School of Information Sciences (SOIS), MAHE, Manipal 576104

First Semester Master of Engineering - ME (Big Data and Data Analytics)

Degree Examination April - 2018

BDA 611 Fundamentals of Machine Learning

Scheme for Evaluation

1A. Write the rule for *Estimating Training Values* in the design of machine learning system. (4 Marks)

In the learning problem, the only training information available to our learner is whether the game was won or lost. We require training examples that assign specific scores to specific board states. It is easy to assign a value to board states that correspond to the end of the game. It is difficult to assign training values to the intermediate board states.

One simple approach to estimate training values for intermediate board states:

- Assign the training value of $V_{train}(b)$ for any intermediate board state ***b*** to be ***V'*** *(Successor(b))*

Rule for estimating training values.

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$

Where V' is the leaner's current approximation to V and (Successor (b) is next board state following b.

1B. Write the approaches involved in *adjusting the weights* to best fit the set of training examples. (6 Marks)

We need to choose the weights w_i to best fit the set of training examples $\{ < b, V_{train}(b) > \}$. Best fit minimize the square error E between V and V'.

$$E \equiv \sum_{\langle b, V_{train}(b) \rangle \in \ training \ examples} (V_{train}(b) - \hat{V}(b))^2$$

The LMS training rule: For each training example $\{< b, V_{train}(b) >\}$

- Use current weight to calculate V' (b)
- For each weight w_i update it as
- Here η is a small constant (e.g., 0.1) that moderates the size of the weight update
- If the error $\mathbf{V_{train}(b) - V'(b)}$ - is zero, no weights are changed

If $V_{train}(b) - V'(b)$ is +ve the each weight is increased in proportion to the value of its corresponding feature

2. Implement Candidate-elimination algorithm to obtain most general and most specific hypotheses for the training examples given in the following table (10 Marks)

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

Initialization

G0 ← {<?, ?, ?, ?, ?, ?>}

S0 ← {<Ø, Ø, Ø, Ø, Ø, Ø >}

(1 Marks)

Iteration 1

d1 = <Sunny, Warm, Normal, Strong, Warm, Same>

G1 ← {<?, ?, ?, ?, ?, ?>}

S1 ← {< Sunny, Warm, Normal, Strong, Warm, Same >}

(1 Marks)

Iteration 2

d2 = <Sunny, Warm, High, Strong, Warm, Same>

G2 ← {<?, ?, ?, ?, ?, ?>}

S2 ← {< Sunny, Warm, ?, Strong, Warm, Same >}

(2 Marks)

Iteration 3

consistent

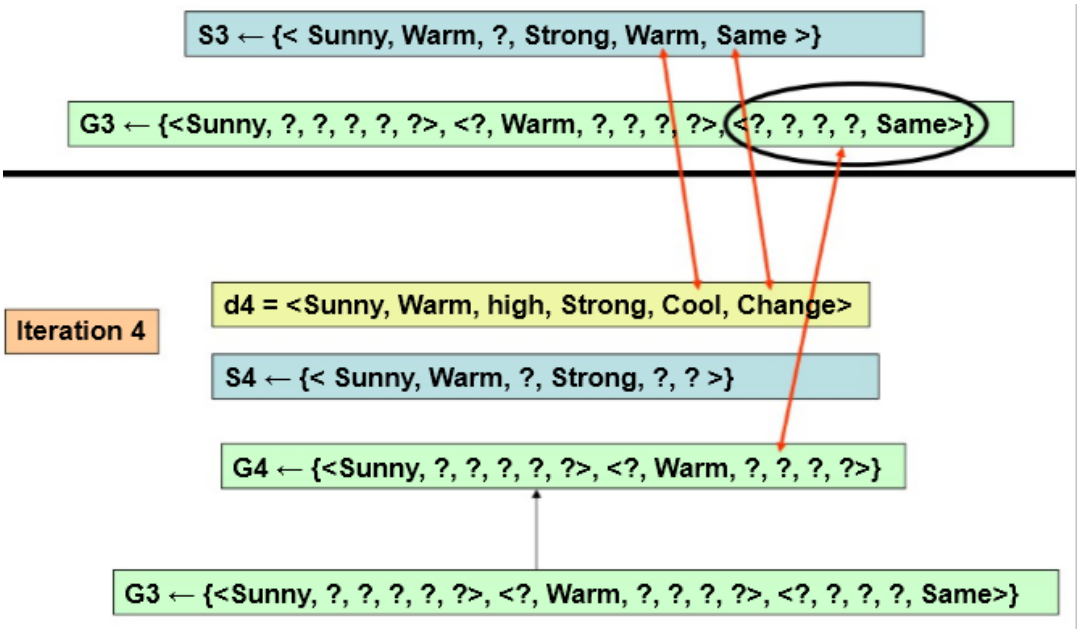
d3 = <Rainy, Cold, High, Strong, Warm, Change>

