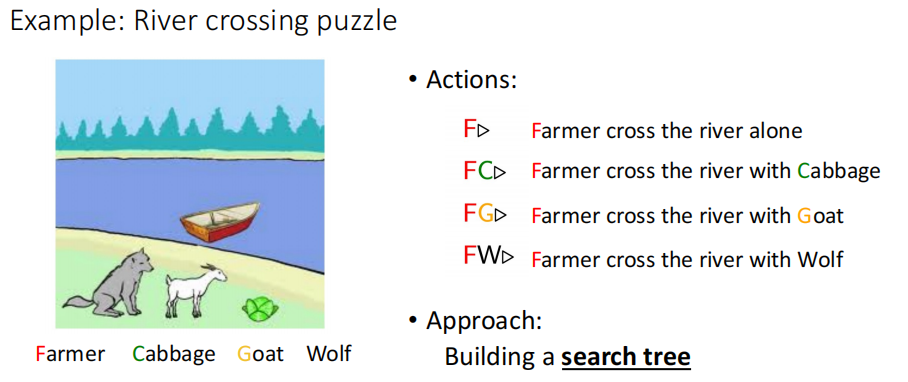
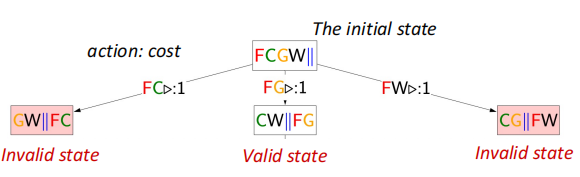
**Lecture1**

Example: River crossing puzzle

Invalid State：FCGW|| , FG||WC, G || FCW and W|| FCG

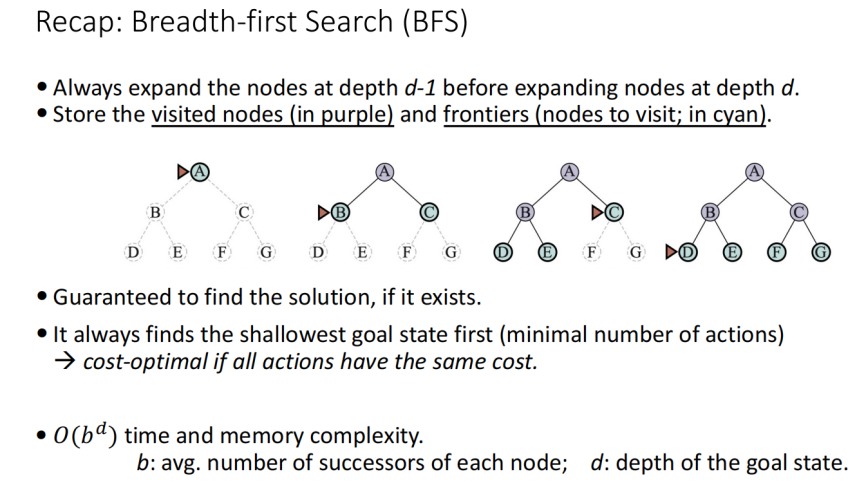




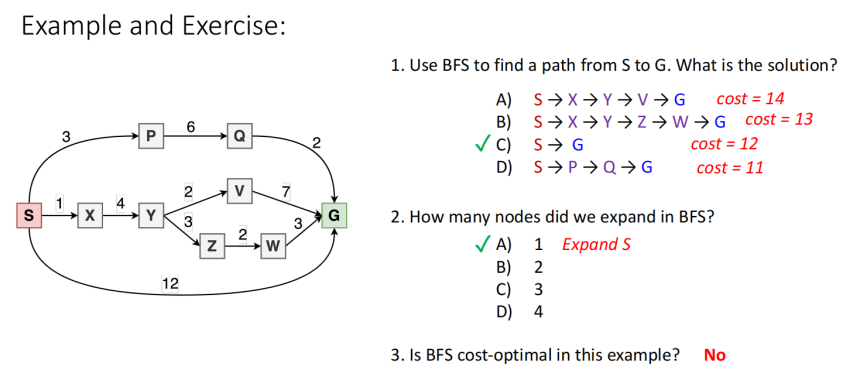


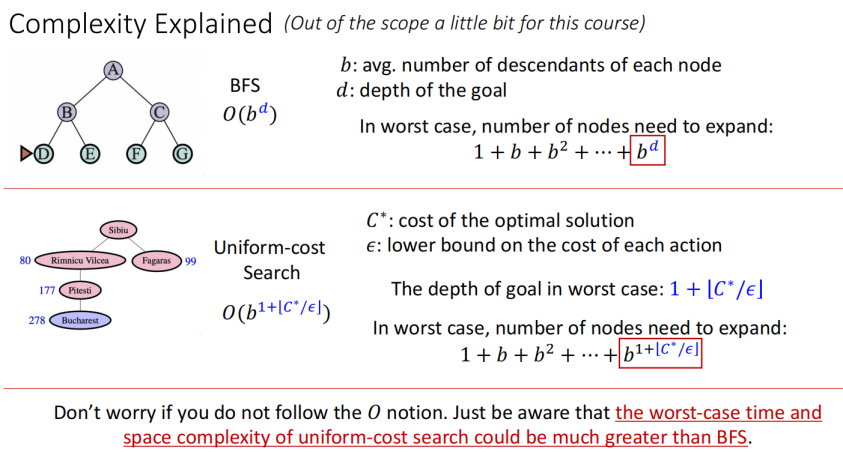
**Lecture2**

Search

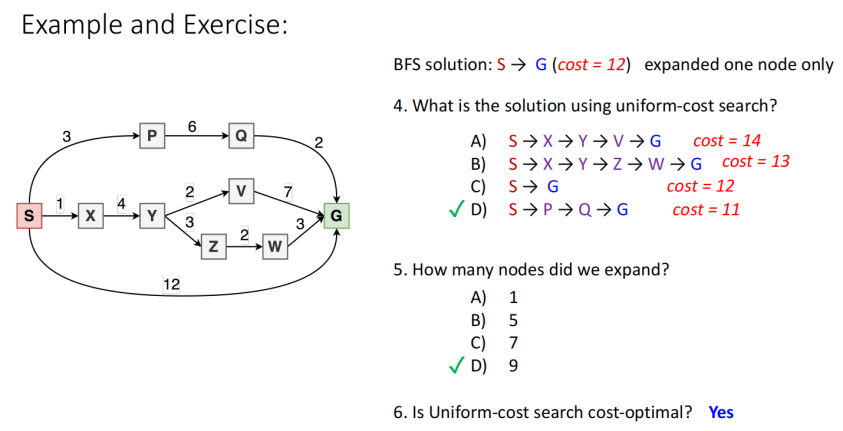


BFS会遍历所有的节点





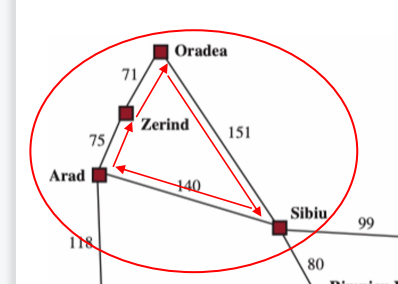
最坏的情况，uniform-cost search是要比BFS的内存和空间复杂度要大得多。





All visited nodes are stored in memory Can we save some memory? Discard visited nodes? a tree-like search (actually more commonly used for DFS)

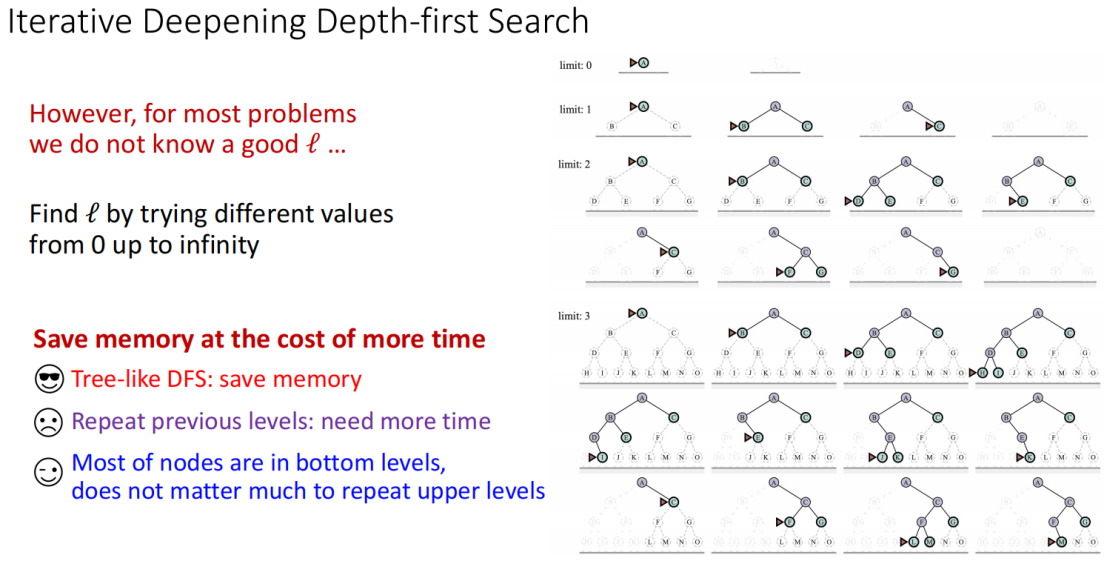
If implemented in tree-like search, discard visited nodes that have no descendants. memory-efficient, but could visit the same node multiple times, we will get stuck in infinite loops

