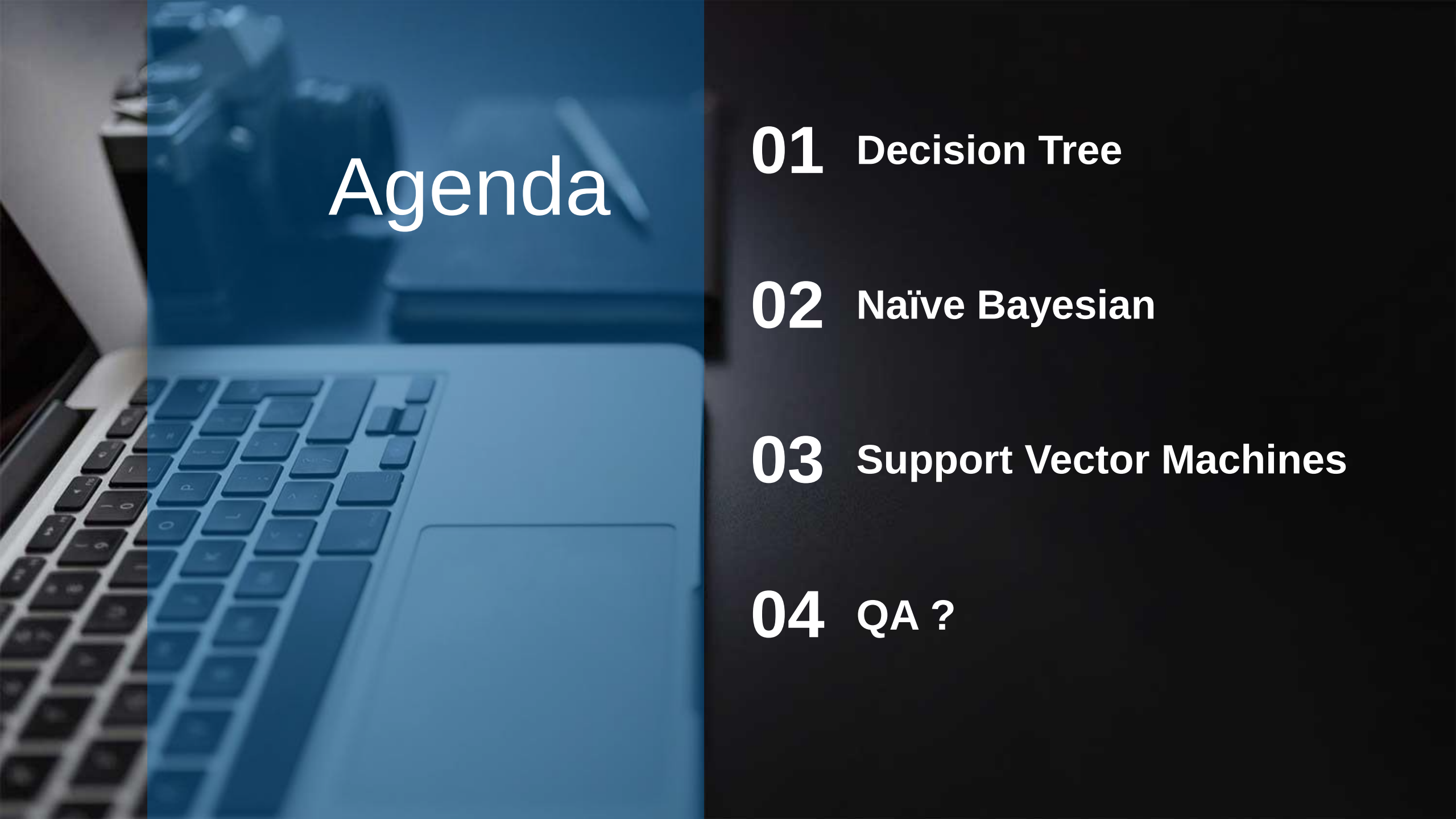
Data Mining

Munawar, PhD

6. Classification



Agenda

01 Decision Tree

02 Naïve Bayesian

03 Support Vector Machines

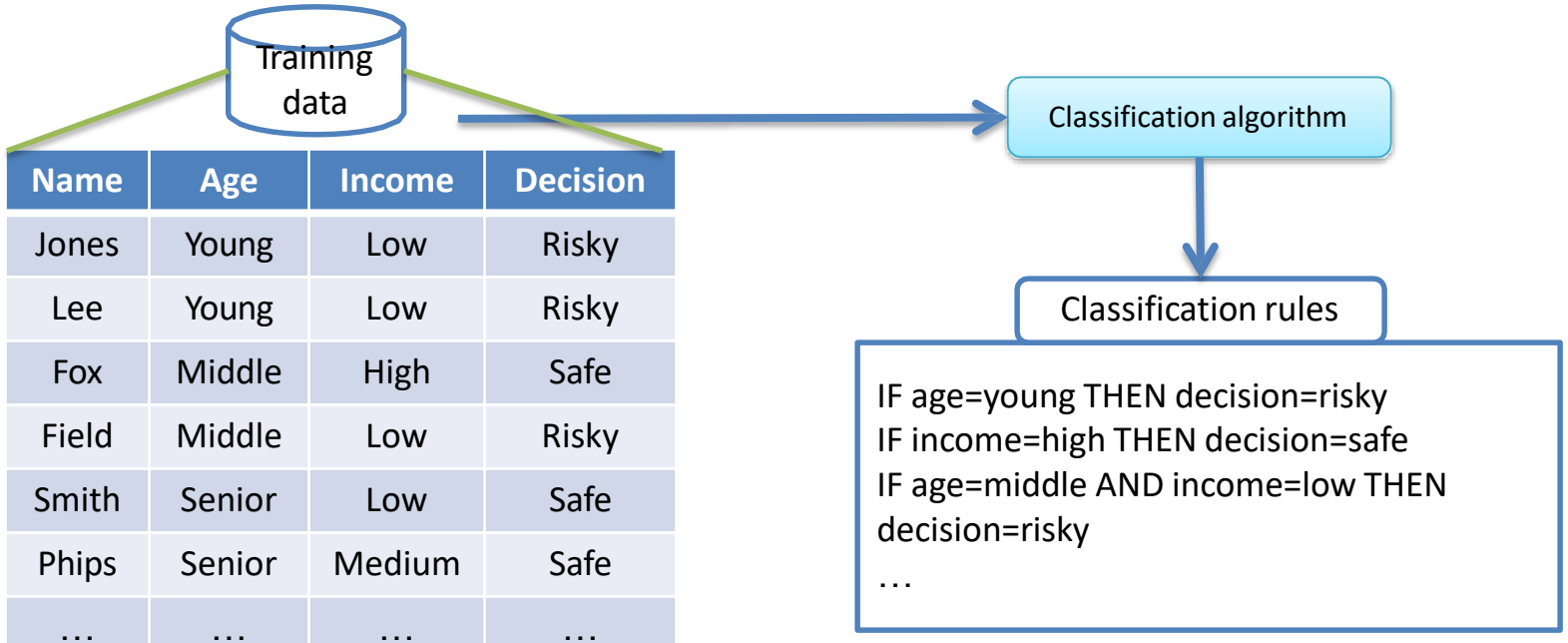
04 QA ?

Classification

- What is **classification**?
  - Given is a collection of records (**training set**)
    - Each record consists of a set of attributes, plus a specific **class attribute**
  - Find a model for the class attribute as a **function** of the values of other attributes
  - **Goal**: new records should be assigned to some class as accurately as possible
    - A test set is used to determine the accuracy of the model
- Usually, the given data set is divided into **training and test sets**, with training set used to **build** the model and test set used to **validate** it

Classification

- Example: credit approval
  - Step 1: **learning** (induction)
    - Training data is analyzed by some classification algorithm and the learned model is coded into **classification rules**

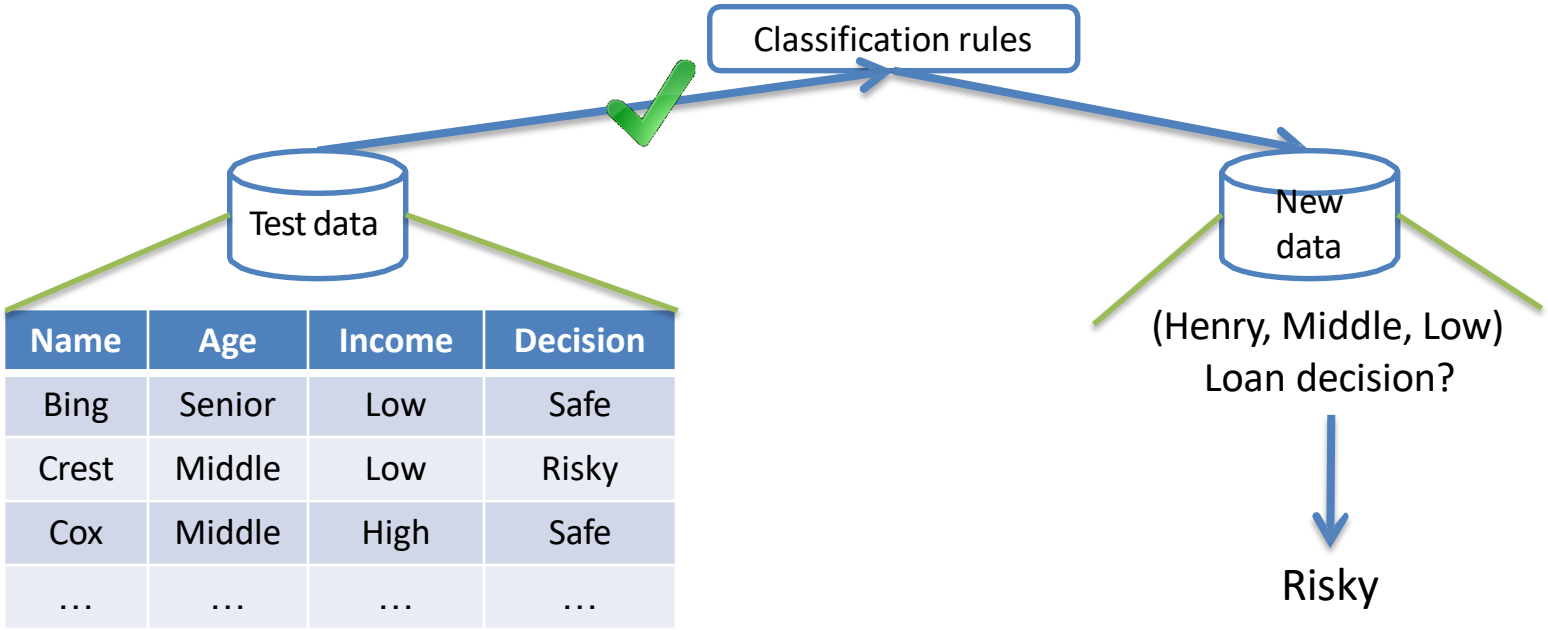


Name	Age	Income	Decision
Jones	Young	Low	Risky
Lee	Young	Low	Risky
Fox	Middle	High	Safe
Field	Middle	Low	Risky
Smith	Senior	Low	Safe
Phips	Senior	Medium	Safe
…	…	…	…

Classification

– Step 2: **classification** (deduction)

- Test data validates the accuracy of the classification rules
- If the accuracy is considered acceptable, then the rules can be applied to the classification of new records



Name	Age	Income	Decision
Bing	Senior	Low	Safe
Crest	Middle	Low	Risky
Cox	Middle	High	Safe
…	…	…	…

