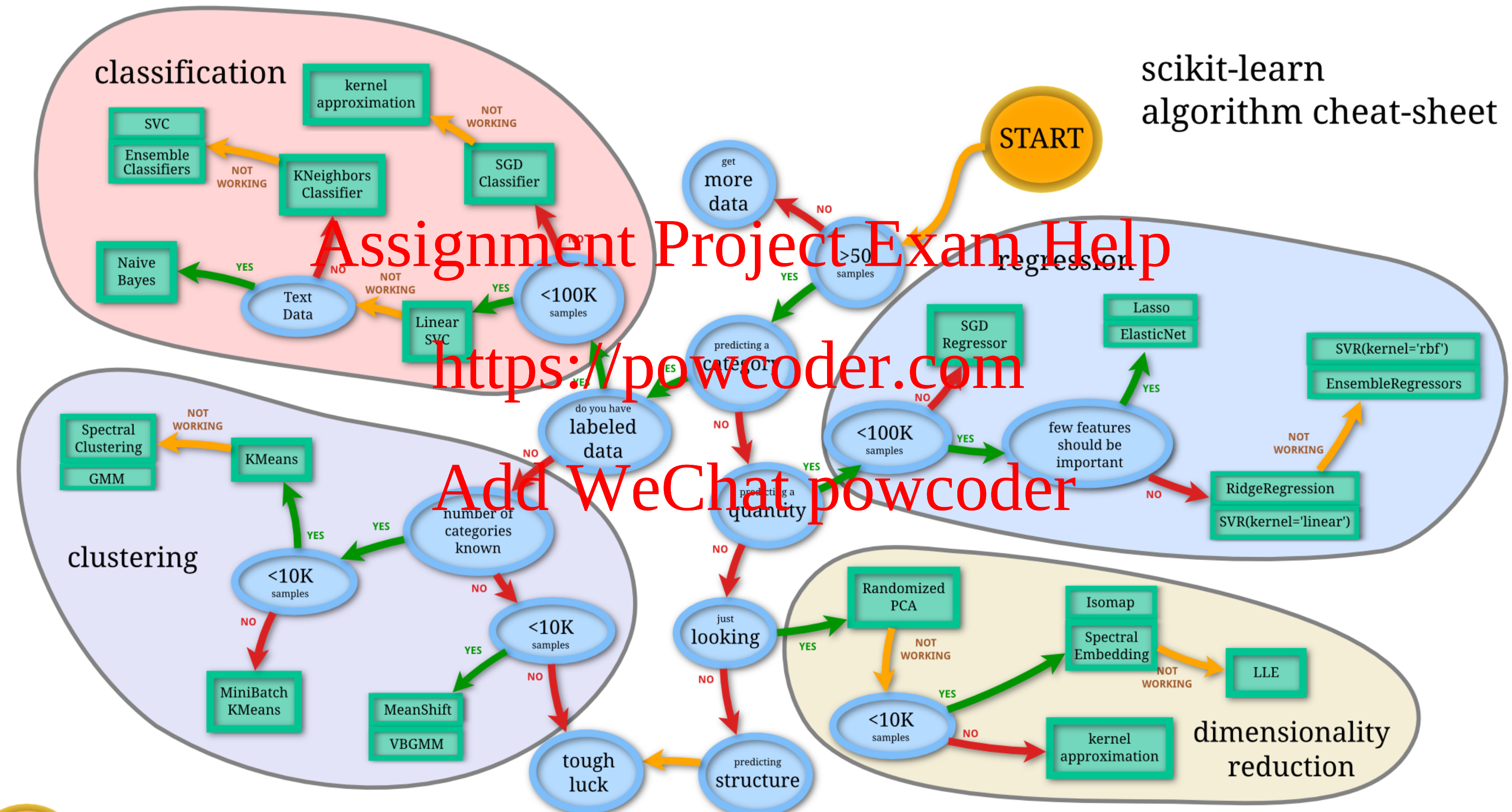
COMP9318: Data Warehousing and Data Mining

— L7: Classification and Prediction —

- Problem definition and preliminaries

ML Map



scikit-learn
algorithm cheat-sheet
START
classification
regression
clustering
dimensionality reduction
kernel approximation
SVC
Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
<100K samples
get more data
>50 samples
predicting a category
do you have labeled data
number of categories known
<10K samples
predicting a quantity
just looking
predicting structure
tough luck
Spectral Clustering
GMM
KMeans
MiniBatch KMeans
MeanShift
VBGMM
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
few features should be important
RidgeRegression
SVR(kernel='linear')
Randomized PCA
Isomap
Spectral Embedding
LLE
kernel approximation
YES
NO
NOT WORKING

Back

Classification vs. Prediction

- Classification:
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction (aka. Regression):
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical Applications
  - credit approval
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis

Classification and Regression

- Given a new **object** o, map it to a **feature vector** $\mathbf{x} = (x_1, x_2, \ldots, x_d)^\top$

- Predict the output (**class label**) $y \in \mathcal{Y}$

  - Binary classification: $\mathcal{Y} = \{-1, +1\}$ Sometimes, $\{0, 1\}$

  - Multi-class classification: $\mathcal{Y} = \{1, 2, \ldots, C\}$

- Learn a classification **function**: $f(\mathbf{x}) : \mathbb{R}^d \mapsto \mathcal{Y}$

- Regression: $f(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}$

Examples of Classification Problem

- Text categorization:

Doc: Months of campaigning and weeks of round-the-clock efforts in Iowa all came down to a final push Sunday, …

→ **Topic**: Politics / Sport

- Input object: a document = a sequence of words

- Input features $\mathbf{x}$ = word frequencies
  - freq(democrats)=2, freq(basketball)= 0, …
  - $\mathbf{x}$ = [1, 2, 0, …]^T
- Class label: y
  - 'Politics': $y = +1$
  - 'Sport': $y = -1$

Examples of Classification Problem

- Image Classification:



Which images are birds, which are not?

- Input object: an image = a matrix of RGB values

- Input features $\mathbf{x}$ = Color histogram

  - pixel_count(red) = 1004, pixel_count(blue) = 23000
  - $\mathbf{x}$ = [1004, 23000, …]^T
- Class label y
  - 'bird image': $y = +1$
  - 'non-bird image': $y = -1$

How to find f()?

1. Input:
   - In supervised learning, we are given a set of training examples:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$$

   - Identical independent distribution (i.i.d) assumption
     - A critical assumption for machine learning theory
     - e.g.,

$$\log P(\mathbf{x} \mid \theta) = \sum_i \log P(x_i \mid \theta)$$

How to find f()?

2. Representation of f()
   - Typically only consider a particular function family F
     - consider **parameterized functions** $f(\mathbf{x}; \theta) \in F$

Examples:

- F: linear functions (for regression) ➔ $f = \mathbf{w}^T\mathbf{x}$
- F: linear functions (for classification) ➔ $f = \sigma(\mathbf{w}^T\mathbf{x})$
- What about more general function families?

How to find f()?

Empirical vs. Structured Risk Minimization (ERM vs. SRM)

3. Criterion for the best f()
   - Non-Bayesian approaches:
     - ERM — Loss function: $\mathcal{G}(\{l(\hat{\mathbf{y}}_i, \mathbf{y}_i)\}) = \mathcal{G}(\{l(f(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i)\})$
     - SRM — Regularization: $\Omega(\boldsymbol{\theta})$

Examples:

- Regression with L2 loss and L1 regularization

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 + \lambda \|\boldsymbol{\theta}\|_1$$

- Classification with cross entropy loss and L2 regularization

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(- \sum_{j=1}^{k} \mathbf{y}_{i,j} \log(\hat{\mathbf{y}}_{i,j}) \right) + \lambda \|\boldsymbol{\theta}\|_2^2$$

Machine Learning Terminologies

- Supervised learning has input labelled data
  - #instances x #attributes matrix/table
  - #attributes = #features + 1
    - 1 (usu. the last attribute) is for the class attribute

- Labelled data split into 2 or 3 disjoint subsets
  - Training data

    - Training error = $1.0 - \frac{\#correctly\_classified}{\#training\_instances}$

→ Build a model

  - Validation/development data

→ Select/refine the model

  - Testing data
    - Testing/generalization error = $1.0 - \frac{\#correctly\_classified}{\#testing\_instances}$

→ Evaluate the model

- We mainly discuss binary classification here
  - i.e., #labels = 2

- Overview of the whole process (not including cross-validation)

In practice, much more than this simple view

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Classification Process (1): Preprocessing & Feature Engineering

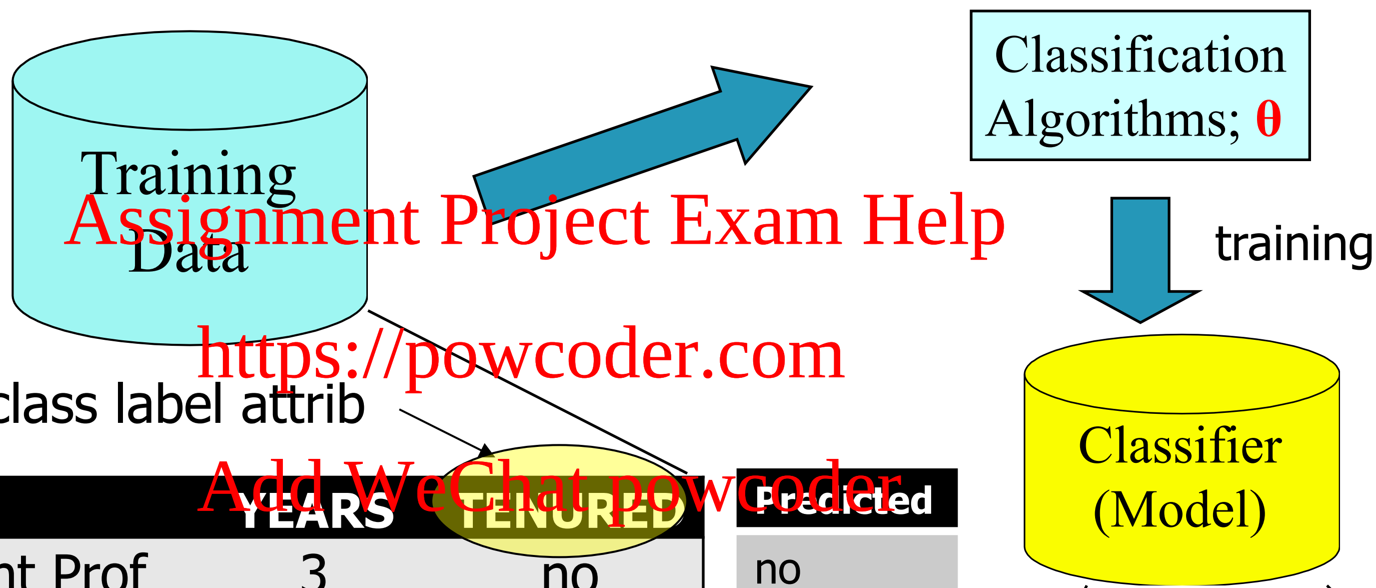
EID	Name	Title	EMP_Date	Track
101	Mike	Assistant Prof	2013-01-01	3-year contract
102	Mary	Assistant Prof	2009-04-15	Continuing
103	Bill	Scientia Professor	2014-02-03	Continuing
110	Jim	Associate Prof	2009-03-14	Continuing
121	Dave	Associate Prof	2009-12-02	1-year contract
234	Anne	Professor	2013-03-21	Future Fellow
188	Sarah	Student Officer	2008-01-17	Continuing

