
    - last equality holds because P(D) is constant
    - So : we need P(D|h) and P(h) for all $h \in H$ to compute h_{MAP}



THOMAS MORE

ML hypothesis

- P(h): a priori probability that h is correct
- What if no preferences for one h over another?
- Then assume P(h) = P(h') for all h, h'$\in$ H
- Under this assumption h_{MAP} is called the *maximum likelihood hypothesis* h_{ML}
  - $\boxed{h_{ML} = argmax_{h \in H} P(D|h)}$ (because P(h) constant)
- How to find h_{MAP} or h_{ML} ?
  - brute force method: compute P(D|h), P(h) for all h$\in$ H
  - usually not feasible



THOMAS MORE

Minimum Description Length (MDL)

- Occam's razor: prefer simplest hypothesis
  - "simplest" = shortest description
- Minimal description length principle:
  - hypothesis+corrections should have shortest description
    - trading off *complexity* and *correctness* of hypothesis
  - given data D, hypothesis space H, and encodings C_1 and C_2 for hypotheses resp. data:
  - $h_{MDL} = argmin_{h \in H} L_{C1}(h) + L_{C2}(D|h)$
    - $L_{C1}(x)$ denotes description length of x under encoding C1
    - $L_{C2}(D|h)$ indicates exceptions to h's predictions
      - ★ $L_{C2}(D|h) = 0$ if h entirely correct



THOMAS MORE

- Interesting observation:

$$h_{MAP} = \arg\max_{h \in H} P(D \mid h)P(h)$$
$$= \arg\max_{h \in H} \left(\log_2 P(D \mid h) + \log_2 P(h)\right)$$
$$= \arg\min_{h \in H} \left(-\log_2 P(D \mid h) - \log_2 P(h)\right)$$

  - h_{MAP} equals h_{MDL} under *optimal* encodings
    - encoding length of message based on its probability
  - But note: no link between *optimal* encoding and *practical* encoding mechanisms! (trees, …)
    - hence, *no reason* for claiming that e.g. shorter trees have higher probability of being correct!



THOMAS MORE

Bayes Optimal Classifier

- Problem considered up till now:
  - given data D and hypothesis space H, find most probable *hypothesis* h in H
- Now consider this problem:
  - given data D, hypothesis space H, and new instance x, what is most probable *classification* of x?
    - equivalent to h(x), with h most probable hypothesis? ***No.***
  - example:
    - $P(h_1|D)=0.4$, $P(h_2|D)=P(h_3|D)=0.3$; and $h_1(x)=+$, $h_2(x)=h_3(x)= -$
    - What is most probable classification of x?



THOMAS MORE

Bayes optimal classifier

- With hypothesis space H and set of classes V, most probable classification is

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

- In our example: $P(+|h_1) = 1$, $P(-|h_1)=0$; etc.



THOMAS MORE

Gibbs classifier

- Bayes Optimal is optimal but expensive
  - uses all hypotheses in H
- What if we approximate it as follows:
  - select one h at random, according to probabilities $P(h_i|D)$
  - predict h(x)
- This method is called the Gibbs classifier
- Surprisingly: $E(\text{error}_{\text{Gibbs}}) \leq 2\, E(\text{error}_{\text{BayesOptimal}})$



THOMAS MORE

Naïve Bayes classifier

- Simple & popular classification method
- Based on Bayes' rule + assumption of *conditional independence*
  - assumption often violated in practice
  - even then, it usually works well
- Successful application: classification of text documents



THOMAS MORE

Classification using Bayes rule

- Given attribute values, what is most probable value of target variable?

$$
\begin{aligned}
v_{MAP} &= \arg\max_{v_j \in V} P(v_j \mid a_1, a_2, ..., a_n) \\
&= \arg\max_{v_j \in V} \frac{P(a_1, a_2, ..., a_n \mid v_j) P(v_j)}{P(a_1, a_2, ..., a_n)} \\
&= \arg\max_{v_j \in V} P(a_1, a_2, ..., a_n \mid v_j) P(v_j)
\end{aligned}
$$