Classification

- **Supervised** learning
  - The training data (observations, measurements, etc.) is accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- **Unsupervised** learning
  - 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

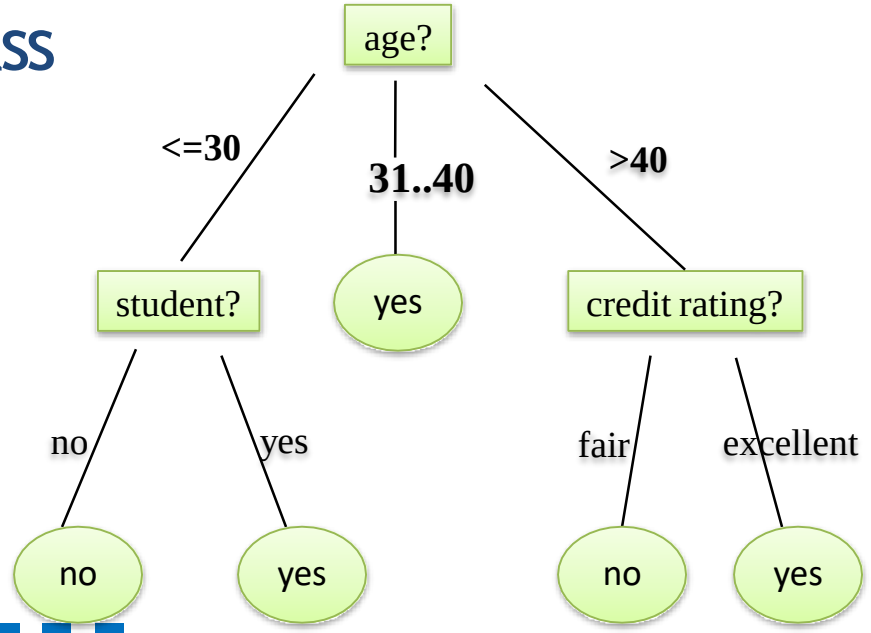
Classification

- Prominent **classification techniques**
  - Decision Tree based Methods
  - Rule-based Methods
  - Naive Bayes and Bayesian Belief Networks
  - Support Vector Machines (SVM)
  - Neural Networks

Decision Tree

Decision Trees

- **Decision tree**
  - A flow-chart-like **tree** structure
  - **Internal node** denotes a test on an attribute
  - **Branch** represents an outcome of the test
  - **Leaf nodes** represent class labels or class distribution
  - E.g., decision making
    - Who buys a computer?

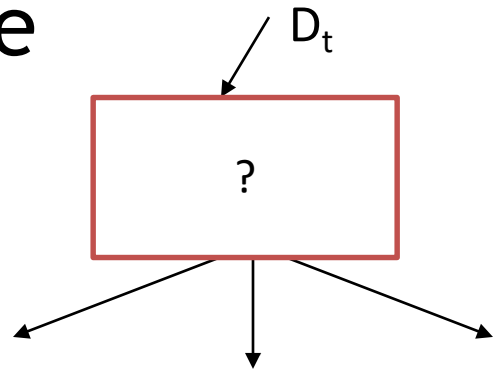


Decision Trees

- Decision tree induction
  - Basis: Hunt's Algorithm
    - One of the earliest methods
  - Different implementations of Hunt's Algorithm
    - ID3 and its successor, C4.5
      - Represents a benchmark to supervised learning algorithms
    - Classification and Regression Trees (CART)

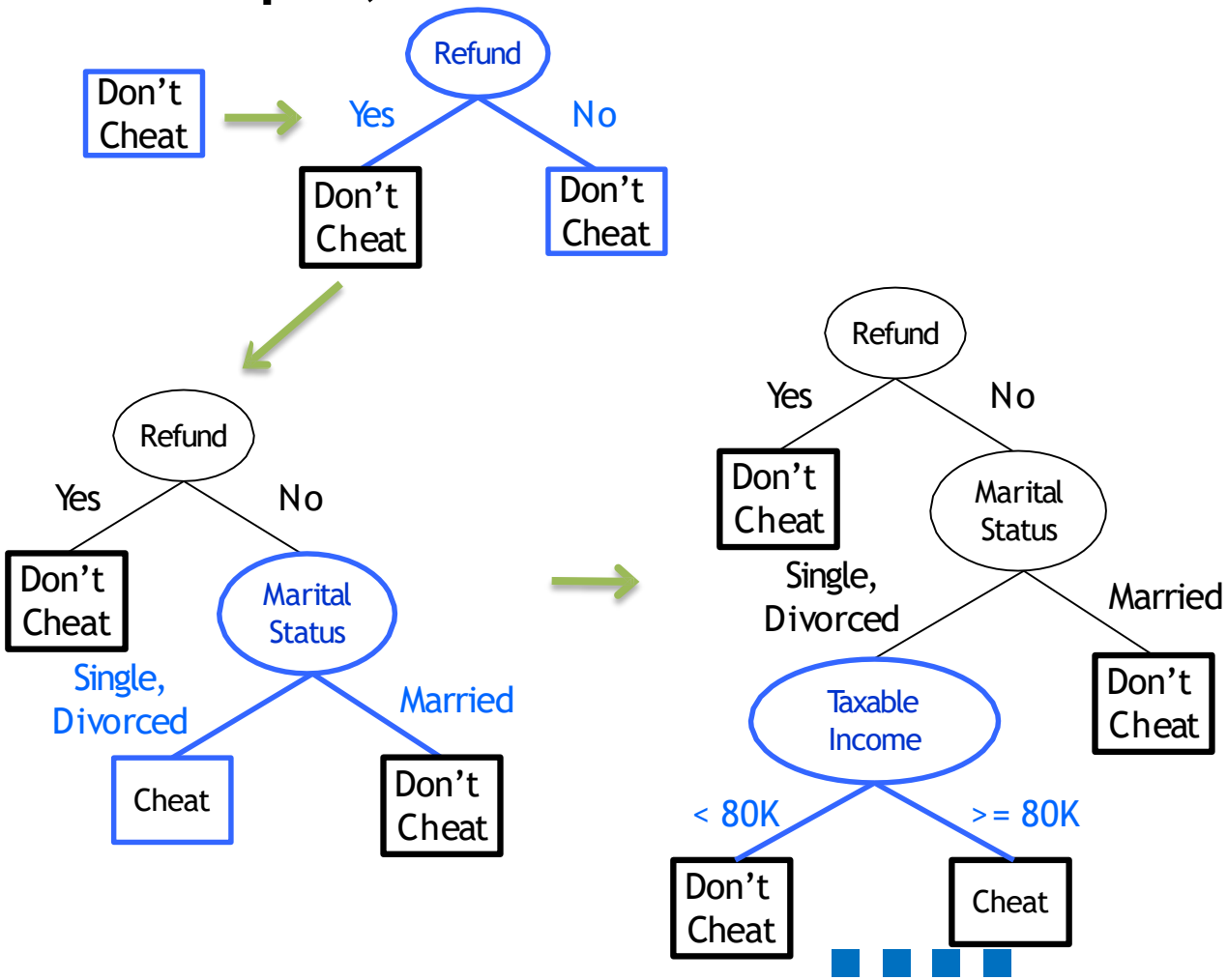
Decision Trees

- **Hunt's algorithm**, general structure
  - Let D_t be the set of training records that reach a node t
  - General Procedure:
    - If D_t contains records that **belong the same class** y_t, then t is a **leaf node** labeled as y_t
    - If D_t contains records that belong to **more than one class**, use an attribute test to split the data into smaller subsets: **recursively** apply the procedure to each subset



Hunts Algorithm

- Example, VAT refund



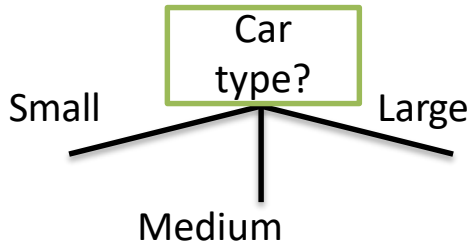
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

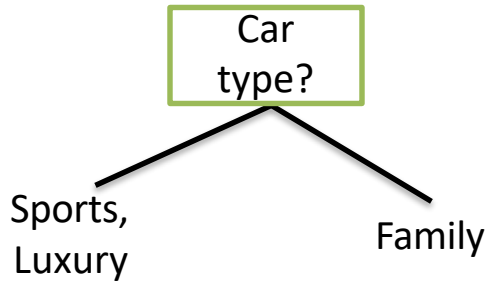
Hunts Algorithm

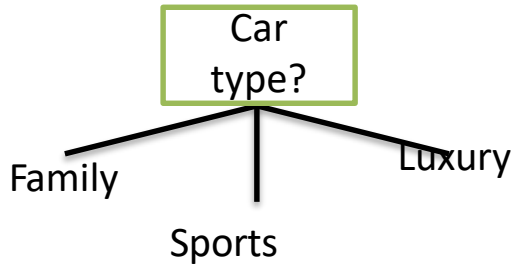
- Greedy strategy
  - **Split** the records based on an **attribute test** that optimizes a certain criterion
- Issues
  - Determine **how to split** the records
    - How to specify the **attribute test condition?**
    - How to **determine the best split?**
  - Determine **when to stop** splitting

Attribute test condition

- **Splitting:** how to specify the attribute test condition?
  - Depends on **attribute types**
    - Nominal e.g., car: sports, luxury, family
    - Ordinal e.g., small, medium large
    - Continuous e.g, age
  - Depends on **number of ways** to split
    - Binary split
    - Multi-way split





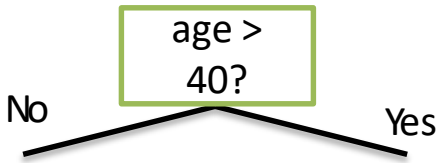


Attribute test condition

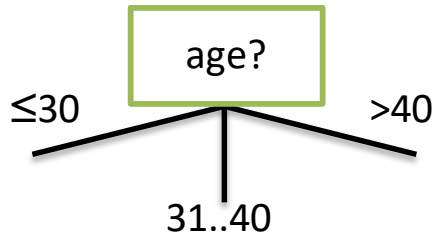
- What about splitting **continuous attributes**?
  - **Discretization** to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), clustering, or supervised clustering
  - Binary decision
    - Consider all possible splits and finds the best cut
    - Can be quite computationally expensive

Attribute test condition

- Example: age
  - Binary split



  - Multi-way split



Determine the best split

- Central question: How to determine the best split?
  - We can split on **any** of the 4 attributes!
  - E.g., income
    - Low – yes:3, no:1
    - Medium – yes:4, no:2
    - High – yes:2, no:2
  - E.g., student
    - No – yes:3, no:4
    - Yes – yes:6, no:1
  - Which split is **better?**

age	income	student	Credit rating	Buys computer
27	high	no	fair	no
28	high	no	excellent	no
31	high	no	fair	yes
45	medium	no	excellent	yes
43	low	yes	excellent	yes
56	low	yes	fair	no
37	low	yes	excellent	yes
20	medium	no	fair	no
20	low	yes	fair	yes
60	medium	yes	excellent	yes
24	medium	yes	excellent	yes
36	medium	no	excellent	yes
31	high	yes	fair	yes
41	medium	no	fair	no

Determine the best split

- What does better mean?
  - Nodes with **homogeneous** class distribution (pure nodes) are preferred
    - E.g., homogeneous nodes
      - Student attribute, Yes - yes:6, no:1
    - E.g., heterogeneous nodes
      - Income attribute, High - yes:2, no:2
- How do we measure node **impurity**?