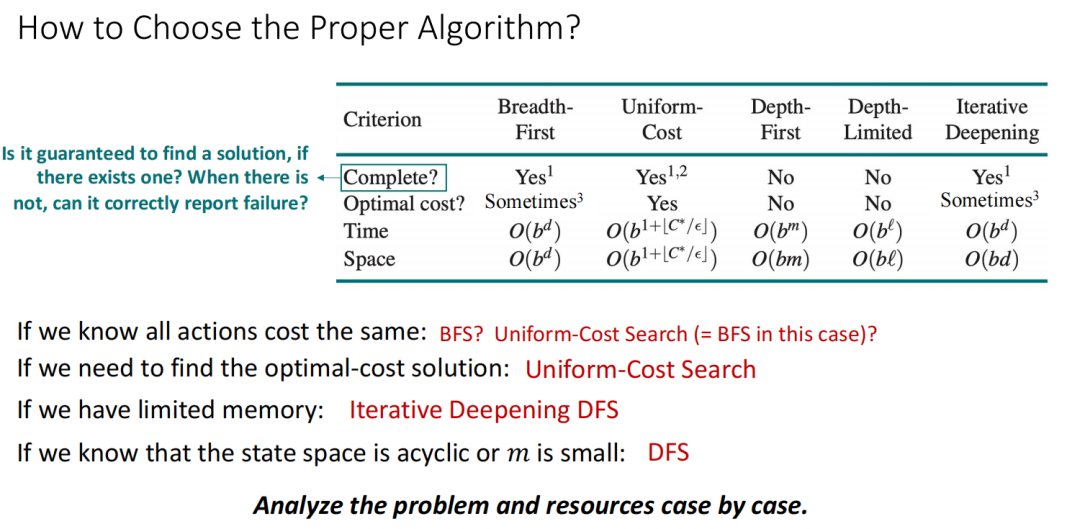


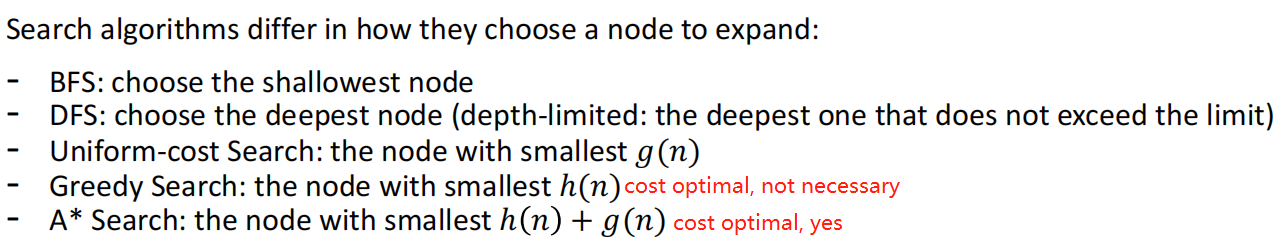
Pros and Cons of Tree-like DFS Pros: Save memory

Cons: visit a node multiple times, and get stuck in wrong path or infinite loops

Depth-limited Search set a depth limit ℓand ignore all nodes exceeding this depth



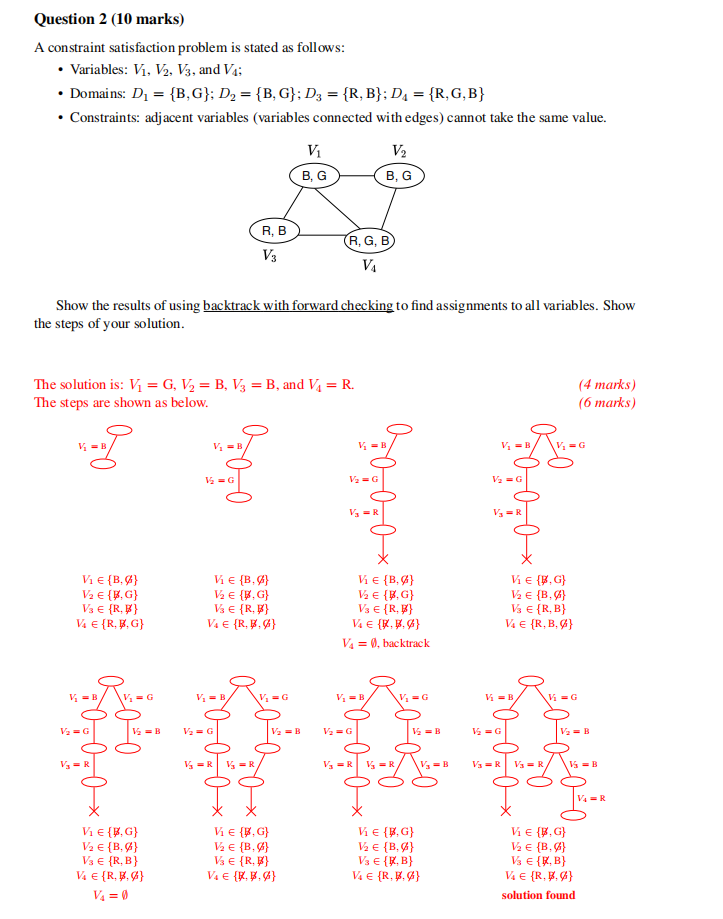




**Lecture3**

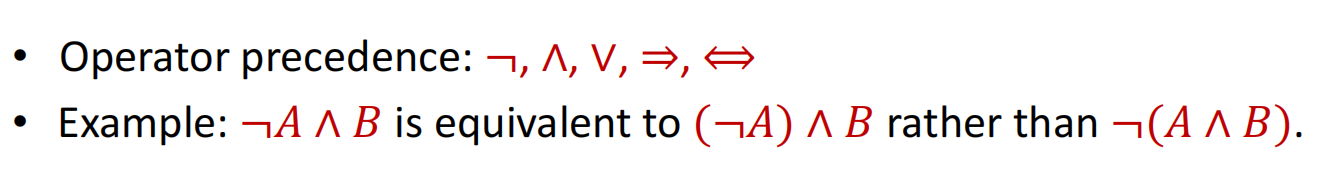
Search

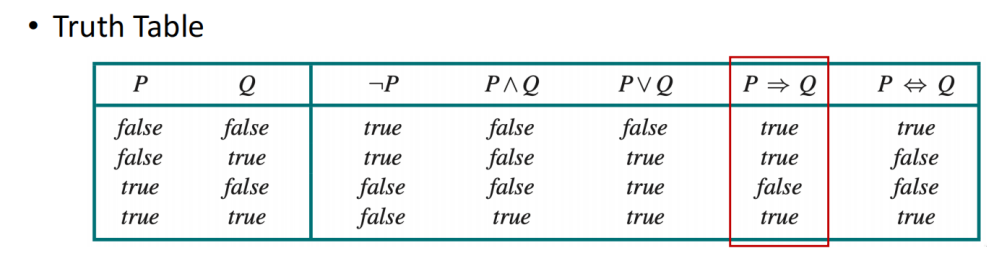
Constraint Satisfaction Problems (CSPs)

