DATA MINING

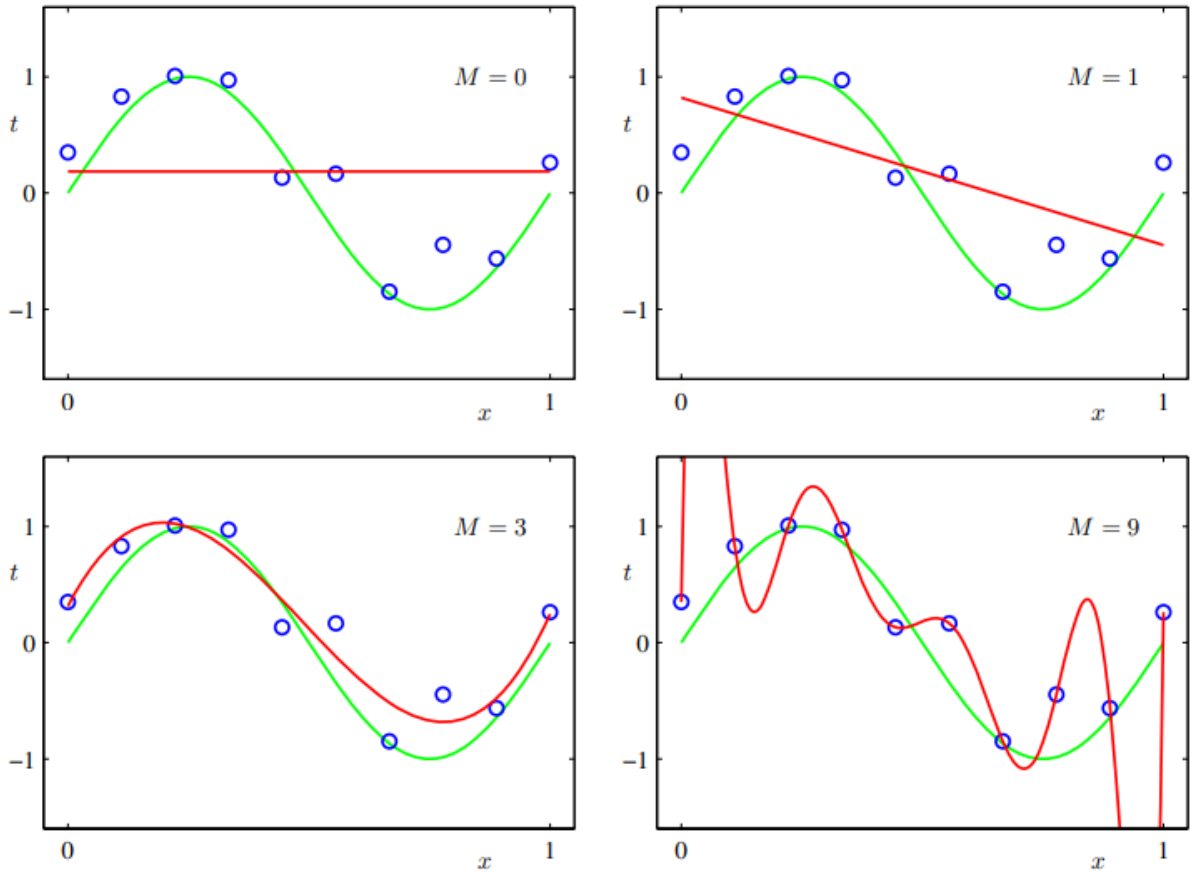
/CISC 873 - Steven Ding
/Week #1/Lecture 3
Magic. Just Magic..

Assignments (Option #1 100% total)	Due Date	Individual Project (Option #2 100% total)	Due Date
Assignment #0	10-02	Project Proposal	10-02
Assignment #1	10-09	Phase 1 source code and results	10-23
Assignment #2	10-23	Phase 2 source code and results	11-13
Assignment #3	11-6	Phase 3 source code and results	11-27
Assignment #4	11-13	Final report	12-18
Assignment #5	11-27		
Assignment #6	12-11		
Assignment #7	12-18		

- Survey available
- Individual project – let's chat (link to be created)

Overfitting

- Introduce more degrees of freedom ==> always fit better (to the observations)

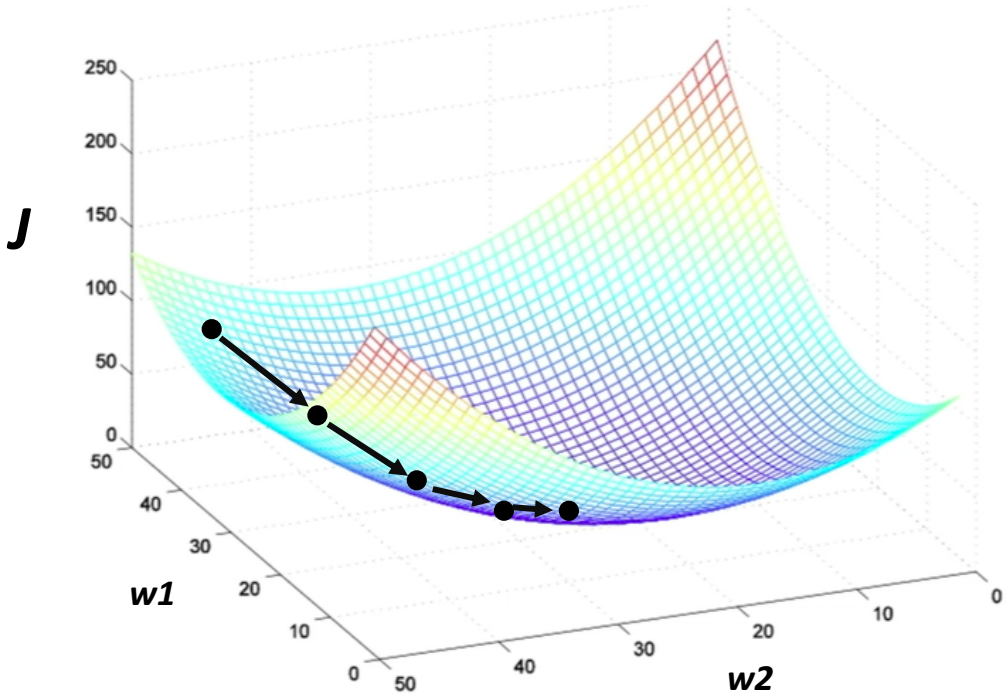


Bishop, C. M. (2006). Pattern recognition and machine learning. springer.