Raw Data

Training Data

Gender	RANK	YEARS	TENURED
M	Assistant Prof	3	no
F	Assistant Prof	7	yes
M	Professor	2	yes
M	Associate Prof	7	yes
M	Associate Prof	6	no
F	Professor	3	no

Classification Process (2): Training

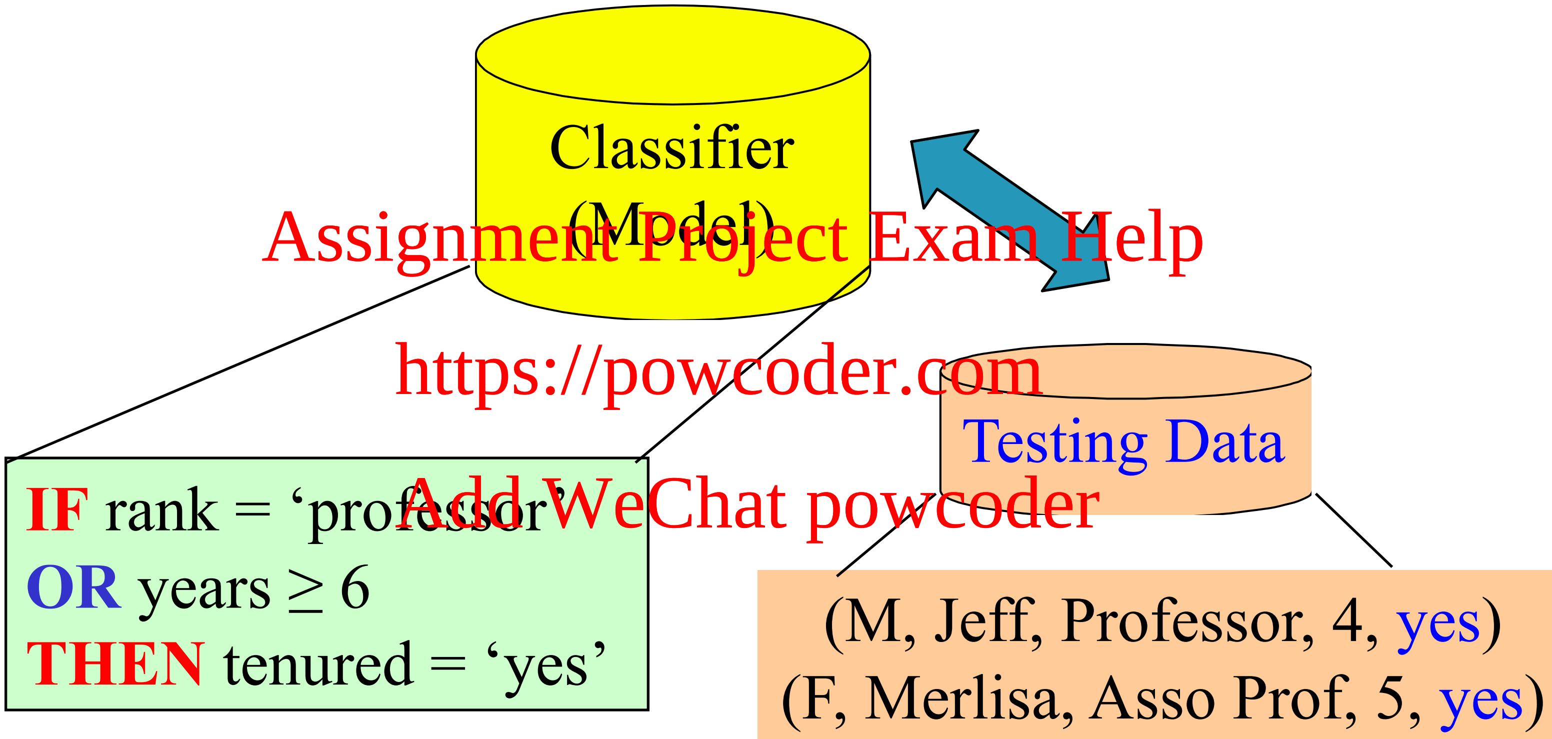


Gender	RANK	YEARS	TENURED	Predicted
M	Assistant Prof	3	no	no
F	Assistant Prof	7	yes	yes
M	Professor	2	yes	yes
M	Associate Prof	7	yes	yes
M	Associate Prof	6	no	**yes**
F	Professor	3	no	**yes**

IF rank = 'professor'
OR years $\geq$ 6
THEN tenured = 'yes'

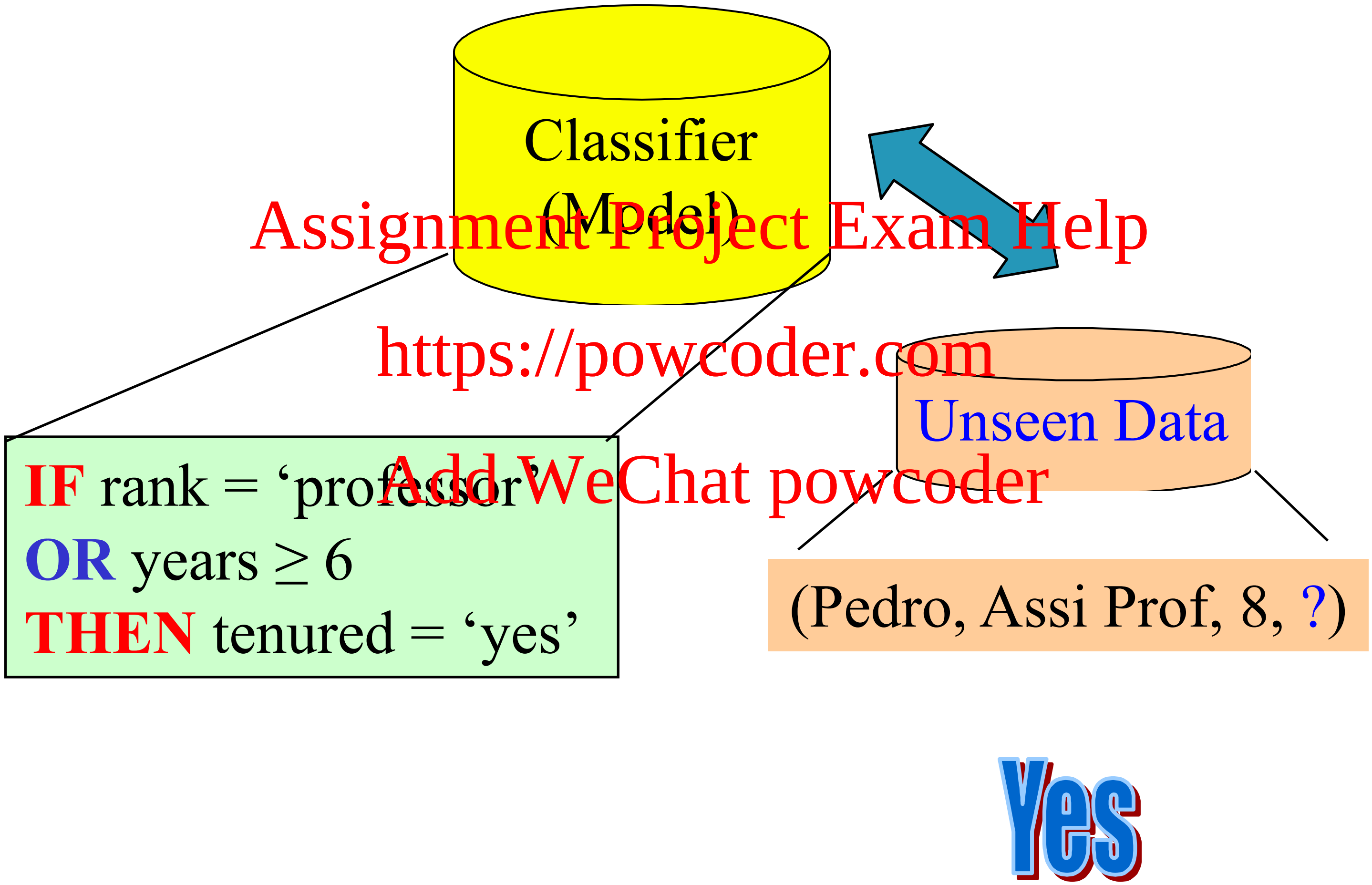
training error = 33.3%

Classification Process (3): Evaluate the Model on Testing Data



testing error = 50%

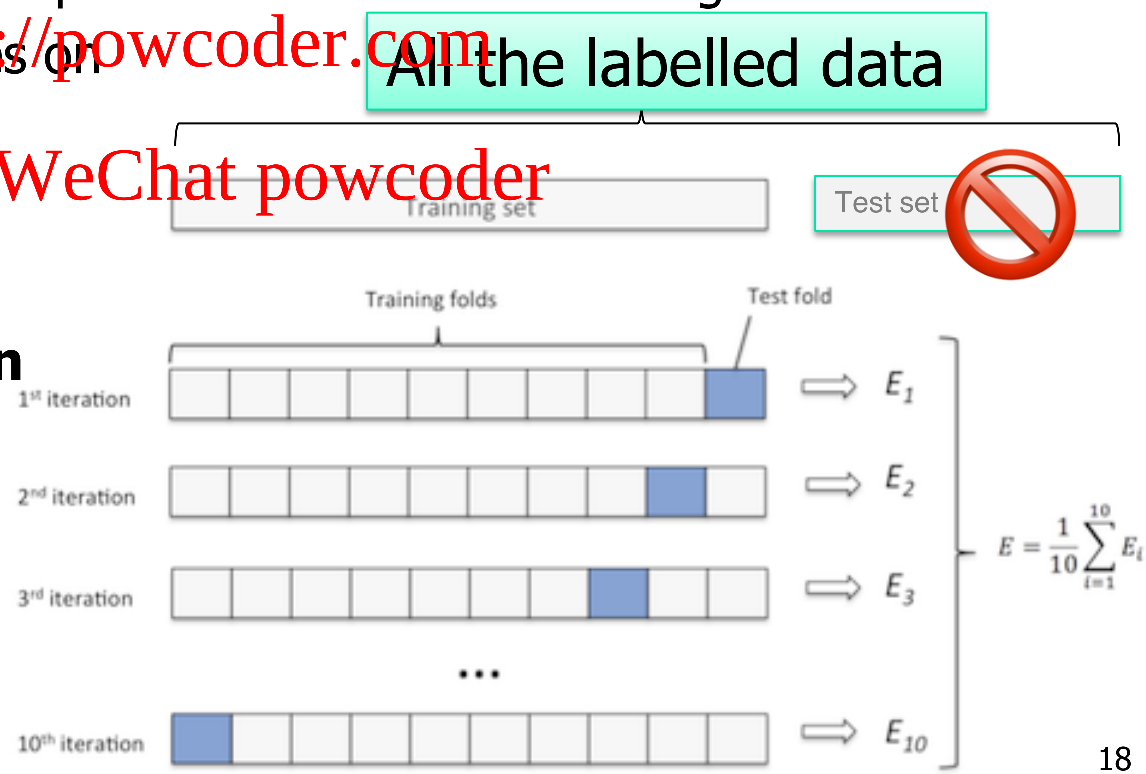
Classification Process (4): Use the Model in Production



How to judge a model?

- Based on training error or testing error?
  - Testing error
    - Otherwise, this is a kind of data scooping ➔ **overfitting**
- What if there are multiple models to choose from?
  - Further split a "test/development set" from the training set
- Can we trust the error values on the development set?
  - Need "large" dev set
  ➔ less data for training
  - **k-fold cross-validation**
  - k=n: leave-one-out

10-fold CV



Exercise: Problem definition and Feature Engineering

- How to formulate the following into ML problems? What kind of resources do you need? What are the features you think may be most relevant?

