2019S1FIT5047 Final Review

Last Edited	@Jun 20, 2019 4:19 PM
Tags	FIT5047 TUTOR

Table of Content

Table of Content
Agents
Search
  Tentative
    Backtrack
    Graphsearch
    Breadth-first search (BFS)
    Uniform-cost Search (UCS)
    Depth-first Search (DFS)
    Depth-limited Search (DLS)
    Iterative Deepening DF Search (ID-DFS)
    Best-first Greedy Search
    Algorithm A
    Algorithm A*
  Irrevocable
    Hill Climbing
    Local Beam Search
    Simulated Annealing
    Genetic Algorithms
  Adversarial Search Algorithms
Knowledge Representation
  Propositional Logic
  First Order Logic
Probability
Bayes Nets
Machine Learning
  Supervised Learning
    Classification
    Decision Trees
    Naive Bayes classifier

- K Nearest Neighbour
- Logistic Regression
- Regression
- Unsupervised Learning
  - K-menas

Agents

- Rationality depends on
  - Performance measure
  - prior knowledge of the environment
  - available actions
  - percept sequence
- For each possible `percept sequence`, a rational agent should select an `action` that is expected to maximize its `performance measure`, given the evidence provided by the `percept sequence` and the agent's `built-in knowledge`
- Rationality is NOT omniscience
- PEAS e.g. (p10)
  - Performance measure
  - Environment
  - Actuators
  - Sensors
- Environment Types (p12-13)
  - Fully/Partially observable
  - Known/Unknown
  - Single/Multi agent
  - Deterministic/stochastic
  - Episodic/Sequential
  - Static/Dynamic
  - Discrete/Continous

- Agent Types (p17)
  - Simple reflex
  - Model based
  - Goal based
  - Utility based
  - Learning

Search

- Control Strategies
  - Tentative
    - Uninformed: Backtracking, Tree- and Graph search
    - Informed: Best-first greedy search, A, A*
  - Irrevocable
    - Informed: Hill climbing, Simulated annealing, Genetic algorithms

Tentative

Backtrack

- At any point in time, we keep one path only.
- `Procedure of Backtrack algorithm`
  1. is current state the goal? → SUCCEED
  2. is current state a deadend → FAIL
  3. get all available operations of current state
  4. if there is no available operations for current state → FAIL
  5. pick one
  6. do it to current state to produce a new state
  7. do Backtrack algorithm on the new state

8. FAIL? → 4
9. Return

Graphsearch

- Search Tree
    - Root of search tree is the initial state
    - Leaves are frontier
    - At each step choose one leaf node to expand
    - `Procedure of basic tree search algorithm`
        1. initial state → frontier
        2. frontier empty? → FAIL
        3. otherwise remove one from frontier
        4. if it is goal → SUCCEED
        5. expand the node
        6. add the resulting nodes to frontier
- `Procedure of Graph Search`
    1. initial state → frontier
    2. empty → closed
    3. frontier empty → FAIL
    4. otherwise remove one from frontier
    5. if it is goal → SUCCEED
    6. add the node to closed
    7. expand the node
    8. add the resulting nodes to frontier
    9. go to 3
- Search Strategies
    - A search strategy is defined by picking the order of node expansion

- Strategies are evaluated along several dimensions
    - completeness
    - time complexity
    - space complexity
    - optimality
- O Notion

Breadth-first search (BFS)

- Expand shallowest unexpanded node
- Complete: Yes (if b finite)
- Time complexity: O(b^d)
- Space: O(b^d)
- Optimal: Yes (if all action have the same cost)

Uniform-cost Search (UCS)

- path in frontier with cost
- expand in increasing order of g(n)

Depth-first Search (DFS)

- Expand the deepest unexpanded node
- Complete: No (if infinite-state spaces), Yes (if finite-state spaces)
- Time complexity: O(b^m)
- Space complexity: O(bm)
- Optimal: No

Depth-limited Search (DLS)

- Search with depth limit L
- Complete: No (if d > L)
- Time complexity: O(b^L)
- Space complexity: O(bL)

- Optimal: No

Iterative Deepening DF Search (ID-DFS)

- Complete: Yes
- Time complexity: O(b^d)
- Space complexity: O(bd)
- Optimal: Yes (if step costs are identical)

Best-first Greedy Search

- f(n)=h(n)
- Complete: No (if infinite-state spaces), Yes (if finite-state spaces)
- Time complexity: O(b^m)
- Space complexity: O(b^m)
- Optimal: No

Algorithm A

- f(n)=g(n)+h(n)

Algorithm A*

- A + admissible h()
- Admissibility (guarantee the optimal solution) 反例

$$\forall n\, h(n) \le h^*(n)$$

- Monotonicity 证明

$$\forall n\, h(n) \le c(n,m) + h(m)$$

- BFS = A* when g(n)=depth and h(n)=0
- UCS = A* with g(n)⩾0 and h(n)=0
- Best-first Greedy with g(n)=0 and h(n)⩾0

Irrevocable

Hill Climbing

- `Procedure`
    - if current state = goal RETURN
    - loop until solution found or no more operators
        - select an operator that has not been applied yet
        - Evaluate new state
            - if new state = goal RETURN
            - else if new state is better than current state
                - current state ← new state
- Problem: stuck in local maxima

Local Beam Search

- Keep track of k states rather than just one
- Start with k randomly generated states
- At each iteration, all the successors of all k states are generated
    - If any one is a goal state, stop
    - Else select the k best successors from the complete list and repeat

Simulated Annealing

- Temperature

$$Pr = e^{-\Delta E/T}$$

- Annealing schedule
- BestSoFar

Genetic Algorithms

- Start with a population of k randomly generated states
- A state is represented as a string over a finite alphabet of genes (often a string of 0s and 1s)

- A successor state is generated by combining two parent states
- Evaluation function (fitness function)
- Produce the next generation of states by selection, crossover and mutation

Adversarial Search Algorithms

- Minimax
- Complete: Yes (if tree is finite)
- Optimal: Yes
- Time complexity: O(b^m)
- Space complexity: O(bm)
- a-b prunning
    - min node → b, max node → a
    - switch a and b when move up, but no need to switch when passing down
    - a cut happens on min nodes, while b cut on max nodes