Optimizers

- Gradient Descend (GD)
- Stochastic Gradient Descend (SGD)
- SGD with Momentum
- Adadelta
- Adagrad
- Adam
- RSMProp
- Early Stopping

GD



Total cost:

$J = \Sigma$ (y' - y)^2

Optimizers

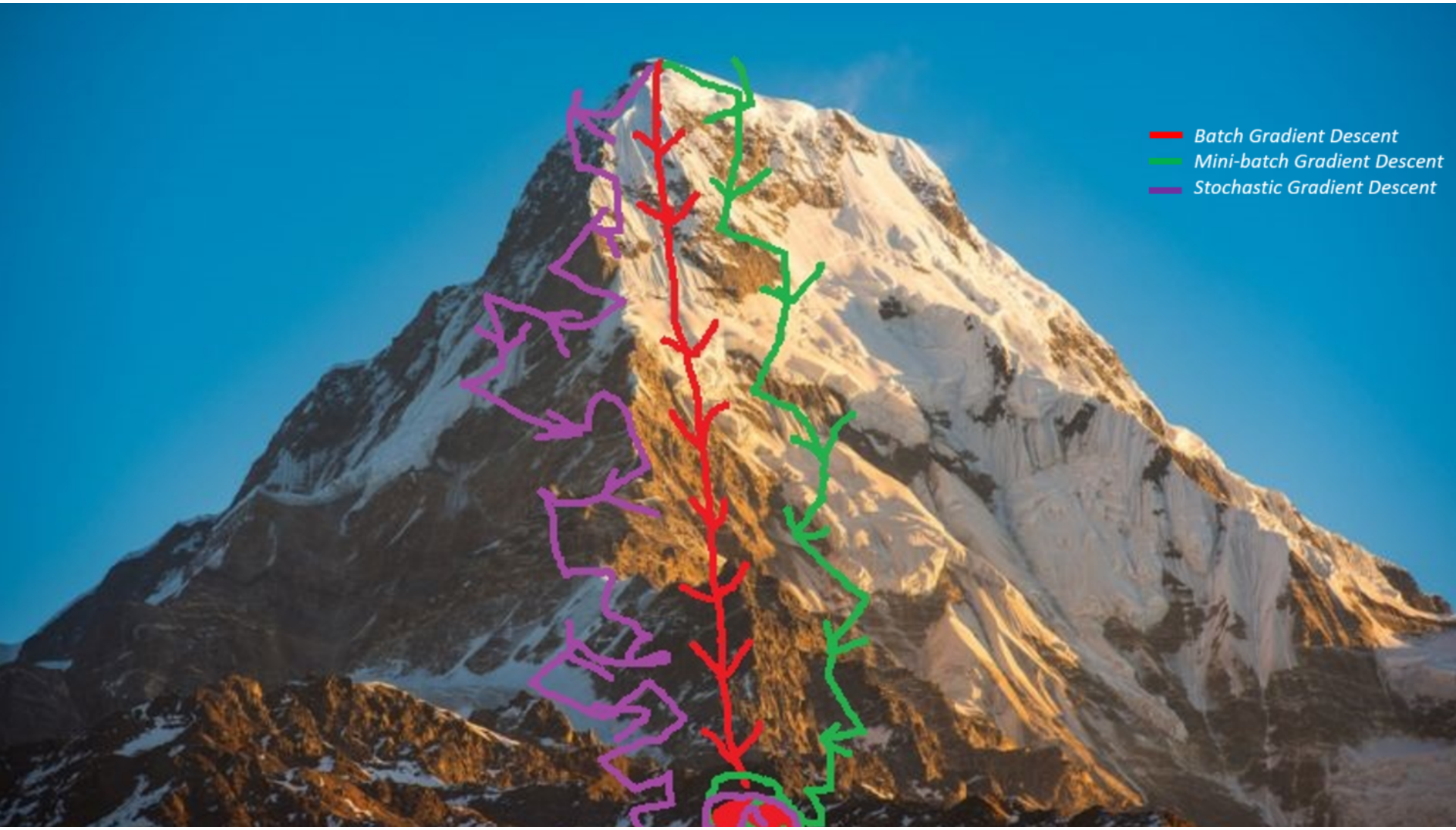
- Gradient Descend (GD)
  - 1 forward-backward pass **with the whole dataset**
- Stochastic Gradient Descend (SGD)
  - SGD
    - 1 forward-backward pass with **1 sample**
    - 1 forward-backward pass with **a mini-batch of sample**
      - A subset that is small enough to fit the memory

Optimizers

- Every backward pass:
  - Calculate Gradient
  - Multiplied by a learning rate (\alpha)

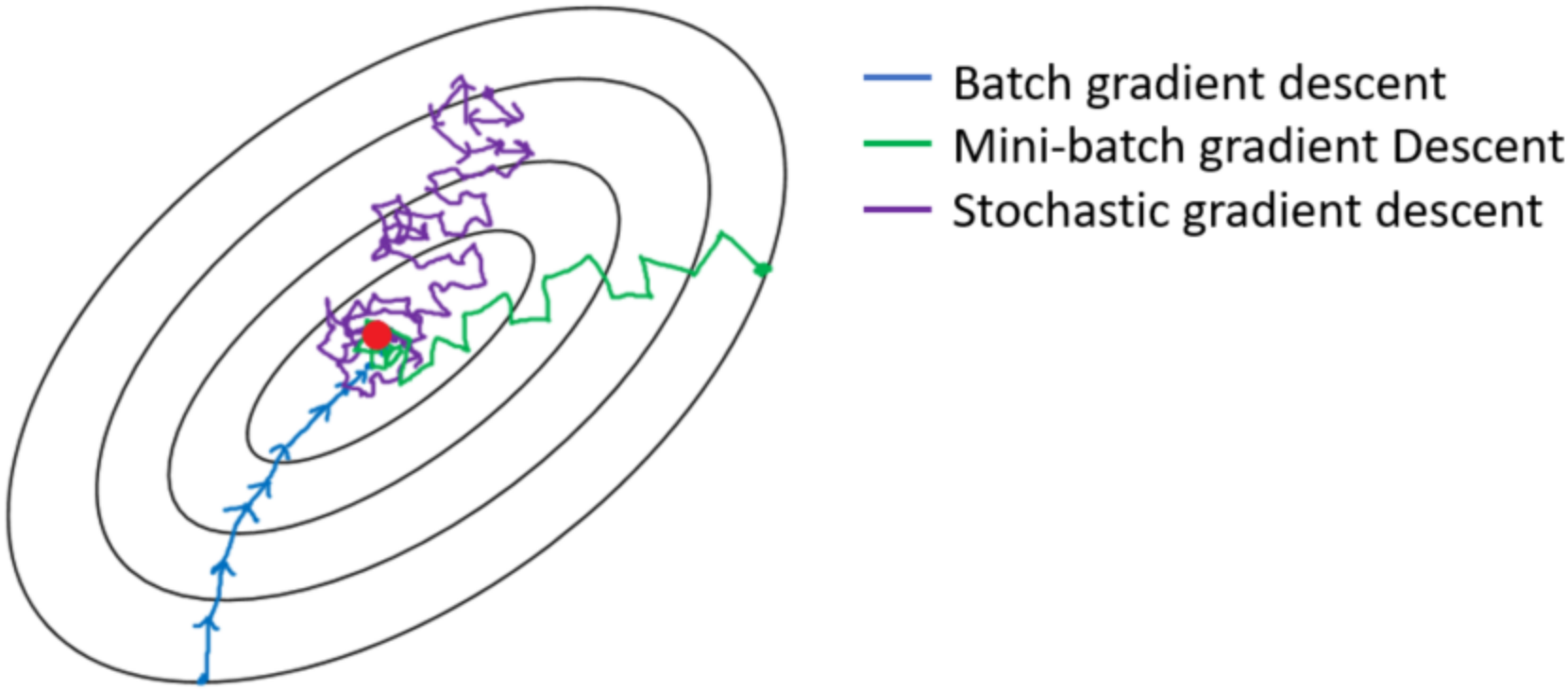
- GD
  - Gradient based on the whole training set => most accurate
  - Mini-batch: subset of samples to calculate gradient => sort-of accurate
  - 1 sample SGD: hm.. A bit high variance. Still can get the job done but just takes a bit longer.