1. Predict the sale trend of a particular product in the next month

2. Design an algorithm to produce better top-10 ranking results for queries

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

- Decision Tree classifier
  - ID3
  - Other variants

Training Dataset

This follows an example from Quinlan's ID3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

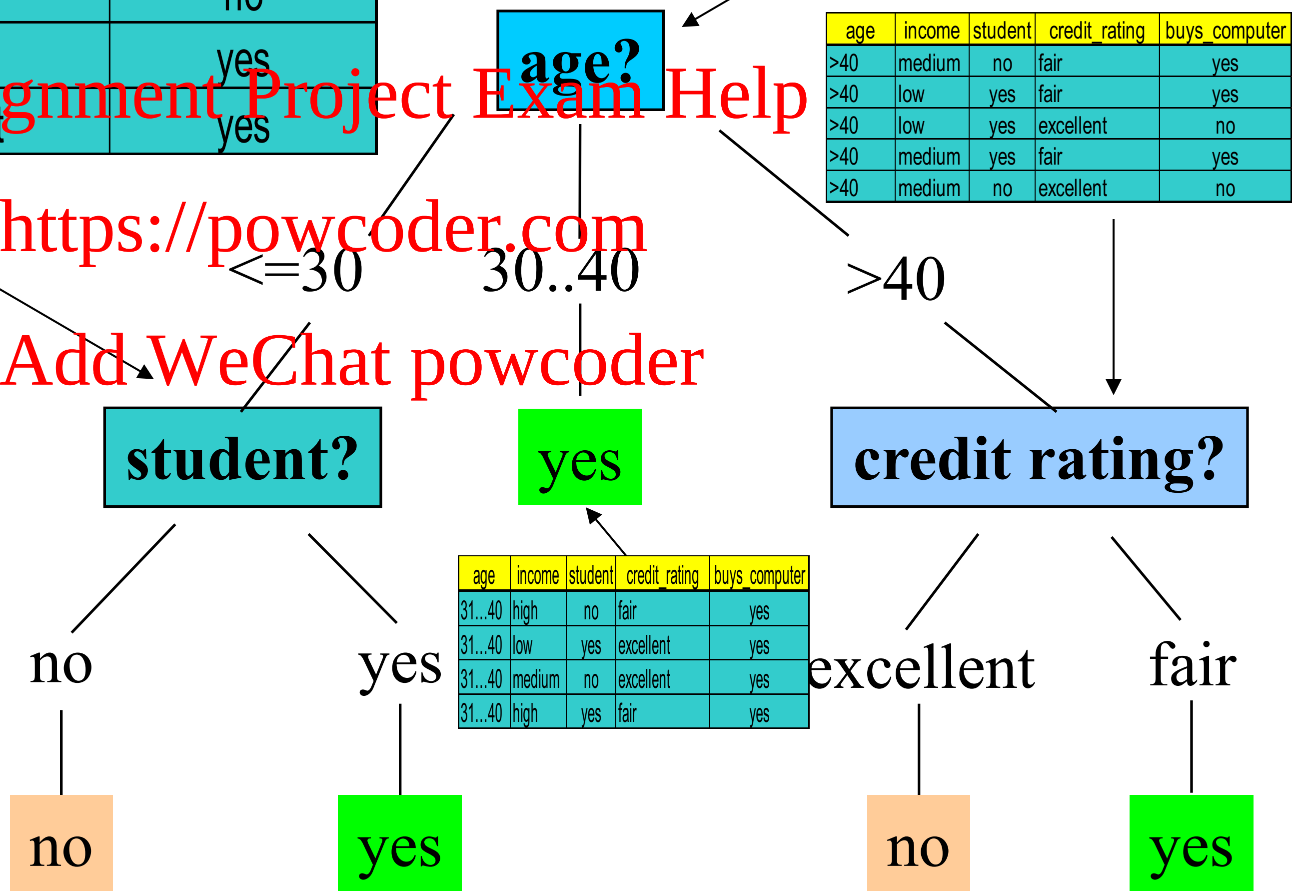
Output: A Decision Tree for

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	credit_rating	buys_computer
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

age	income	student	credit_rating	buys_computer
31...40	high	no	fair	yes
31...40	low	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes



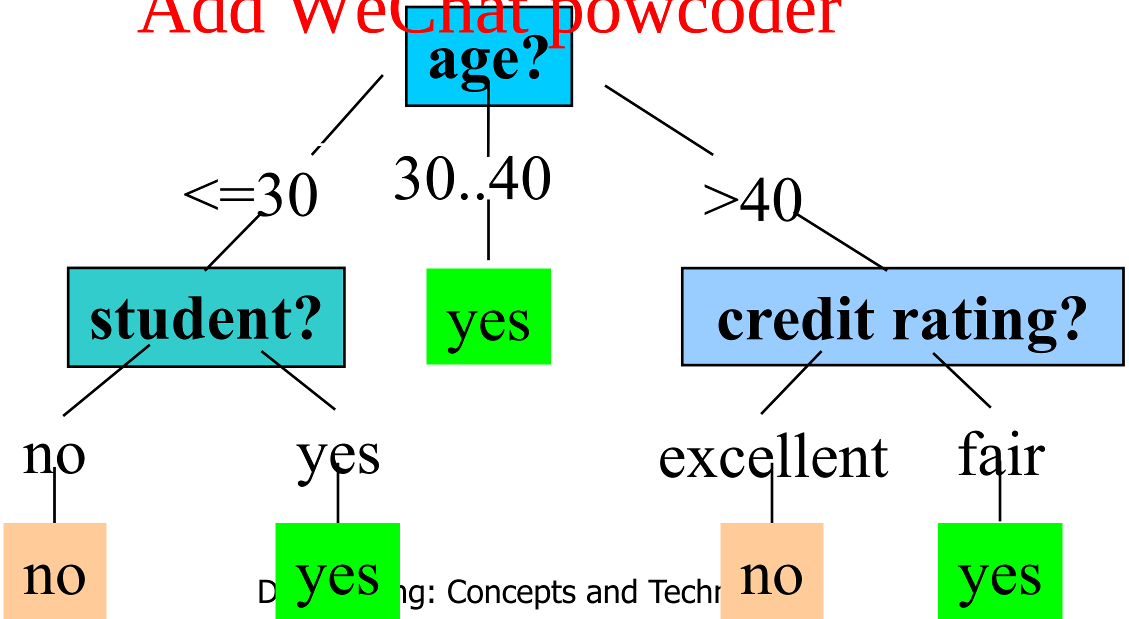
Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
  - Rules are easier for humans to understand
- One rule is created for each path from the root to a leaf
  - Each attribute-value pair along a path forms a conjunction
  - The leaf node holds the class prediction
  - Example

IF *age* = "<=30" **AND** *student* = "*no*" **THEN** *buys_computer* = "*no*"

IF *age* = "<=30" **AND** *student* = "*yes*" **THEN** *buys_computer* = "*yes*"

… … …



Exercise: Write down the pseudo-code of the induction algorithm

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Input: Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Overview: Tree is constructed in a top-down recursive <u>divide-and-conquer</u> manner
    - At start, all the training examples are at the root
    - Samples are partitioned recursively based on selected *test-attributes*
    - *Test-attributes* are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Three conditions for stopping partitioning (i.e., boundary conditions)
  - There are no samples left, OR
  - All samples for a given node belong to the same class, OR
  - There are no remaining attributes for further partitioning (majority voting is employed for classifying the leaf)

Decision Tree Induction Algorithm

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- S contains s_i tuples of class C_i for i = {1, …, m}
- information measures info required to classify any arbitrary tuple

$$I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}$$

- entropy of attribute A with values $\{a_1, a_2, \ldots, a_v\}$

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{s} I(s_{1j}, \ldots, s_{mj})$$

- information gained by branching on attribute A

$$Gain(A) = I(s_1, s_2, \ldots, s_m) - E(A)$$

Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- I(p, n) = I(9, 5) =0.940
- Compute the entropy for *age*:

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

$\frac{5}{14} I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = I(p,n) - E(age) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

income	p_i	n_i	$I(p_i, n_i)$
high	2	2	1
medium	4	2	0.918

Q: what's the extreme/worst case?

Other Attribute Selection Measures and Splitting Choices

- Gini index (CART, IBM IntelligentMiner)
  - All attributes are assumed continuous-valued

  - Assume there exist several possible split values for each attribute

  - May need other tools, such as clustering, to get the possible split values

  - Can be modified for categorical attributes
- Induces binary split => binary decision trees

Gini Index (IBM IntelligentMiner)

- If a data set T contains examples from n classes, gini index, $gini(T)$ is defined as

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2 \qquad = \sum_j p_j \cdot (1 - p_j)$$

  where p_j is the relative frequency of class j in T.

- If a data set T is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from n classes, the *gini* index $gini(T)$ is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

Case I: Numerical Attributes

Age	Car	Class
20	...	Y
20	...	N
20	...	N
25	...	N
25	...	Y
30	...	Y
30	...	Y
30	...	Y
40	...	Y
40	...	Y

Split value

22.5

27.5

...

Cut=22.5	Y	N
<	1	2
>=	6	1

$Gini_{split}(S)$ = 3/10*Gini(1,2) + 7/10*Gini(6,1) = 0.30

Cut=27.5	Y	N
<	2	3
>=	5	0

$Gini_{split}(S)$ = 5/10*Gini(2,3) + 5/10*Gini(5,0) = 0.24

... ...

$$Gini(S) = 1 - \sum p_j^2$$

$$Gini_{split}(S) = \frac{n_1}{n} Gini(S_1) + \frac{n_2}{n} Gini(S_2)$$

Exercise: compute gini indexes for other splits

Case II: Categorical Attributes

attrib list for **Car**

Age	Car	Class
20	M	Y
30	M	Y
25	T	N
30	S	Y
40	S	Y
20	T	N
30	M	Y
25	M	Y
40	M	Y
20	S	N

count matrix

	Class=Y	Class=N
M	5	0
T	0	2
S	2	1

Need to consider all possible splits !