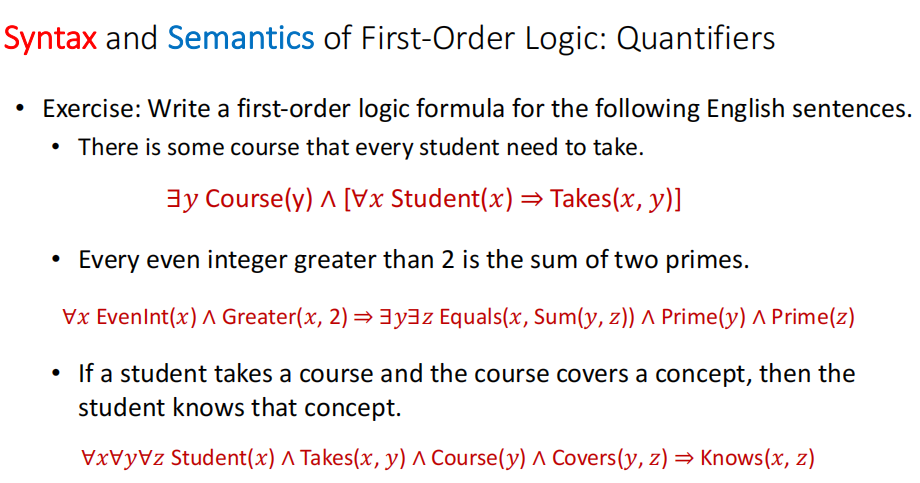


**Lecture5**

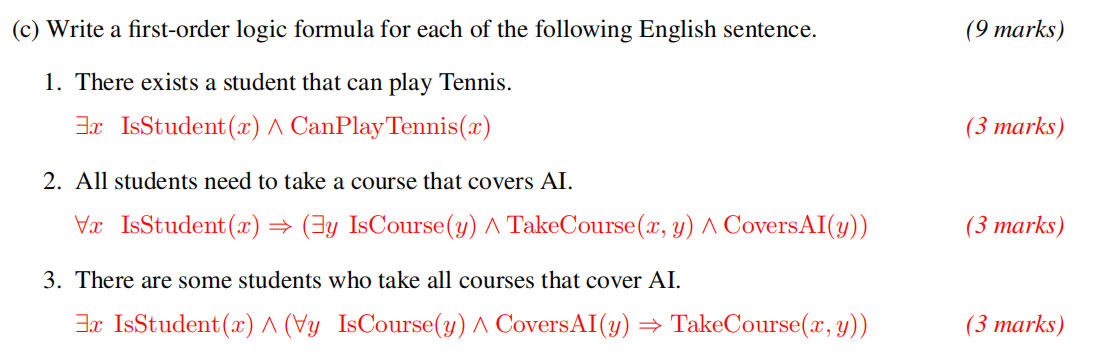
Logic, 非 not，并and，或 or，非P或Q，全等于

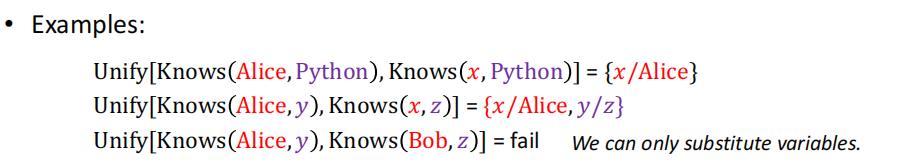


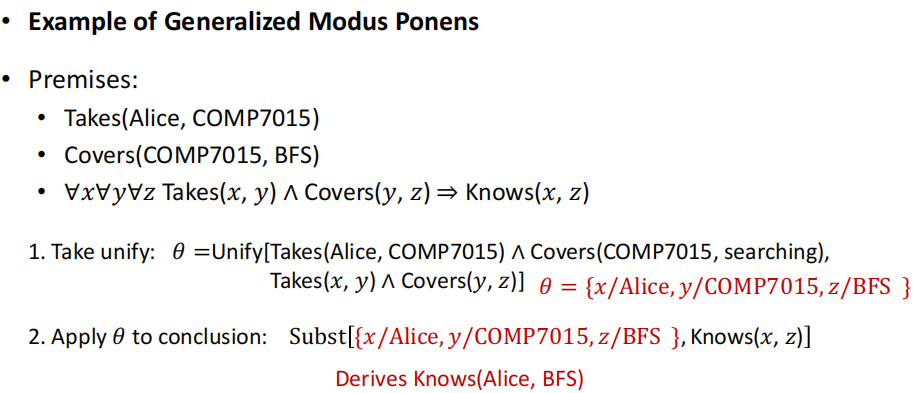




∃存在，∀所有

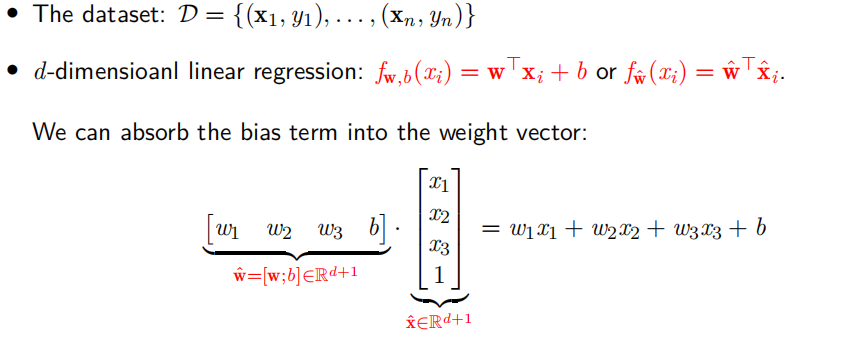


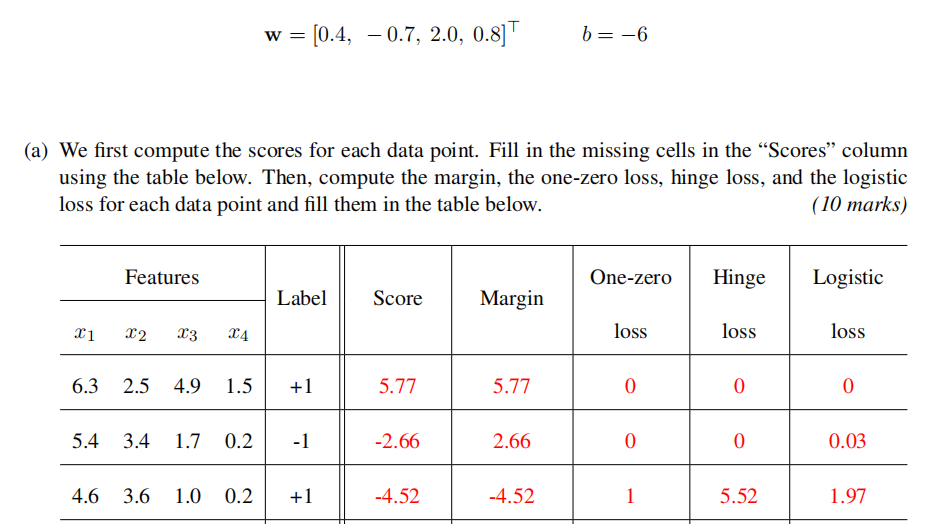




**Lecture5**

ML

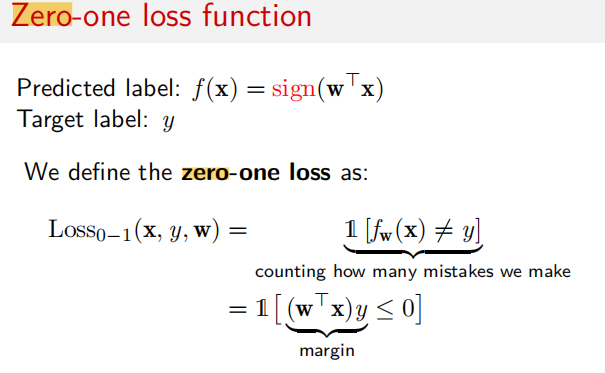




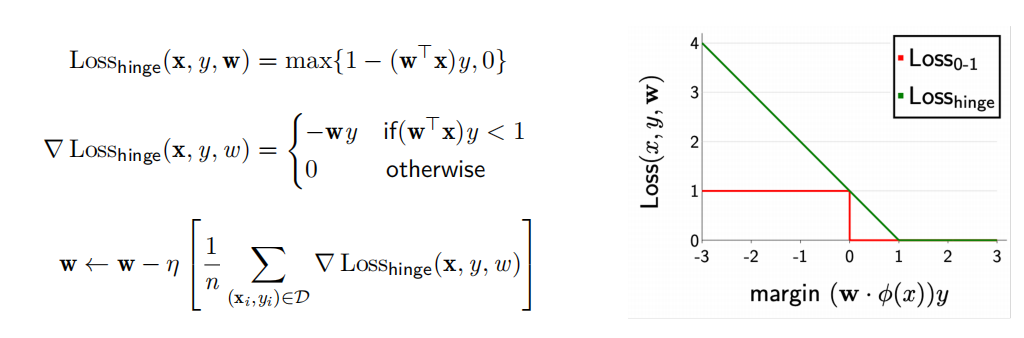
6.3\*0.4 - 2.5\*0.7 + 4.9\*2+1.5\*0.8 - 6 = 5.77 score

5.4\*0.4-3.4\*0.7+1.7\*2+0.2\*0.8 - 6 = -2.66 Score = -2.66, margin = score \* label = -2.66\*-1 = 2.66