Optimizers



https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Optimizers



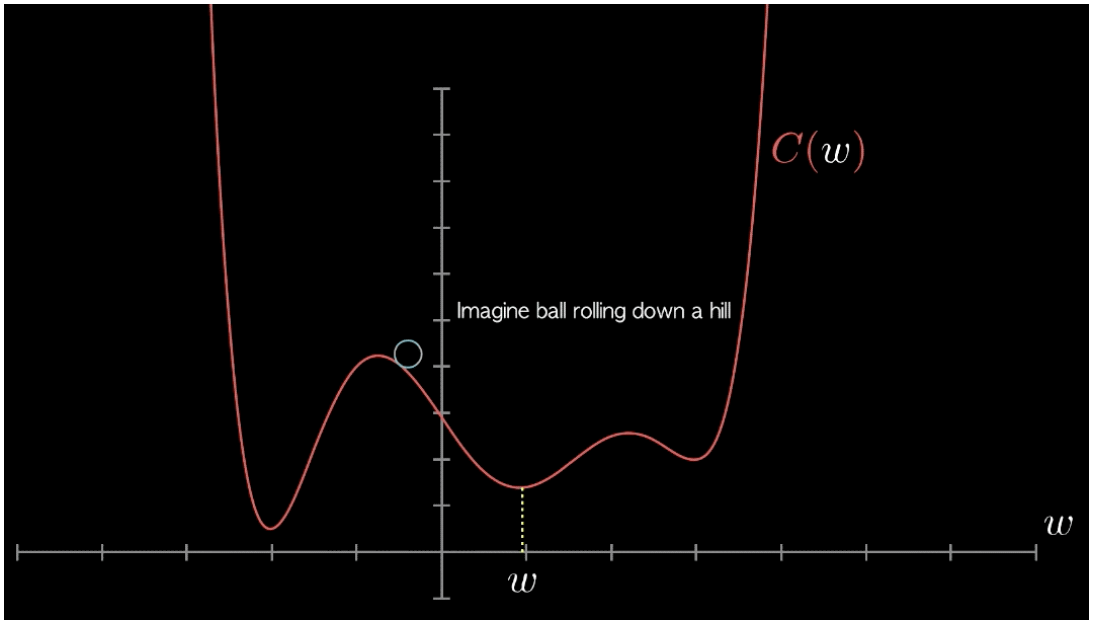
https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Optimizers – when?

- Gradient Descend (GD)
  - Convex/smooth loss function
  - Small dataset/model
- Stochastic Gradient Descend (SGD)
  - SGD
    - 1 forward-backward pass with **1 sample**
      - Fast. Low memory requirement
    - 1 forward-backward pass with **a mini-batch of sample**
      - Optimal subset that fits the memory

SGD+momentum

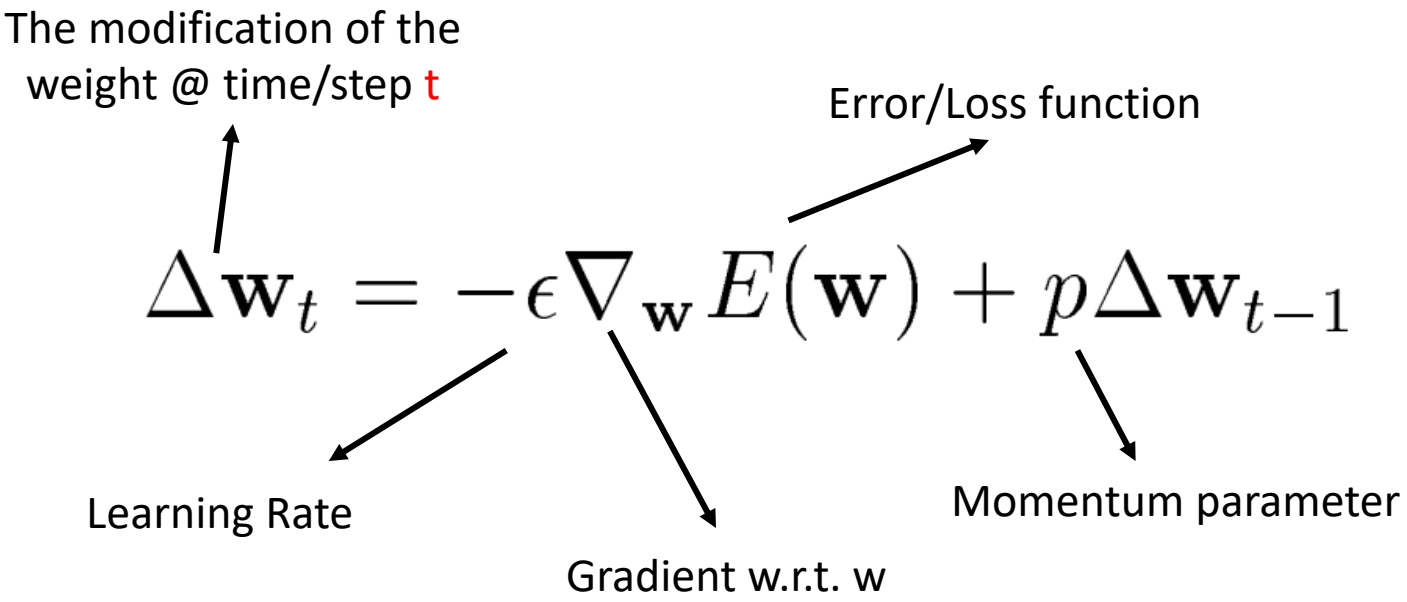
- Accelerate!!
  - With the concept of `speed`



- Try this:
  - https://distill.pub/2017/momentum/

SGD+momentum

- How?
  - Use a moving average of gradient.