S3 ← {< Sunny, Warm, ?, Strong, Warm, Same >}

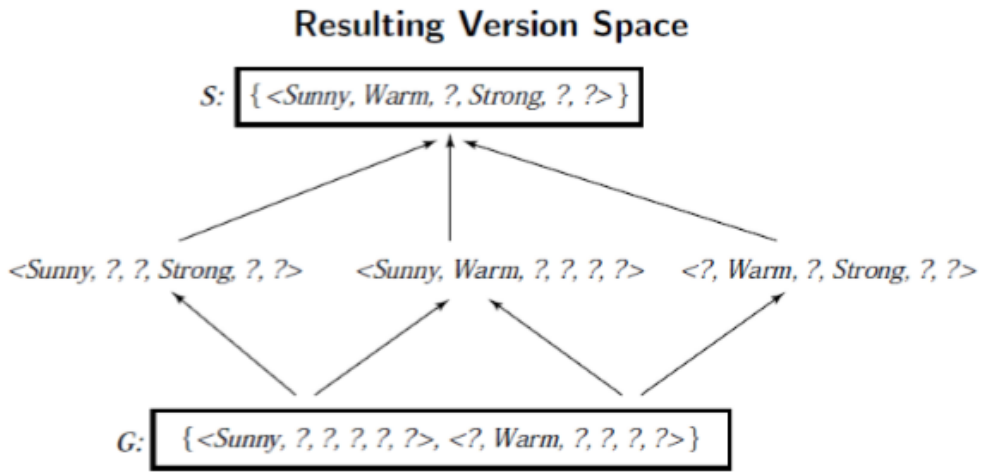
G3 ← {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, Same>}

G2 ← {<?, ?, ?, ?, ?, ?>}

(2 Marks)



(2 Marks)



(2 Marks)

3. State and prove the *ε-Exhausted Version Space* theorem to determine the number of training examples required to reduce this probability of failure below some desired level δ. (10 marks)

ε-Exhausted Version Space

Definition: The version space $VS_{H,D}$ is said to be ϵ-**exhausted** with respect to c and $\mathcal{D}$, if every hypothesis h in $VS_{H,D}$ has error less than ϵ with respect to c and $\mathcal{D}$.

$$(\forall h \in VS_{H,D})\ error_{\mathcal{D}}(h) < \epsilon$$

(2 Marks)

Theorem: [Haussler, 1988].

If the hypothesis space H is finite, and D is a sequence of $m \geq 1$ independent random examples of some target concept c, then for any $0 \leq \epsilon \leq 1$, the probability that the version space with respect to H and D is not ϵ-exhausted (with respect to c) is less than

$$|H|e^{-\epsilon m}$$

(2 Marks)

Proof:

- Let $h_1, h_2, \ldots h_k$ be all the hypotheses in H that have true error greater than ε with respect to c. [$H_{bad} = \{h_1, h_2, \ldots h_k\}$]
- We fail to ε-exhaust the version space if and only if at least one of these k hypotheses happens to be consistent with all ***m*** independent random training examples.
- The probability that any single hypothesis having true error greater than ε would be consistent with one randomly drawn example is at most (1 - ε).