Determine the best split

- Methods to measure impurity
  - **Information gain** (e.g. C4.5)
    - All attributes are assumed to be categorical
    - Can be modified for continuous-valued attributes
    - Also called Kullback-Leibler divergence
  - Other possibilities (e.g. Gini index)

Decision Tree Induction

- **Information gain**
  - Method
    - Assume there are two classes, P and N
      - Let the set of examples S contain p elements of class P and n elements of class N
      - The amount of information needed to decide whether an arbitrary example in S belongs to P or N is defined as

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

    - Select the attribute with the **highest** information gain

estimates the probability that label is p

estimates the probability that label is n

Information Gain

- Information gain in decision tree induction
  - Assume that using attribute A a set S will be partitioned into sets $\{S_1, S_2, \ldots, S_v\}$
  - If S_i contains p_i examples of P and n_i examples of N, the **entropy**, or the expected information needed to classify objects in all subtrees S_i is

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

  - The encoding information that would be gained by branching on A is

$$Gain(A) = I(p, n) - E(A)$$

Information Gain

- Attribute selection by gain computation, example:
  - Class P: buys_computer = "yes"
  - Class N: buys_computer = "no"

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

  - I(p, n) = I(9, 5) =0.94

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	excellent	yes
>40	low	yes	excellent	yes
>40	low	yes	fair	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	excellent	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	fair	no

Information Gain

- Compute the entropy for the following 3 age partitions age <=30, 30<age<=40 and 40<age

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0,971
31…40	4	0	0
>40	3	2	0,971

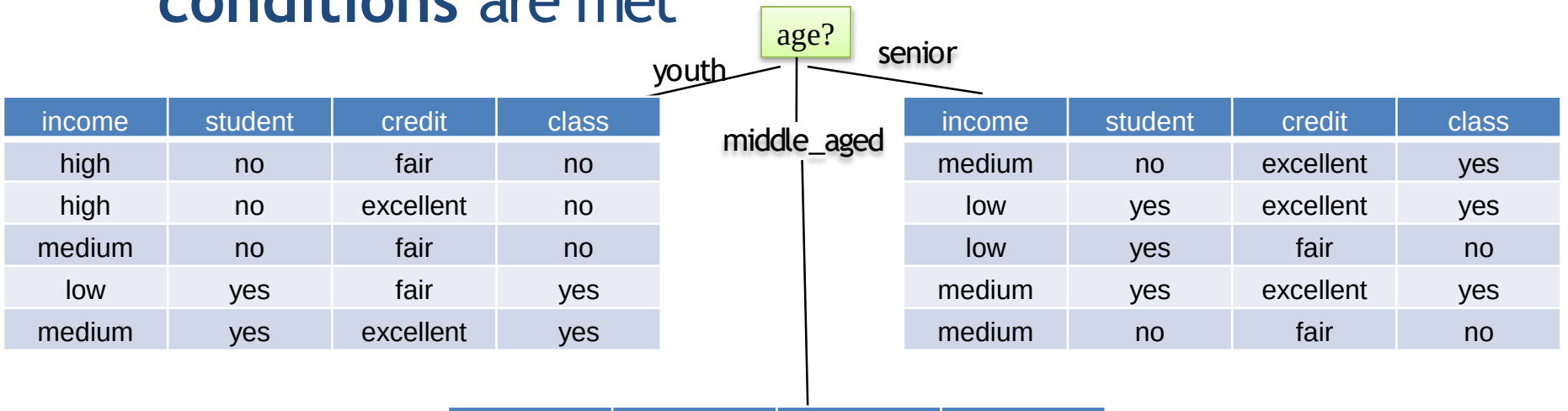
$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

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

  - Gain(age) = I(p, n) − E(age) $\Longrightarrow$ 0.94 − 0.694 = 0.246
- Analogously we can calculate also:
  - Gain(income) = 0.029
  - Gain(student) = 0.151
  - Gain(credit_rating) = 0.048

Information Gain

- Since age promises the highest information gain, it becomes the splitting node
- Continue recursively to grow the tree until **stop conditions** are met



income	student	credit	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

income	student	credit	class
medium	no	excellent	yes
low	yes	excellent	yes
low	yes	fair	no
medium	yes	excellent	yes
medium	no	fair	no

Decision Tree Induction

- Stop conditions
  - All the records **belong to the same class**
    - E.g.,

income	credit	class
high	fair	no
high	excellent	no
medium	fair	no

    - In this case a leaf node is created with the corresponding class label (here "no")

Decision Tree Induction

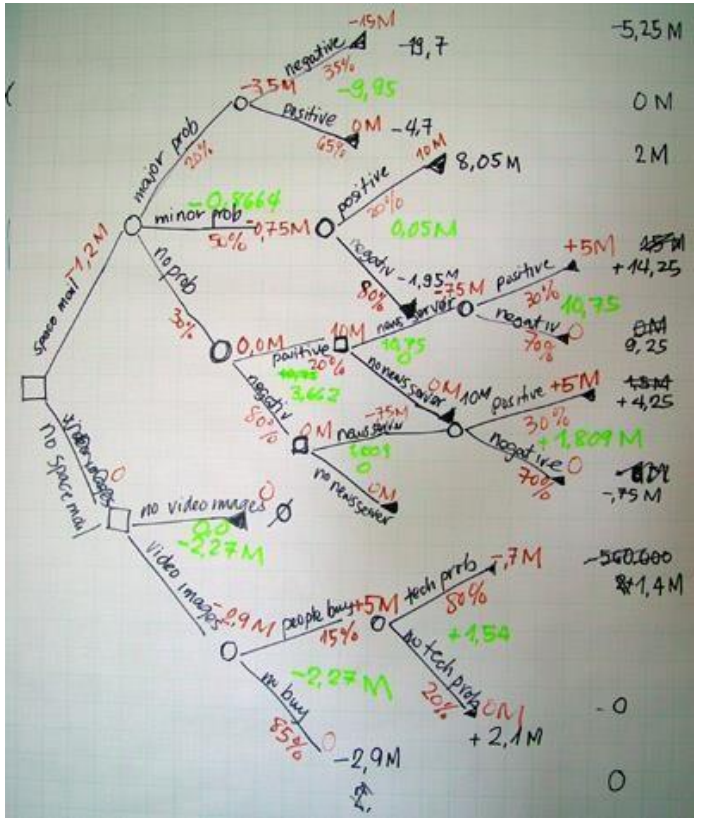
- Stop conditions
  - All the records **have similar attribute values**
    - E.g., perform split by student but all records are students

student	credit	class
yes	fair	no
yes	fair	no
yes	fair	no
yes	fair	yes
yes	excellent	yes

    - In this case instead of performing the split, a leaf node is created with the **majority** class as label (here "no")

Decision Trees

- Decision tree **deduction**
  - Use the decision tree rules to classify new data
  - Exemplified together with induction in the **detour section**



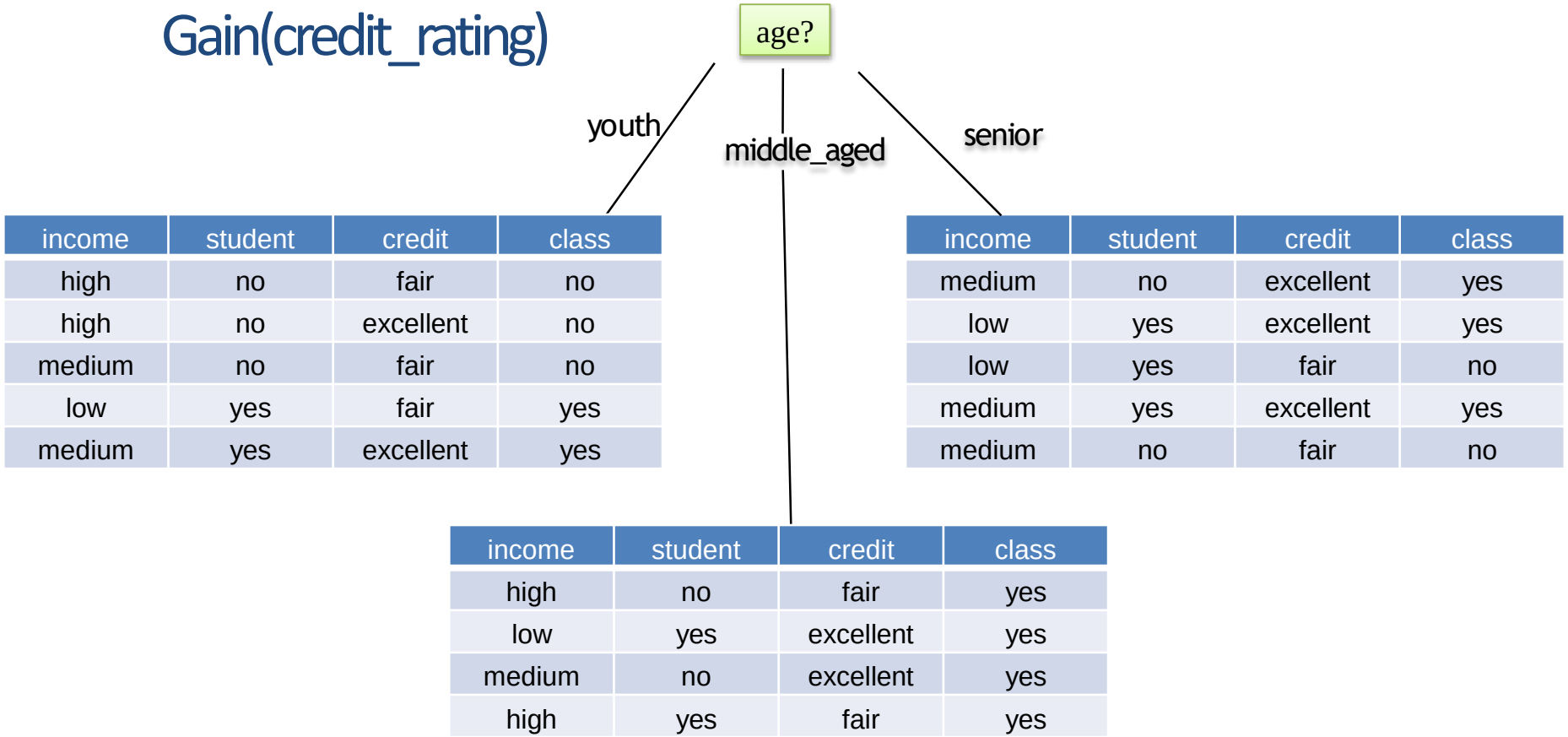
Decision Trees

- Classification based on decision tree example
  - Step 1 - Induction
    - Generate the decision tree
    - Input: **training data set**, attribute list used for classification, attribute selection method
    - Output: the decision tree
  - Step 2 - Deduction
    - Predict the classes of the **new data**
    - Input: the decision tree from step 1 and the new data
    - Output: classified new data

Decision Trees

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

- First node was already calculated
  - Gain(age) = 0.246 > Gain(income), Gain(student), Gain(credit_rating)

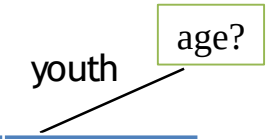


income	student	credit	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit	class
medium	no	excellent	yes
low	yes	excellent	yes
low	yes	fair	no
medium	yes	excellent	yes
medium	no	fair	no

income	student	credit	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Decision Trees



income	student	credit	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

- Subnode age = youth
  - For the income attribute
    - I(2, 3) =0.97
    - E(income) = 1/5*I(1,0)+ 2/5 * I(1, 1) +2/5 * I(0, 2) =0.4
    - Thus Gain(youth,income) =0.97 – 0.4=0.57
  - For the student attribute
    - I is the same
    - E(student) =2/5 * I(2,0) +3/5 * I(0,3) =0
    - Thus Gain(youth, student) =0.97