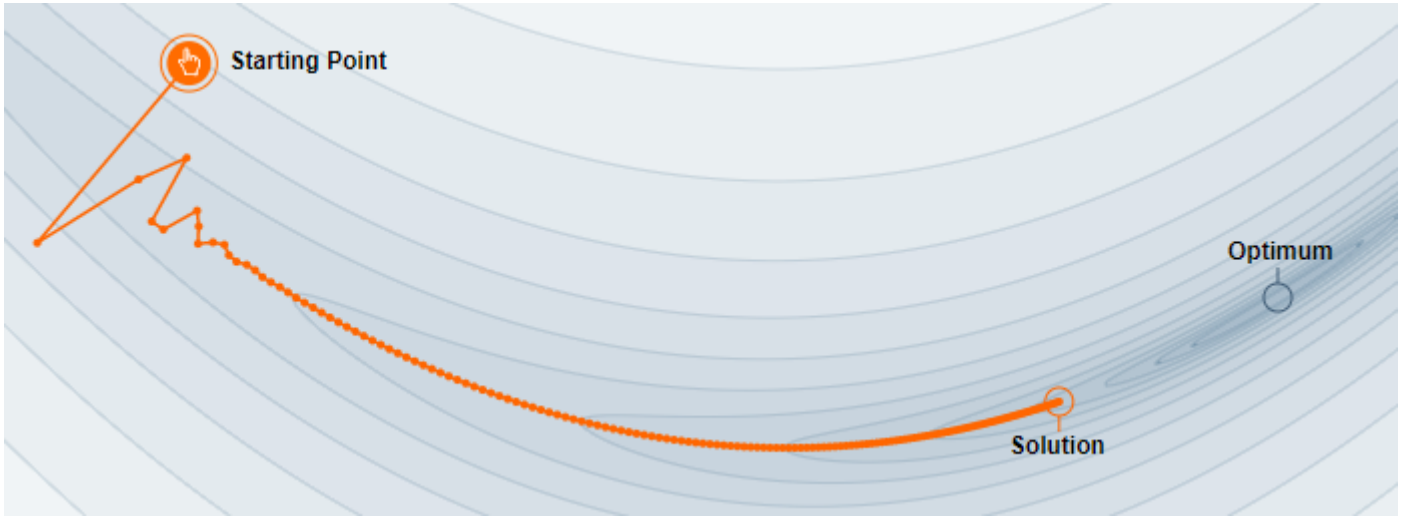


$$w_t = w_t + \Delta w_t$$

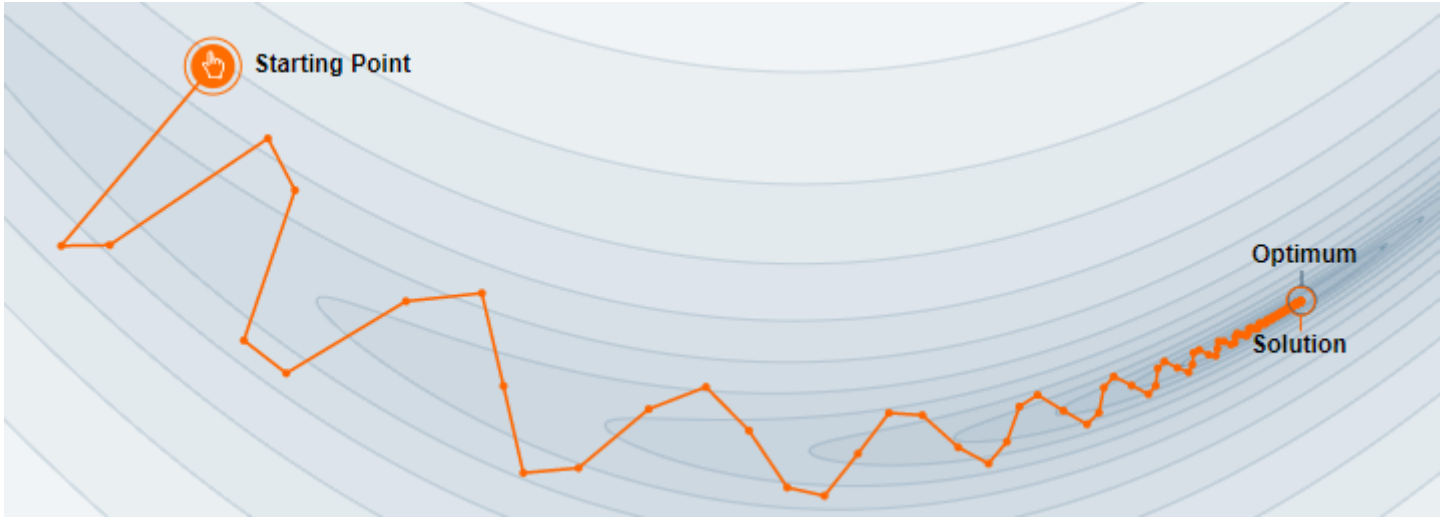
Ning Qian. On the momentum term in gradient descent learning algorithms. Neural networks : the official journal of the International Neural Network Society, 12(1):145–151, 1999

SGD+momentum

Momentum=0.5



Momentum=0.8



Ning Qian. On the momentum term in gradient descent learning algorithms. Neural networks : the official journal of the International Neural Network Society, 12(1):145–151, 1999

Adagrad

- Use the idea of Momentum
- Scale learning rate according to the history of gradient
- Learning rate is reduced if the gradient is very large
- Different learning rate for different parameter
- Normalized by exponentially decaying average of past squared gradients
- **Eliminate the need to manually tune the learning rate**

$$g_t = g_{t-1} + \nabla_w E(w)^2$$

$$w_t = w_t - \frac{\varepsilon}{\sqrt{g_t} + \beta} \nabla_w E(w)^2$$

AdaDelta

- Use the idea of Momentum
- Less aggressive than Adagrad
- Adagrad issues:
  - **Accumulation of the squared gradients**
  - **learning rate is always decreasing**
- AdaDelta tries to solve the above issue
- Restrict the history into a specific size
- **No need to set initial learning rate**

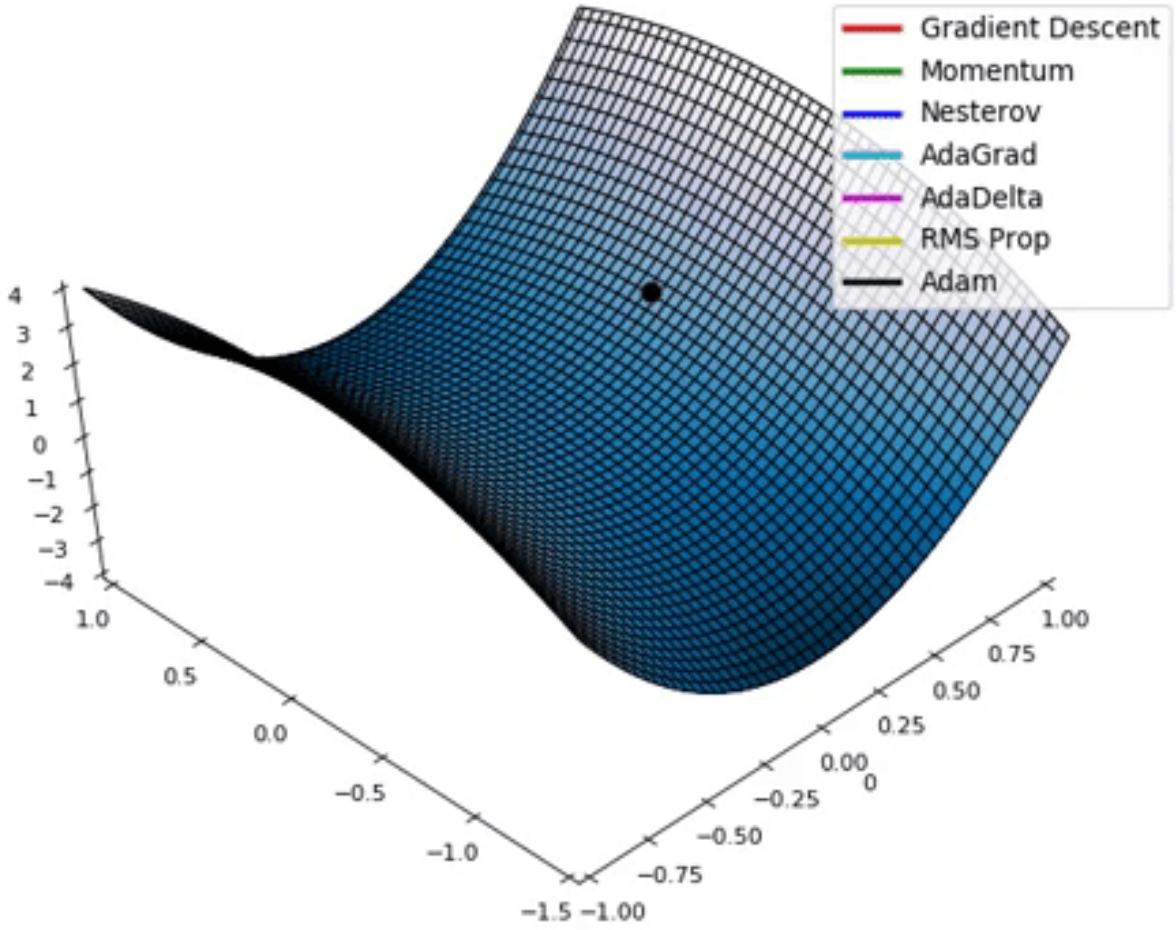
RMSProp

- Use the idea of Momentum
- Also tries to solve Adagrad's issue
- Similar idea to AdaDelta:
  - Normalize learning rate by the magnitudes of recent gradient of a weight.
  - But with different formulations.