- Probability that this hypothesis will be consistent with ***m*** independently drawn examples is at most $(1 - \varepsilon)^m$
- Given that we have *k* hypotheses with error greater than ε, the probability that at least one of these will be consistent with all ***m*** training examples is at most

$$k(1-\varepsilon)^m$$

since k is ≤ |H|, this is at most $H(1-\varepsilon)^m$

- We use a general inequality stating that if $0 \leq \varepsilon \leq 1$; then then $(1 - \varepsilon) \leq e^{-\varepsilon}$.

$$k(1-\epsilon)^m \leq |H|(1-\epsilon)^m \leq |H|e^{-\epsilon m}$$

(4 Marks)

Let us use this result to determine the number of training examples required to reduce this probability of failure below some desired level δ.

$$|H|e^{-\epsilon m} \leq \delta$$

Rearranging terms to solve for m, we find

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta))$$

- Number of training examples ***m*** is sufficient to assure that any consistent hypothesis will be **probably approximately correct**.

(2 Marks)

4. What is *Agnostic Learning*? Obtain the equation for number of training examples *"m"* required in this case.
(7 Marks)

Agnostic Learning:

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta))$$

(2 Marks)

- Every hypothesis in H having training error = zero and true error of at most ε.
- If H does not contain the target concept ***c,*** then a zero-error hypothesis cannot always be found
- In this case, our learner **L** is to output the hypothesis from ***H*** that has the minimum error over the training examples
- A learner that makes no assumption that the target concept is representable by ***H*** and that simply finds the hypothesis with minimum training error, is often called an **agnostic learner**.

(2 Marks)

So far, assumed $c \in H$

Agnostic learning setting: don't assume $c \in H$

- The Hoeffding bounds state that if the training error **$error_D(h)$** is measured over the set **D** containing **m** randomly drawn examples, then

$$\Pr[error_D(h) > error_D(h) + \epsilon] \leq e^{-2m\epsilon^2}$$

$$\Pr[(\exists h \in H)(error_D(h) > error_D(h) + \epsilon)] \leq |H|e^{-2m\epsilon^2}$$

$$m \geq \frac{1}{2\epsilon^2}(\ln|H| + \ln(1/\delta))$$

(3 Marks)

5A. Define *Vapnik-Chervonenkis (VC)* Dimension **(3Marks)**

Vapnik-Chervonenkis Dimension:

An unbiased hypothesis space shatters the entire instance space. The larger the subset of *X* that can be shattered, the more expressive the hypothesis space is, i.e. the less biased.

Definition:

The Vapnik-Chervonenkis dimension, VC(*H*) of hypothesis space *H* defined over instance space *X* is the size of the largest finite subset of *X* shattered by *H*. If arbitrarily large finite subsets of *X* can be shattered then VC(*H*) = ∞

5B. Consider a medical diagnosis problem in which there are two alternative hypotheses:

(1) the patient has a particular disease (denoted by *cancer*)

(2) the patient does not (denoted by ¬*cancer*)

Prior knowledge over the entire population of people only *0.008* have this disease. The available data is from a particular laboratory test with two possible outcomes (positive and negative). Furthermore, the lab test is only an imperfect indicator of the disease. The test returns a correct positive result in only 98% of the cases in which the disease is actually present and a correct negative result in only 97% of the cases in which the disease is not present. In other cases, the test returns the opposite result. Suppose, a new patient is observed for whom the lab test returns a *positive* result. *Should you diagnose the patient as having cancer or not?* **(10 Marks)**

$\oplus$ (positive) and $\ominus$ (negative)

$$P(cancer) = .008 \qquad P(\neg cancer) = 0.992$$
$$P(\oplus|cancer) = .98 \qquad P(\ominus|cancer) = .02$$
$$P(\oplus|\neg cancer) = .03 \qquad P(\ominus|\neg cancer) = .97$$

(3 Marks)

$$P(\oplus|cancer)P(cancer) = (.98).008 = .0078$$
$$P(\oplus|\neg cancer)P(\neg cancer) = (.03).992 = .0298$$
$$\Rightarrow h_{MAP} = \neg cancer$$