Class P: buys_computer = “yes”
Class N: buys_computer = “no”

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

$$E(A) = \sum_{i=1}^{v}\frac{p_i + n_i}{p+n} I(p_i, n_i)$$

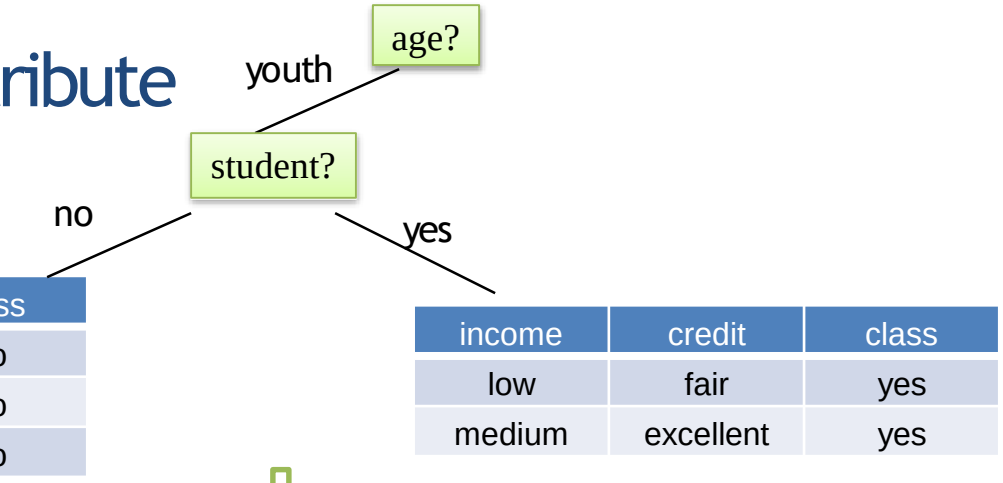
Decision Trees

income	student	credit	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

- Subnode age = youth
  - For the credit attribute
    - I(2, 3) =0.97
    - E(credit) = 3/5*I(1,2)+ 2/5 * I(1, 1) =0.95
    - Thus Gain(youth,credit) =0.97 – 0.95 = 0.02
  - Largest gain was of 0.97 for the **student** attribute

Decision Trees

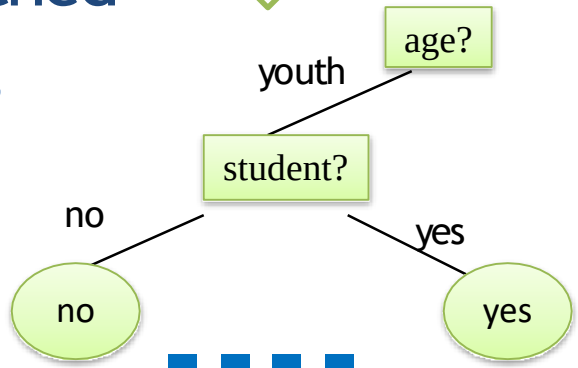
- Subnode age = youth
  - Split by student attribute



income	credit	class
high	fair	no
high	excellent	no
medium	fair	no

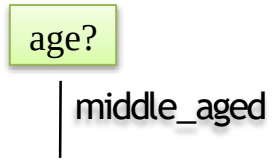
income	credit	class
low	fair	yes
medium	excellent	yes

  - Stop condition reached
    - Resulting subtree is

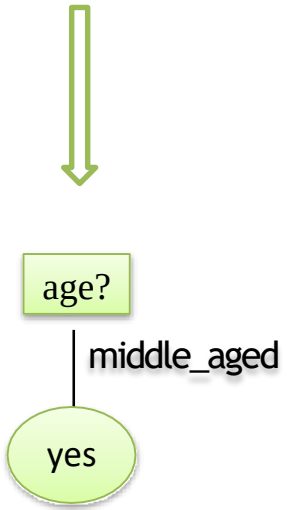


Decision Trees

- Subnode age = middle_aged
  - **Stop condition reached**
    - We have just one class

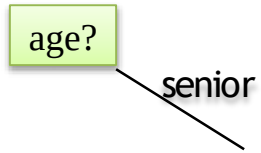


income	student	credit	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes



Decision Trees

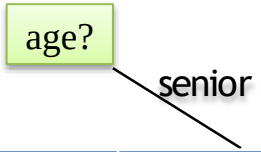
- Subnode age = senior
  - For the income attribute
    - I(3, 2) =0.97
    - E(income) = 2/5*I(1,1)+ 3/5 * I(2,1) =0.95
    - Thus Gain(senior,income) =0.97 – 0.95=0.02
  - For the student attribute
    - I is the same
    - E(student) =3/5 * I(2,1) +2/5 * I(1,1) =0.95
    - Thus Gain(senior, student) =0.02



income	student	credit	class
medium	no	excellent	yes
low	yes	excellent	yes
low	yes	fair	no
medium	yes	excellent	yes
medium	no	fair	no

Decision Trees

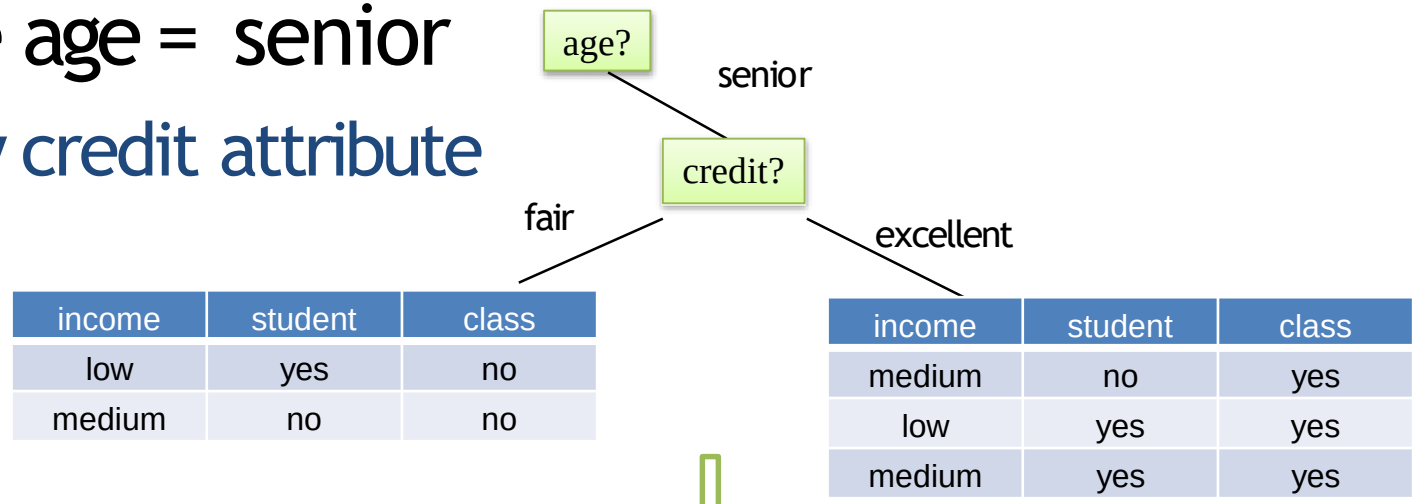
- Subnode age = senior
  - For the credit attribute
    - I is the same
    - E(income) = 2/5*I(0,2)+ 3/5 * I(3,0) =0
      - Thus Gain(senior,credit) =0.97
  - Thus **split by credit** attribute



income	student	credit	class
medium	no	excellent	yes
low	yes	excellent	yes
low	yes	fair	no
medium	yes	excellent	yes
medium	no	fair	no

Decision Trees

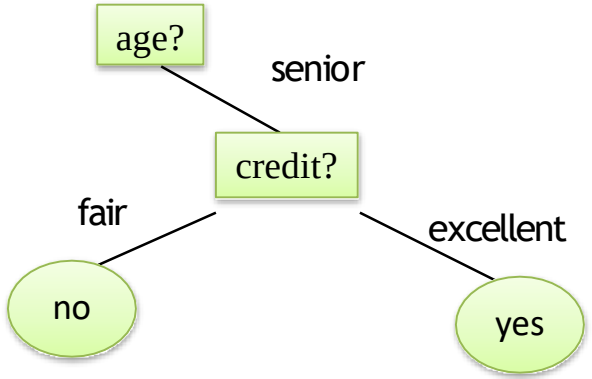
- Subnode age = senior
  - Split by credit attribute



income	student	class
low	yes	no
medium	no	no

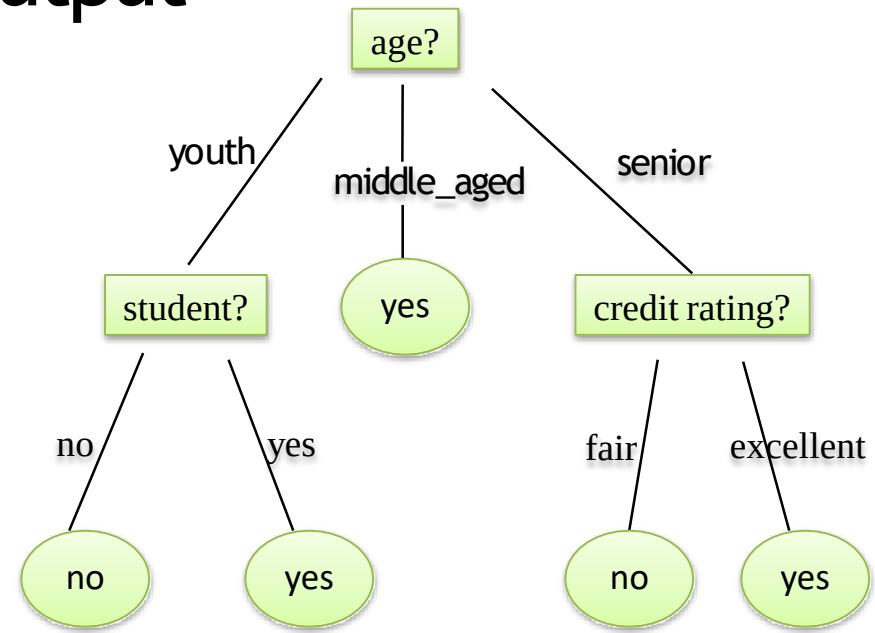
income	student	class
medium	no	yes
low	yes	yes
medium	yes	yes

  - Stop condition reached



Decision Trees

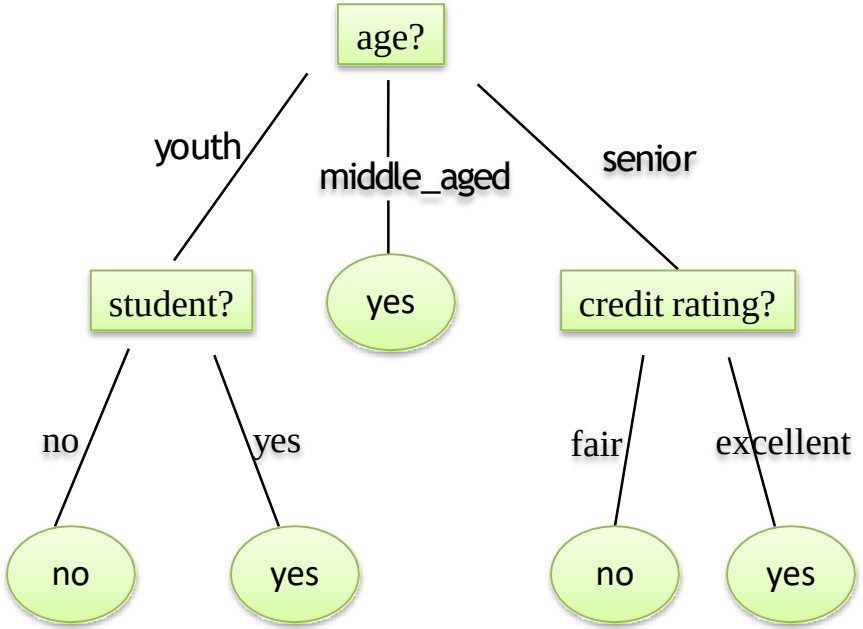
- Step 1 has finished with the following decision tree as output