Adaptive Moment Estimation (Adam)

- Similar idea to Adadelta and RMSprop
- Keep track of exponentially decaying average of past gradients
- Also keeps an exponentially decaying average of past gradients, similar to momentum

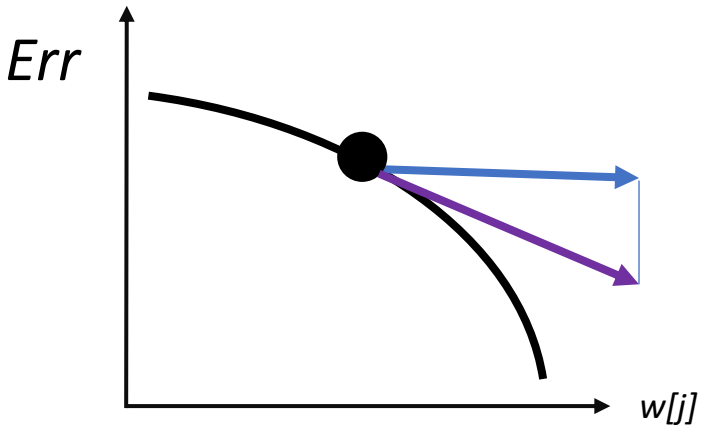
Adaptive Moment Estimation (Adam)



https://rnrahman.com/blog/visualising-stochastic-optimisers/

Solving with Gradient Descent

- Recall: $\partial Err(\theta)/\partial \boldsymbol{w} = -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$
  - Given specific \theta we can know its gradients to improve it!
  - To improve = to reduce Err(…)
  - Complexity is ~ O(|D|^2)



- Algorithm:
  - Initialize \theta as w0
    - For t in range(n):

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \times \partial Err(\theta)/\partial \boldsymbol{w}_t$$

      - Stop if:

$$\left|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\right| \leq \epsilon$$

Automatically Differentiation engine

- Chain Rule:

$$z = \omega x + b$$
$$\hat{y} = \sigma(z)$$
$$L = \frac{1}{2}(y - \hat{y})^2 + \lambda\frac{1}{2}\omega^2$$

$$L = \frac{1}{2}(y - \sigma(\omega x + b))^2 + \lambda\frac{1}{2}w^2$$

$$\frac{\partial L}{\partial_\omega} = \frac{\partial}{\partial\omega}\left(\frac{1}{2}(y - \sigma(\omega x + b))^2 + \lambda\frac{1}{2}w^2\right)$$

$$\frac{\partial L}{\partial_\omega} = (\sigma(\omega x + b)^{\cdot} - t)\sigma'(wx + b) + \lambda w$$

$$\frac{\partial L}{\partial_b} = \frac{\partial}{\partial w}\left(\frac{1}{2}(y - \sigma(\omega x + b))^2 + \lambda\frac{1}{2}w^2\right)$$

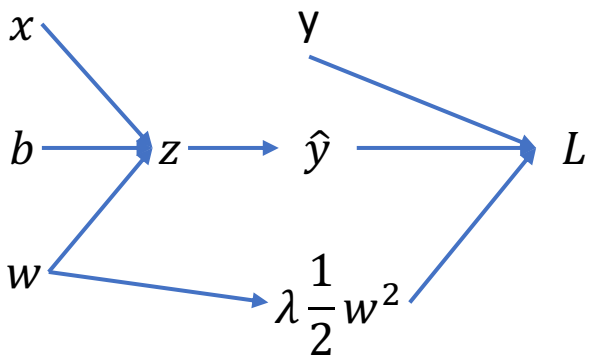
$$\frac{\partial L}{\partial_b} = (6(\omega x + b) - y)\sigma'(\omega x + b)$$

Automatically Differentiation engine

- Chain Rule:

$$\frac{d}{d_t} f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dk}$$

computation graph



TensorFlow

[5]:
```
import tensorflow as tf
x = tf.Variable(3.0)
with tf.GradientTape() as tape:
  y = x**2
dy_dx = tape.gradient(y, x)
dy_dx.numpy()
```