5.4\*0.4-3.4\*0.7+1.7\*2+0.2\*0.8 = 3.34 > 0 => 1 \* -1 = -1 => 0

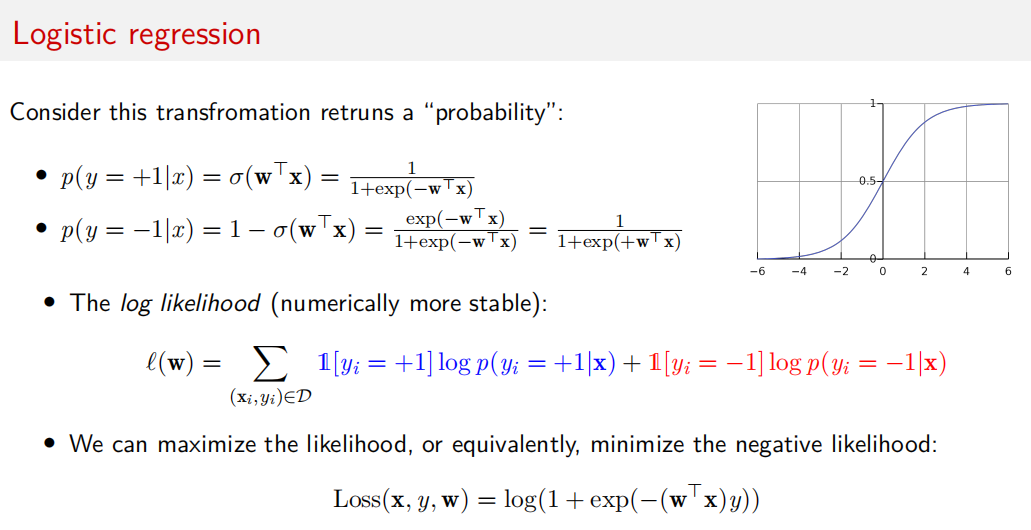


(4.6\*0.4 - 3.6\*0.7 + 1\*2+0.2\*0.8) = 1.48 1.48>0 = 1 \* 1(label) = 1

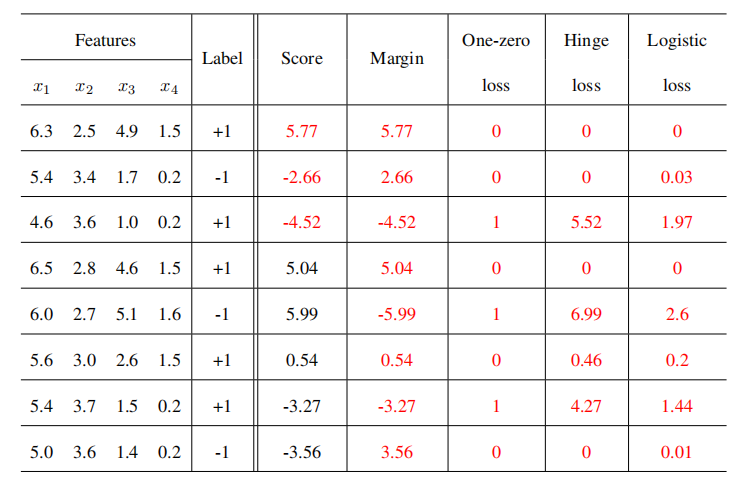


hingLoss = 1 - (-4.52) margin = max 5.52 0 => 5.52,只有在zero-loss=1时出现

1 - 2.66 = max (-1.66 0) => 0



- 算出margin后，log(1 + )，margin前面有一个负号



对比Score 与 Label的正负号

TP就是预测和实际都是正的，5.77 +1, 5.04 +1, 0.54 +1, TP=3，positive true，正-对了

FN就是预测是负的，与实际比是错误的 -4.52 +1, -3.27 -1 FN = 2, negative false, 负-反了

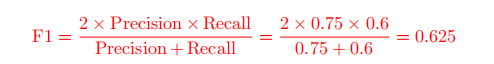
TN 就是预测是负的，实际是对的，-3.56 -1, -2.66 -1 TN=2，negative true，反-对了

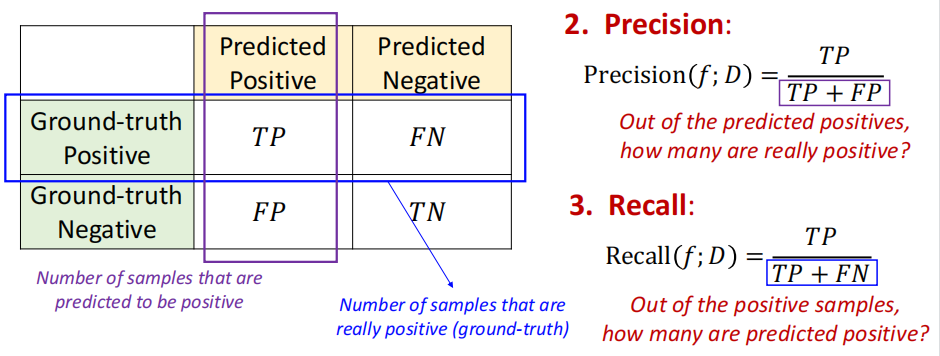
FP 就是预测是正的，与实际是错误的, 5.99 -1 , FP = 1，positive false， 正-反了

Accuracy 都是对的情况 = TP+FN / TP+FN+TN+FP = 5 / 8

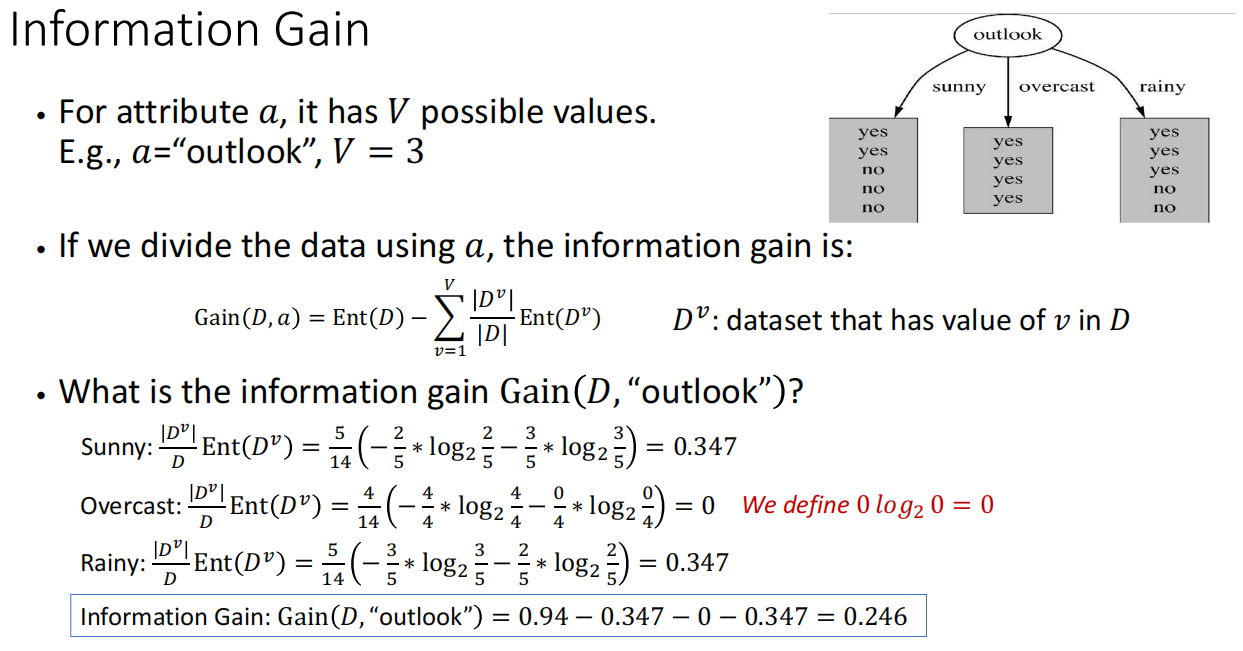
Precision 都是正的情况为N = TP / TP + FP = 3 / 3+1 = 3/4