- Problem: too much data needed to estimate $P(a_1 \ldots a_n | v_j)$



THOMAS MORE

The Naïve Bayes classifier

- *Naïve Bayes assumption*: attributes are *independent, given the class*
  - $P(a_1,\ldots,a_n|v_j) = P(a_1|v_j)P(a_2|v_j)\ldots P(a_n|v_j)$
  - also called *conditional independence* (given the class)
- Under that assumption, v_{MAP} becomes

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$



THOMAS MORE

- What if assumption violated?
  - i.e. $P(a_1,\ldots,a_n|v_j) \neq P(a_1|v_j)P(a_2|v_j)\ldots P(a_n|v_j)$
- Prediction still equivalent to Bayes prediction as long as the following (weaker) condition holds:

$$\arg\max_{v_j \in V} P(a_1 \mid v_j)P(a_2 \mid v_j)...P(a_n \mid v_j)P(v_j)$$
$$= \arg\max_{v_j \in V} P(a_1, a_2, ..., a_n \mid v_j)P(v_j)$$

- But *probabilities* associated with prediction may be unrealistically close to 0 or 1



THOMAS MORE

Learning a Naïve Bayes classifier

- To *learn* such a classifier: just estimate $P(v_j)$, $P(a_i|v_j)$ from data

$$v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i \mid v_j)$$

- How to estimate?
  - simplest: standard estimate from statistics
    - estimate probability from sample proportion
    - e.g., estimate P(A|B) as count(A and B) / count(B)
  - in practice, something more complicated needed…



THOMAS MORE

Estimating probabilities

- Problem:
  - What if attribute value a_i never observed for class v_j?
  - Estimate $P(a_i|v_j)=0$ because count(a_i and v_j) = 0 ?
    - Effect is too strong: this 0 makes the whole product 0!
- Solution: use m-estimate
  - interpolates between observed value n_c/n and a priori estimate p -> estimate may get close to 0 but never 0
    - m is weight given to a priori estimate

$$\hat{P}(a_i \mid v_j) = \frac{n_c + mp}{n + m}$$



THOMAS MORE

Exercice

- Healthy:
  - <single,dark,one>
  - <single,light,two>
  - <double,light,one>
- Virulent:
  - <single,dark,two>
  - <double,dark,one>
  - <double,light,two>
- Classify <single,light,one> using Naive Bayes
  - assume m=0



THOMAS MORE

Solution (1)

- most likely hypothesis is:

$$argmax_H P(single|H)P(light|H)P(one|H)P(H)$$

- Compute
  - P(single|Healthy)=2/3, P(double|Healthy)=1/3
  - P(dark|Healthy)=1/3, P(light|Healthy)=2/3
  - P(one|Healthy)=2/3, P(two|Healthy)=1/3
    - P(Healthy)=1/2
  - P(single|Virulent)=1/3, P(double|Virulent)=2/3
  - P(dark|Virulent)=2/3, P(light|Virulent)=1/3
  - P(one|Virulent)=1/3, P(two|Virulent)=2/3
    - P(Virulent)=1/2



THOMAS MORE

Solution (2)

- Likelihood(Healthy)
  - = P(single|Healthy) P(light|Healthy)P(one| Healthy)P(Healthy)
  - =2/3 x 2/3 x 2/3 x 1/2
  - =4/27
- Likelihood(Virulent)
  - = P(single|Virulent) P(light|Virulent)P(one| Virulent)P(Virulent)
  - =1/3 x 1/3 x 1/3 x 1/2
  - =1/54
- Most likely class is Healthy



THOMAS MORE

Learning to classify text

- Example application:
  - given text of newsgroup article, guess which newsgroup it is taken from
  - Naïve bayes turns out to work well on this application
  - How to apply NB?
  - Key issue :
    - how do we represent examples?
    - what are the attributes?