	Y	N
{M, T}	5	2
{S}	2	1

	Y	N
{M, S}	7	1
{T}	0	2

	Y	N
{T, S}	2	3
{M}	5	0

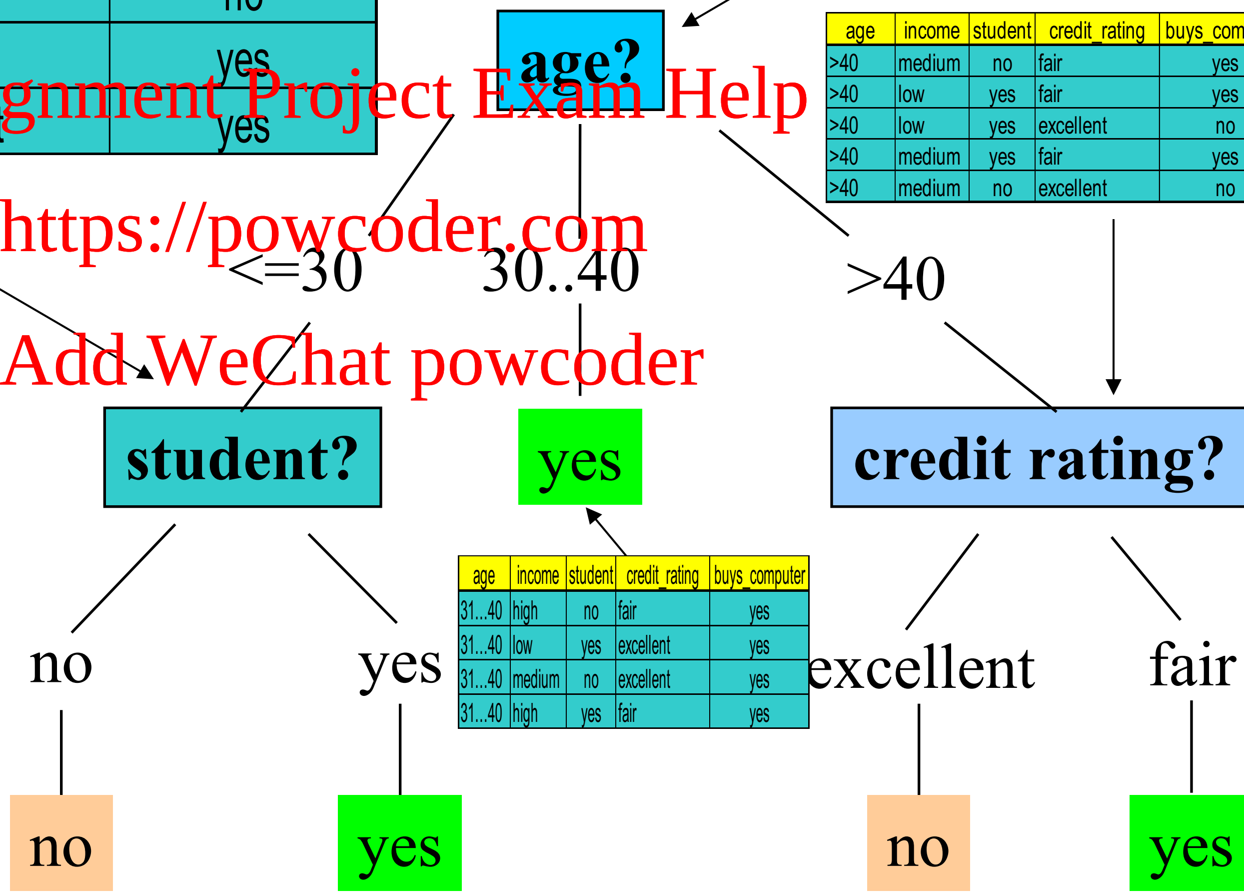
$Gini_{split}(S)$ = 7/10*Gini(5,2) + 3/10*Gini(2,1) = 0.42

$Gini_{split}(S)$ = 8/10*Gini(7,1) + 2/10*Gini(0,2) = 0.18

$Gini_{split}(S)$ = 5/10*Gini(2,3) + 5/10*Gini(5,0) = 0.24

ID3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



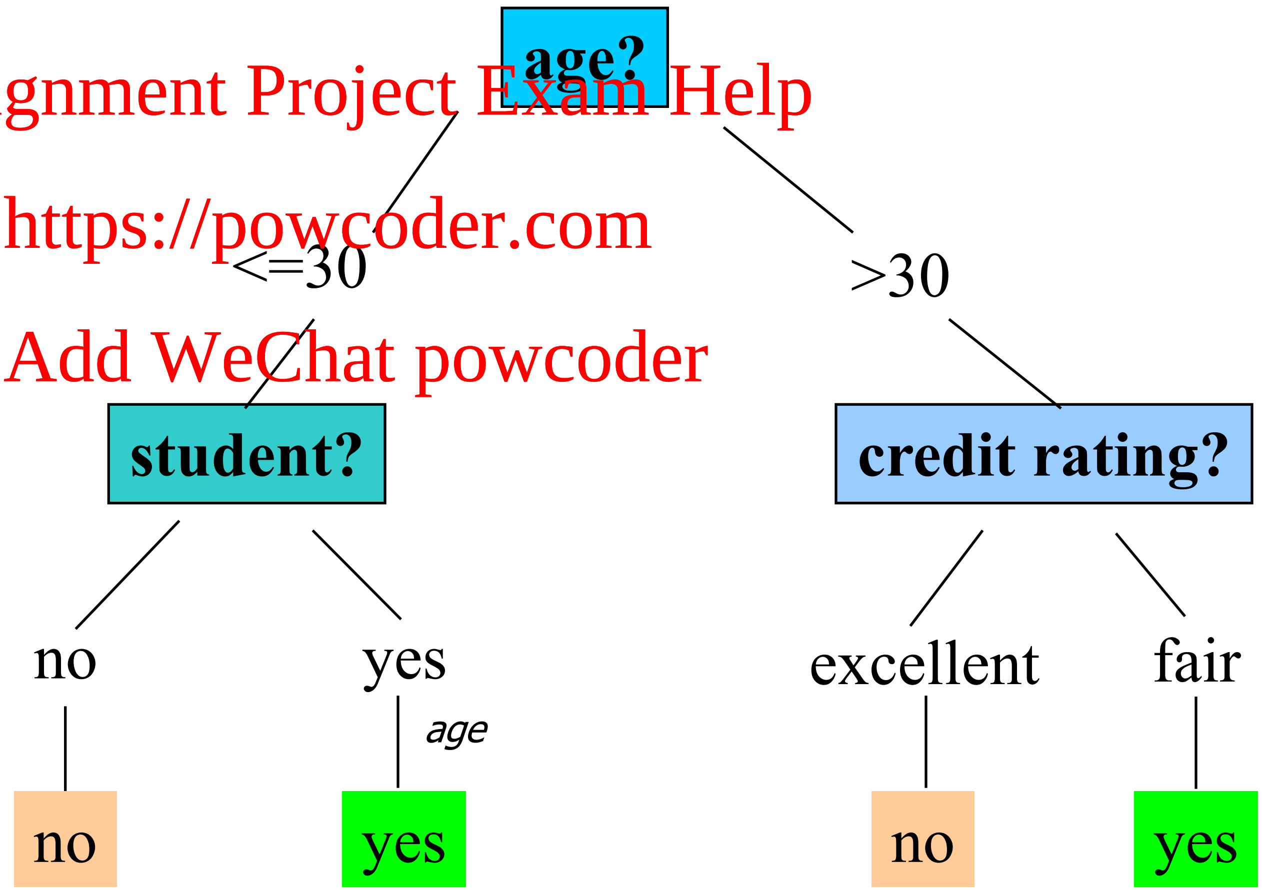
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	credit_rating	buys_computer
31...40	high	no	fair	yes
31...40	low	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes

age	income	student	credit_rating	buys_computer
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

CART/SPRINT

Illustrative of
the shape only



Avoid Overfitting in Classification

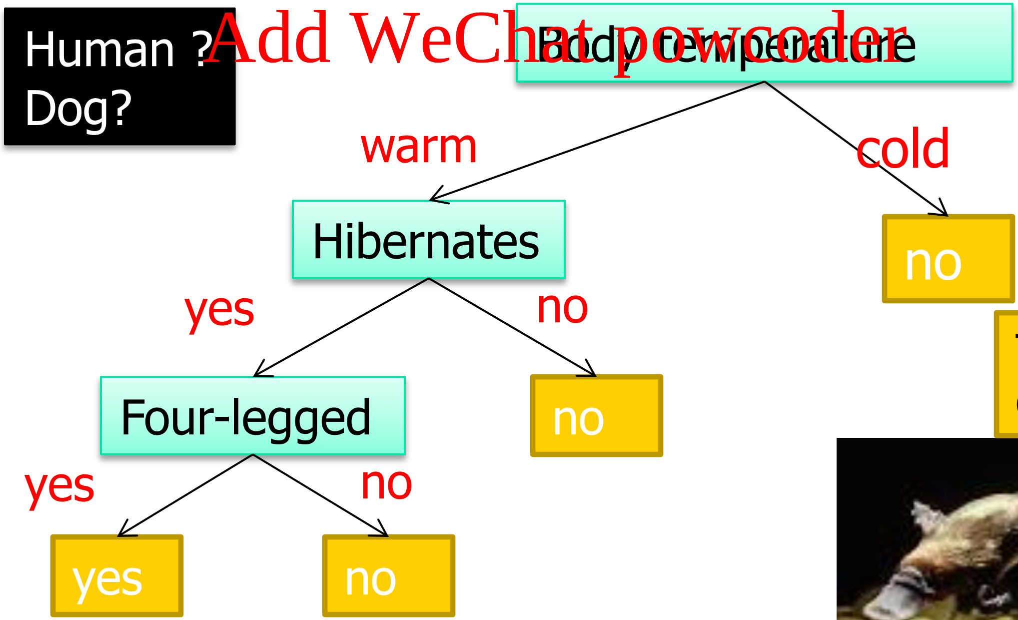
- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

- Lack of representative samples
- Existence of noise

Overfitting Example /1

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Is Mammal?
Salamander	Cold-blooded	No	Yes	Yes	No
Guppy	Cold-blooded	Yes	No	No	No
Eagle	Warm-blooded	No	No	No	No
Poorwill	Warm-blooded	No	No	Yes	No
Platypus	Warm-blooded	No	Yes	Yes	Yes

Human ?
Dog?



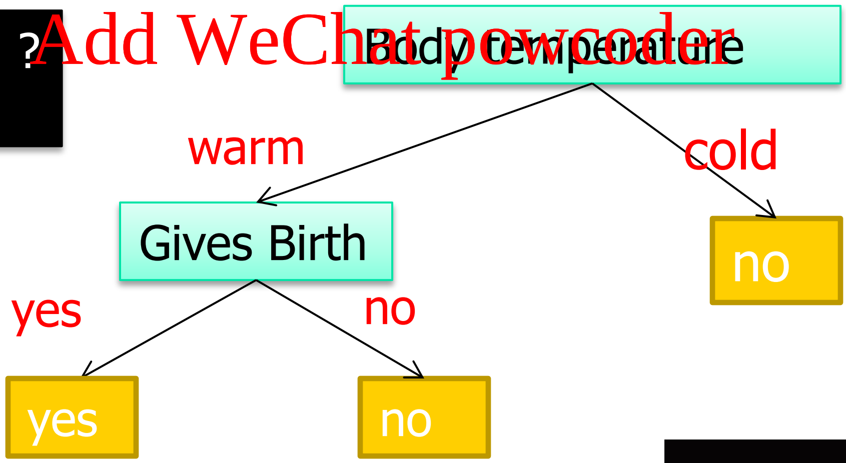
Training error = 0%, but does **not** generalize!