Knowledge Representation

- Syntax: defines the sentence in the language
- Semantics: defines the meaning of sentences
- Entailment means that one thing follows logically from another

$$KB \models \alpha$$

- Inference means that sentence a can be derived from KB by procedure i

$$KB \vdash_i \alpha$$

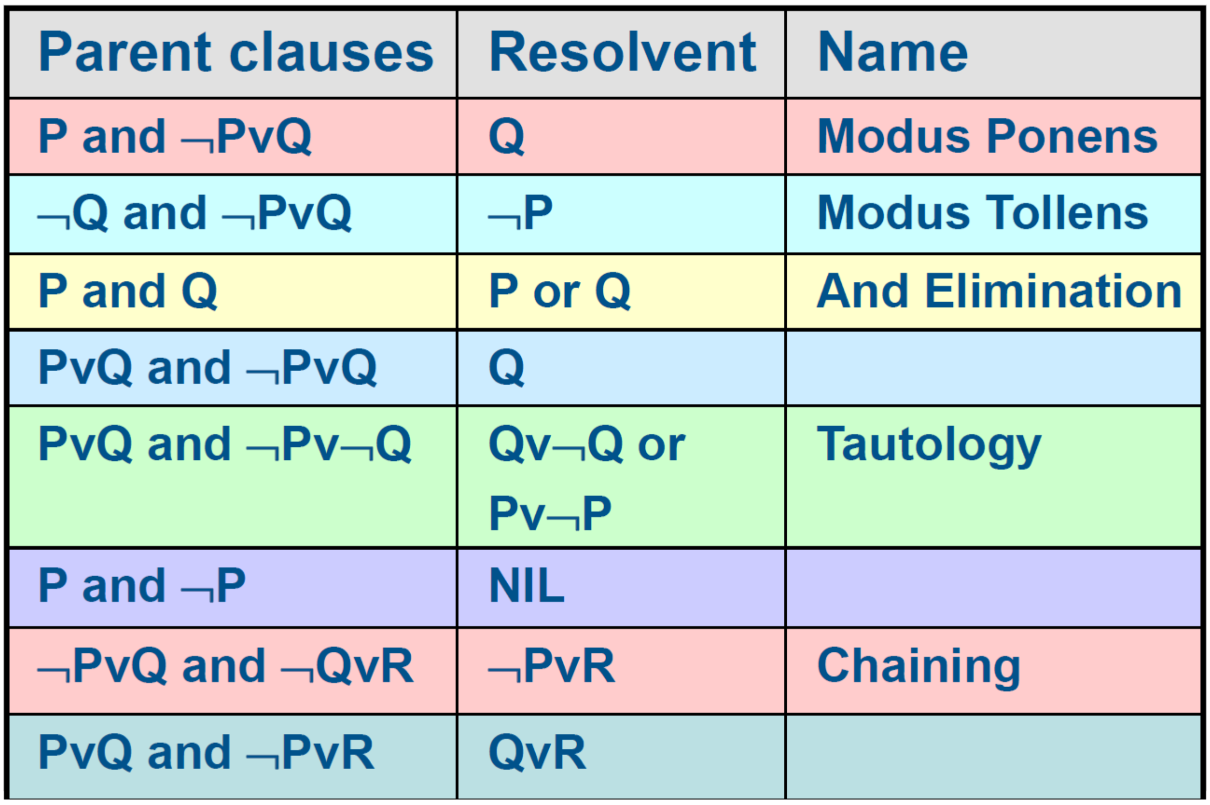
- soundness: procedure i is sound if whenever KB├ i α, it is also true that KB╞ α
- Completeness: procedure i is complete if whenever KB╞ α, it is also true that KB├ i α

Propositional Logic

- Logical Equivalence

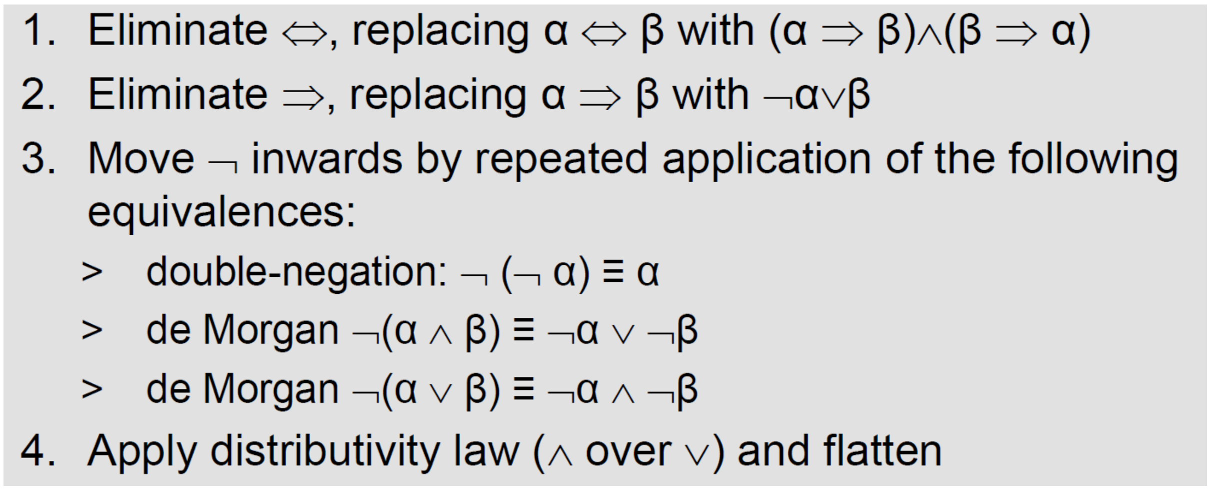
$$\begin{aligned}
(\alpha \land \beta) &\equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) &\equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) &\equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) &\equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) &\equiv (\neg\alpha \lor \neg\beta) \quad \text{de Morgan} \\
\neg(\alpha \lor \beta) &\equiv (\neg\alpha \land \neg\beta) \quad \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) &\equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) &\equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{aligned}$$

- Validity
- Satisfiability
- Common Inference Rules



Parent clauses	Resolvent	Name
P and ¬PvQ	Q	Modus Ponens
¬Q and ¬PvQ	¬P	Modus Tollens
P and Q	P or Q	And Elimination
PvQ and ¬PvQ	Q	
PvQ and ¬Pv¬Q	Qv¬Q or Pv¬P	Tautology
P and ¬P	NIL	
¬PvQ and ¬QvR	¬PvR	Chaining
PvQ and ¬PvR	QvR	

- Conjunctive Normal Form



1. Eliminate ⇔, replacing α ⇔ β with (α ⇒ β)∧(β ⇒ α)
2. Eliminate ⇒, replacing α ⇒ β with ¬α∨β
3. Move ¬ inwards by repeated application of the following equivalences:
   > double-negation: ¬ (¬ α) ≡ α
   > de Morgan ¬(α ∧ β) ≡ ¬α ∨ ¬β
   > de Morgan ¬(α ∨ β) ≡ ¬α ∧ ¬β
4. Apply distributivity law (∧ over ∨) and flatten

- Resolution

$$\frac{\overset{if\ l_i = \neg m_j}{l_1 \vee l_2 \ldots \vee l_i \vee \ldots \vee l_k \quad m_1 \vee m_2 \vee \ldots \vee m_j \vee \ldots \vee m_n}}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

- Resolution Refutation (p33例)

1. Negate the goal and add the negation to the set of clauses
2. Apply resolution to the clauses in the set of clauses until a contradiction is reached

- Definite Clause: a disjunction of literals of which exactly one is positive
- Horn Clause: a disjunction of literals of which at most one is positive
- Forward Chaining: data-driven, may do lots of work that is irrelevant to the goal
- Backward Chaining: goal-driven, complexity can be much less than linear in size of KB