THOMAS MORE

Representation

- Binary classification (+/-) or multiple classes possible
- Attributes = word positions
  - I.e. attribute i represents i-th word in text
  - value of attribute = word that occurs there
    - Note: could have chosen other representations; e.g. attribute = specific word, value = its frequency in the text ("bag of words" representation)
  - further assumption: probability of having a specific word is independent of position
    - $P(a_i=w_k|v_j) = P(a_m=w_k|v_j)\ \forall i,m$



THOMAS MORE

Algorithm

procedure learn_naïve_bayes_text(*E*: set of articles, V: set of classes)

Voc = all words and tokens occurring in E

estimate $P(v_j)$ and $P(w_k|v_j)$ for all w_k in E and v_j in V:

N_j = number of articles of class j

N = number of articles

$P(v_j) = N_j/N$

n_{kj} = number of times word w_k occurs in text of class j

n_j = number of words in class j (counting doubles)

$P(w_k|v_j) = (n_{kj}+1)/(n_j+|Voc|)$

procedure classify_naïve_bayes_text(A: article)

remove from A all words/tokens that are not in Voc

return $\text{argmax}_{vj \in V}\ P(v_j) \prod_i P(a_i|v_j)$



THOMAS MORE

Exercice

- Classify the sentence ``Cats eat mice and dogs bury bones." given the following two examples:
  - Class A: ``The cat crabs the curls off the stairs."
  - Class B: ``It's raining cats and dogs."



THOMAS MORE

Solution

- Voc = {Cat, Crab, Curl, Stairs, Rain, Dog}
- $$P(w_k|v_j) = \frac{n_{k,j} + 1}{n_j + 6}$$
- return $\text{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$
  - A) $P(A) \cdot P(\text{Cat}|A) \cdot P(\text{Dog}|A) = 0.5 \cdot 0.2 \cdot 0.1 = 0.01$
  - B) $P(B) \cdot P(\text{Cat}|B) \cdot P(\text{Dog}|B) = 0.5 \cdot 0.22 \cdot 0.22 = 0.0247$
- Conclude class B



THOMAS MORE

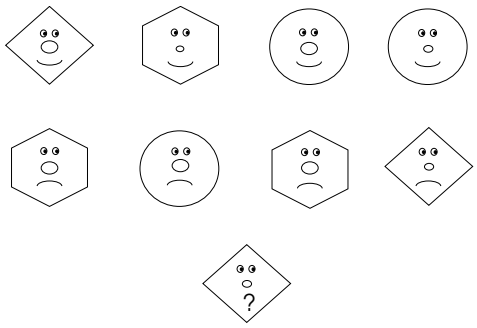
- Some (old) experimental results:
  - 1000 articles taken from 20 newsgroups
  - guess correct newsgroup for unseen documents
  - 89% classification accuracy with previous approach
- Note: more recent approaches based on SVMs, … have been reported to work better
  - But Naïve Bayes still used in practice, e.g., for spam detection



THOMAS MORE

Additional Exercice

- Predict wether the last face is smiling or not using Naive Bayes.





THOMAS MORE

Recall previous slide: Classification using Bayes rule

- Given attribute values, what is most probable value of target variable?

$$
\begin{aligned}
v_{MAP} &= \arg\max_{v_j \in V} P(v_j \mid a_1, a_2, ..., a_n) \\
&= \arg\max_{v_j \in V} \frac{P(a_1, a_2, ..., a_n \mid v_j) P(v_j)}{P(a_1, a_2, ..., a_n)} \\
&= \arg\max_{v_j \in V} P(a_1, a_2, ..., a_n \mid v_j) P(v_j)
\end{aligned}
$$

- Problem: too much data needed to estimate $P(a_1 \ldots a_n | v_j)$



THOMAS MORE

Bayesian Belief Networks

- Consider two extremes of spectrum:
  - guessing joint probability distribution
    - would yield optimal classifier
    - but infeasible in practice (too much data needed)
  - Naïve Bayes
    - much more feasible
    - but strong assumptions of conditional independence
- Can we find something in between?
  - make some independence assumptions, but only where reasonable