- Lack of representative samples
- Existence of noise

Overfitting Example /2

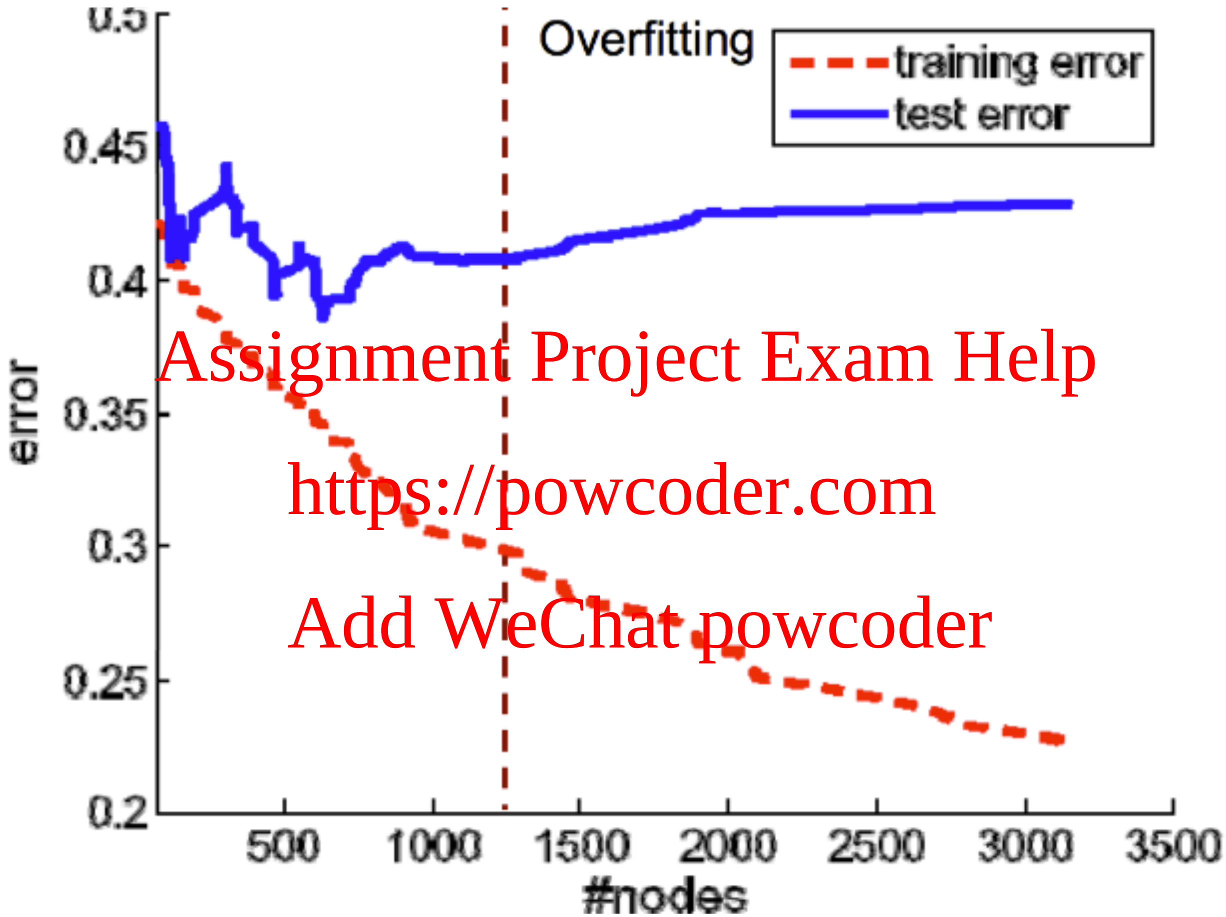
Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Is Mammal?
Salamander	Cold-blooded	No	Yes	Yes	No
Guppy	Cold-blooded	Yes	No	No	No
Eagle	Warm-blooded	No	No	No	No
Poorwill	Warm-blooded	No	No	Yes	No
Platypus	Warm-blooded	No	Yes	Yes	Yes

Human ?
Dog?



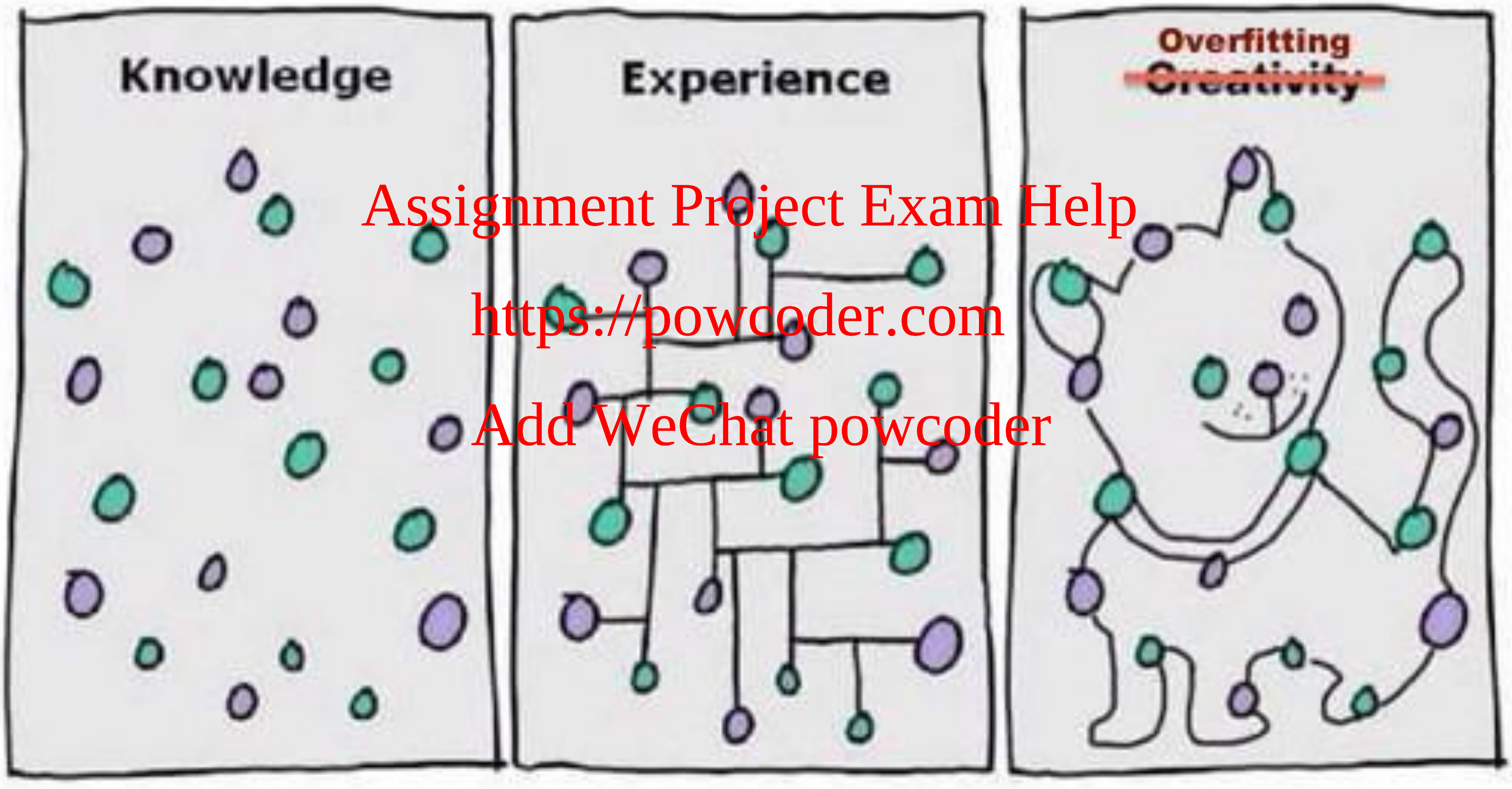
Training error = 20%, but **generalizes**!

Overfitting



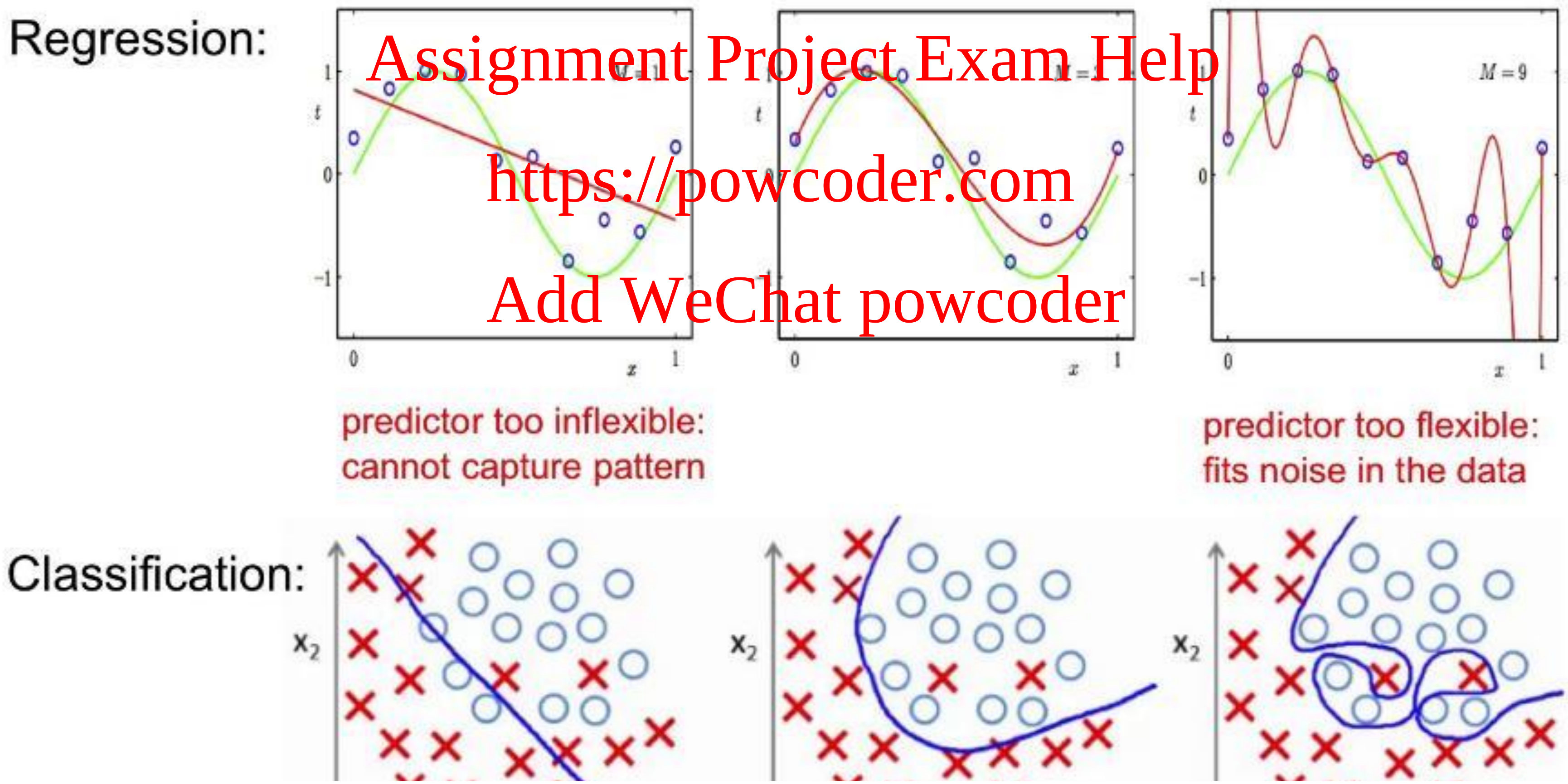
- Overfitting: model too complex ➔ training error keep decreasing, but testing error increases
- Underfitting: model too simple ➔ both training and testing has large errors.