First Order Logic

- Some Equivalences

$$\neg(\exists x)P(x) \equiv (\forall x)[\neg P(x)]$$

$$\neg(\forall x)P(x) \equiv (\exists x)[\neg P(x)]$$

$$(\forall x)[P(x) \wedge Q(x)] \equiv (\forall x)P(x) \wedge (\forall y)Q(y)$$

$$(\exists x)[P(x) \vee Q(x)] \equiv (\exists x)P(x) \vee (\exists y)Q(y)$$

- Rules of Inference
  - Modus Ponens:

$$[P \; and \; P \Rightarrow Q] \rightarrow Q$$

  - Modus Tollens:

$$[\neg Q \; and \; P \Rightarrow Q] \rightarrow \neg P$$

  - Universal Specialization

$$\forall x \, W(x) \rightarrow W(A) \quad where \; A \; is \; a \; constant$$

- Substitution

$$s = \{v_1|t_1, v_2|t_2, ..., v_n|t_n\}$$

- all elements are applied simultaneously

$$P(w, y, g(z), x)\{x|g(y), y|h(z), z|x\} \to P(w, h(z), g(x), g(y))$$

- Composition of Substitutions
  - term & variable
  - apply sj to the `terms` of si
  - add any pairs of sj having `variables` not in si
  - Properties of compositions of substitutions:
    - L(s1s2)=(ls1)s2
    - (s1s2)s3=s1(s2s3)
    - s1s2≠s2s1
- Unification
  - mgu(most general unifier)
  - Algorithm Unify
- Converting wffs into Clauses
  1. Eliminate implication symbols
  2. Reduce scopes of negation symbols
  3. standardize variables (for each quantifier)
  4. Eliminate existential quantifiers
  5. Move all universal quantifiers to the front
  6. put result in conjunctive normal form
  7. Eliminate universal quantifiers
  8. Eliminate `and` symbols
  9. Rename variables
- General Resolution

  - convert wffs to clauses first
  - lj and -mi have a mgu

Probability

- Joint Distribution
- Conditional Distribution
- Marginal Distribution
- Chain Rule

$$Pr(x_1, ..., x_n) = \prod_{i=1}^{n} Pr(x_i | x_1, ..., x_{i-1})$$

- Bayes' Rule

$$Pr(x, y) = Pr(x|y)Pr(y) = Pr(y|x)Pr(x)$$

- Independence

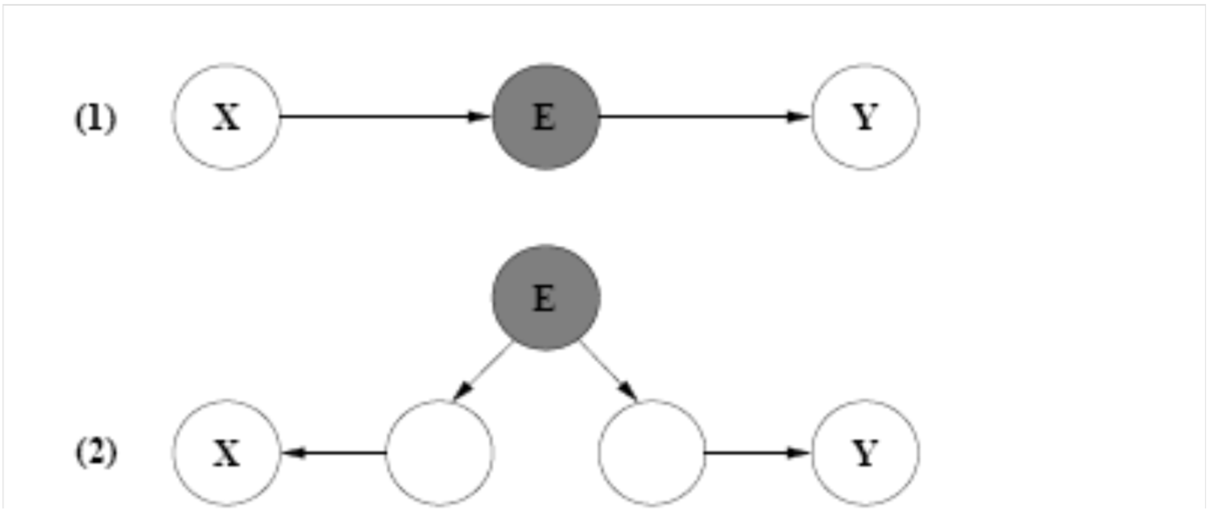
$$Pr(X, Y) = Pr(X)Pr(Y)$$
$$\forall x, y \; Pr(x, y) = Pr(x)Pr(y) \; or \; Pr(x|y) = Pr(x)$$

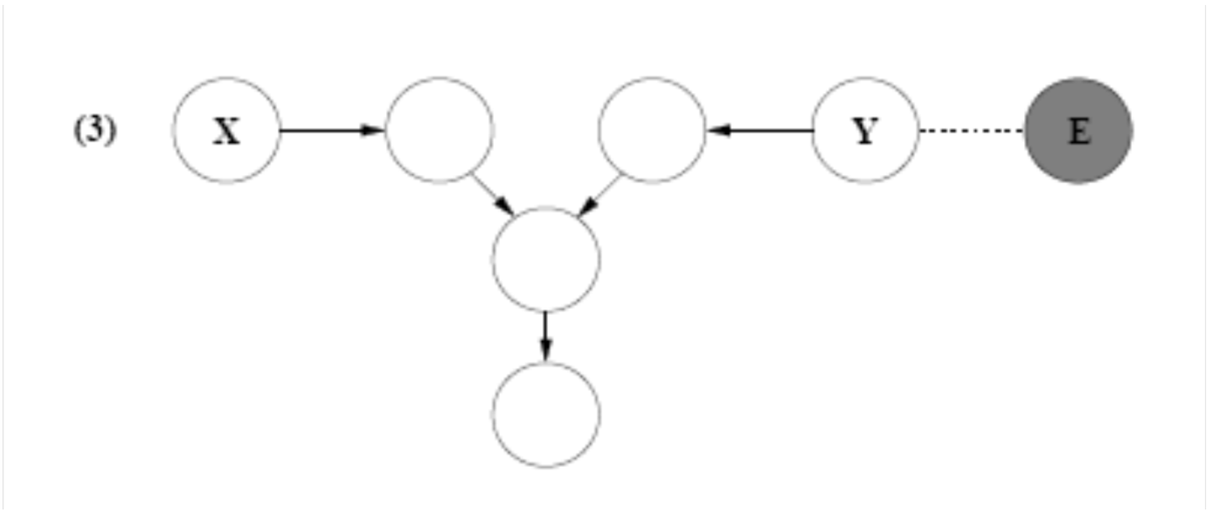
- Conditional Independence

Bayes Nets

- Notation
  - Nodes: variables
  - Arcs:interactions (dependence)
- Conditional Probability Tables (CPTs)
  - Each row contains the conditional probability of each node value for each possible combination of values in its parent nodes
  - Each row must sum to 1