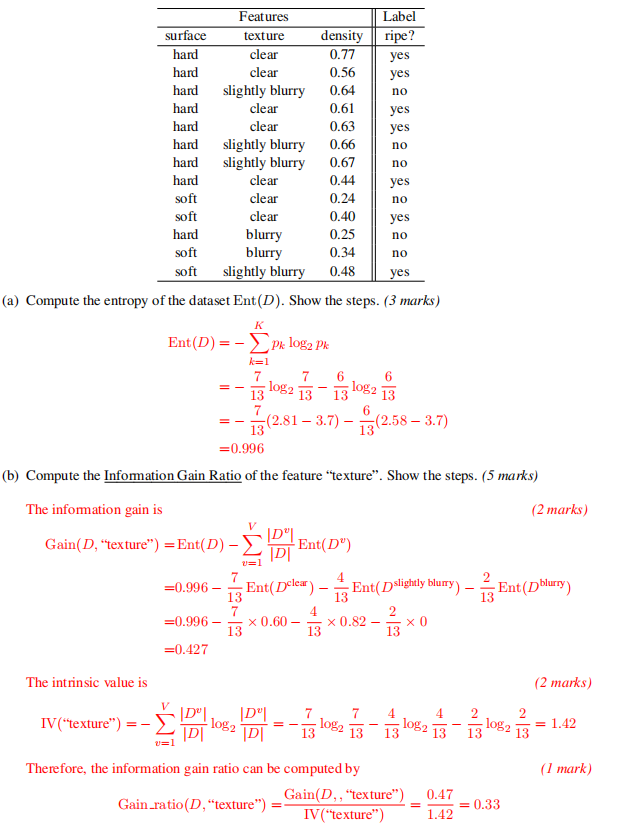
Recall TP / TP + FN = 3 / 3 + 2 = 3/5

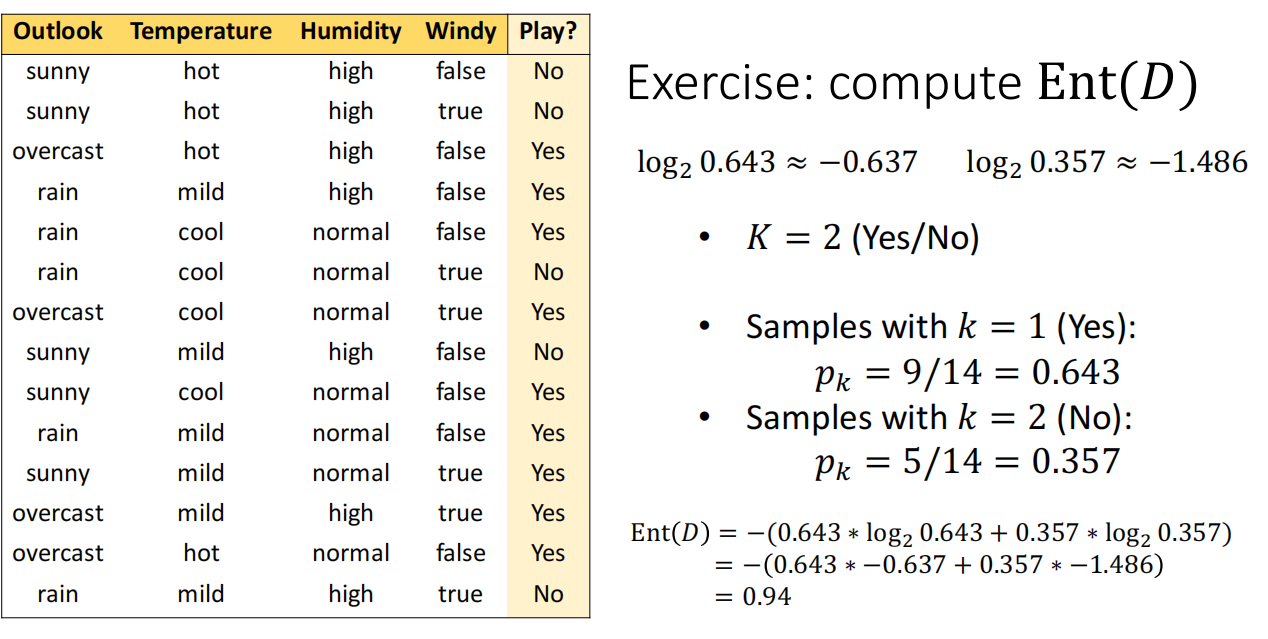




Entropy





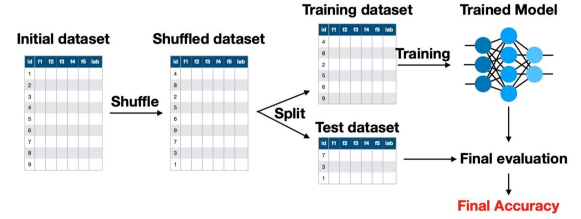


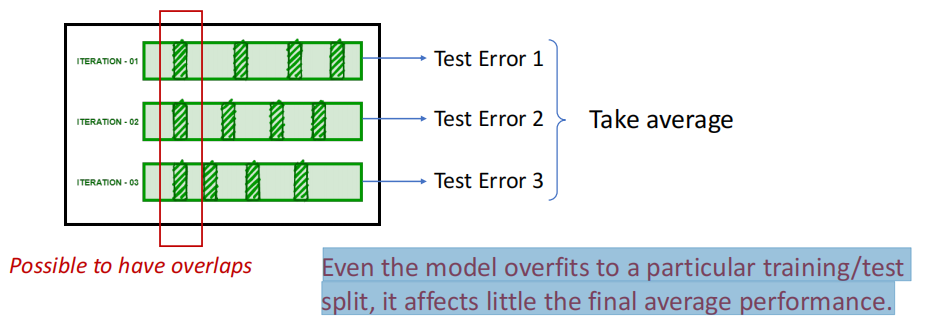
如果是数值column，先求数值的两两平均值。排序后。

第一个值的: Ent(D) - D+number / Number \* End(D+) - D-number / Number \* End(D-)

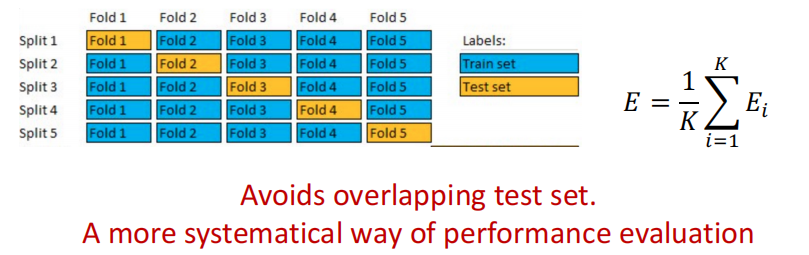
End(D+) = -log(2,yes-number/D+number) - log(2,no-number/D+number)

Method for Performance Evaluation

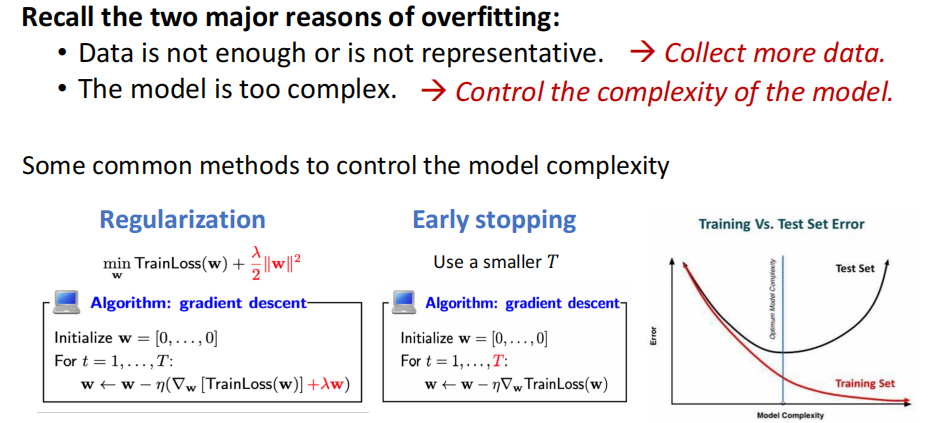


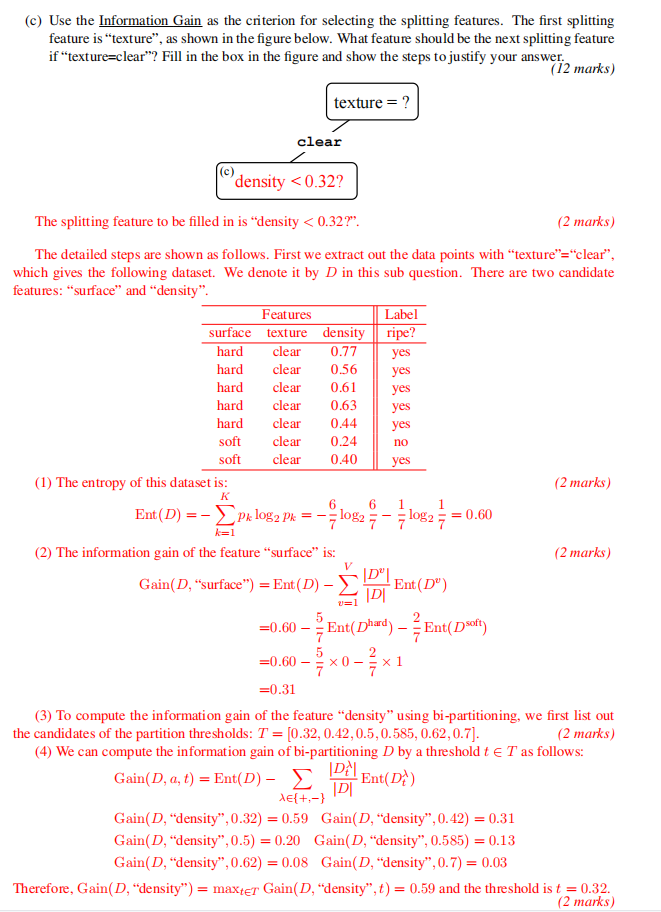


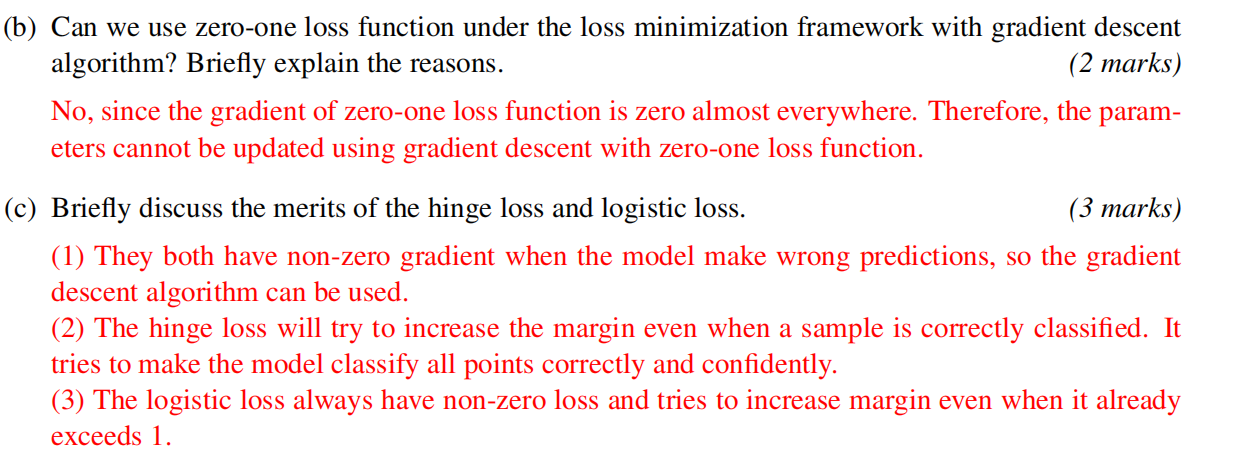
Idea: Split the data into k subsets of equal size, use one subset for testing and others for training in each iteration