(4 Marks)

the exact posterior probabilites can be determined by normalizing the above properties to 1

$$P(cancer|\oplus) = \frac{.0078}{.0078+0.0298} = .21$$
$$P(\neg cancer|\oplus) = \frac{.0298}{.0078+0.0298} = .79$$

$\Rightarrow$ the result of Bayesian inference depends strongly on the prior probabilities, which must be available in order to apply the method directly

(3 Marks)

6A. Derive Bayes Optimal Classifier. **(4 Marks)**

Most probable classification of the new instance is obtained by combining the predictions of all hypotheses, weighted by their posterior probabilities. If the possible classification of new example take any value v_j from some set V, then the probability **P(v_j|D)** for new instance is,

$$P(v_j|D) = \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

The optimal classification of the new instance is the value v_j, for which **P (v_j|D)** is maximum.

Bayes optimal classification:

$$\underset{v_j \in V}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j|h_i) P(h_i|D)$$

Any system that classifies new instances according to above equation is called a ***Bayes*** optimal ***classifier*** or Bayes optimal learner.

6B. Consider a hypothesis space containing three hypotheses: h_1, h_2, and h_3. Posterior probabilities of h_1, h_2, and h_3 given the training data are 0.4, 0.3, and 0.3 respectively. New instance x is encountered, which is classified positive by h_1, but negative by h2 and h3. Use Bayes Optimal Classifier to obtain the most probable classification of the new instance given the training data? (6 marks)

$$\underset{v_j \in \{\oplus, \ominus\}}{\operatorname{argmax}} \sum_{h_i \in H} P(v_j|h_i) P(h_i|D) = \ominus$$

To illustrate in terms of the above example, the set of possible classifications of the new instance is $V = \{\oplus, \ominus\}$, and

Posterior probabilities of these hypotheses

$$\begin{array}{ll} P(h_1|D) = .4, & P(\ominus|h_1) = 0, \quad P(\oplus|h_1) = 1 \\ P(h_2|D) = .3, & P(\ominus|h_2) = 1, \quad P(\oplus|h_2) = 0 \\ P(h_3|D) = .3, & P(\ominus|h_3) = 1, \quad P(\oplus|h_3) = 0 \end{array}$$

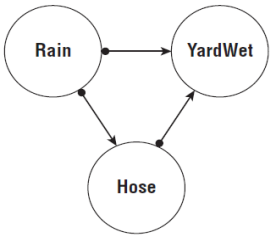
New instance V is classified as +ve and −ve.

therefore

$$\sum_{h_i \in H} P(\oplus|h_i) P(h_i|D) = .4 \qquad (1*.4 + 0*.3 + 0*.3 = 0.4)$$

$$\sum_{h_i \in H} P(\ominus|h_i) P(h_i|D) = .6 \qquad (0*.4+1*.3+1*.3 = 0.6)$$

7. Expert assigned some basic outcomes to the nodes given in the table below for Bayesian Networks shown in the following figure. What's the probability that it's raining when the yard is wet? (10 Marks)



YARD			
Hose	**Rain**	**True**	**False**
False	False	0.0	1.0
False	True	0.8	0.2
True	False	0.9	0.1
True	True	0.99	0.01

HOSE		
Value of Rain Node	**True**	**False**
False	0.4	0.6
True	0.01	0.99

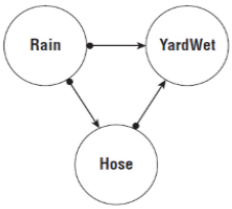
RAIN	
True	**False**
0.2	0.8

Assigning Probabilities

Probability values are between 0 and 1

Expert assign some basic outcomes to the nodes.

Node = Rain (It doesn't have any parent nodes.)