- A CPT for a Boolean variable with n Boolean parents contains 2 n+1 probabilities
- A node with no parents has one row (its prior probabilities)
- How big is a joint distribution over N Boolean variables? (2^N)
- How big is an N-node net if each node has up to k parents? (O(N*2^(k+1))
- Factors that affect conditional independence
  - + Causal chain
  - + Common causes
  - - Common effects
- D-separation





- Decision Networks
  - Expected utility

$$EU(A|E) = \sum_i Pr(O_i|E, A)U(O_i|A)$$

  - E=available evidence,A=a non-deterministic action,Oi=a possible outcome state, U=utility
  - Types of Nodes
    - Chance nodes (ovals) random variables
    - Decision nodes (rectangles)
    - Utility nodes (diamonds)
  - Types of Links
    - Informational Links: indicate when a chance node needs to be observed before a decision is made
      - Any link entering a decision node is an informational link
    - Conditional Links: indicate the variables on which the probability assignment to a chance node will be conditioned

Machine Learning

- Why learning? (p5)
    - Learning is essential for unknown environments
    - Learning is necessary in dynamic environments
    - Learning is useful as a system construction method
    - Learning modifies the agent's decision mechanisms to improve performance
- What is Machine Learning? (p6)
- What is inductive learning? (p6)
- Types of Learning Approaches (p8)
    - Verification Driven (Hypothesis first)
    - Discovery Driven (Data first)
        - Supervised learning
        - Unsupervised learning
        - Reinforcement learning
- Completeness and Consistency in Inductive Learning (p10) 结合p9定义
    - complete
    - consistent
- Some different angles to understand ML (p9-13)
    - definition
    - search
    - Hypothesis
    - Compression
- ML vs DM (p14) 理论vs实践
- Types of Data
    - Data: (un)labeled instances
    - Training Data: Used to train the model
    - Validation Data: Used to fine-tune the model

    - Test Data: Used to measure the generalization of the model
- Experimentation cycle
    - Learn model parameters on Training Data
    - Fine tune the model on Validation Data
    - Compute performance on Test Data
    - `Never "peek" at the test set!`
- Evaluation Metrics
    - Accuracy

$$\frac{\text{correctly predicted}}{\text{predicted}}$$

    - Recall

$$\frac{\text{correctly predicted as class c}}{\text{instances in class c}}$$

    - Precision

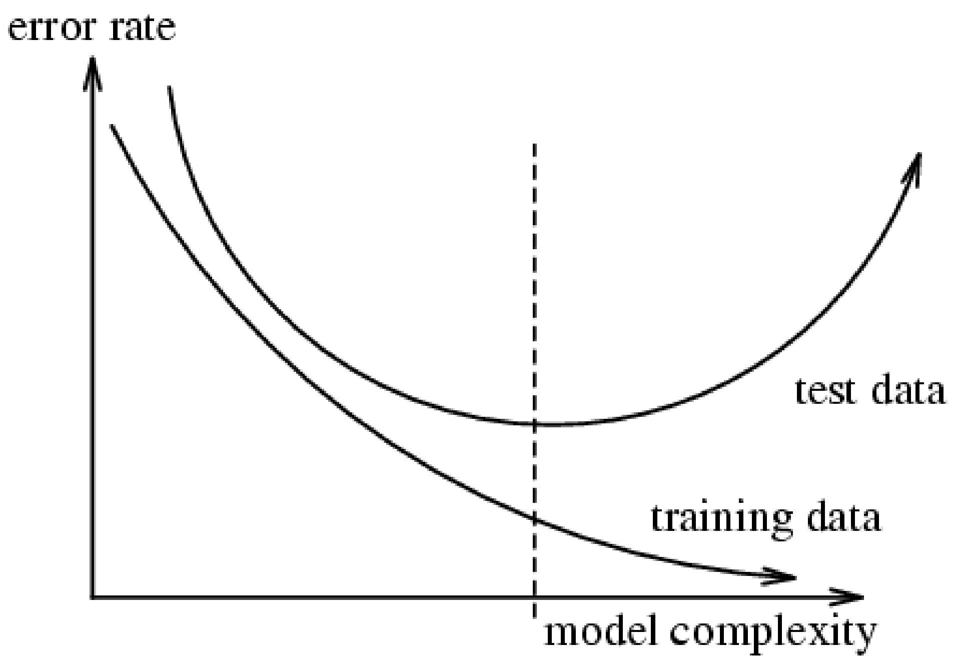
$$\frac{\text{correctly predicted as class c}}{\text{predicted as class c}}$$

- Overfitting and Generalization
    - Overfitting: fitting the training data very closely, but not generalizing well
    - Cause:noisy data, small training data set, complex model

Supervised Learning

- Build patterns by inferring an unknown attribute given the values of known attributes
    - Type of Inferred Value
        - Classification (categorical or discrete)
        - Regression (continuous or ordered)

    - Time of Inferred Value
        - Classification
        - Prediction
- Supervised Inductive Learning (p20)
- Big Idea 1:Pick h from the space H which agrees with f on the training set.
- Big Idea 2:Prefer a simpler hypothesis to complex ones provided both explain the data equally well. (Ockham's Razor)
    - Simpler: improve generalizability, prevent overfitting
- Bias and Variance
    - Bias（偏差）：算法的期望预测与真实结果的偏差，刻画了算法本身的拟合能力
    - Variance（方差）：数据集变动所导致的学习性能的变化，刻画了数据扰动所造成的影响
    - Bias is the true error of the best classifier in the concept class
    - Variance is the error of a trained classifier with respect to the best classifier in the concept class



- High bias: both training and test error are high (too simple)
- High Variance: training error is low but test error is high (overfitting)
- MATH Learning is indeed Search/Optimization

$$h^* = argmin_{h \in H}\{[\sum_{(x,y)\in D} error(h(x), y)] + \lambda Complexity(h)\}$$

Classification

- Classification Process
  - Model construction (training data)
  - Model evaluation (test data)
  - Model usage (new data)
- Cross-validation
  - x-validation: 10-fold
  - stratified x-validation: 类别按比例