Overfitting



Overfitting examples in Regression & Classification

Under- and Over-fitting examples



DT Pruning Methods

- Use a separate validation set
- Estimation of generalization/test errors
- Use all the data for training
  - but apply a statistical test (e.g., chi-square) to estimate whether expanding or pruning a node may improve the entire distribution
- Use minimum description length (MDL) principle
  - halting growth of the tree when the encoding is minimized

Pessimistic Post-pruning

Better estimate of generalization error in C4.5:
Use the upper 75% confidence bound from the training error of a node, assuming a binomial distribution

- Observed on the training data
  - $e(t)$: #errors on a leaf node t of the tree T
  - $e(T) = \Sigma_{t \text{ in } T}\, e(t)$
- What's the ***generalization errors*** (i.e., errors on testing data) on T?
  - Use pessimistic estimates
  - $e'(t) = e(t) + 0.5$
  - $E'(t) = e(T) + 0.5N$, where N is the number of leaf nodes in T
- What's the generalization errors on root(T) only?
  - $E'(root(T)) = e(T) + 0.5$
- Post-pruning from bottom-up
  - If generalization error reduces after pruning, replace sub-tree by a leaf node
  - Use majority voting to decide the class label

Example

Class = Yes	20
Class = No	10

Error = 10/30

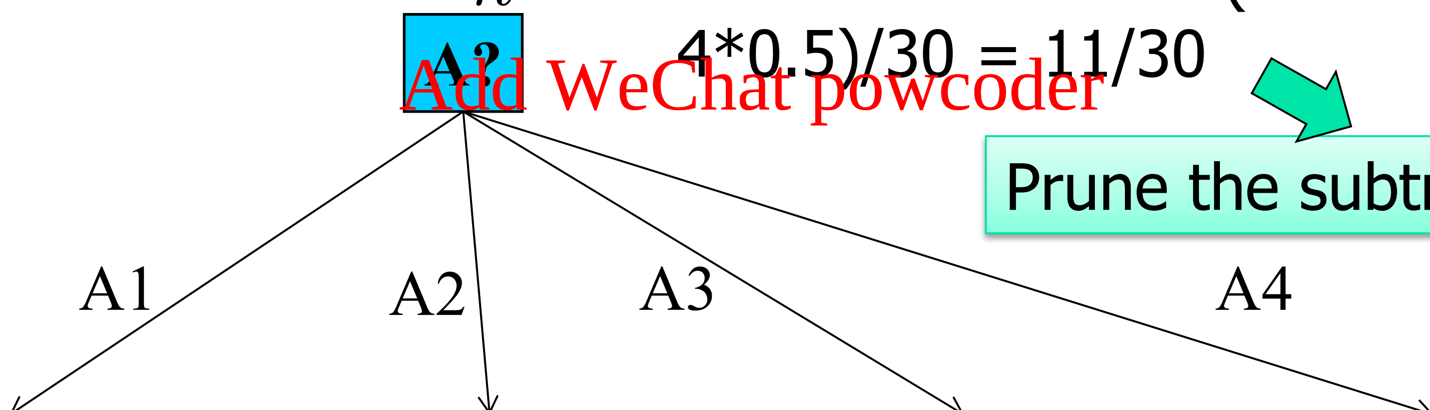
$\hat{\varepsilon}_h \rightarrow \mathcal{E}_h$

- Training error before splitting on A = 10/30
- Pessimistic error = (10+0.5)/30

$\hat{\varepsilon}_{h'} \rightarrow \mathcal{E}_{h'}$

- Training error after splitting on A = 9/30
- Pessimistic error = (9 + 4*0.5)/30 = 11/30

Prune the subtree at A



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why decision tree induction in data mining?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods

Enhancements to basic decision tree induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals, one-hot encoding, or specialized DT learning algorithms
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

- Bayesian Classifiers

Bayesian Classification: Why?

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems

- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.

- Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem: Basics

- Let X be a data sample whose class label is unknown
- Let h be a hypothesis that X belongs to class C

- For classification problems, determine P(h|X): the probability that the hypothesis holds given the observed data sample X

- P(h): prior probability of hypothesis h (i.e. the initial probability before we observe any data, reflects the background knowledge)
- P(X): probability that sample data is observed
- P(X|h) : probability of observing the sample X, given that the hypothesis holds

Bayesian Theorem

- Given training data *X, posteriori probability of a hypothesis h, P(h|X)* follows the Bayes theorem

$$P(h \mid X) = \frac{P(X|h)P(h)}{P(X)}$$

- Informally, this can be written as

posterior = likelihood x prior / evidence

- MAP (maximum posteriori) hypothesis

$$h_{\mathrm{MAP}} = \arg\max_{h \in H} P(h \mid X) = \arg\max_{h \in H} P(X|h)P(h)$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Training dataset

Hypotheses:
C_1:buys_computer= 'yes'
C_2:buys_computer= 'no'

Data sample
X =(**age**<=30,
Income=medium,
Student=yes,
Credit_rating=
Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Sparsity Problem

$$h_{\text{MAP}} = \arg\max_{h \in H} P(X|h)P(h)$$

- Maximum likelihood Estimate of P(h)
  - Let p be the probability that the class is C1
  - Consider a training example x and its label y
  - $L(x) = p^y(1-p)^{1-y}$
  - $L(X) = \Pi_{x \text{ in } X}\, L(x)$
  - $l(X) = \log(L(X)) = \Sigma_{x \text{ in } X}\, y\log(p) + (1-y)\log(1-p)$
  - To maximize $l(X)$, let $dl(X)/dp = 0$ ➔ $p = (\Sigma y)/n$
  - $P(C_1) = 9/14$, $P(C_2) = 5/14$
- ML estimate of P(X|h) = ?
  - Requires $O(2^d)$ training examples, where d is the #features.

Data likelihood

Log Data likelihood

Curse of dimensionality

Naïve Bayes Classifier

- Use a model
  - Assumption: attributes are conditionally independent:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

  - The product of occurrence of say 2 elements x_1 and x_2, given the current class is C, is the product of the probabilities of each element taken separately, given the same class $P([x_1,x_2],C) = P(x_1,C) * P(x_2,C)$

  - No dependence relation between attributes given the class
  - Greatly reduces the computation cost, only count the class distribution ➔ Only need to estimate $P(x_k \mid C_i)$

Naïve Bayesian Classifier: Example

- Compute P(X|Ci) for each class

X=(age<=30 , income =medium, student=yes, credit_rating=fair)

P(age=“<30” | buys_computer=“yes”) = 2/9=0.222

P(age=“<30” | buys_computer=“no”) = 3/5 =0.6
P(income=“medium” | buys_computer=“yes”)= 4/9 =0.444

P(income=“medium” | buys_computer=“no”) = 2/5 = 0.4
P(student=“yes” | buys_computer=“yes)= 6/9 =0.667
P(student=“yes” | buys_computer=“no”)= 1/5=0.2

P(credit_rating=“fair” | buys_computer=“yes”)=6/9=0.667
P(credit_rating=“fair” | buys_computer=“no”)=2/5=0.4

P(X|Ci) : P(X|buys_computer=“yes”)= 0.222 x 0.444 x 0.667 x 0.0.667 =0.044
P(X|buys_computer=“no”)= 0.6 x 0.4 x 0.2 x 0.4 =0.019
P(X|Ci)*P(Ci) : P(X|buys_computer=“yes”) * P(buys_computer=“yes”)=0.028
P(X|buys_computer=“no”) * P(buys_computer=“no”)=0.007

X belongs to class “buys_computer=yes”

likelihood

The Need for Smoothing

- $\Pr[X_i = v_j \mid C_k]$ could still be 0, if not observed in the training data
  - makes $\Pr[C_k \mid X] = 0$, regardless of other likelihood values of $\Pr[C_t = v_t \mid C_k]$
- Add-1 Smoothing
  - reserve a small amount of probability for unseen probabilities
  - (conditional) probabilities of observed events have to be adjusted to make the total probability equals 1.0

Many other kinds of smoothing exist (esp. in Natural Language Modelling)

Add-1 Smoothing

- $\Pr[X_i = v_j \mid C_k] = \text{Count}(X_i = v_j, C_k) \quad / \text{Count}(C_k)$
- $\Pr[X_i = v_j \mid C_k] = [\text{Count}(X_i = v_j, C_k) + 1] / [\text{Count}(C_k) + B]$

- What's the right value for B?
  - make $\Sigma_{vj} \Pr[X_i = v_j \mid C_k] = 1$
  - Explain the above constraint
    - $\Pr[X_i \mid C_k]$: Given Ck, the (conditional) probability of X_i taking a specific value
    - X_i must take one of the values, hence summing over all possible value for X_i, the probabilities should sum up to 1.0
  - B = dom(X_i), i.e., # of values X_i can take

Smoothing Example

no instance of age <= 30 in the “No” class

Class:
C1:buys_computer= ‘yes’
C2:buys_computer= ‘no’

Data sample
X =(**age**<=30,
Income=medium,
Student=yes,
Credit_rating= Fair)

age	income	student	credit_rating	buys_computer
30…40	high	no	fair	no
30…40	high	no	excellent	no
30…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
30…40	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

B=3 B=3 B=2 B=2

Consider the “No” Class

- Compute P(X|Ci) for the “No” class

X=(age<=30 , income =medium, student=yes, credit_rating=fair)

P(age=“<30” | buys_computer=“no”) = (0 +1) / (5 +3) = 0.125

P(income=“medium” | buys_computer=“no”) = (2 +1) / (5 +3) = 0.375

P(student=“yes” | buys_computer=“no”)= (1 +1) / (5 +2) = 0.286

P(credit_rating=“fair” | buys_computer=“no”)= (2 +1) / (5 +2) = 0.429

P(X|Ci) : P(X|buys_computer=“no”)= 0.125 x 0.375 x 0.286 x 0.429 = 0.00575

P(X|Ci)*P(Ci) : P(X|buys_computer=“no”) * P(buys_computer=“no”) = 0.00205

Probabilities	Without Smoothing	With Smoothing
Pr[<=30 | No]	0 / 5	1 / 8
Pr[30..40 | No]	3 / 5	4 / 8
Pr[>=40 | No]	2 / 5	3 / 8

sum up to 1

How to Handle Numeric Values

- Need to model the distribution of $\Pr[X_i \mid C_k]$
- Method 1:
  - Assume it to be Gaussian ➔ Gaussian Naïve Bayes
  - $\Pr[X_i = v_j \mid C_k] = \dfrac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\dfrac{(v_j - \mu_{ik})^2}{2\sigma_i^2}\right)$

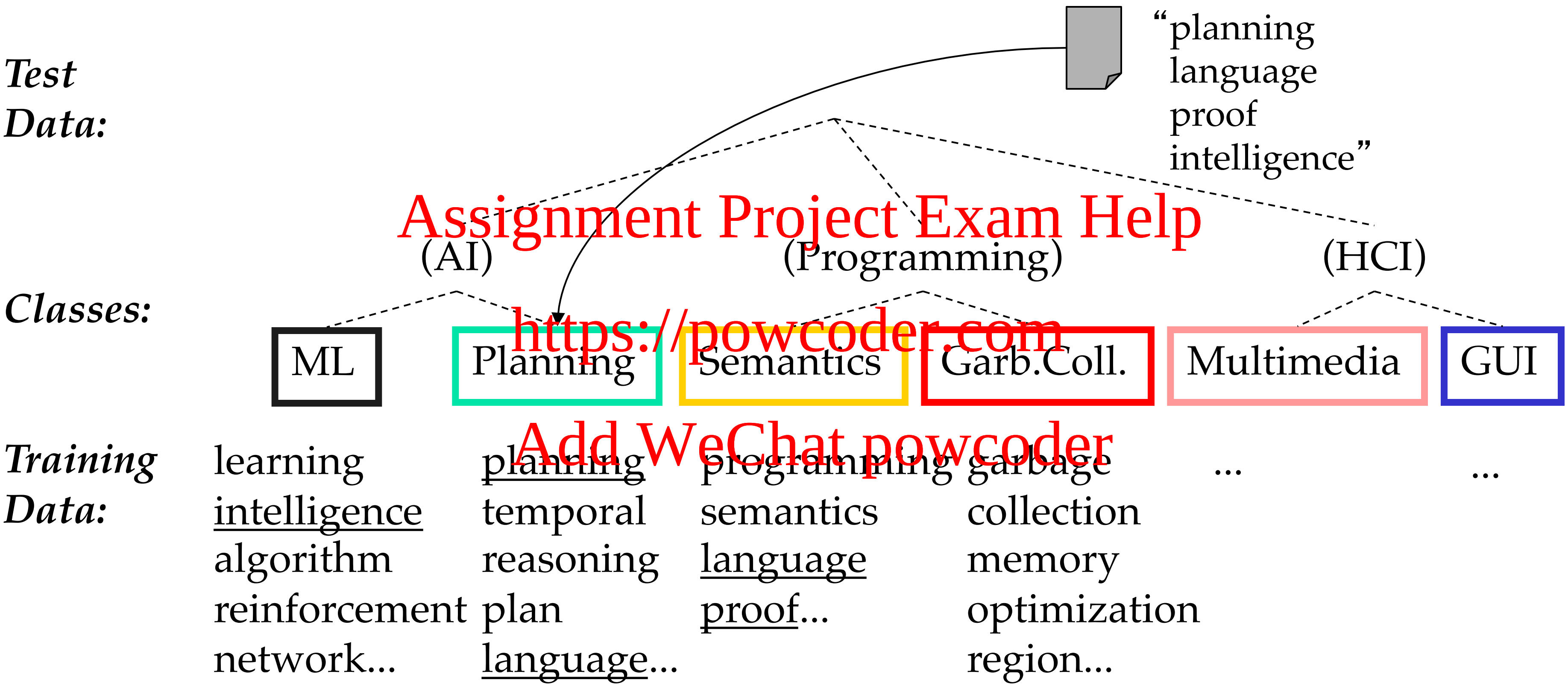
- Method 2:
  - Use binning to discretize the feature values

Text Classification

- NB has been widely used in **text classification** (aka., text categorization)
- Outline:
  - Applications
  - Language Model
  - Two Classification Methods

Based on "Chap 13: Text classification & Naive Bayes" in Introduction to Information Retrieval
- http://nlp.stanford.edu/IR-book/

Document Classification



(Note: in real life there is often a hierarchy, not present in the above problem statement; and also, you get papers on ML approaches to Garb. Coll.)