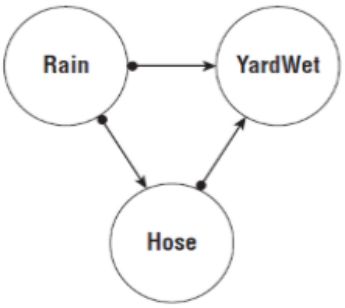


A basic Bayesian Network

RAIN	
True	**False**
0.2	0.8

Hose node: (uses the Rain node as a parent node)

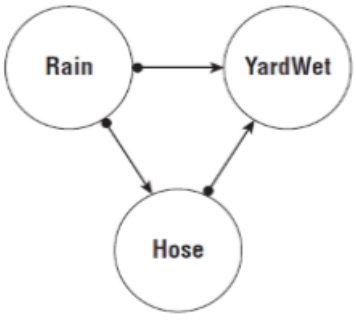
- Need to assign probabilities for each of the outcomes of the parent node.



A basic Bayesian Network

HOSE		
Value of Rain Node	**True**	**False**
False	0.4	0.6
True	0.01	0.99

Node: Yard, which has two parents: Rain and Hose. You need to ensure that all the outcomes are taken into account



A basic Bayesian Network

YARD			
Hose	**Rain**	**True**	**False**
False	False	0.0	1.0
False	True	0.8	0.2
True	False	0.9	0.1
True	True	0.99	0.01

“Given that the yard is wet, what’s the probability that it’s raining?”

Start with what you know.

- The yard is wet (True), and we want to know if it’s raining (True).
- The only variable you’re not certain about is the state of the hose; it could be true or false as it stands.

The joint probability function:

p(Y=True, H=True, R=True)

p(Y=True|H=True, R=True) × p(H=True|R=True) × p(R=True)

Multiply the values of the wet Yard probability, *where the values for Hose and Rain are also true*, *by the probability of Hose being true with the value of Rain being true*, *by the probability of Rain being true*.

The values you need are

0.99 (Y=T, H=T, R=T) × 0.01 (H=T, R=T) × 0.2 (R=T) = 0.00198

Next, work out the value of the probability if the hose was false.

This is the variable you don’t know, so it’s important to work out the probability for it.

p(Y=True|H=False, R=True) × p(H=False|R=True) × p(R=True)

0.8 × 0.99 × 0.2 = 0.1584

T,T,T = we know is 0.00198

T,F,T = we know is 0.1584

T,T,F = 0.9 × 0.4 × 0.8 = 0.288

T,F,F = 0.0 × 0.6 × 0.8 = 0.0

So the final equation looks like this:

$$\frac{0.00198 + 0.1584}{0.00198 + 0.288 + 0.1584 + 0.0} = 0.3577 \quad = 35.77\%$$

8. Write the algorithm and explain *Distance-Weighted Nearest Neighbor Learning* algorithms for learning the discrete-valued and real-valued target functions. (5+5 Marks)

Training algorithm:

- For each training example $\langle x, f(x) \rangle$, add the example to the list *training_examples*

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from *training_examples* that are nearest to x_q
  - Return

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

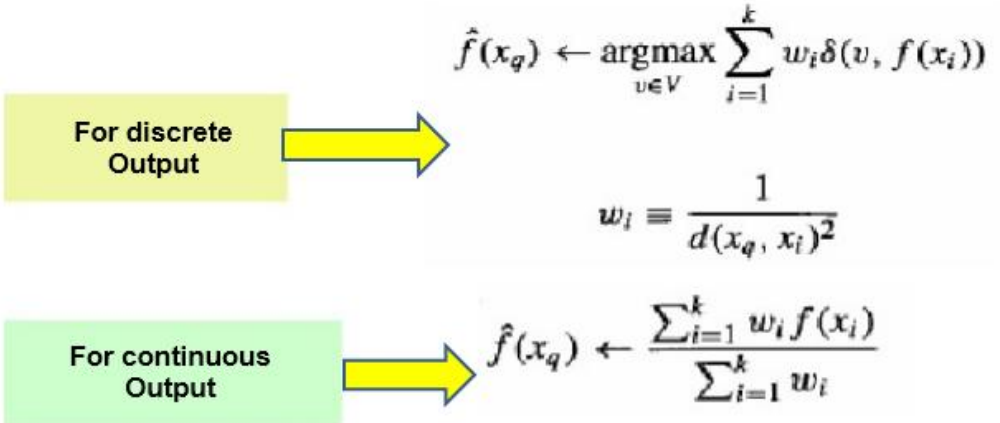
where $\delta(a, b) = 1$ if $a = b$ and where $\delta(a, b) = 0$ otherwise.