Decision Trees

- Step 2
  - New data

age	income	student	Credit rating		Buys computer
35 (31…40)	low	yes	fair	→	yes
29 (<=30)	low	yes	fair	→	yes
25 (<=30)	low	yes	excellent	→	yes
55 (>40)	low	no	fair	→	no



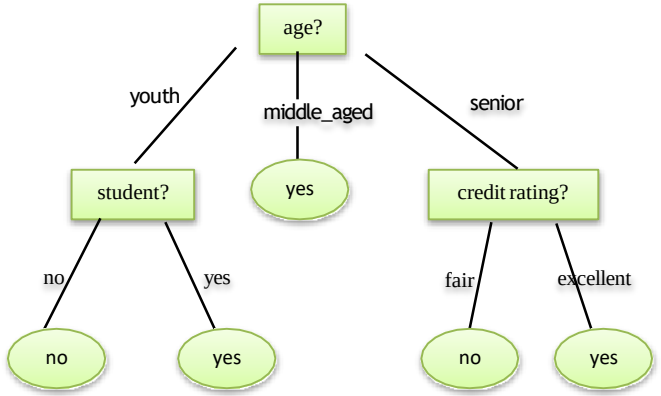
Classification Rules

- Extracting classification rules from trees
  - Represent the knowledge in the form of **IF-THEN rules**
  - **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
  - Rules are easier for humans to understand

Extracting Rules from Trees

– Example

- IF age = "≤ 30" AND student = "no" THEN buys_computer = "no"
- IF age = "≤ 30" AND student = "yes" THEN buys_computer = "yes"
- IF age = "31…40" THEN buys_computer = "yes"
- IF age = ">40" AND credit_rating = "excellent" THEN buys_computer = "yes"
- IF age = ">40" AND credit_rating = "fair" THEN buys_computer = "no"



Summary: Decision Trees

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets
  - Very good average performance over many datasets

Over-Fitting

- Avoid over-fitting in classification
  - The generated tree may over-fit the training data
    - Too many branches, some may reflect anomalies due to noise or outliers
    - Result is in poor accuracy for unseen samples
  - Two approaches to avoid **over-fitting**
    - Prepruning
    - Postpruning

Avoiding Over-Fitting

- Prepruning
  - Halt tree construction early
    - Do not split a node if this would result in the information gain 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"

Enhancements

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- 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

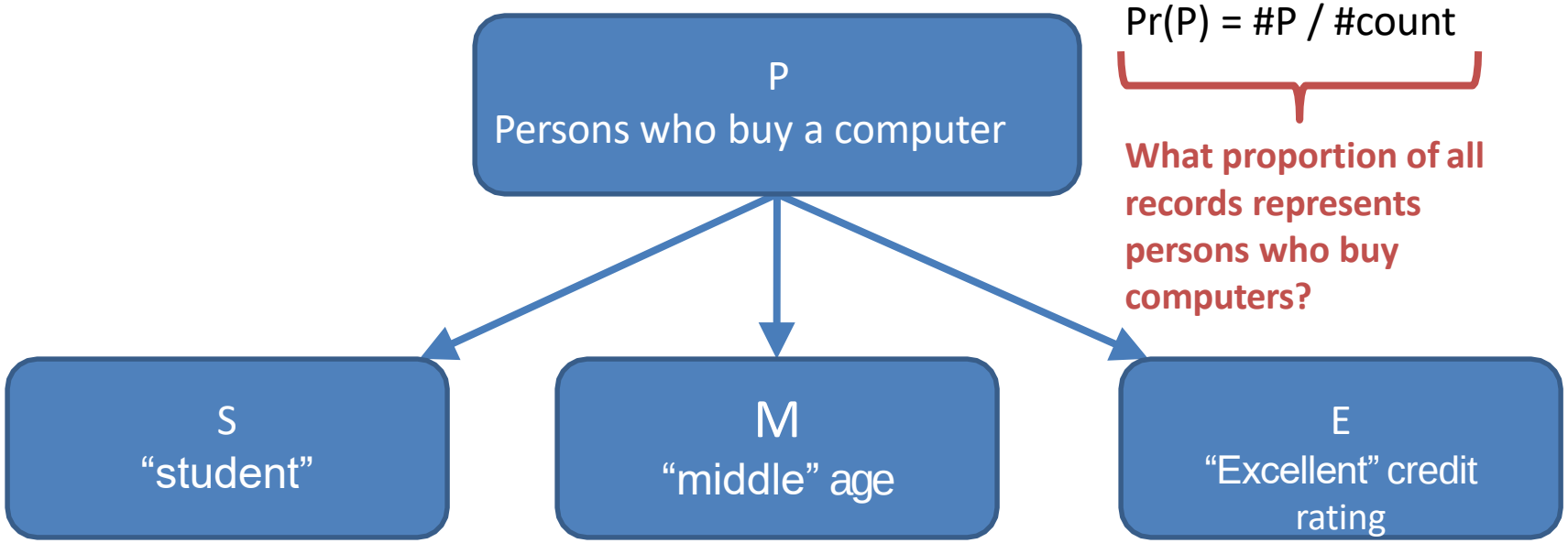
Naïve Bayesian

Naive Bayesian Classification

- Bayesian Classification
  - 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

Naive Bayesian Classification

- A simple **Bayesian network:**



Pr(P) = #P / #count

What proportion of all records represents persons who buy computers?

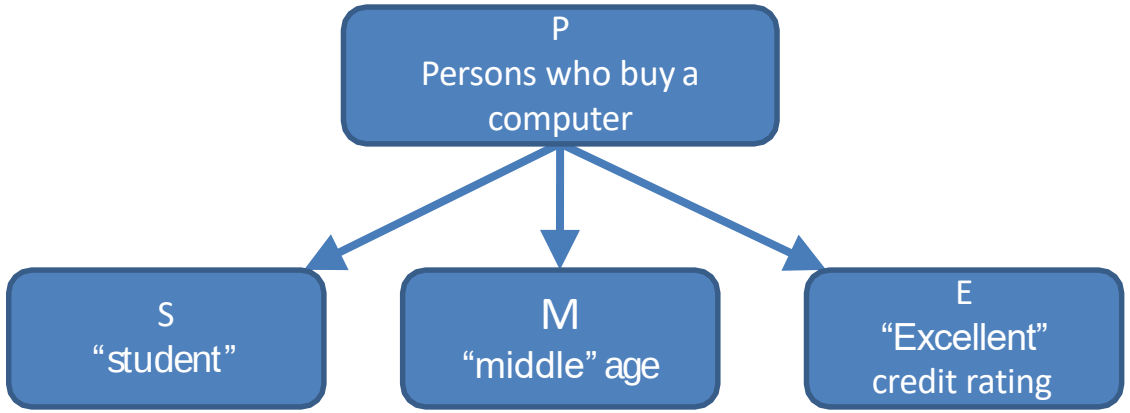
Pr(S) = #S / #count Pr(S|P) = #(S and P) / #P
Pr(S|¬P) = #(S and ¬P) / #(¬P)

Pr(M) = …
Pr(M|P) = …
Pr(M|¬P) = …

Pr(E) = …
Pr(E|P) = …
Pr(E|¬P) = …

All these probabilities can be estimated from the training set (possibly using smoothing)!

Naive Bayesian Classification



- For new records to be classified:
  - We know whether each of the events S, M, and E occurred
  - **We want to find out whether event P is true**
- This can be done using **Bayes' Theorem:**

$$\Pr(A|B) = \frac{\Pr(A)}{\Pr(B)} \cdot \Pr(B|A)$$

Naive Bayesian Classification

- Assume that the test set to be classified represents a young student with fair credit rating
  - Consequently, **we want to find Pr(P |S, ¬M, ¬E)**
- **Bayes Theorem** yields:

$$\Pr(P \mid S, \neg M, \neg E) = \frac{\Pr(P)}{\Pr(S, \neg M, \neg E)} \cdot \Pr(S, \neg M, \neg E \mid P)$$

Naive Bayesian Classification

$$\Pr(P \mid S, \neg M, \neg E) = \frac{\Pr(P)}{\Pr(S, \neg M, \neg E)} \cdot \Pr(S, \neg M, \neg E \mid P)$$

- In naive Bayes (sometimes called **idiot Bayes**), **statistical independence** is assumed:

$$\Pr(P \mid S, \neg M, \neg E) = \frac{\Pr(P)}{\Pr(S) \cdot \Pr(\neg M) \cdot \Pr(\neg E)} \cdot \Pr(S \mid P) \cdot \Pr(\neg M \mid P) \cdot \Pr(\neg E \mid P)$$

- **How to classify a new record d?**
  - Estimate Pr(c |d), for any class c ∈C
  - Assign d to the class having the **highest probability**

Naive Bayes

- Example:
  - Positive (p)
    - Buys_computer = yes
  - Negative (n)
    - Buys_computer = no
  - P(p) = 9/14
  - P(n) = 5/14
  - Calculate the probabilities for each attribute
    - E.g.:

Age attribute	
P(youth|p) = 2/9	P(youth|n) = 3/5
P(middle|p) = 4/9	P(middle|n) = 0/5
P(senior|p) = 3/9	P(senior|n) = 2/5

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

Naive Bayes

– Continue with the other attributes:

Income attribute	
P(low|p) = 3/9	P(low|n) = 1/5
P(medium|p) = 4/9	P(medium|n) = 2/5
P(high|p) = 2/9	P(high|n) = 2/5

Student attribute	
P(yes|p) = 6/9	P(yes|n) = 1/5
P(no|p) = 3/9	P(no|n) = 4/5

Credit attribute	
P(fair|p) = 6/9	P(fair|n) = 2/5
P(excellent|p) = 3/9	P(excellent|n) = 3/5

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

Naïve Bayes

- Classify an unseen set
- X=<Age: youth, Income: low, Student: yes, Credit: fair>
  - Compare P(p|X) and P(n|X)

[P(X|p)*P(p)]/P(X) = P(youth|p) * P(low|p)*
*P(yes|p)*P(fair|p) *P(p) =
= 2/9 * 3/9 * 6/9 * 6/9 * 9/14
= 0.0211

[P(X|n)*P(n))]/P(X) = P(youth|n) * P(low|n)*
*P(yes|n)*P(fair|n) *P(n) =
= 3/5 * 1/5 * 1/5 * 2/5 * 5/14
= 0.0034

- **Since P(X|p)*P(p) > P(X|n)*P(n) X can be classified as buys a computer**

P(p) = 9/14 P(n) = 5/14

Age attribute	
P(youth|p) = 2/9	P(youth|n) = 3/5
P(middle|p) = 4/9	P(middle|n) = 0/5
P(senior|p) = 3/9	P(senior|n) = 2/5

Income attribute	
P(low|p) = 3/9	P(low|n) = 1/5
P(medium|p) = 4/9	P(medium|n) = 2/5
P(high|p) = 2/9	P(high|n) = 2/5