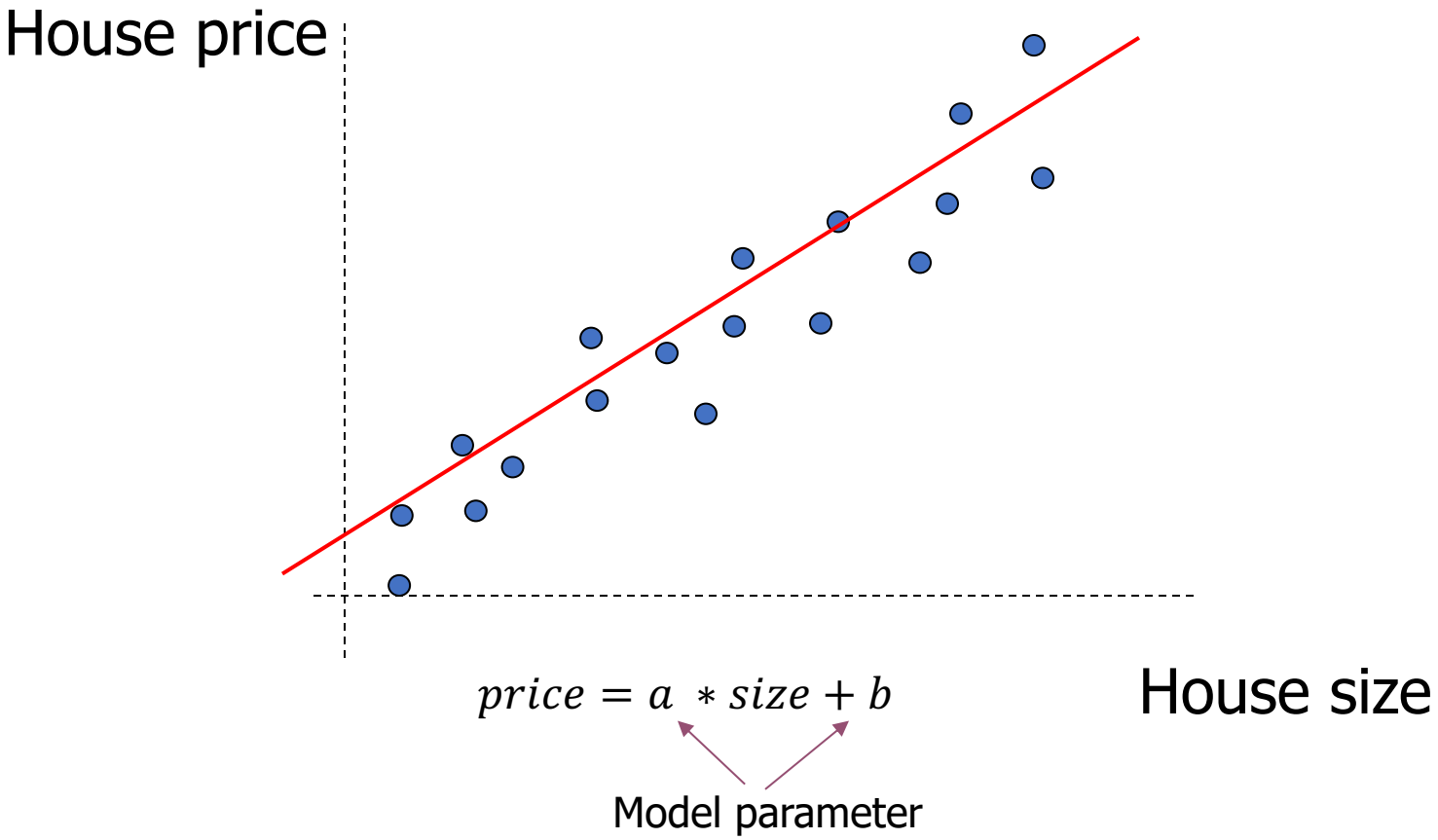
[5]:
```
6.0
```

DATA MINING

/CISC 873 - Steven Ding
/Week #1/Lecture 3
Classification

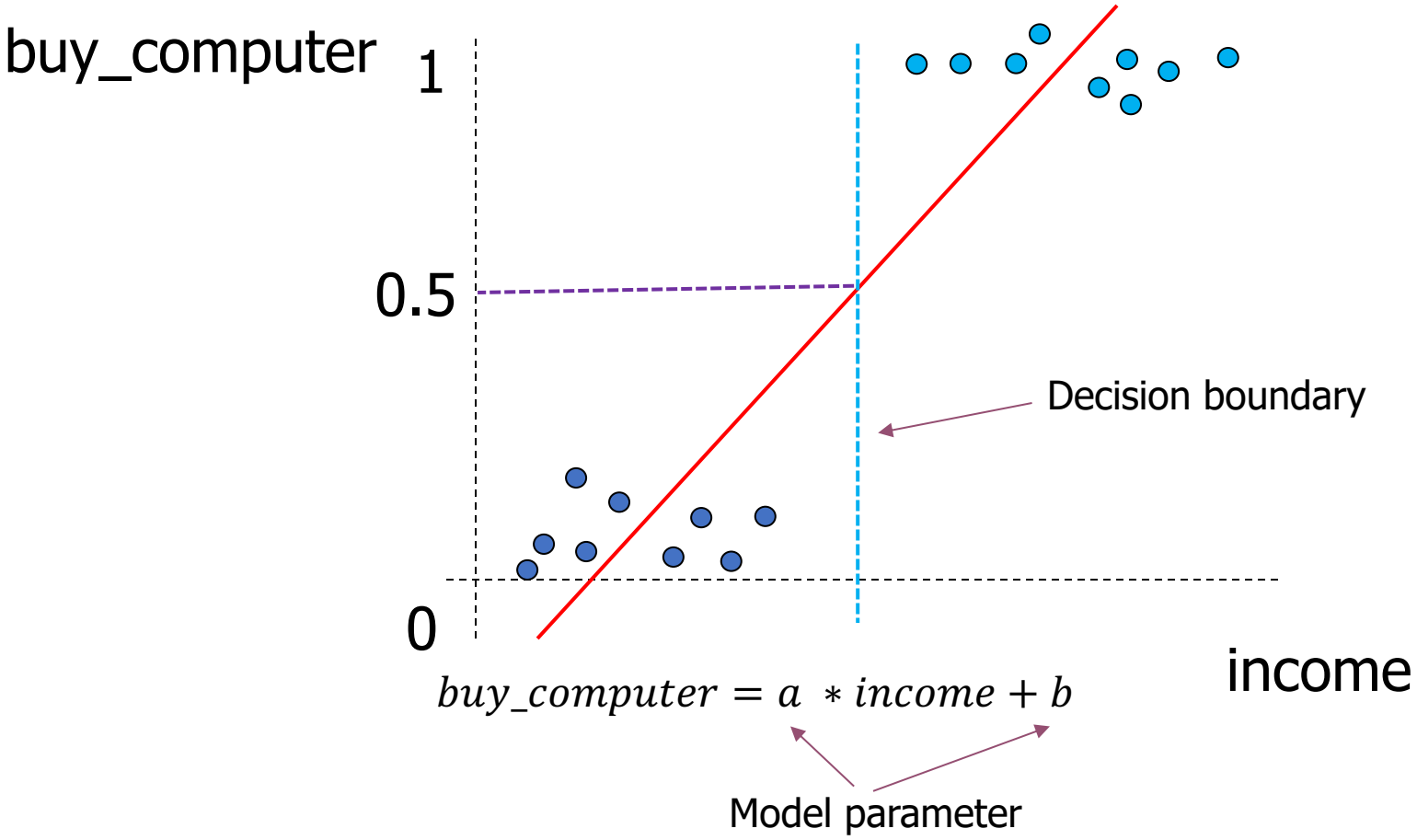
Linear Regression

- House Price Prediction (a prediction problem)