How to Prevent Overfitting



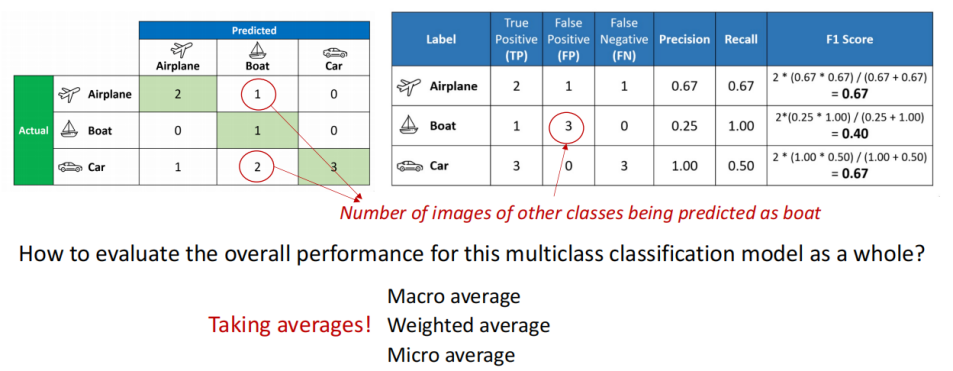


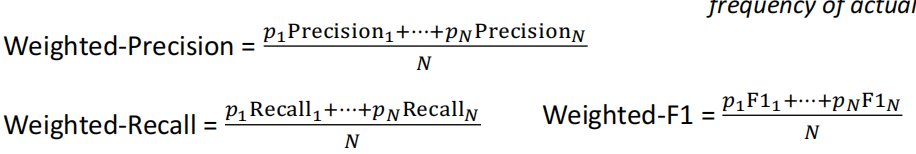


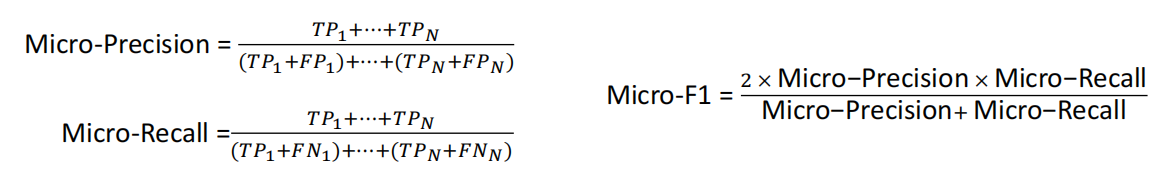
**Lecture7**

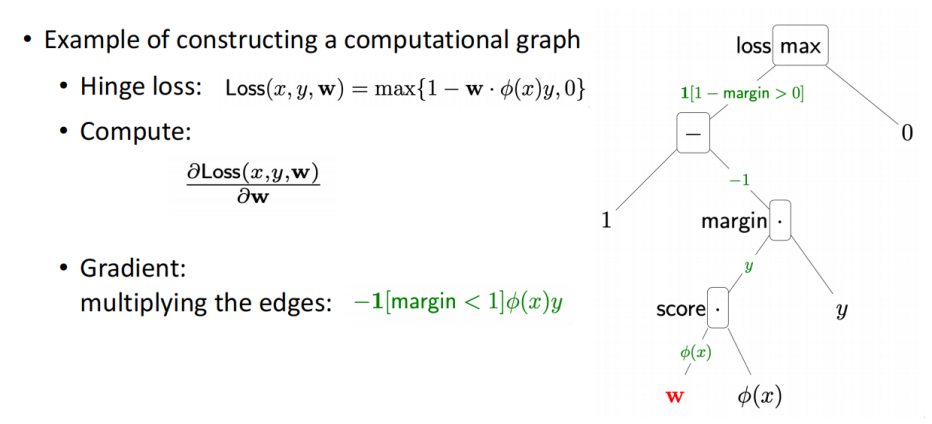
Gradient Decent & Computational graph

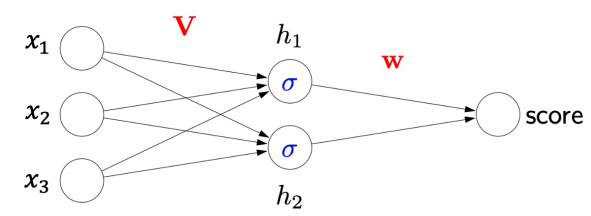
Multiple class confusion matrix,算 accuracy, recall and precision的平均

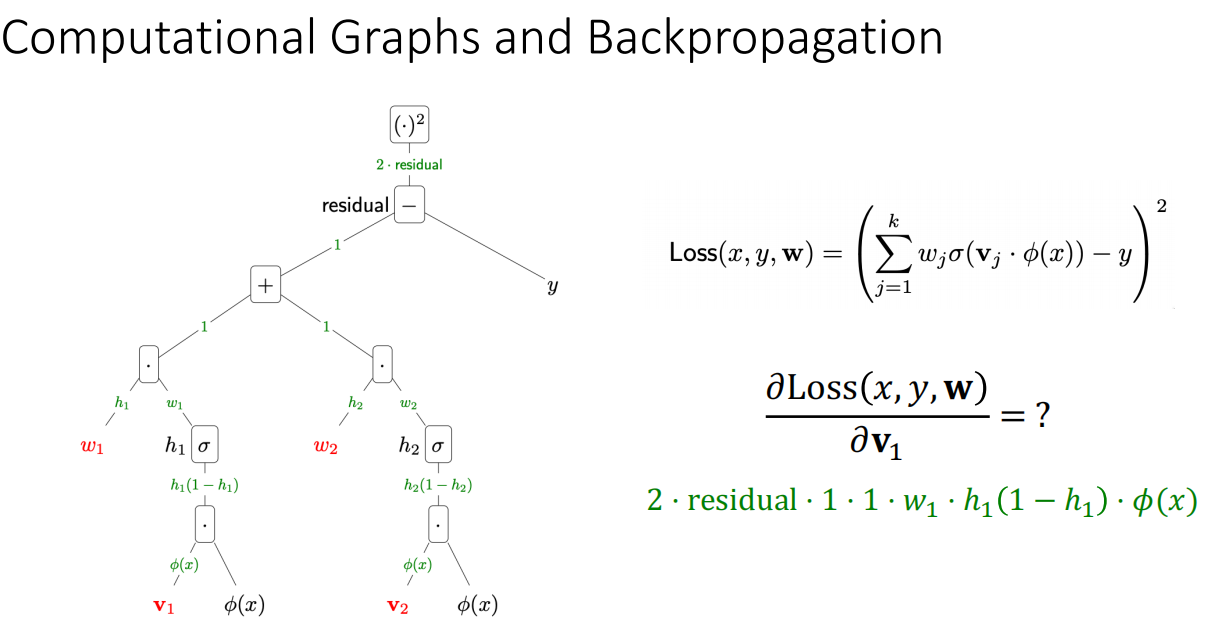








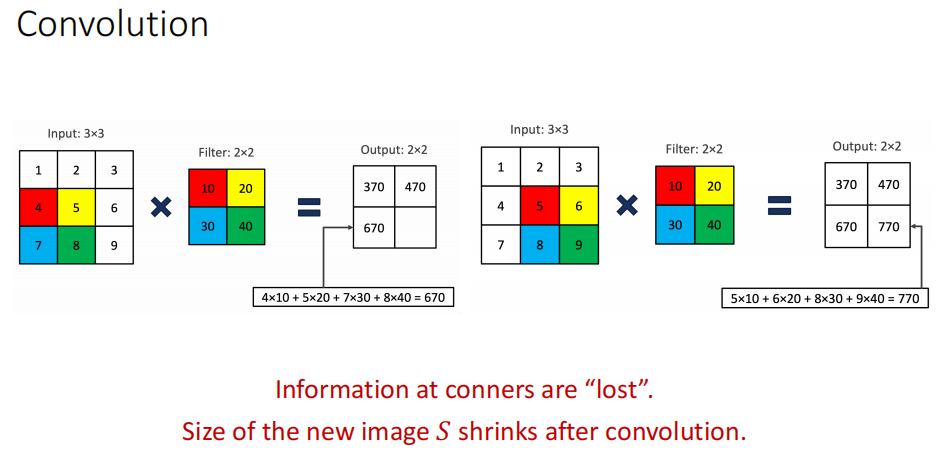




**Lecture 8 & 9**

CNN 图像处理 & RNN 时序数据处理，语言文字nlp，以及股票

Information at conners are lost. Size of the new image shrinks after convolution.