Student attribute	
P(yes|p) = 6/9	P(yes|n) = 1/5
P(no|p) = 3/9	P(no|n) = 4/5

Credit attribute	
P(fair|p) = 6/9	P(fair|n) = 2/5
P(excellent|p) = 3/9	P(excellent|n) = 3/5

Naive Bayesian Classification

- Summary
  - Robust to isolated noise points
  - Handle missing values by ignoring the instance during probability estimate calculations
  - Robust to irrelevant attributes
  - **Independence assumption may not hold for some attributes**

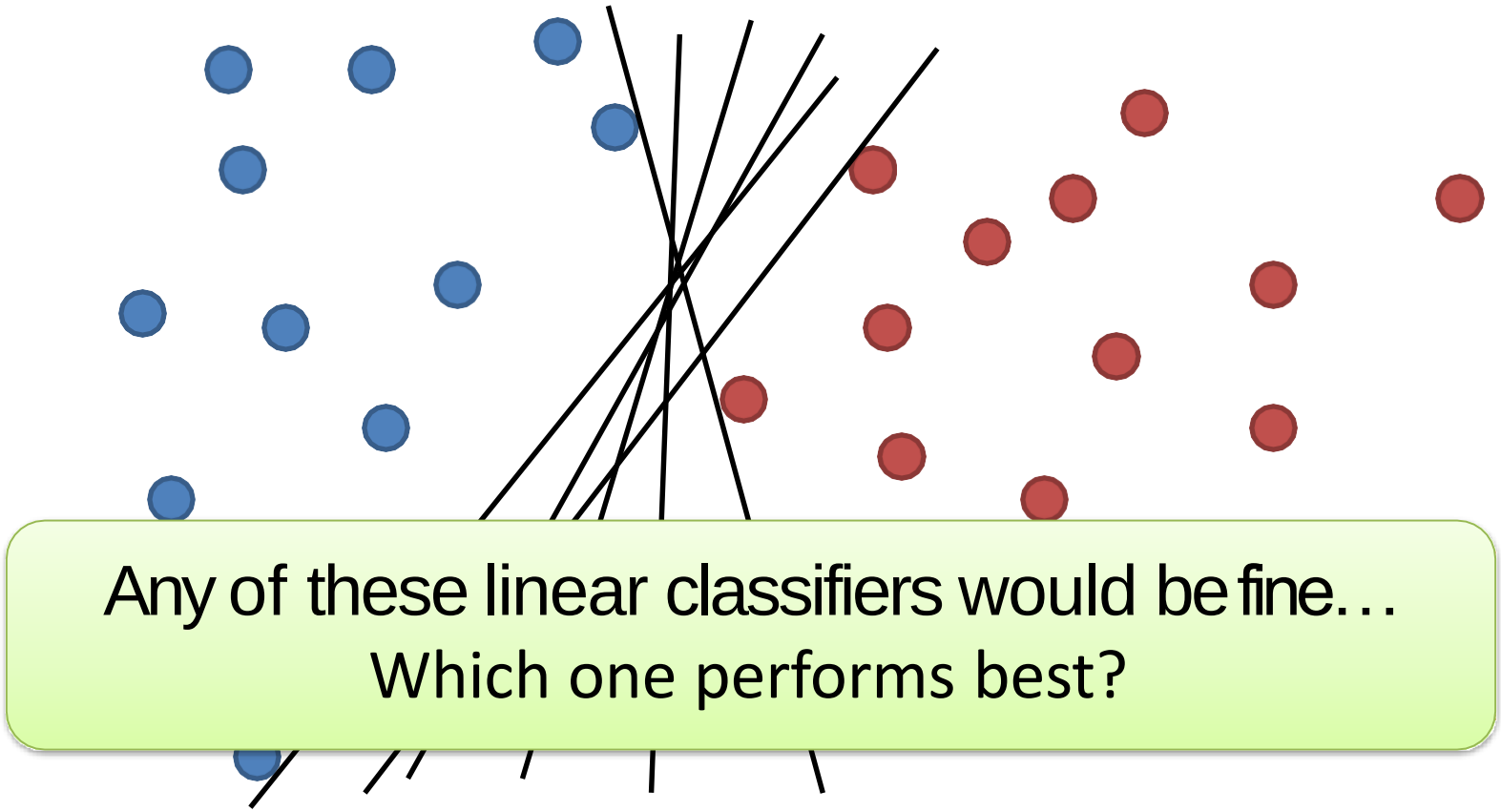
Support Vector Machines (SVM)

Support Vector Machines

- Main tool for classification today
  - **Assumptions:**
    - **Binary classification:**
      Let's assume there are only **two classes**
    - **Vector representation:**
      Any item to be classified can be represented as a ***d*-dimensional real vector**
  - **Task:**
    - Find a **linear classifier** (i.e. a hyperplane) that divides the $\mathbb{R}^d$ into two parts

Support Vector Machines

- Example: A two-dimensional example training set
  - **Task: Separate it by a line!**

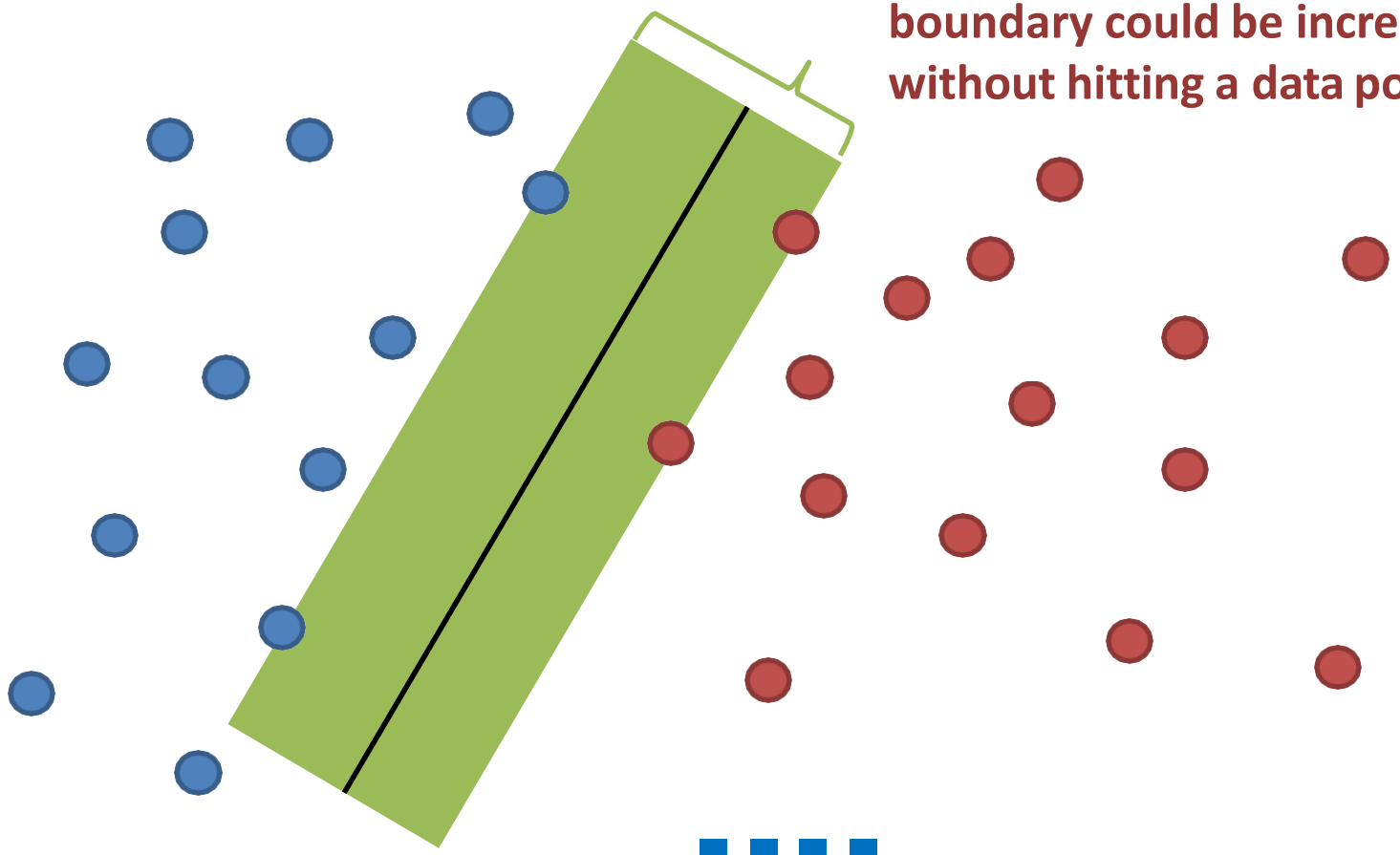


Any of these linear classifiers would be fine…
Which one performs best?

SVM Margin

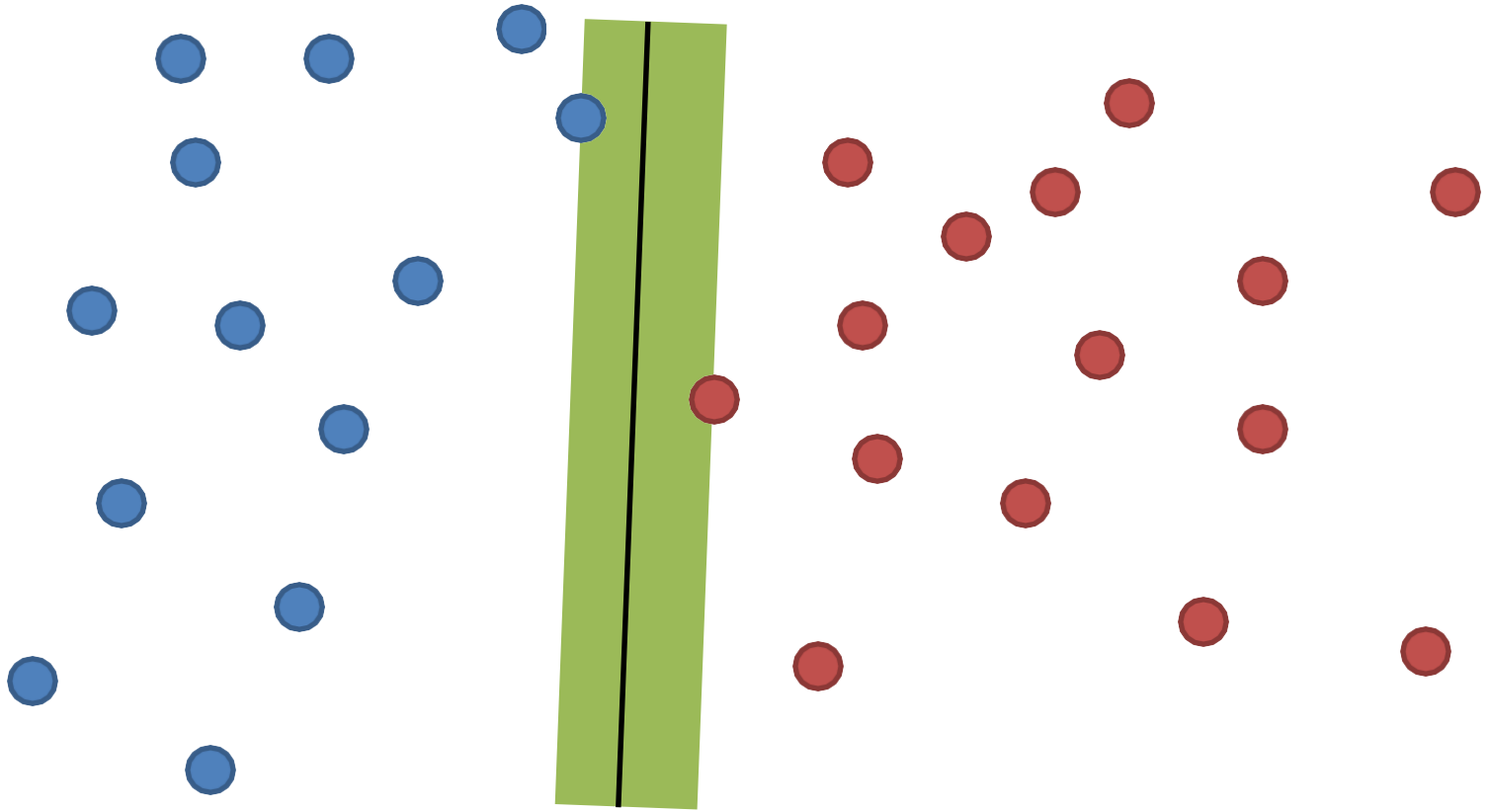
- Which line is better? Idea: measure the **quality** of a linear classifier by its **margin!**