Distance-Weighted Nearest Neighbor Learning

One refinement to the k-NN algorithm is to weight the contribution of each of the k neighbors according to their distance to the query point x_q, giving greater weight to closer neighbors.

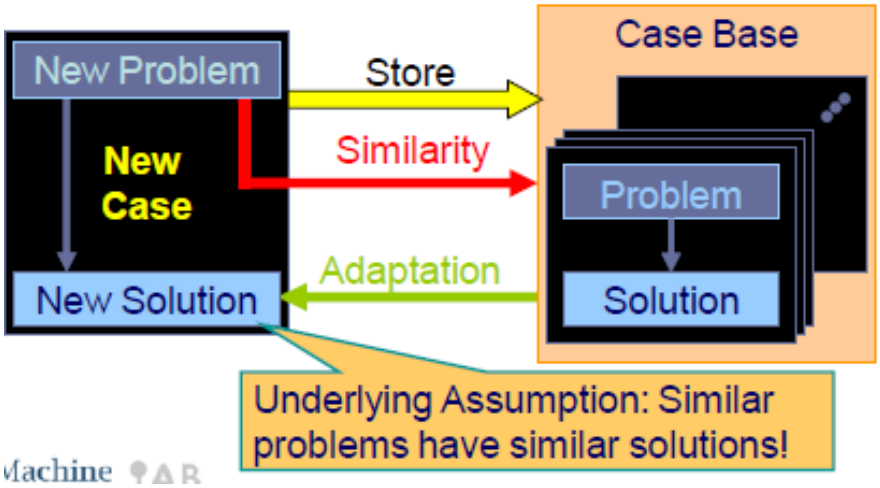
$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$



9. Explain the learning concept of *Case-Based Reasoning* with suitable example. (10 Marks)

- It is an approach
  - To model the way human think
  - To build intelligent systems
- Basic Idea:
  - store experiences made → as cases
  - solving a new problem do the following
    - **recall** similar experiences (made in the past) from memory
    - **reuse** that experience in the context of the new situation(reuse it partially, completely or modified)
    - new experience obtained this way is **stored** to memory again
- Solve new problems by selecting cases used for similar problems and by (eventually) adapting the belonging solution.



Contents of a Case

Mandatory	Optionally
problem part	context
solution part	pointer to other relevant cases
	solution quality assessment
	steps of the solution

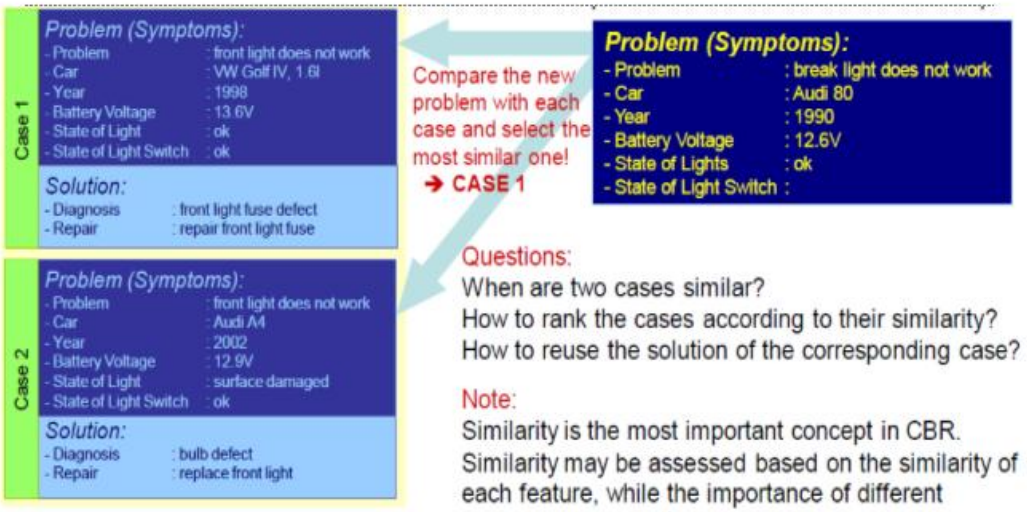
New Problem (Query) has To Be Solved

Problem (Symptoms):