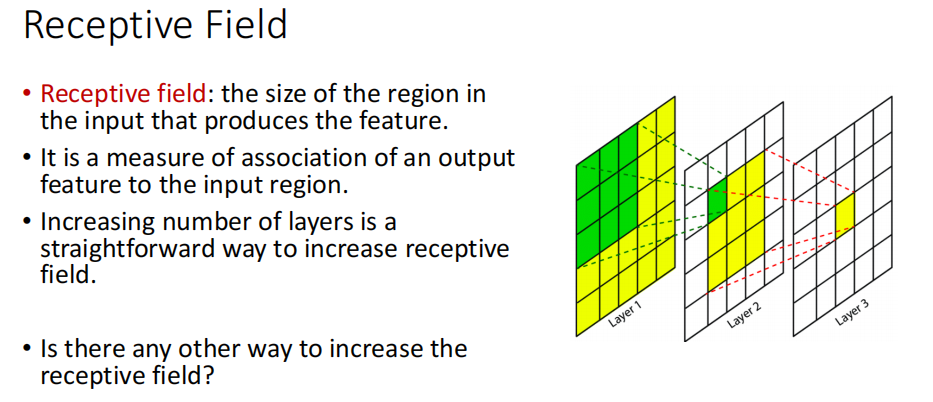
• Padding keeps the dimension of input and output matrix the same



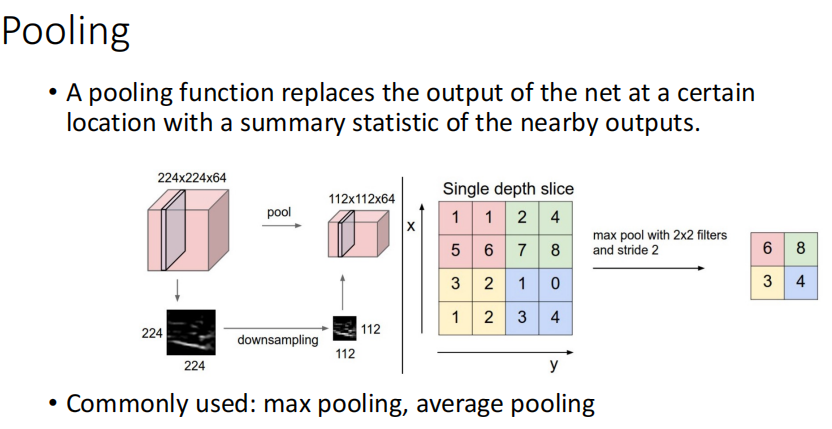
Padding: to avoid loss of information and the output has the same size as input.

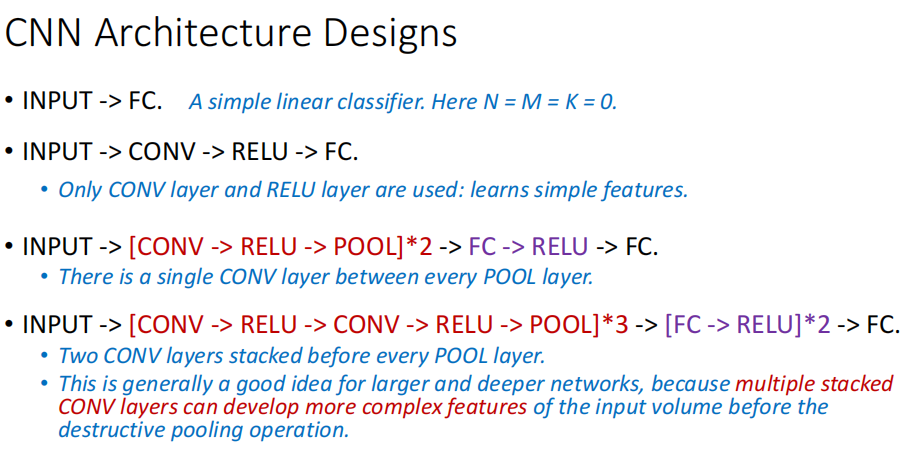
Stride: to compress information and the output has smaller size than input

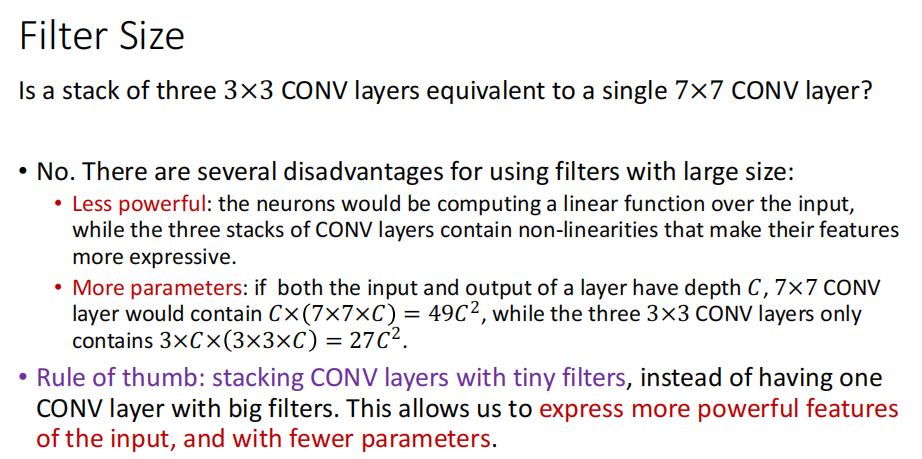
More stride, more receptive Field.Increasing number of layers is a way to increase receptive field