Margin = The width that the boundary could be increased without hitting a data point



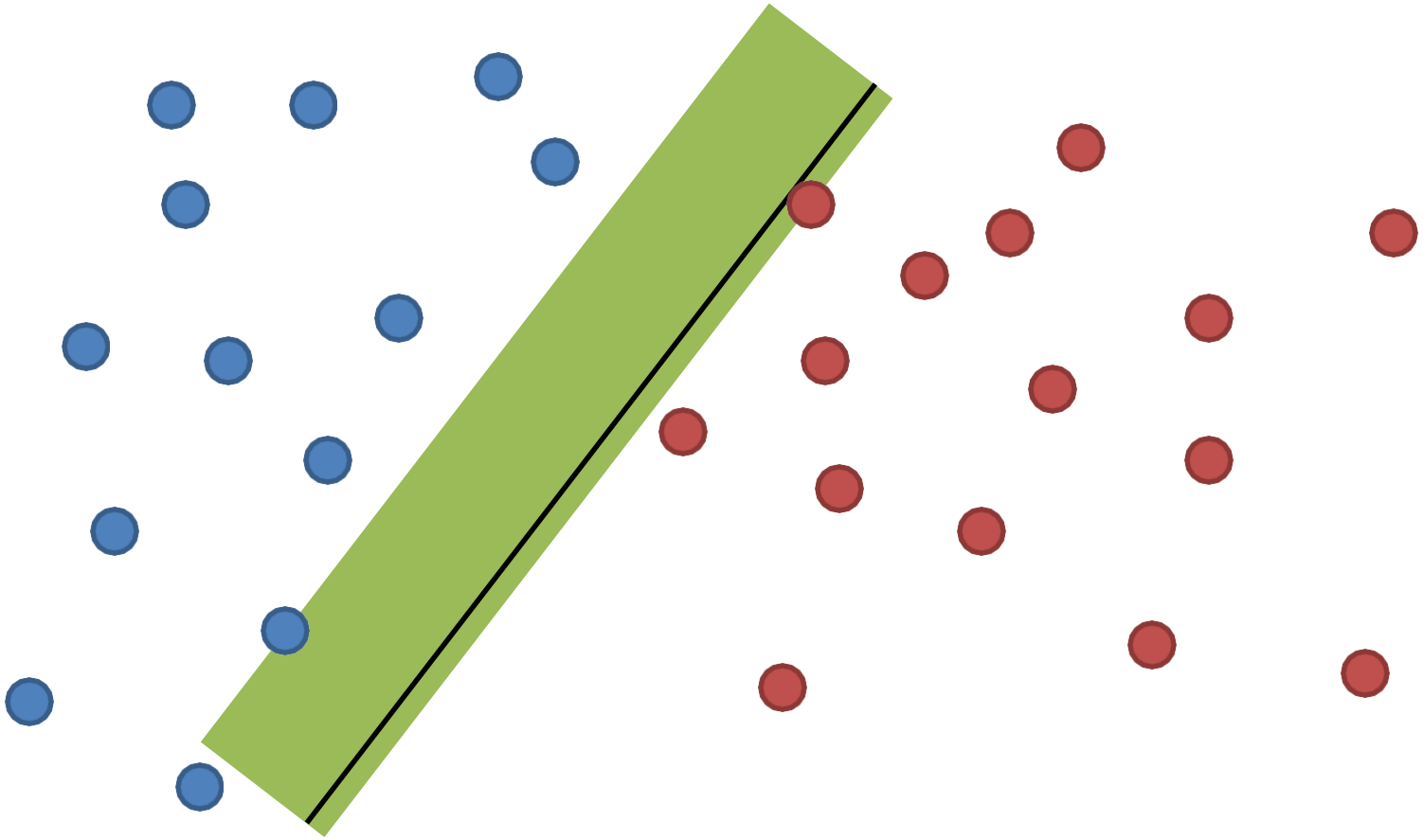
SVM Margin

- Margins (2)



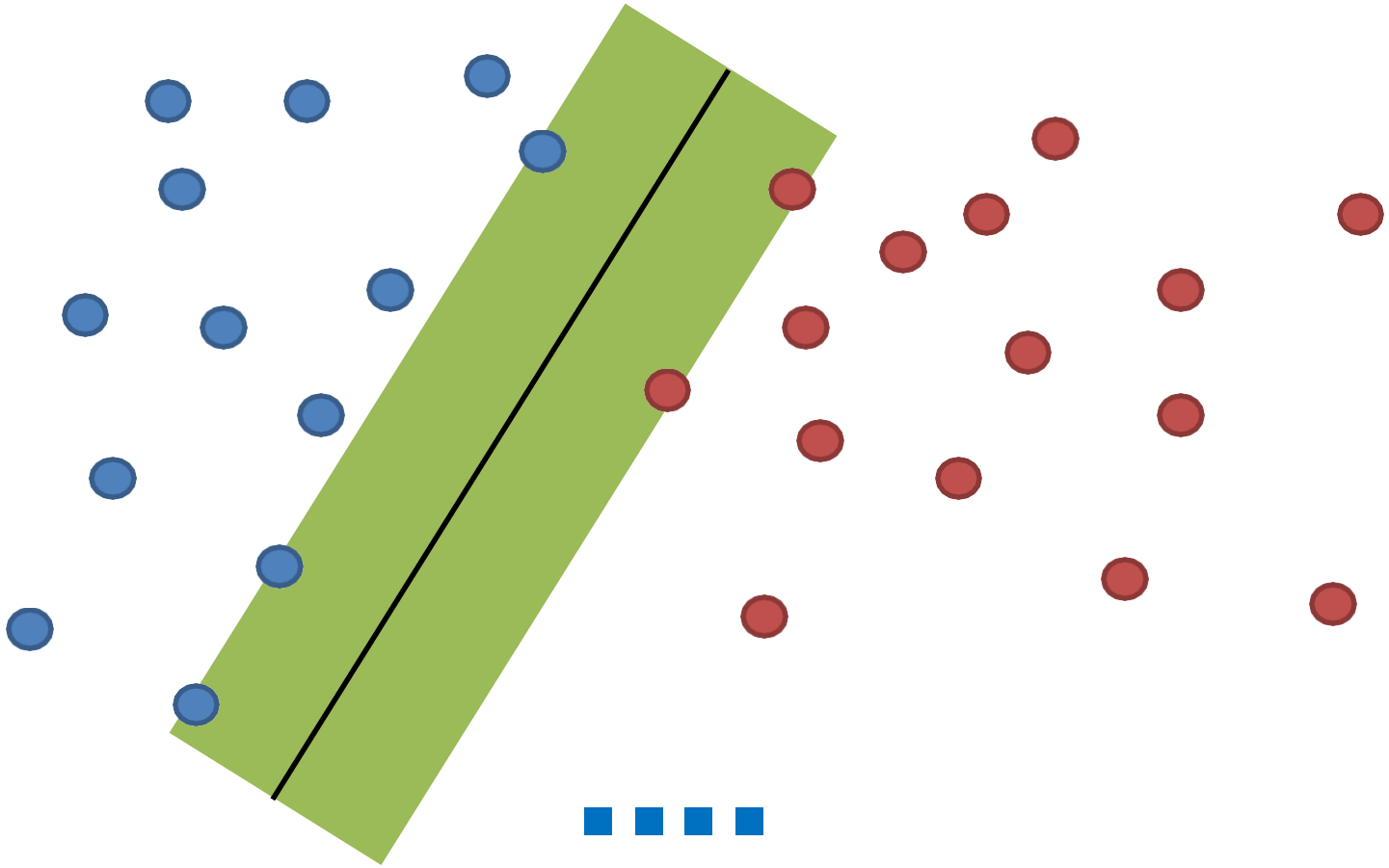
SVM Margin

- Margins (3)



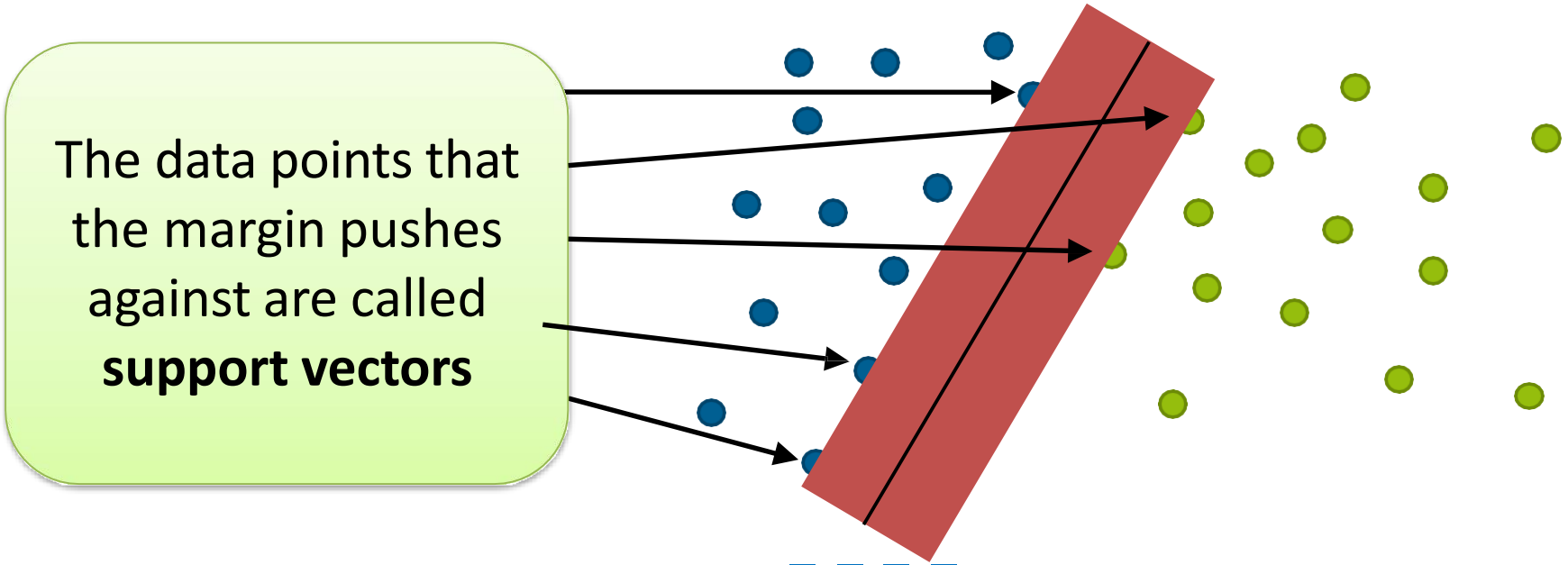
Support Vector Machines

- A **maximum margin classifier** is the linear classifier with a maximum margin



Maximum Margin Classifier

- The **maximum margin classifier** is the simplest kind of support vector machine, called a **linear SVM**
  - **Let's assume for now that there always is such a classifier, i.e. the trainingset is linearly separable!**