More Text Classification Examples:

Many search engine functionalities use classification

Assign labels to each document or web-page:

- Labels are most often topics such as Yahoo-categories

  e.g., "finance," "sports," "news>world>asia>business"

- Labels may be genres

  e.g., "editorials" "movie-reviews" "news“

- Labels may be opinion on a person/product

  e.g., “like”, “hate”, “neutral”

- Labels may be domain-specific

  e.g., "interesting-to-me" : "not-interesting-to-me”

  e.g., “contains adult language”: “doesn’t”

  e.g., language identification: English, French, Chinese, …

  e.g., search vertical: about Linux versus not

  e.g., “link spam”: “not link spam”

Challenge in Applying NB to Text

- Need to model P(text | class)
  - e.g., P("a b c d" | class)
- Extremely sparse → Need a model to help
- 1st method:
  - Based on a statistical language model
    - Turns out to be the **bag** of words model if using unigrams
  - → multinomial NB

sklearn.naive_bayes.MultinomialNB

- 2nd method:
  - View a text as a **set** of tokens → a Boolean vector in $\{0, 1\}^{|V|}$, where V is the vocabulary.
  - → Bernoulli NB

sklearn.naive_bayes.BernoulliNB

Unigram and higher-order Language models in Information Retrieval

- P("a b c d") =
  - P(a) * P(b | a) * P(c | a b) * P(d | a b c)

- Unigram model: 0-th order Markov Model
  - P(d | a b c) = P(d)

Easy. Effective!

- Bigram Language Models: 1st order Markov Model
  - P(d | a b c) = P(d | c)
  - P(a b c d) = P(a) * P(b | a) * P(c | b) * P(d | c)

- The same with class-conditional probabilities, i.e., P("a b c d" | C)

Two Models /1

- Model 1: Multinomial = Class conditional unigram
  - One feature X_i for each word pos in document
    - feature's values are all words in dictionary
  - Value of X_i is the word in position i
  - Naïve Bayes assumption:
    - Given the document's topic, word in one position in the document tells us nothing about words in other positions
  - Second assumption:
    - Word appearance does not depend on position

$$P(X_i = w \mid c) = P(X_j = w \mid c)$$

for all positions *i*,*j*, word *w*, and class *c*

    - Just have one multinomial feature predicting all words

Using **Multinomial** Naive Bayes Classifiers to Classify Text: Basic method

$$P(C_j \mid w_1 w_2 w_3 w_4) \propto P(C_j) P(w_1 w_2 w_3 w_4 \mid C_j)$$

an example text of 4 tokens

$$\propto P(c_j) \prod_{i=1}^{4} P(w_i \mid C_j)$$

unigram model

$$\propto P(c_j) \prod_{i=1}^{|V|} P(x_i \mid C_j)^{\theta_i}$$

bag of word; multinomial

- Essentially, the classification is *independent* of the positions of the words
  - Use same parameters for each position
  - Result is **bag** of words model, i.e. $\text{text} \equiv \{x_i : \theta_i\}_{i=1}^{|V|}$

e.g., "to be or not to be"

Naïve Bayes: Learning

- From training corpus, extract $V = Vocabulary$
- Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms
  - For each c_j in C do
    - $docs_j \leftarrow$ subset of documents for which the target class is c_j
    - $$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$
  - $Text_j \leftarrow$ single document containing all $docs_j$
  - for each word x_k in $Vocabulary$
    - $n_k \leftarrow$ number of occurrences of x_k in $Text_j$
    - $$P(x_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Naïve Bayes: Classifying

- positions ← all word positions in current document which contain tokens found in *Vocabulary*

- Return c_{NB}, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Naive Bayes: Time Complexity

- Training Time: $O(|D|L_d + |C||V|))$
  where L_d is the average length of a document in D.
  - Assumes V and all D_i, n_i, and n_{ij} pre-computed in $O(|D|L_d)$ time during one pass through all of the data.
  - Generally just $O(|D|L_d)$ since usually $|C||V| < |D|L_d$
- Test Time: $O(|C|\ L_t)$
  where L_t is the average length of a test document.
- Very efficient overall, linearly proportional to the time needed to just read in all the data.

Why?

Bernoulli Model

- V = {a, b, c, d, e} = $\{x_1, x_2, x_3, x_4, x_5\}$
- Feature functions f_i(text) = if text contains x_i

- The feature functions extract a vector of $\{0, 1\}^{|V|}$ from any text

- Apply NB directly

"a b c d"
"d e b"

a	b	c	d	e	C
1	1	1	1	0	+
0	1	0	1	1	-

Two Models /2

- Model 2: Multivariate Bernoulli
  - One feature X_w for each word in dictionary
  - X_w = true in document d if w appears in d
  - Naive Bayes assumption:
    - Given the document's topic, appearance of one word in the document tells us nothing about chances that another word appears
- This is the model used in the binary independence model in classic probabilistic relevance feedback in hand-classified data (Maron in IR was a very early user of NB)

Parameter estimation

- Multivariate Bernoulli model:

$$\hat{P}(X_w = t \mid c_j) = \text{fraction of documents of topic } c_j \text{ in which word } w \text{ appears}$$

- Multinomial model:

$$\hat{P}(X_i = w \mid c_j) = \text{fraction of times in which word } w \text{ appears across all documents of topic } c_j$$

  - Can create a mega-document for topic *j* by concatenating all documents in this topic
  - Use frequency of *w* in mega-document

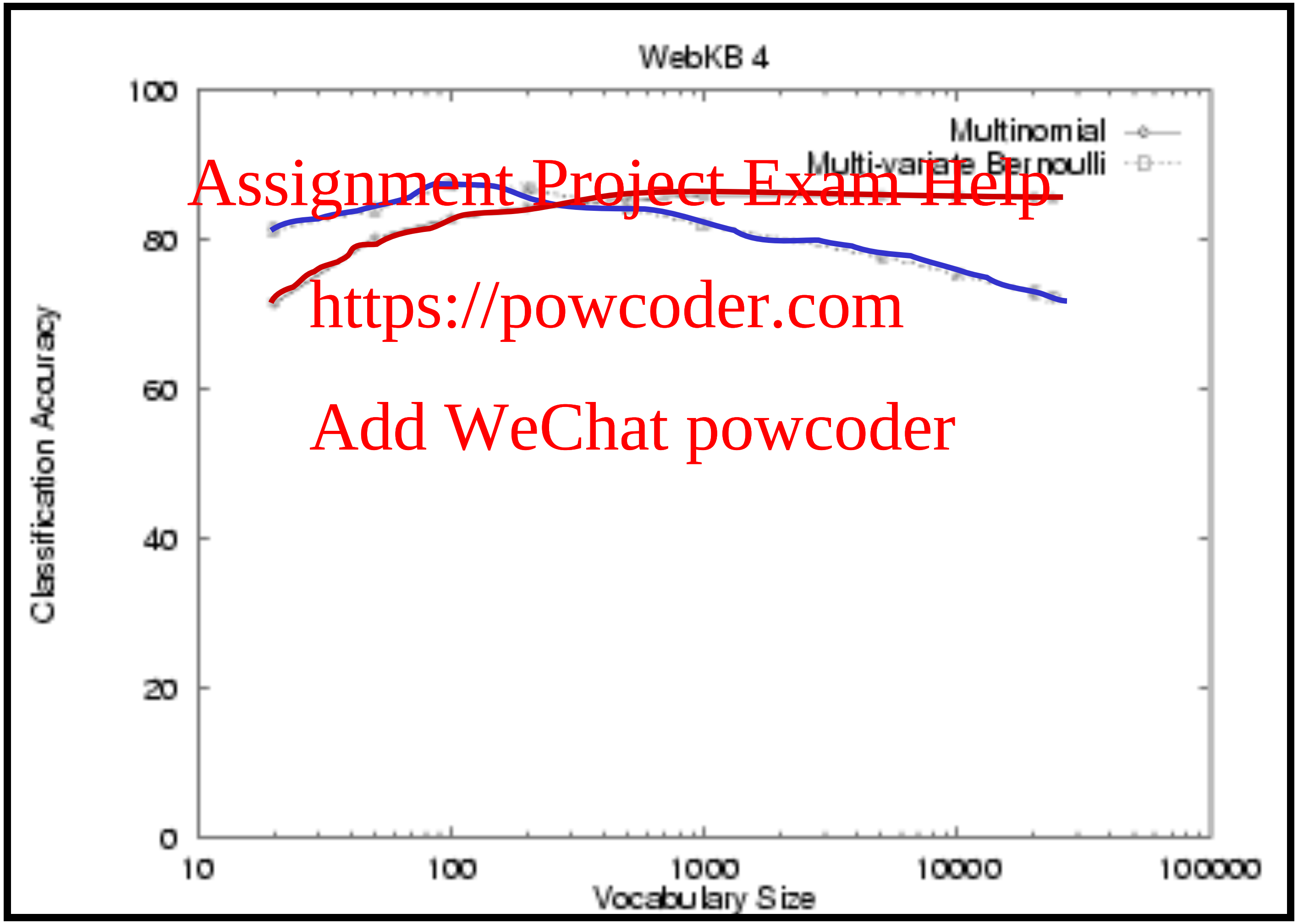
Classification

- Multinomial vs Multivariate Bernoulli?