THOMAS MORE

Bayesian belief networks

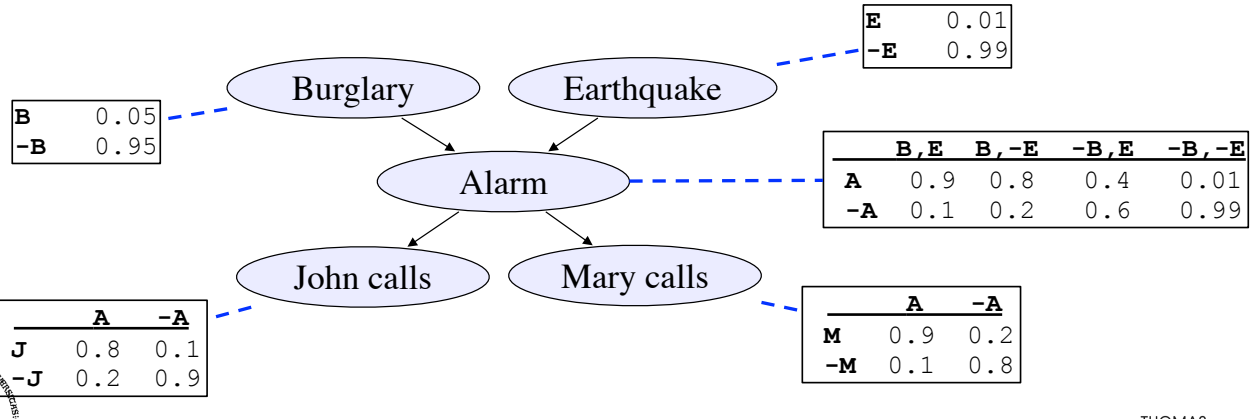
- Bayesian belief network consists of
  - 1: *graph*
    - intuitively: indicates which variables "directly influence" which other variables
      - arrow from A to B: A has direct effect on B
      - parents(X) = set of all nodes directly influencing X
        - » X is influenced only by its parents
    - formally: each node is *conditionally independent of each of its non-descendants, given its parents*
      - conditional independence: cf. Naïve Bayes
      - *X conditionally independent of Y given Z iff P(X|Y,Z) = P(X|Z)*
  - 2: *conditional probability tables*
    - for each node X : P(X|parents(X)) is given



THOMAS MORE

Example

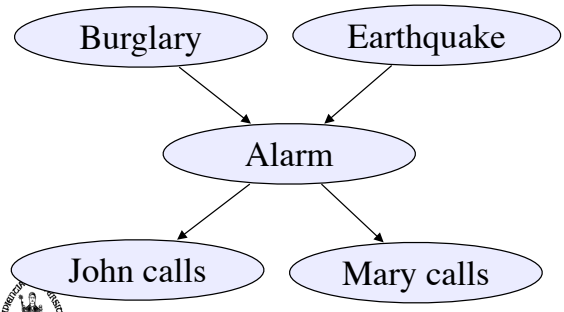
- Burglary or earthquake may cause alarm to go off
- Alarm going off may cause one of neighbours to call

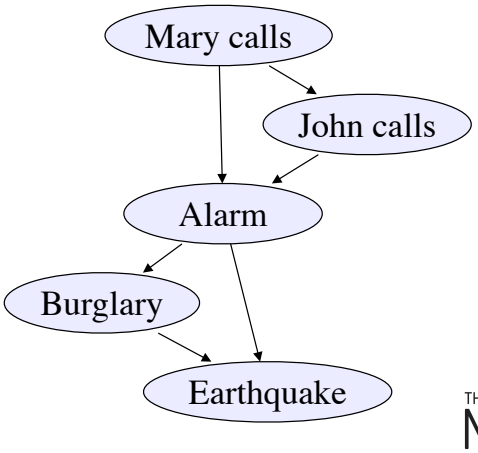




THOMAS MORE

- Network topology usually reflects *direct causal* influences
  - other structure also possible
  - but may render network more complex