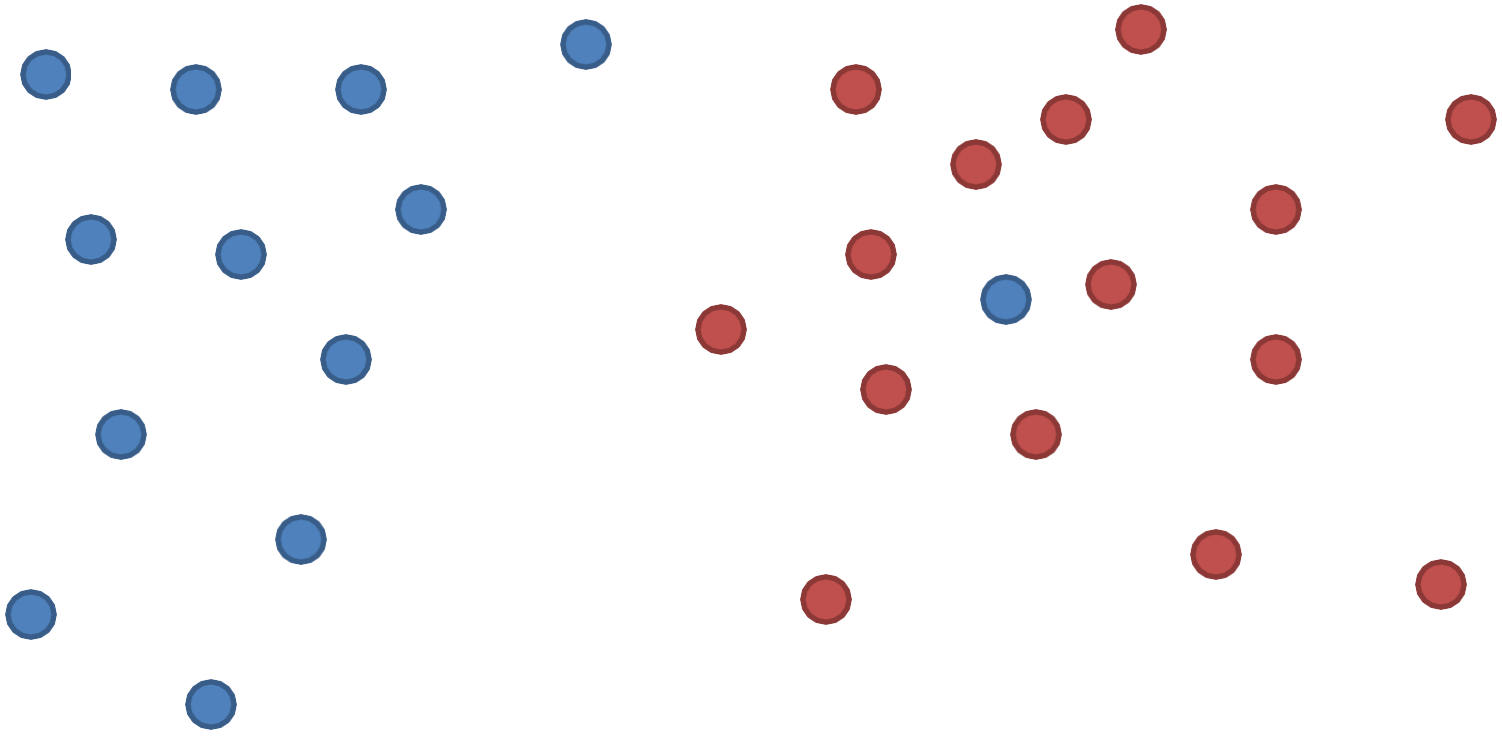


Maximum Margin Classifier

- Why **maximum margin?**
  - It's **intuitive** to divide the two classes by a large margin
  - The largest margin **guards best against small errors** in choosing the "right" separator
  - This approach is **robust** since usually only a small fraction of all data points are support vectors
  - There are some **theoretical arguments** why this is a good thing
  - Empirically, **it works very well**

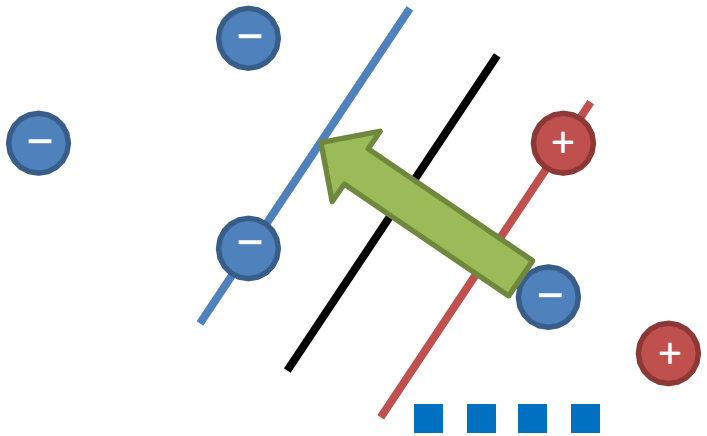
SVM

- At the beginning we assumed that our training data set is **linearly separable…**
- What if it looks like this?



Soft Margin

- So-called **soft margins** can be used to handle such cases
- We allow the classifier to make some mistakes on the training data
- Each **misclassification** gets assigned an **error**, the **total classification error** then is to be minimized

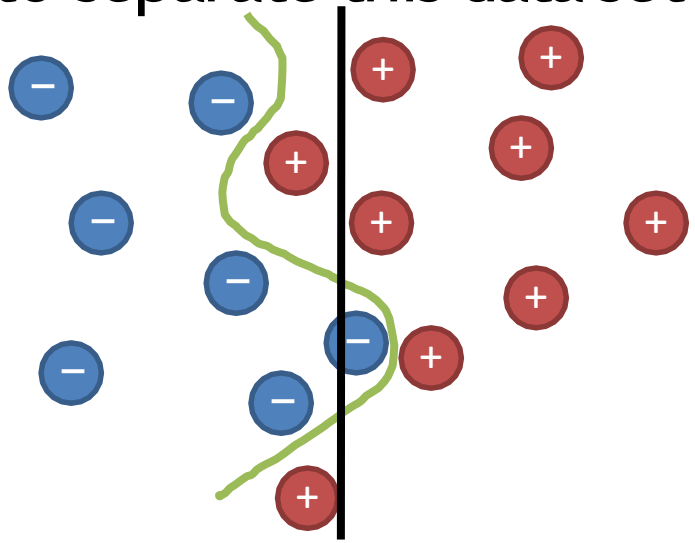


SVM

- At the beginning, we also assumed that there are only **two classes** in the training set
- How to handle more than that?
- Some ideas:
  - **One-versus-all classifiers:**
    Build an SVM for any class that occurs in the training set; To classify new items, choose the greatest margin's class
  - **One-versus-one classifiers:**
    Build an SVM for any pair of classes in the training set; To classify new items, choose the class selected by most SVMs
  - **Multiclass SVMs**
    …

Overfitting

- **One problem in using SVMs remains:**
  If we use a mapping to a high-dimensional space that is "complicated enough," we could find a perfect linear separation in the transformed space, for any training set
- So, what type of SVM is the "right" one?
- **Example:** How to separate this data set into two parts?



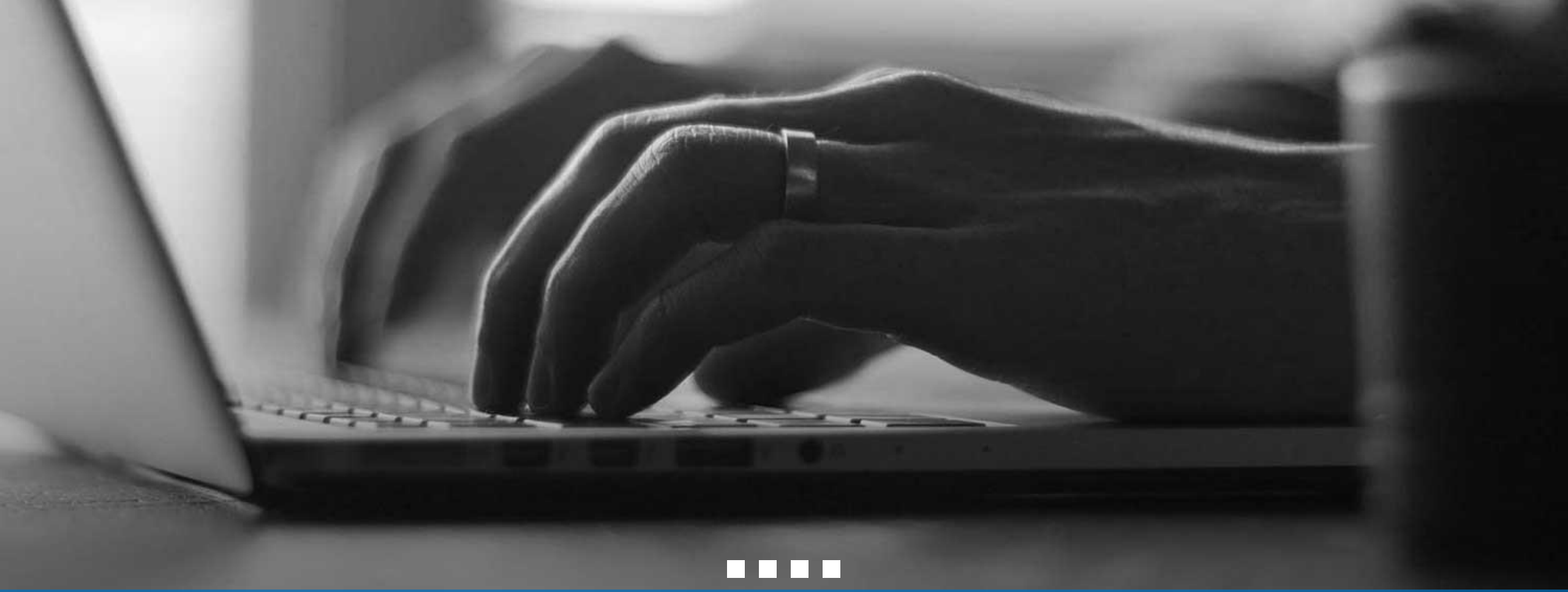
Overfitting

- **A perfect classification for the training set could generalize badly on new data**
- Fitting a classifier too strongly to the specific properties of the training set is called **overfitting**
- What can we do to avoid it?
- **Cross-validation:**
  - **Randomly split the available data** into two parts (training set + test set)
  - Use the first part for **learning the classifier** and the second part for **checking the classifier's performance**
  - Choose a classifier that maximizes performance on the test set

Overfitting

- **Regularization:**
  - If you know how a "good" classifier roughly should look like (e.g. polynomial of low degree) you could introduce a penalty value into the optimization problem
  - Assign a large penalty if the type of classifier is far away from what you expect, and a small penalty otherwise
  - Choose the classifier that minimizes the overall optimization goal (original goal + penalty)
  - An example of regularization is the **soft margin technique** since classifiers with large margins and few errors are preferred


QA ???



THANK YOU

Munawar, PhD – moenawar@gmail.com – www.moenawar.web.id