- Problem	: break light does not work
- Car	: Audi 80
- Year	: 1990
- Battery Voltage	: 12.6V
- State of Lights	: ok
- State of Light Switch	:

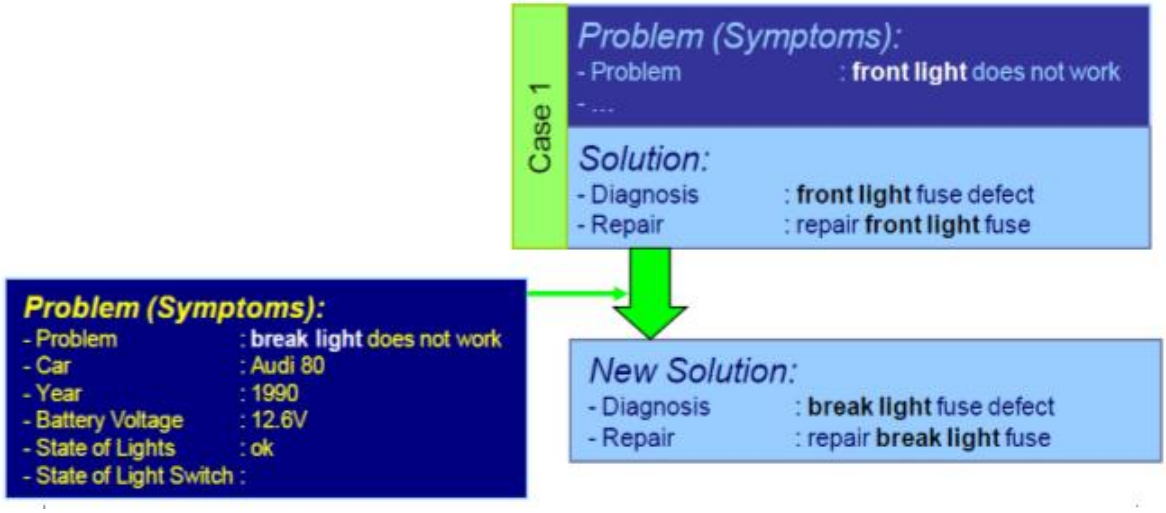
- We make several observations in the current situation
- observations define a new problem
- not all attribute values have to be known
- Note: The new problem is a ``case'' without solution part

Solving a New Diagnostic Problem

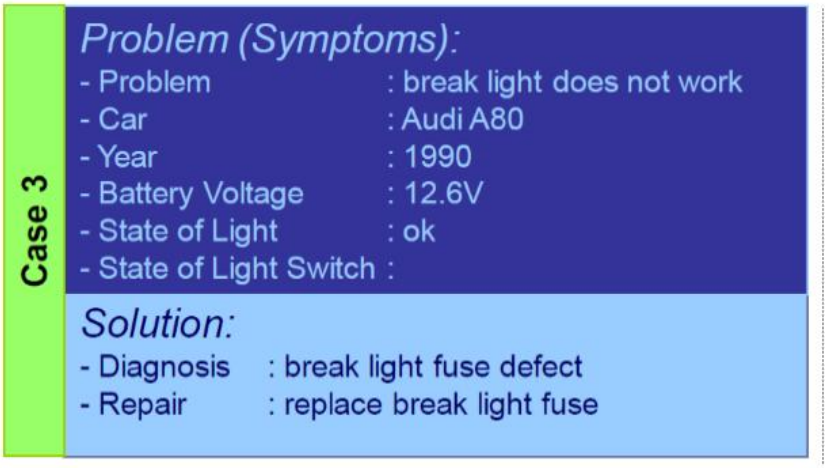


Questions:
When are two cases similar?
How to rank the cases according to their similarity?
How to reuse the solution of the corresponding case?