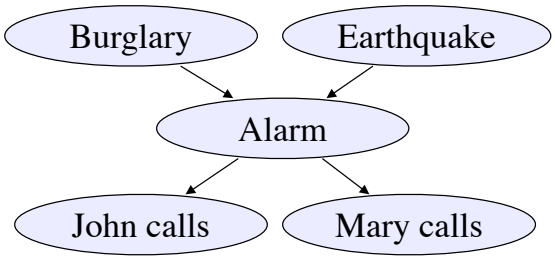
THOMAS MORE

- Graph + conditional probability tables allow to construct joint probability distribution of all variables
  - $P(X_1,X_2,\ldots,X_n) = \prod_i P(X_i|\text{parents}(X_i))$
  - In other words: *bayesian belief network carries full information on joint probability distribution*



THOMAS MORE

Example



- Joint probability distribution from conditional ones:
  - P(J,M,A,B,E) = P(J|A) P(M|A) P(A|B,E) P(B) P(E)
    - to see this: start with P(B) and P(E)
      - "conditionally independent from each other given their parents" = unconditionally independent, hence P(B,E) = P(B) P(E)
    - P(A,B,E) = P(A|B,E) P(B,E) (by definition)
    - P(J,M,A,B,E) = P(J,M|A,B,E) P(A,B,E) (by def.)
      = P(J|A) P(M|A) P(A,B,E)



THOMAS MORE

Inference

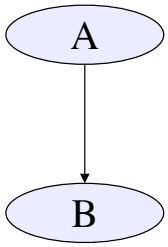
- Given values for certain nodes, infer probability distribution for values of other nodes
- General algorithm quite complicated
  - See, e.g., Russel & Norvig, 1995: *Artificial Intelligence, a Modern Approach*



THOMAS MORE

Simplest case: 2 nodes

- Given: p(A), p(B|A)
  - A known, infer $p_{A=a}(B)$
    - directly from p(B|A)
  - A unknown, infer $p_{A=?}(B)$
    - "total probability" rule
  - B known, infer $p_{B=b}(A)$
    - Bayes' rule
  - B unknown, infer $p_{B=?}(A)$
    - = p(A)





THOMAS MORE

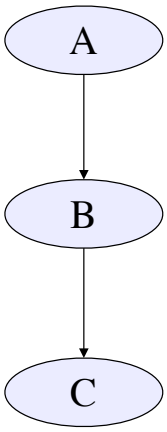
A simple 3 nodes network

- Given: p(A), p(B|A), p(C|B)
- E.g., A=a and C=c known:

$$P(B=b|A=a,C=c)$$

$$= \frac{P(B=b,A=a,C=c)}{\sum_i P(A=a,B=b_i,C=c)}$$

$$= \frac{P(A=a)P(B=b|A=a)P(C=c|B=b)}{\sum_i P(A=a)P(B=b_i|A=a)P(C=c|B=b_i)}$$

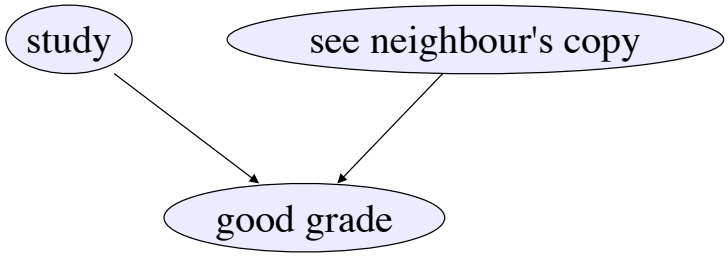




THOMAS MORE

More nodes

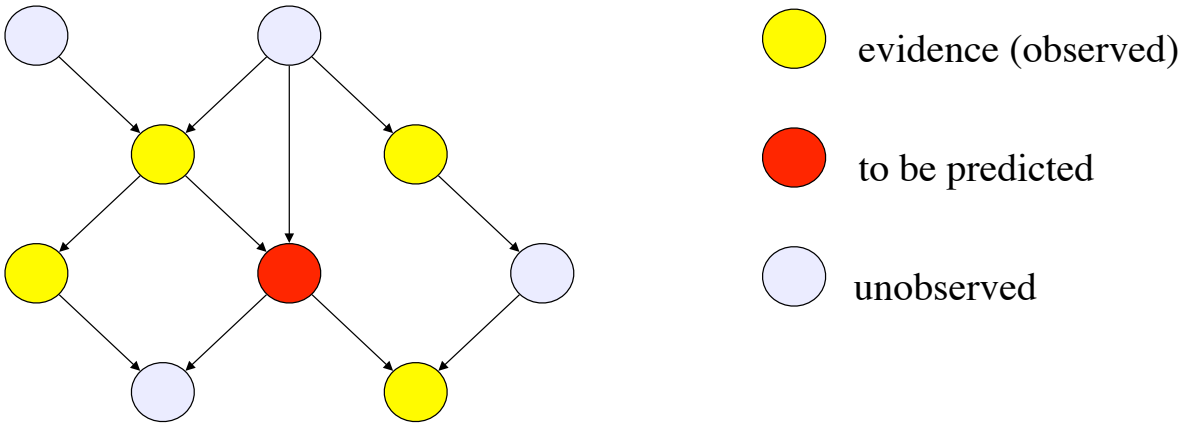
- In general, relatively complex reasoning can be achieved
  - forward and backward reasoning
  - "explaining away“ possible causes
    - "good grade" is evidence for "studied hard"
    - but is less strong evidence if known that person looked at neighbour's copy





THOMAS MORE

General case



- In general: inference is NP-complete
  - approximating methods, e.g. Monte-Carlo



THOMAS MORE

Learning bayesian networks

- Assume structure of network given:
  - only conditional probability tables to be learnt
  - training examples may include values for all variables, or just for some of them
  - when all variables observable:
    - estimating probabilities as easy as for Naïve Bayes
    - e.g. estimate P(A|B,C) as count(A,B,C)/count(B,C)
  - when not all variables observable:
    - notice similarity with training neural networks (hidden units)
    - methods exist, based on gradient descent or EM



THOMAS MORE

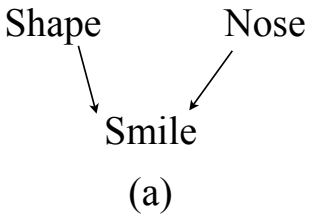
- When structure of network not given:
  - search for structure + tables
    - e.g. propose structure, learn tables
    - propose change to structure, relearn, see whether better results
  - active research topic



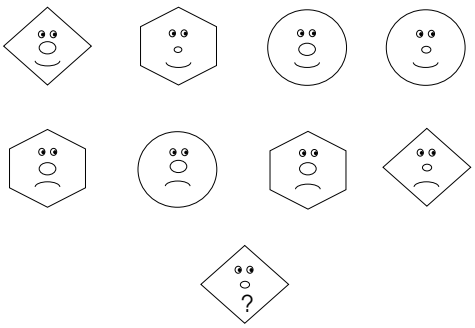
THOMAS MORE

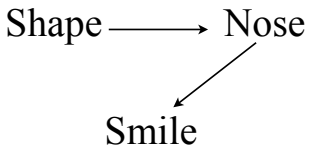
Additional Exercice

- Calculate all probability tables for the following Bayesian network. What is the prediction made by this network



(a)





(b)



THOMAS MORE

To remember

- Importance of Bayes' theorem
- MAP, ML, MDL
  - definitions, characterising learners from this perspective, relationship MDL-MAP
- Bayes optimal classifier, Gibbs classifier
- Naïve Bayes: how it works, assumptions made, application to text classification
- Bayesian networks: representation, inference, learning



THOMAS MORE