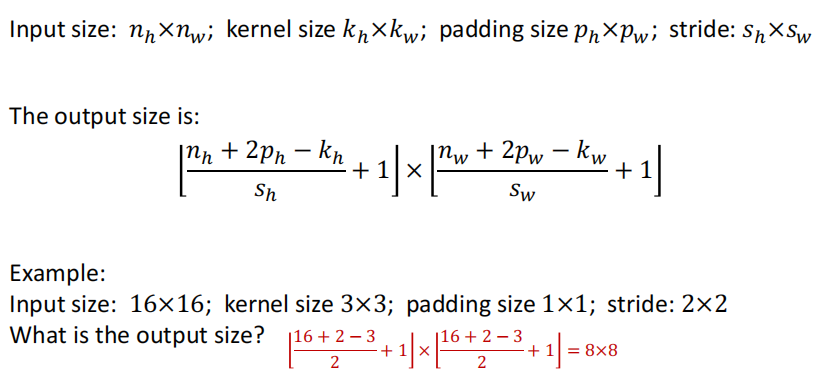


MaxPooling and average pooling

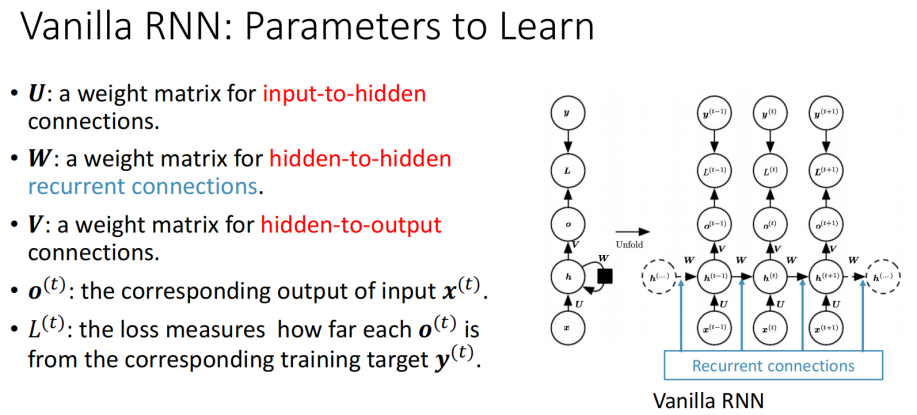


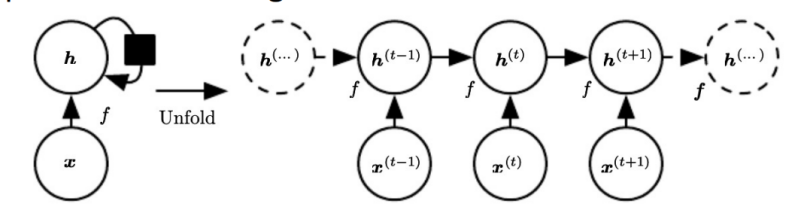






Vanilla RNN: Backpropagation Through Time

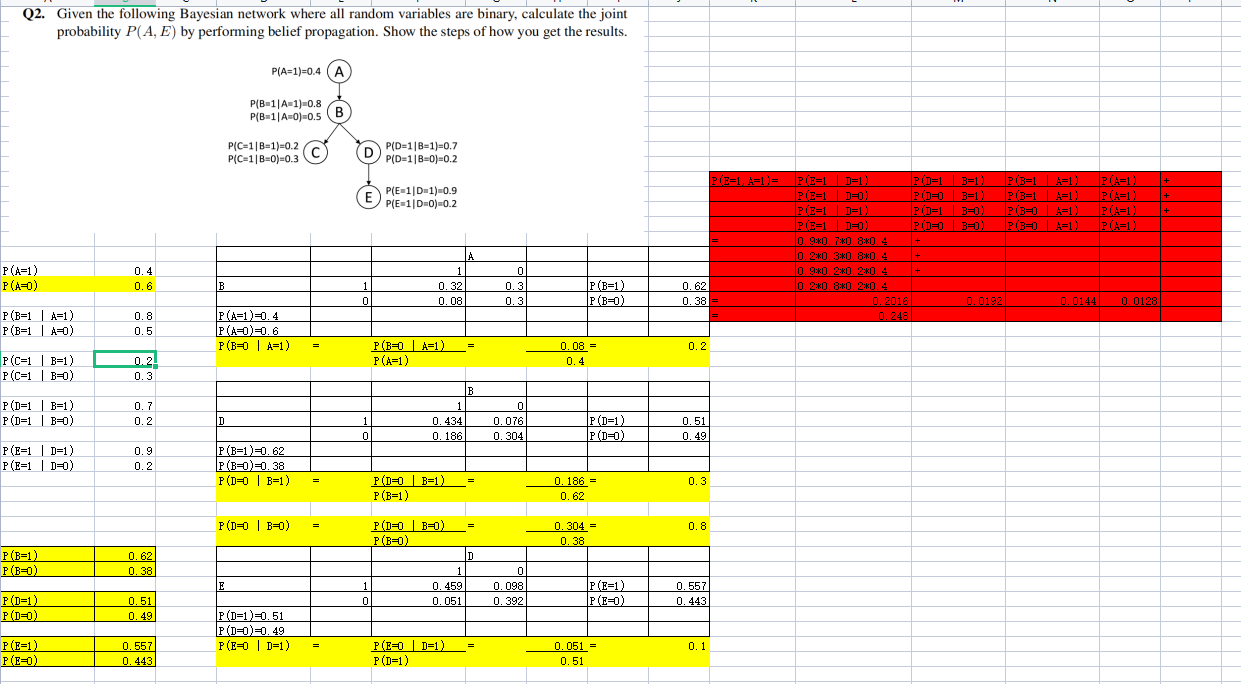


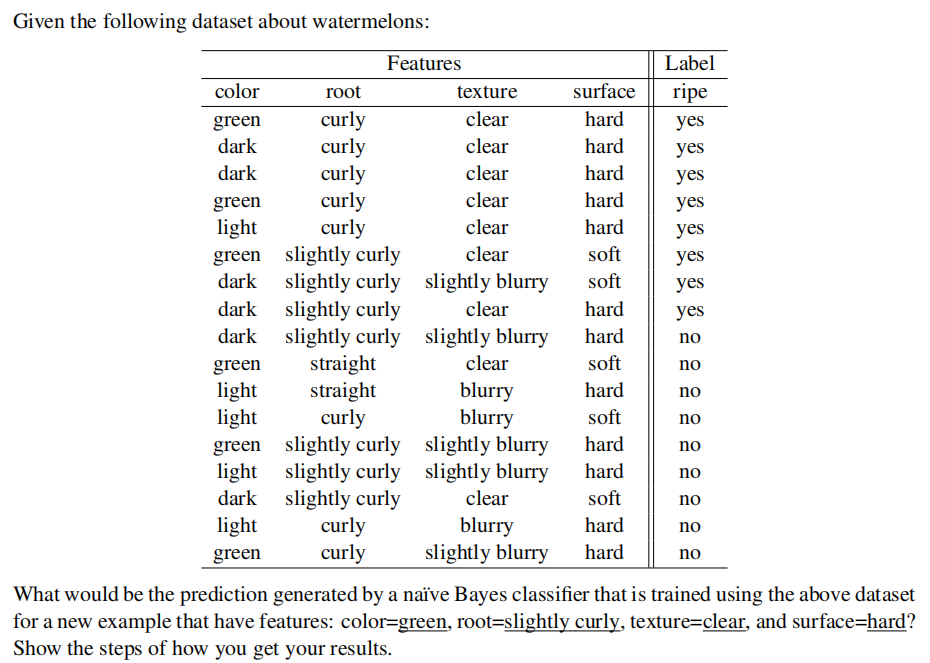


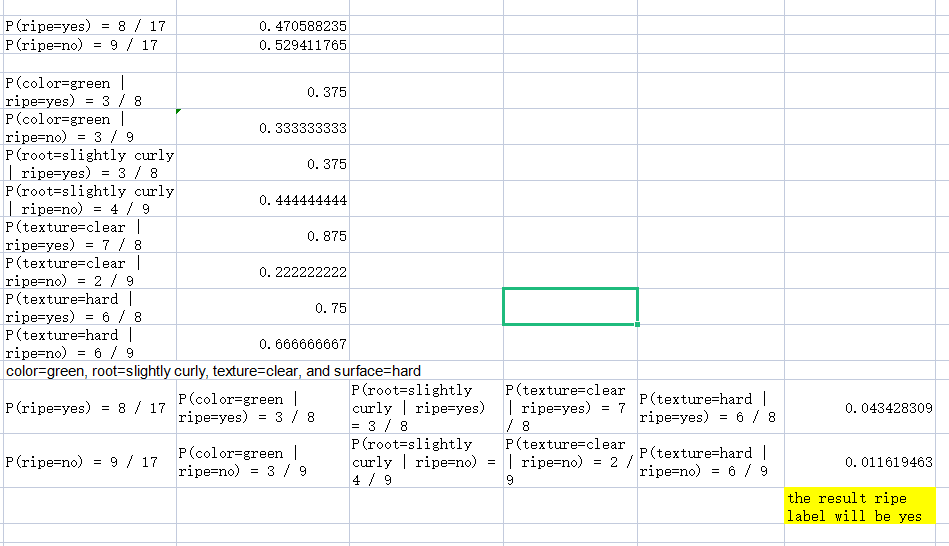
Long Short Term Memory networks (LSTMs) is a special kind of RNN, explicitly designed for learning long-term dependencies

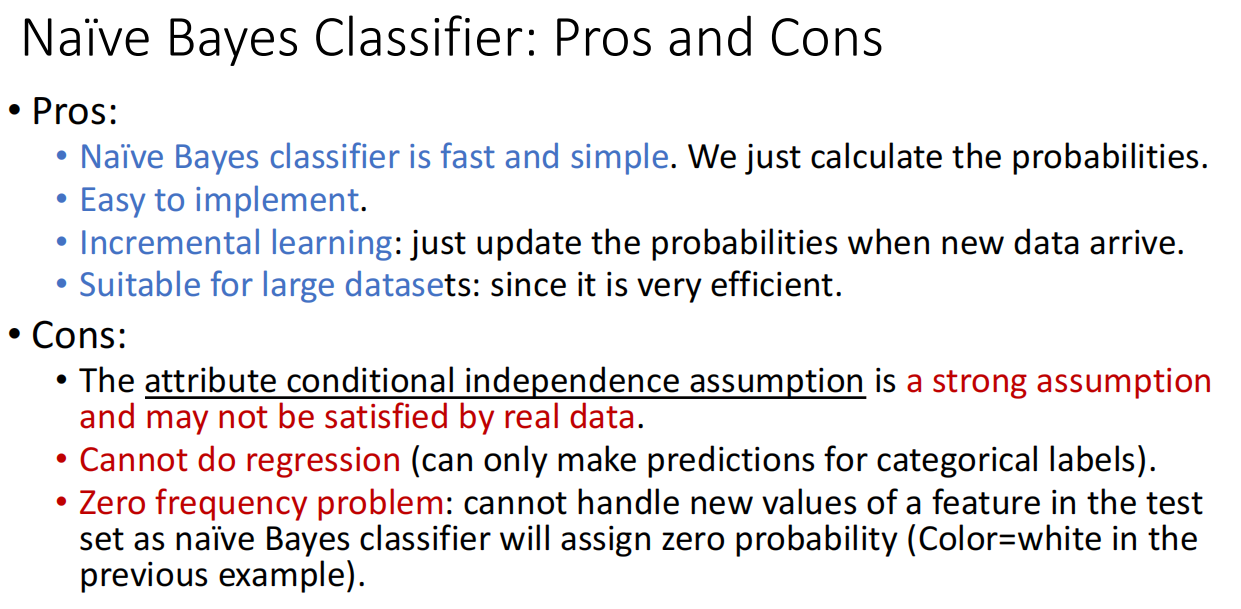
**Lecture 10**

Bayesian Decision









**Lecture 11**

Enforcement learning