Note:
Similarity is the most important concept in CBR. Similarity may be assessed based on the similarity of each feature, while the importance of different features may vary (feature weighting).

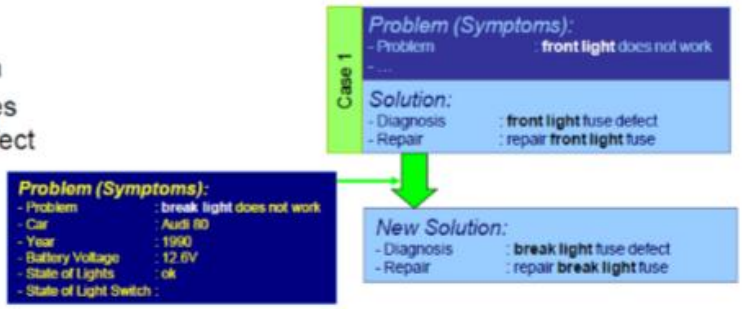


Solution

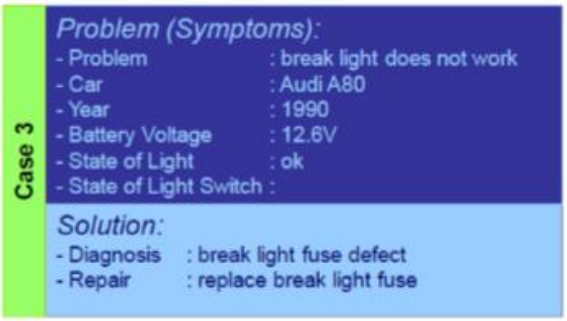


Reuse and Retain

- Reuse
  - adapt the solution
  - how do differences in the problem affect the solution



- Retain
  - if diagnosis is correct: store new case
  - add case to case base



10. Find the covariance matrix and principal components (PCs) for the data showing relationship between numbers of hours studied against the mark received. (10 Marks)

	Hours(H)	*Mark(M)*
Data	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80

- The covariance matrix has 3 rows and 3 columns, and the values are this:

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

H	M	(Hi - H')	(Mi-M')	(Hi-H')2	(Mi-M')	(Hi-H')*(Mi-M')
9	39	-4.91667	-23.42	24.17361	548.4964	115.1483333
15	56	1.083333	-6.42	1.173611	41.2164	-6.955
25	93	11.08333	30.58	122.8403	935.1364	338.9283333
14	61	0.08	-1.42	0.0064	2.0164	-0.1136
10	50	-3.92	-12.42	15.3664	154.2564	48.6864
18	75	4.08	12.58	16.6464	158.2564	51.3264
0	32	-13.92	-30.42	193.7664	925.3764	423.4464
16	85	2.08	22.58	4.3264	509.8564	46.9664
5	42	-8.92	-20.42	79.5664	416.9764	182.1464
19	70	5.08	7.58	25.8064	57.4564	38.5064
16	66	2.08	3.58	4.3264	12.8164	7.4464
20	80	6.08	17.58	36.9664	309.0564	106.8864
167	749	153.08	686.58	524.9651	4070.917	1352.419267
13.91667	**62.41667**			**43.74709**	**339.2431**	**112.7016056**

cov (x, y) = $\begin{bmatrix} 43.75 & 112.7 \\ 112.7 & 339.24 \end{bmatrix}$

Eigen values $\lambda = \begin{pmatrix} 377.31 \\ 5.6728 \end{pmatrix}$

Eigenvector with the highest eigenvalue is the principle component of the data set.

Solving for Eigen vectors corresponding to λ1 = $\begin{pmatrix} 1 \\ 2.9 \end{pmatrix}$

Solving for Eigen vectors corresponding to λ2 = $\begin{pmatrix} 1 \\ 0.34 \end{pmatrix}$