Decision Trees

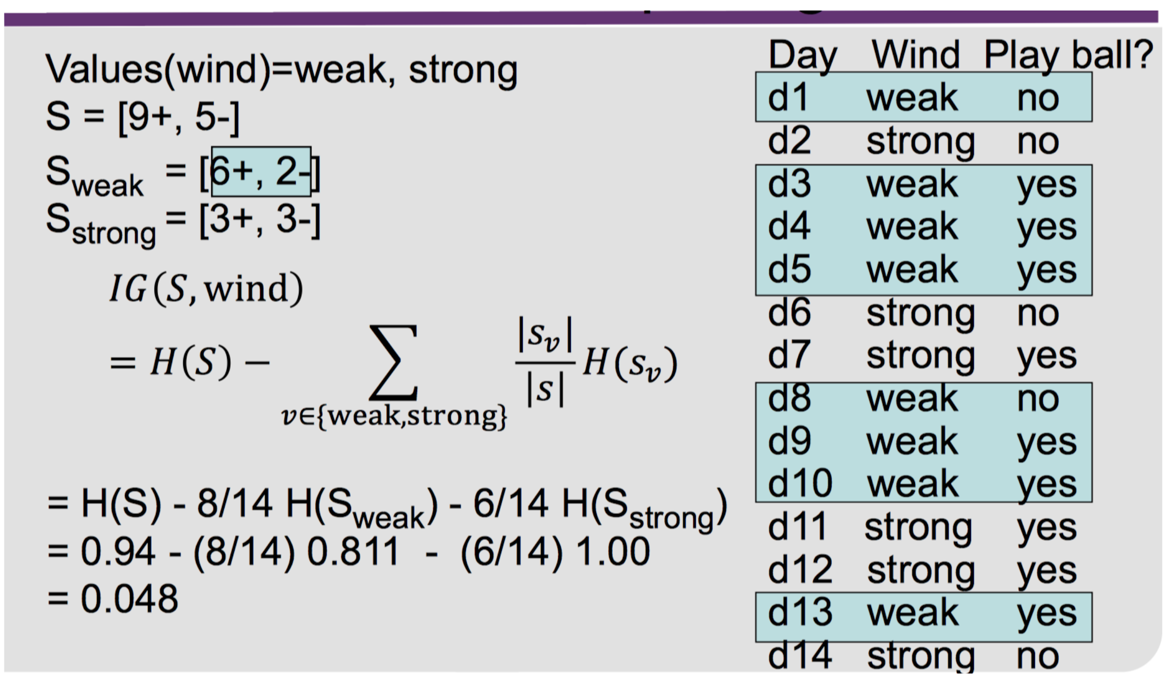
- Classification is based on a tree structure
    - each non-leaf node is a decision node
    - each leaf node represents a class
    - to classify an object
        - each decision node (starting from the root) compares an attribute of the object with a specific attribute value (or range)
        - a path from the root to a leaf node gives the class of the object
- can split on the same attribute again (continuous case)
- Entropy ({1,0},{0.9,0.1},{0.5,0.5})

$$H(X) = -\sum_{i=1}^{n} Pr(x_i) log_2 Pr(x_i)$$

- information gain

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{S_v}{S} H(S_v)$$

例子



Values(wind)=weak, strong

S = [9+, 5-]

S_{weak} = [6+, 2-]

S_{strong} = [3+, 3-]

$$IG(S, \text{wind}) = H(S) - \sum_{v \in \{\text{weak,strong}\}} \frac{|s_v|}{|s|} H(s_v)$$

= H(S) - 8/14 H(S_{weak}) - 6/14 H(S_{strong})

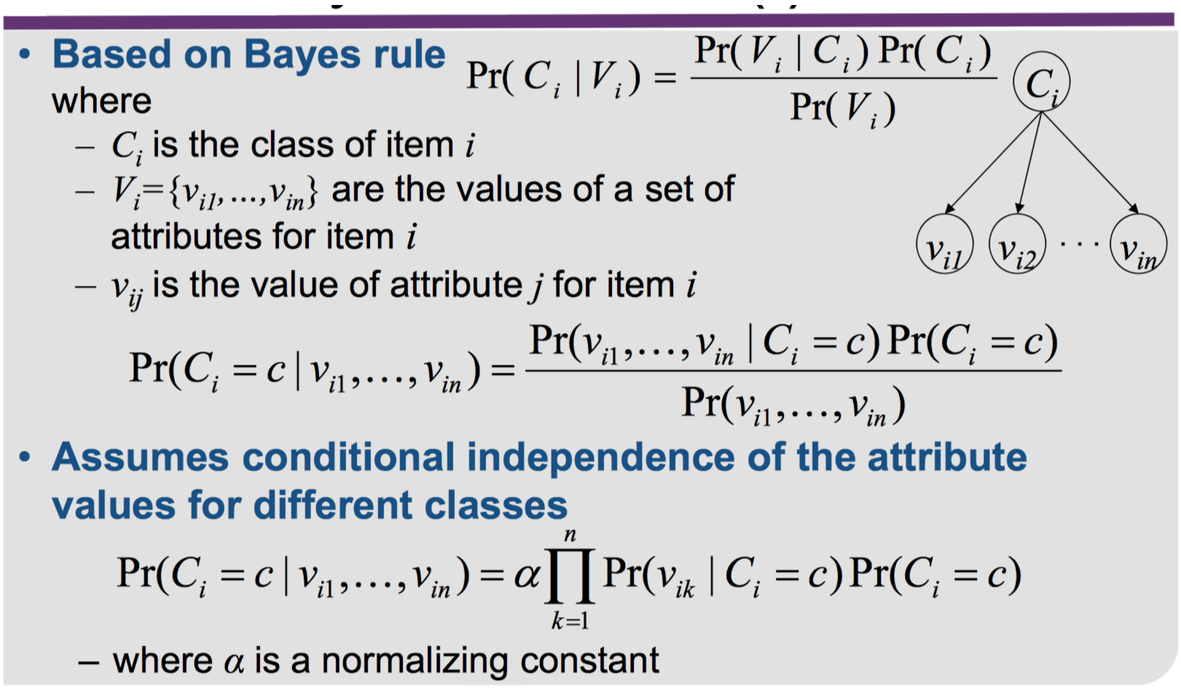
= 0.94 - (8/14) 0.811 - (6/14) 1.00

= 0.048

Day	Wind	Play ball?
d1	weak	no
d2	strong	no
d3	weak	yes
d4	weak	yes
d5	weak	yes
d6	strong	no
d7	strong	yes
d8	weak	no
d9	weak	yes
d10	weak	yes
d11	strong	yes
d12	strong	yes
d13	weak	yes
d14	strong	no

- Avoid overfitting
  - Pre-pruning: Stop growing the tree if the goodness measure falls below a threshold
  - Post-pruning: Grow the full tree, then prune
  - Regularization: Add complexity penalty to the performance measure
- Missing attribute values
  - Categorical
    - use the most common value of the attribute
    - probabilistic selection according to the distribution of values
  - Coninuous
    - use the mean of the attribute values
- Extract classification rules from DTs
  - IF-THEN rules
  - One rule for one path
  - conjunction

Naive Bayes classifier



- **Based on Bayes rule** $\Pr(C_i \mid V_i) = \frac{\Pr(V_i \mid C_i)\Pr(C_i)}{\Pr(V_i)}$

  where
  - C_i is the class of item i
  - $V_i=\{v_{i1},\ldots,v_{in}\}$ are the values of a set of attributes for item i
  - v_{ij} is the value of attribute j for item i

$$\begin{tikzcd} & & C_i \arrow[dll] \arrow[dl] \arrow[dr] & \\ v_{i1} & v_{i2} & \cdots & v_{in} \end{tikzcd}$$