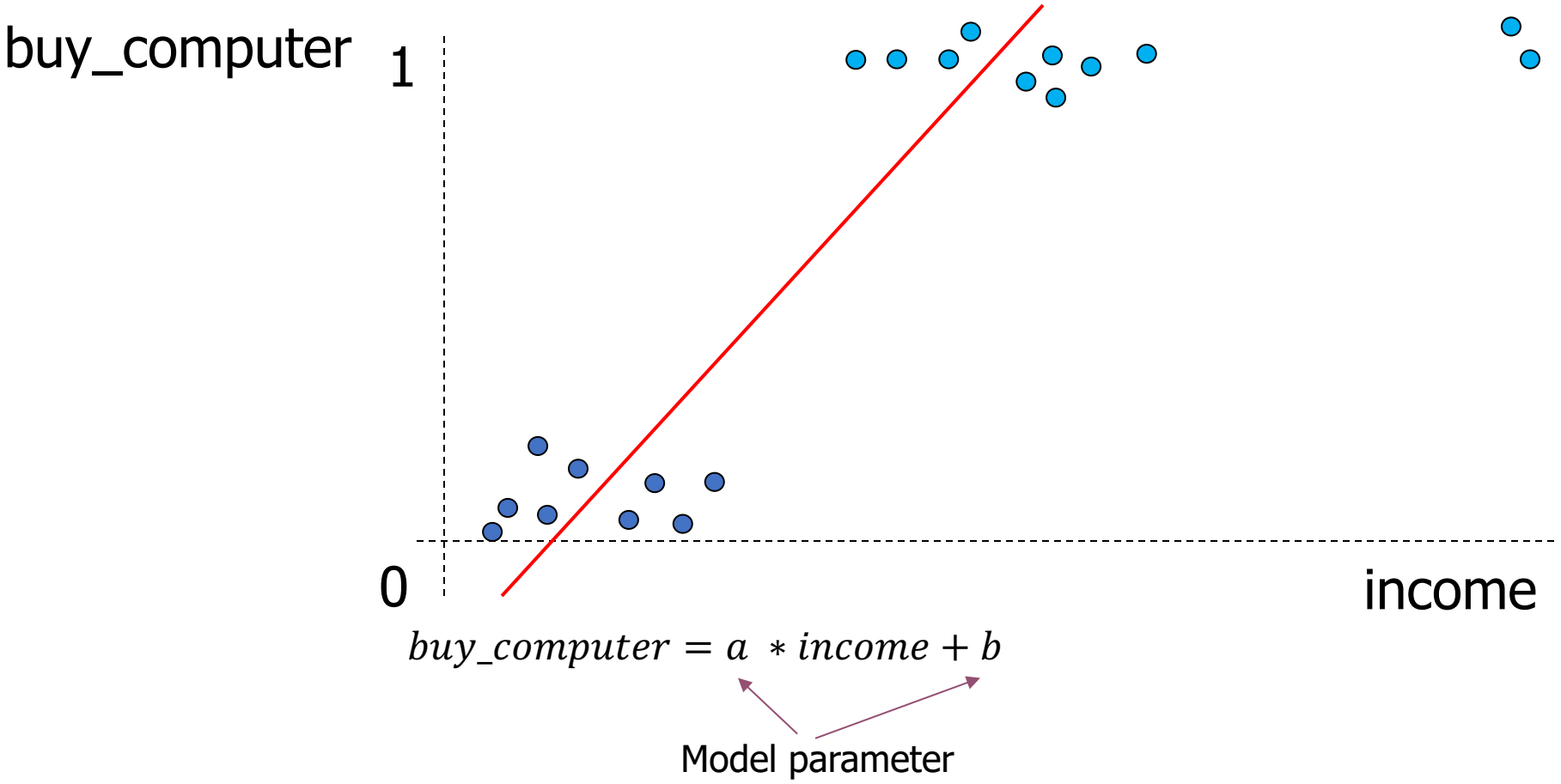
Linear Regression

- Linear Regression for Classification



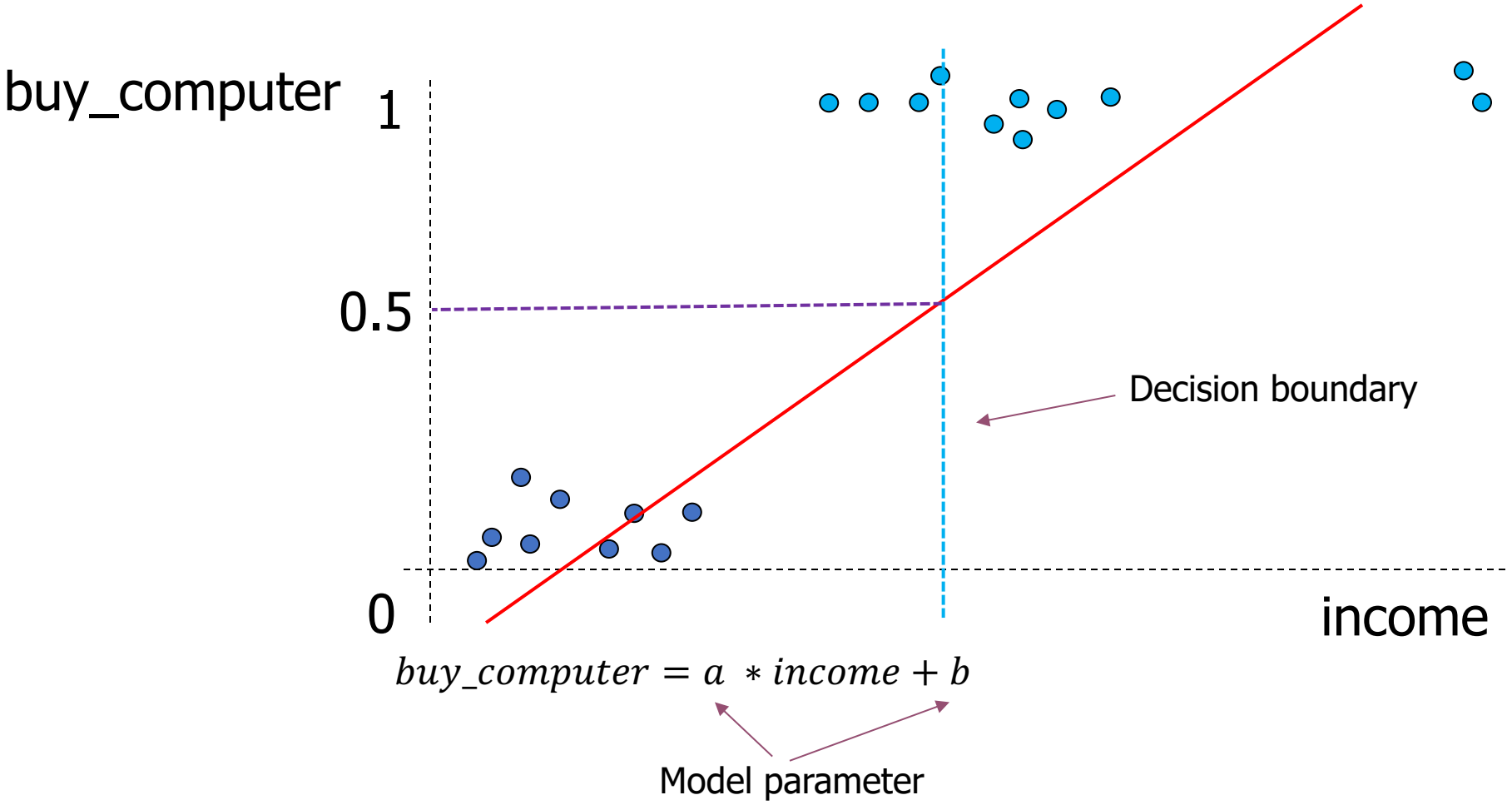
Linear Regression

- Linear Regression for Classification



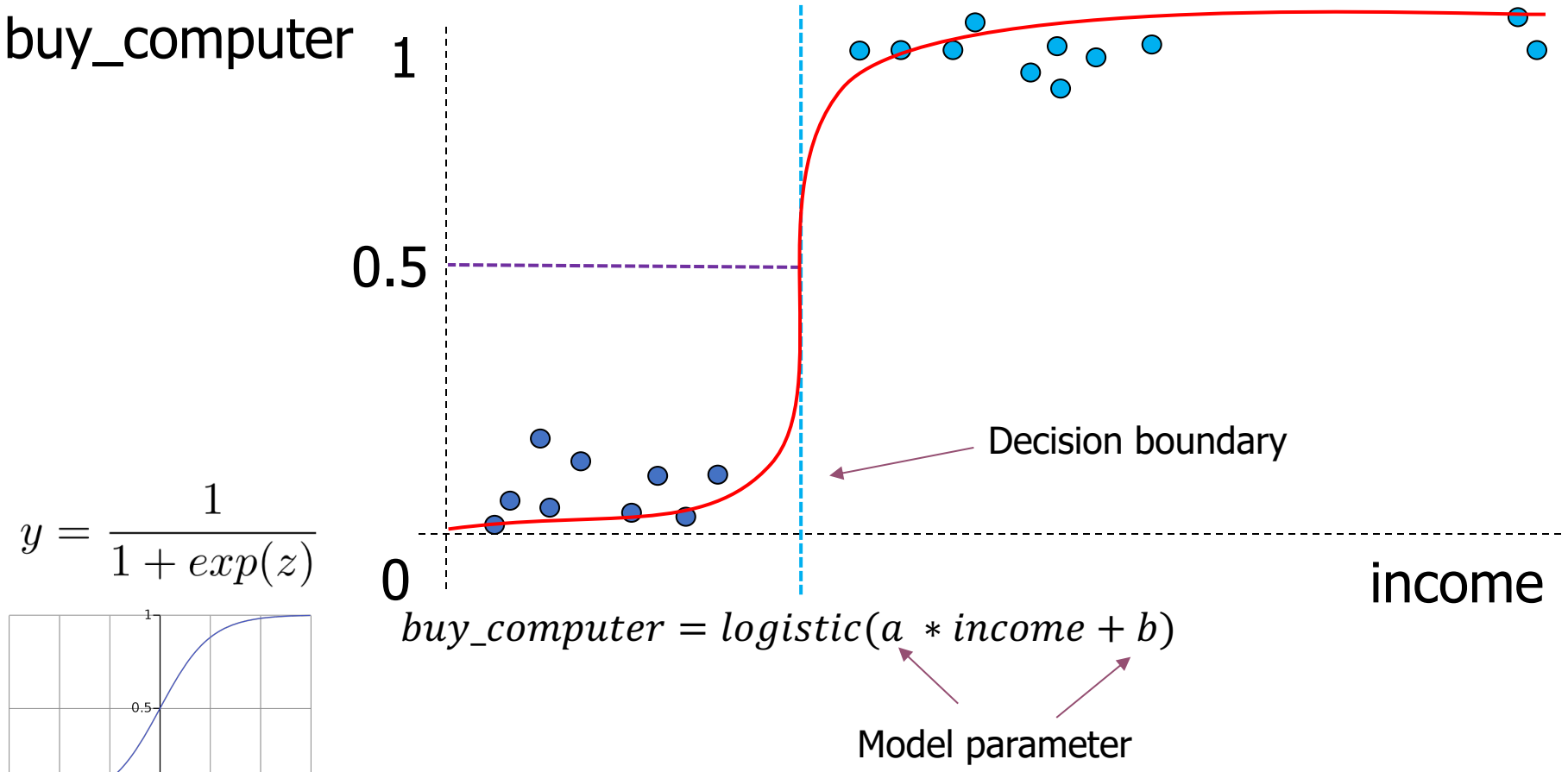
Linear Regression

- Linear Regression for Classification



Logistic Regression

- Logistic Regression for Classification



Logistic Regression

- Logistic Regression for Classification
- The logistic function rescales its input to between 0 and 1.
- The logistic function transforms a straight curve into a ‘S’-shape curve.
- By default, it only handles binary classification task.
- The prediction can be interpreted as probability
- For multinominal class attribute, we use generalized linear regression with multinominal family (skip, but available in RapidMiner).

Generative vs Discriminative Model

- Generative Model
  - Separately model $P(x|y)$ and $P(y)$ from the training set. Classifying using the Bayesian rule:

$$P(y = 1|x) = \frac{P(x|y = 1)\, p(y = 1)}{p(x)}$$

  - Generally converge faster (efficient in training) and requires less data.
  - Naïve Bayesian, Linear discriminant analysis, …

- Discriminative Model
  - Direct estimation of $P(y|x)$