- Multinomial model is almost always more effective in text applications!
  - See results figures later

- See *IIR* sections 13.2 and 13.3 for worked examples with each model

NB Model Comparison: WebKB



Underflow Prevention: log space

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since log(xy) = log(x) + log(y), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

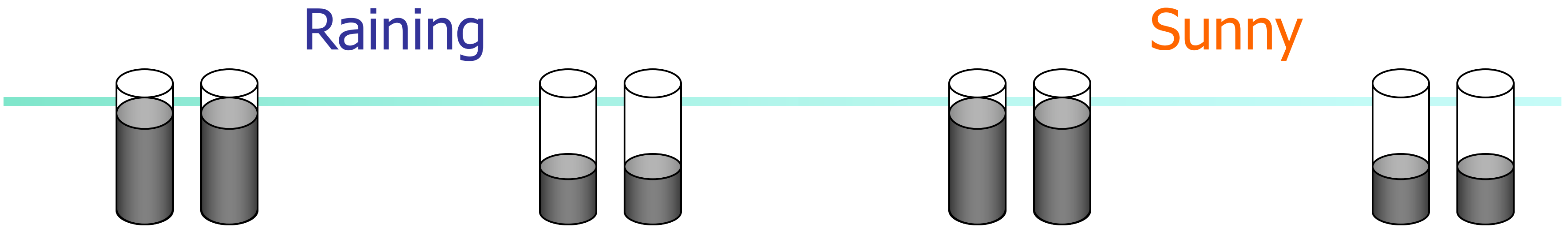
$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

- Note that model is now just max of sum of weights…

Violation of NB Assumptions

- **Conditional independence**
- "Positional independence"
- Examples?

Observations



$P(+,+,r) = 3/8$ $P(-,-,r) = 1/8$ $P(+,+,s) = 1/8$ $P(-,-,s) = 3/8$

- Two identical sensors at the same location
- Equivalent training dataset
- Note: $P(s_1|C) = P(s_2|C)$ no matter what

S1	S2	Class
+	+	r
+	+	r
+	+	r
-	-	r
+	+	s
-	-	s
-	-	s
-	-	s

$P(s_i = + \mid r) =$

$P(s_i = - \mid r) =$

$P(r) =$

$P(s) =$

$P(s_i = + \mid s) =$

$P(s_i = - \mid s) =$

Prediction

- $P(r \mid ++) \propto$

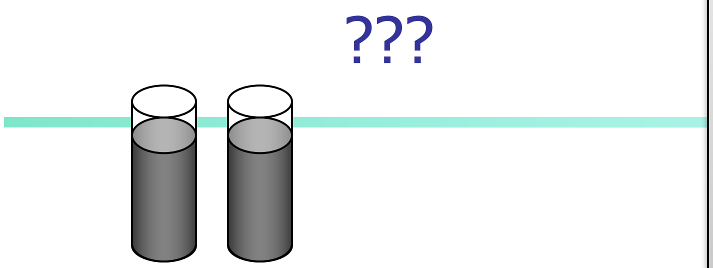
S1	S2	Class
+	+	???

- $P(s \mid ++) \propto$

$P(s_i = + \mid r) = 3/4$
$P(s_i = - \mid r) = 1/4$

$P(s_i = + \mid s) = 1/4$
$P(s_i = - \mid s) = 3/4$

$P(r) = 1/2$
$P(s) = 1/2$



Problem: Posterior probability estimation not accurate

Reason: Correlation between features

Fix: Use logistic regression classifier

- $P(r \mid ++) \propto 9/32$
- $P(s \mid ++) \propto 1/32$

- $P(r \mid ++) = 9/10$
- $P(s \mid ++) = 1/10$

$P(s_i = + \mid r) = 3/4$
$P(s_i = - \mid r) = 1/4$

$P(s_i = + \mid s) = 1/4$
$P(s_i = - \mid s) = 3/4$

$P(r) = 1/2$
$P(s) = 1/2$

S1	S2	Class
+	+	r
+	+	r
+	+	r
-	-	r
+	+	s
-	-	s
-	-	s
-	-	s

Naïve Bayes Posterior Probabilities

- Classification results of naïve Bayes (the class with maximum posterior probability) are usually fairly accurate.

- However, due to the inadequacy of the conditional independence assumption, the actual posterior-probability numerical estimates are not.
  - Output probabilities are commonly very close to 0 or 1.

- Correct estimation $\Rightarrow$ accurate prediction, but correct probability estimation is NOT necessary for accurate prediction (just need right ordering of probabilities)

Naïve Bayesian Classifier: Comments

- Advantages :
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence , therefore loss of accuracy
  - Practically, dependencies exist among variables
  - E.g., hospitals: patients: Profile: age, family history etc

    Symptoms: fever, cough etc., Disease: lung cancer, diabetes etc
  - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- Better methods?
  - Bayesian Belief Networks
  - Logistic regression / maxent

- (Linear Regression and) Logistic Regression Classifier
  - See LR slides

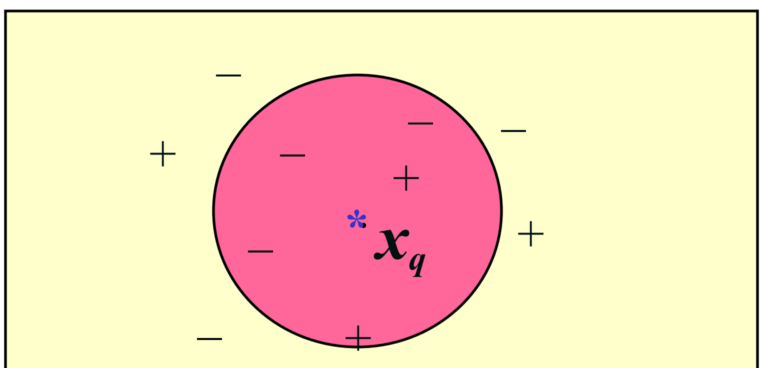
- Other Classification Methods

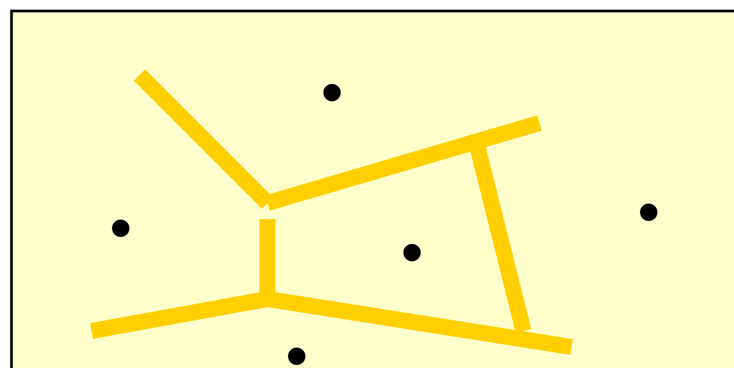
Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.

The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the *n*-D space.
- The nearest neighbor are defined in terms of Euclidean distance.
- The target function could be discrete- or real- valued.
- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to x_q.
- Vonoroi diagram: the *decision boundary* induced by 1-NN for a typical set of training examples.





Bayesian Perspective

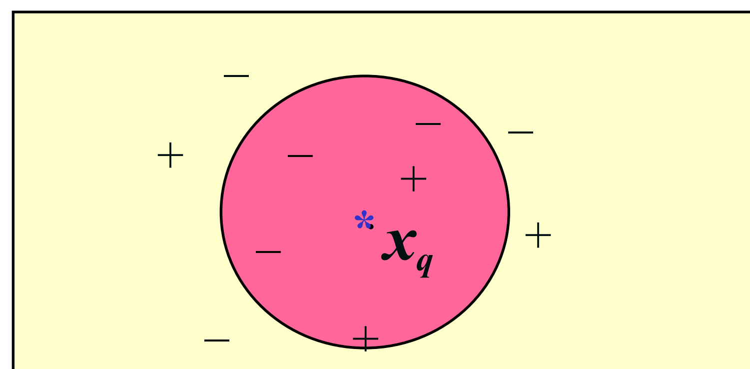
- An informal way to view kNN classifier as a Bayesian model
- Set up:
  - kNN Ball around a test instance $\mathbf{x}$: $B_k(\mathbf{x})$
  - Class i ➔ has N_i training instances
  - Class I ➔ has k_i instances within $B_k(\mathbf{x})$
  - Volume of $B_k(\mathbf{x})$: V

$$P(C_i \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_i)P(C_i)}{P(\mathbf{x})} \qquad P(C_i) = \frac{N_i}{N}$$

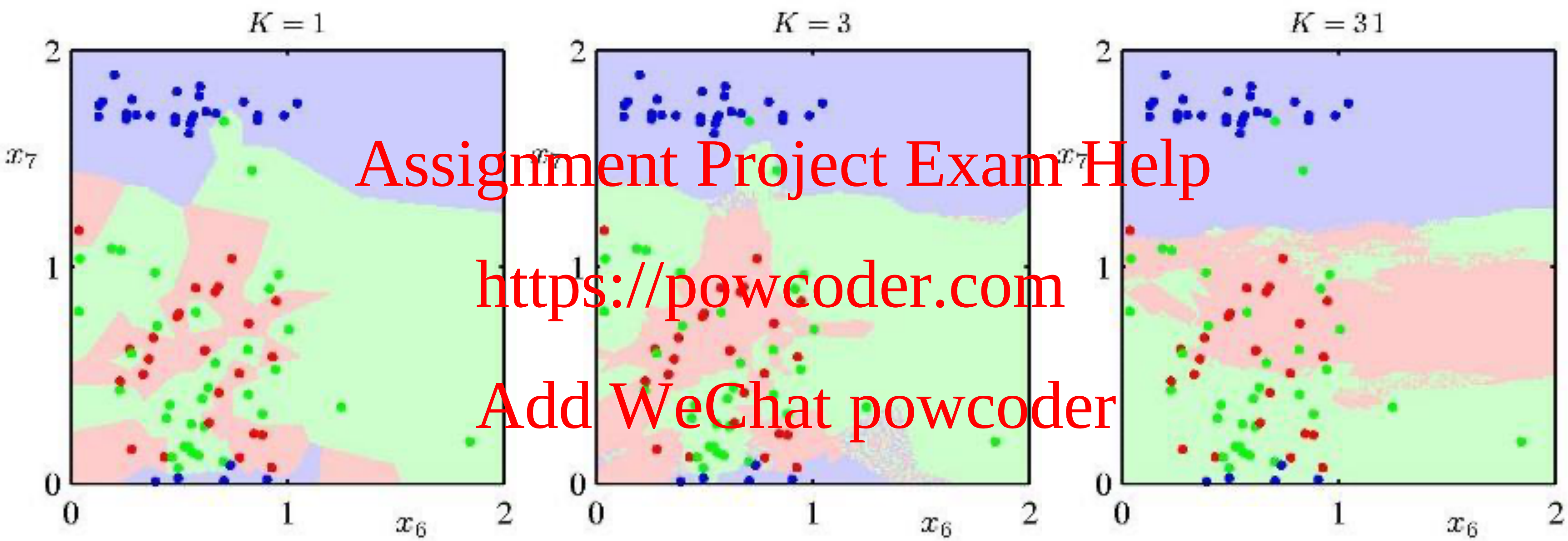
$$P(\mathbf{x}) \cdot V \approx \frac{k}{N} \qquad P(\mathbf{x} \mid C_i) \cdot V \approx \frac{k_i}{N_i}$$

estimated probability (uniform density within $B_k(\mathbf{x})$)

observed probability



Effect of k



- k acts as a smoother

Discussion on the *k*-NN Algorithm

- The k-NN algorithm for continuous-valued target functions
  - Calculate the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - 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

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

  - Similarly, for real-valued target functions
- Robust to noisy data by averaging k-nearest neighbors
- Curse of dimensionality:
  - kNN search becomes very expensive in high dimensional space
    - High-dimensional indexing methods, e.g., **LSH**
  - distance between neighbors could be dominated by irrelevant attributes.
    - To overcome it, axes stretch or elimination of the least relevant attributes.

Remarks on Lazy vs. Eager Learning

- Instance-based learning: lazy evaluation
- Decision-tree and Bayesian classification: eager evaluation
- Key differences

  - Lazy method may consider query instance x_q when deciding how to generalize beyond the training data D

  - Eager method cannot since they have already chosen global approximation when seeing the query

- Efficiency: Lazy - less time training but more time predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space