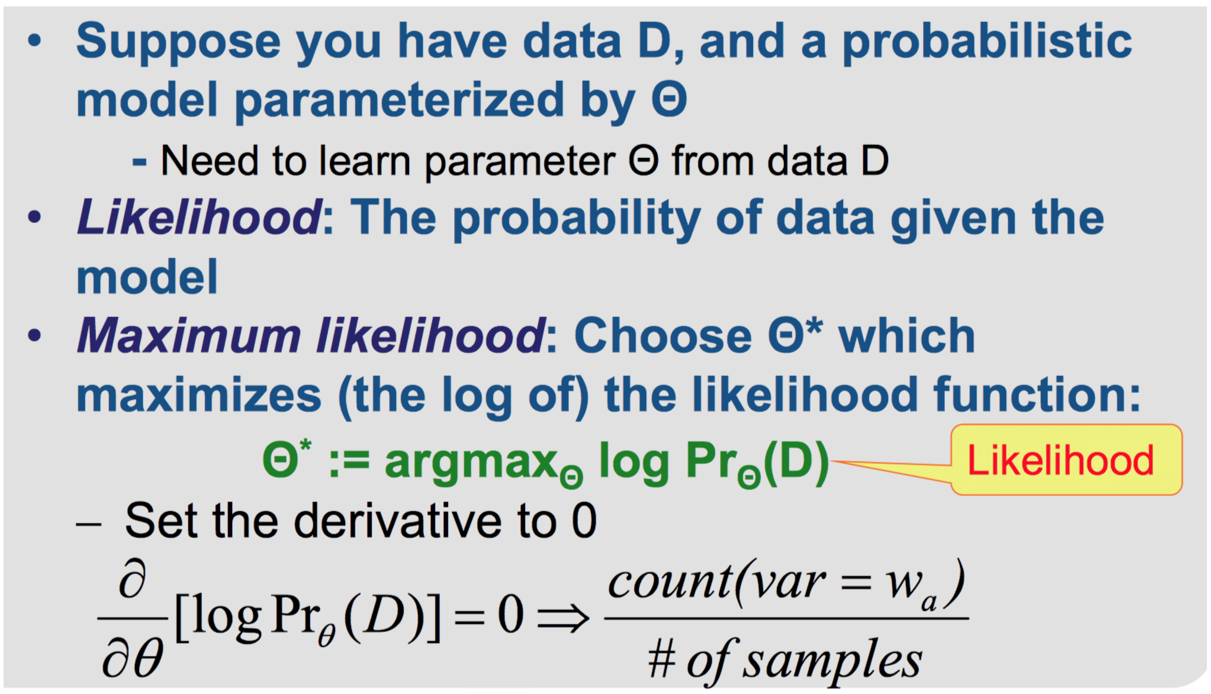
$$\Pr(C_i = c \mid v_{i1},\ldots,v_{in}) = \frac{\Pr(v_{i1},\ldots,v_{in} \mid C_i = c)\Pr(C_i = c)}{\Pr(v_{i1},\ldots,v_{in})}$$

- **Assumes conditional independence of the attribute values for different classes**

$$\Pr(C_i = c \mid v_{i1},\ldots,v_{in}) = \alpha\prod_{k=1}^{n}\Pr(v_{ik} \mid C_i = c)\Pr(C_i = c)$$

  - where α is a normalizing constant

- Learning the parameters: Estimating the CPTs Pr(C) and PR(v|c) from training data
- Maximum likelihood principle



- **Suppose you have data D, and a probabilistic model parameterized by Θ**
  - Need to learn parameter Θ from data D
- ***Likelihood*: The probability of data given the model**
- ***Maximum likelihood*: Choose Θ* which maximizes (the log of) the likelihood function:**

$$\Theta^* := \text{argmax}_\Theta \log \Pr_\Theta(D)$$

Likelihood

  - Set the derivative to 0

$$\frac{\partial}{\partial\theta}[\log \Pr_\theta(D)] = 0 \Rightarrow \frac{count(var = w_a)}{\#\, of\, samples}$$

- Sparse data problem

- If value w_a is not found in class c, w_a =0 → MLE for Pr(w_a|c)=0
- If variable *var* is not found in the training set, MLE for Pr(w_a) is undefined (denominator is zero)

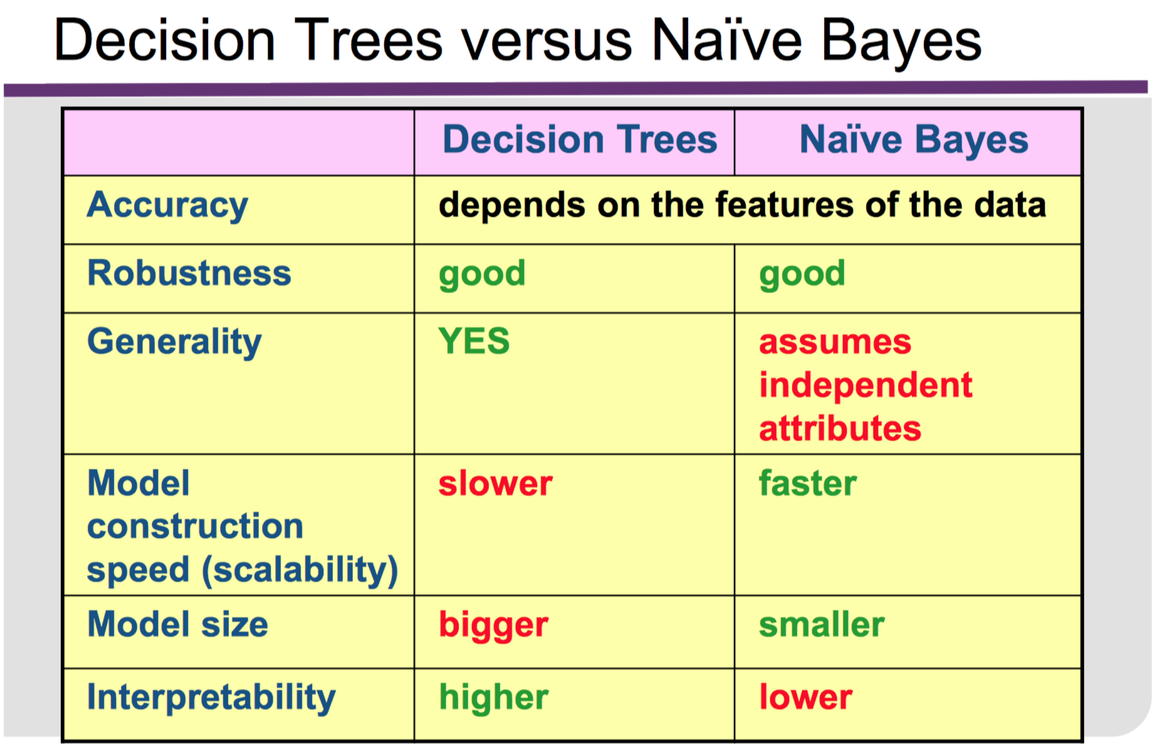
- Expected likelihood Estimator

$$\Pr_{EL}(var = w_a) \cong \frac{|w_a| + \varepsilon}{\sum_{i=1}^{m} \{|w_i| + \varepsilon\}} = \frac{|w_a| + \varepsilon}{\sum_{i=1}^{m} |w_i| + m\varepsilon}$$

- Missing attribute values
    - Categorical
        - ignore missing values
        - consider a missing value (?) to be an additional value
    - Continuous
        - use the mean of the attribute values
- Continuous-valued attributes (commonly normal distribution)
    - Normal desity function

$$\Pr(v_{iA} = t \mid C_i = c) \approx \Pr(t - \frac{\varepsilon}{2} \le v_{iA} \le t + \frac{\varepsilon}{2} \mid C_i = c)$$
$$\approx \varepsilon \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$$

- Comparison with DT



Decision Trees versus Naïve Bayes

	Decision Trees	Naïve Bayes
Accuracy	depends on the features of the data	
Robustness	good	good
Generality	YES	assumes independent attributes
Model construction speed (scalability)	slower	faster
Model size	bigger	smaller
Interpretability	higher	lower

K Nearest Neighbour

- All instances correspond to points in an n-dimensional space
- Compare features of the new instance with features of k training instances that are closest to it in the space (nearest neighbours)
- The target function may be discrete or continuous
  - Discrete: majority vote of the new instance's neighbours
  - Continuous: mean value of the k nearest training examples
- Distance between two instance
  - continuous feature: Euclidean distance (smaller is better)
  - Categorical features: Jaccard coefficient (larger is better)
- Distance weight

    - 1-distance
    - 1/distance
- `THERE IS NO TRAINING FOR KNN`
    - memorize all
    - scale increases

Logistic Regression

- binary classification
- logistic / sigmoid function

$$g(z) = \frac{1}{1+e^{-z}}$$

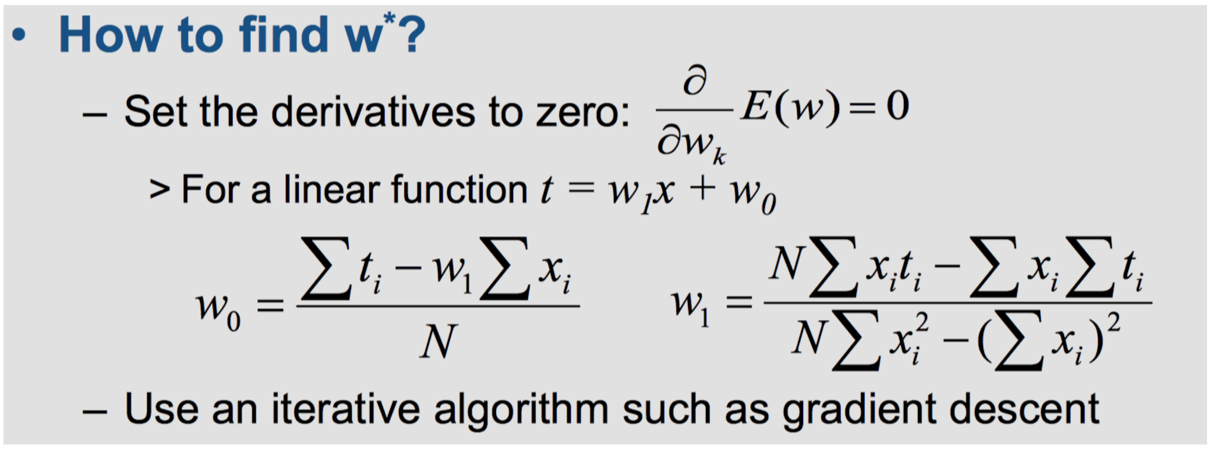
Regression

- Linear regression model: t=w*x+w0 (展开)
- Error function (convex function)

$$E(w) = \sum_{i=1}^{m}(t_i - (w * x_i + w_0))^2$$

- Learning the parameters: find w* that minimizes the error function (例p110 & p115)

$$w^* = argmin_w E(w)$$

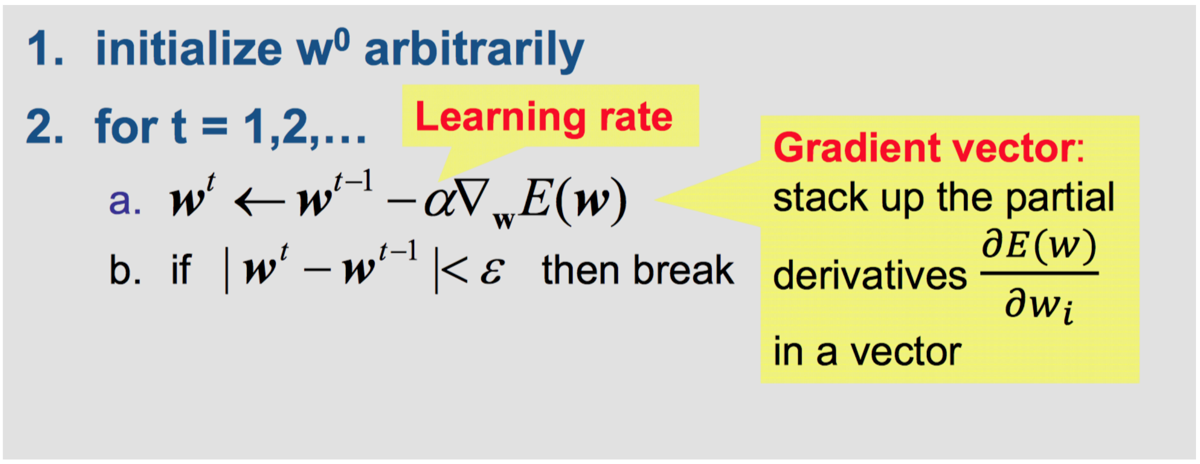


- **How to find w*?**
    - Set the derivatives to zero: $\frac{\partial}{\partial w_k} E(w) = 0$
        > For a linear function $t = w_1 x + w_0$

$$w_0 = \frac{\sum t_i - w_1 \sum x_i}{N} \qquad w_1 = \frac{N\sum x_i t_i - \sum x_i \sum t_i}{N\sum x_i^2 - (\sum x_i)^2}$$

    - Use an iterative algorithm such as gradient descent

- Gradient Descent



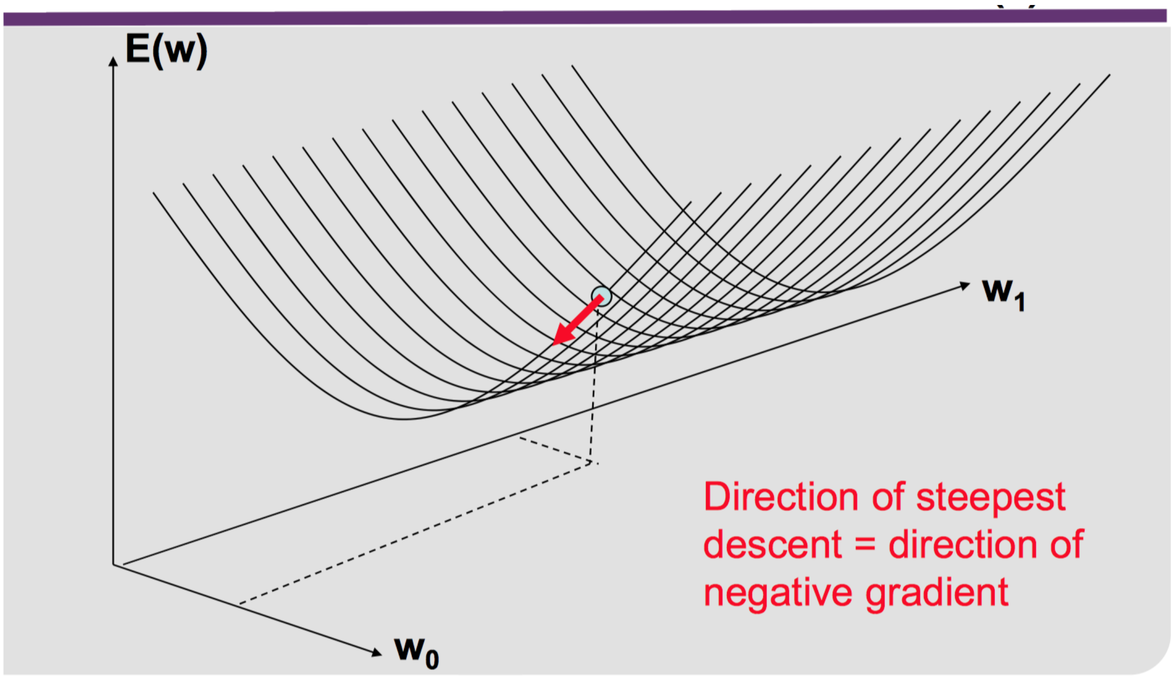
1. initialize w^0 arbitrarily
2. for t = 1,2,…

    a. $w^t \leftarrow w^{t-1} - \alpha \nabla_w E(w)$

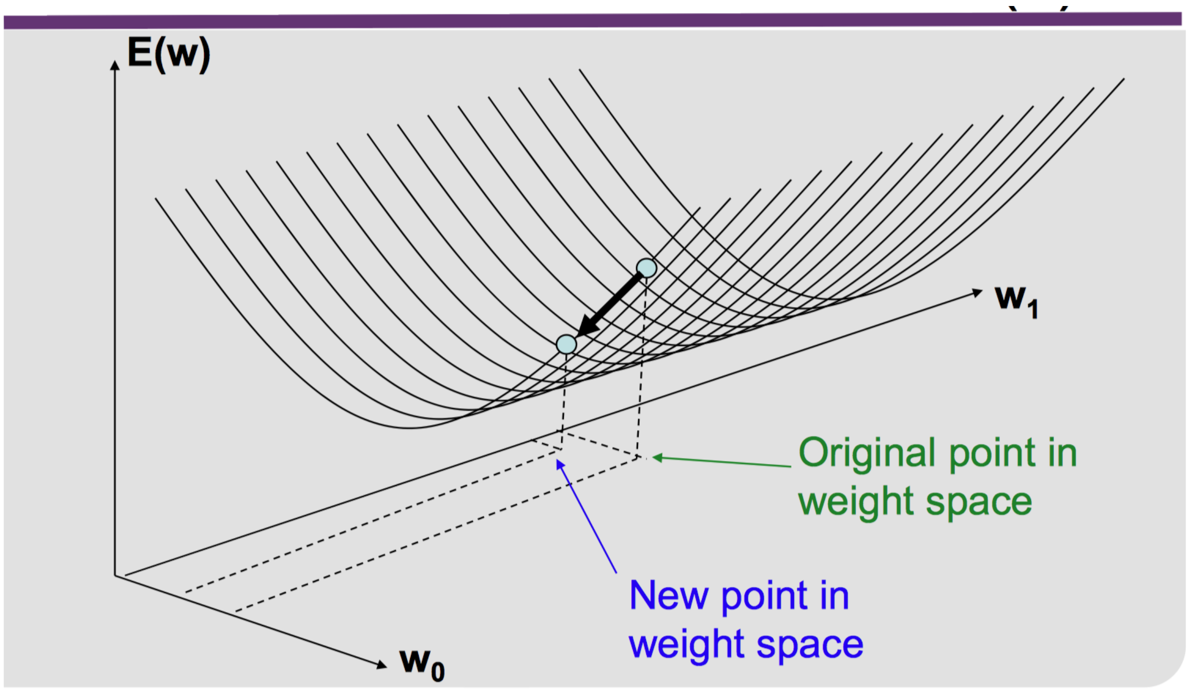
    b. if $|w^t - w^{t-1}| < \varepsilon$ then break

Learning rate

Gradient vector: stack up the partial derivatives $\frac{\partial E(w)}{\partial w_i}$ in a vector



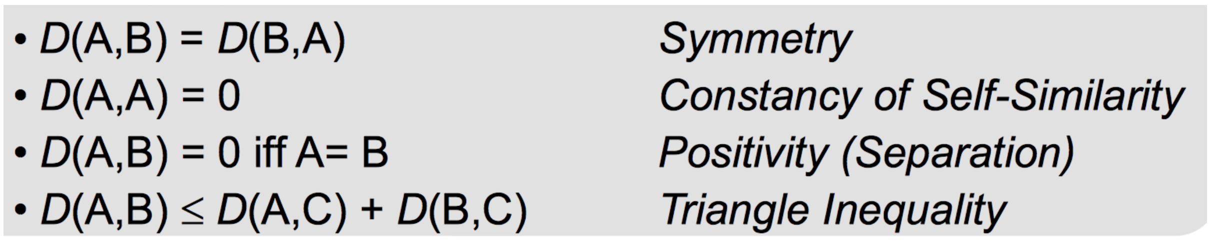
Direction of steepest descent = direction of negative gradient



- Summary of supervised learning
    - Various techniques for supervised machine learning
    - Discrete:
        - Decision trees (non-parametric)
        - Naive-Bayes (parametric)
        - K nearest neighbour (non-parametric)
    - Continuous:
        - Regression
            - Models based on least square error (parametric)

Unsupervised Learning

- no labels, no pre-defined classes
- Clustering and Association Analysis
    - Clustering: Groups records of similar items
    - Association analysis:
- Distance



• *D*(A,B) = *D*(B,A)	*Symmetry*
• *D*(A,A) = 0	*Constancy of Self-Similarity*
• *D*(A,B) = 0 iff A= B	*Positivity (Separation)*
• *D*(A,B) ≤ *D*(A,C) + *D*(B,C)	*Triangle Inequality*

- Edit distance: Transform one of the objects into the other, and measure how much effort it took
- Two types of clustering
    - Partitional
    - Hierarchical

K-menas

- Partitional
- Squared error for cluster Ki

$$se_{K_i} = \sum_{j=1}^{m_i} ||x_{ij} - C_i||^2$$

- Objective Function (minimize seK!)

$$se_K = \sum_{i=1}^{K} se_{K_i}$$

- Try different Ks (poker)
- Normalization
- Summary of the K-Means
    - Advantages
        - Relatively efficient
        - Global optimum may be found using techniques such as deterministic annealing and genetic algorithms
    - Disadvantages
        - Need to specify k in advance

- Applicable only when mean is defined (categorical data)
- Does not deal well with overlapping clusters
- unable to handle noisy data and outliers