  - Requires more data but has lower asymptotic error (performs better).
  - Decision Tree, SVM, Logistic Regression, K-NN, …

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems* (pp. 841-848).

Probabilistic view of discriminative learning

$$P(y=1|x) = \frac{P(x, y=1)}{P(x)}$$

$$\dots$$

$$= \frac{1}{1+exp(a)}$$

Logistic function
(what is the predicted probability)

$$a = ln \frac{P(y=1|x)}{P(y=0|x)}$$

Log-odds ratio
(given x, how much more likely y is 1 compared to 1?)

DATA MINING

/CISC 873 - Steven Ding
/Week #1/Lecture 3
Preprocessing – Dirty Works

Machine Learning is 80% preprocessing and 20% model making.

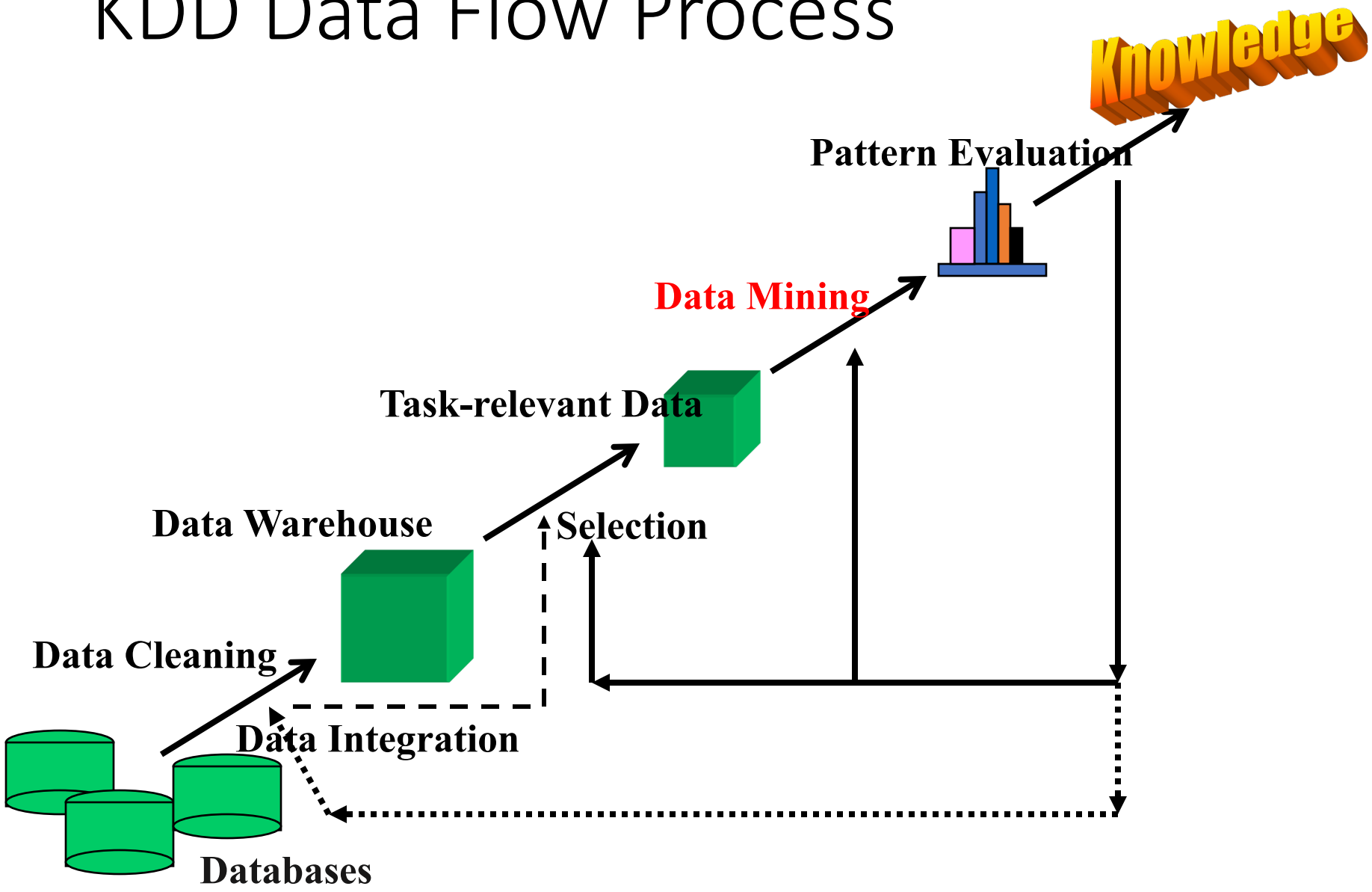
Linear Regression – Tabular Data

- Boston Housing Data
  - CRIM per capita crime rate by town
  - ZN proportion of residential land zoned…
  - INDUS proportion of non-retail business acres per town
  - NOX nitric oxides concentration (parts per 10 million)
  - RM average number of rooms per dwelling
  - AGE proportion of owner-occupied units built prior to 1940
  - DIS weighted distances to five Boston employment centres
  - …
  - MEDV Median value of owner-occupied homes in $1000's

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.

KDD Data Flow Process



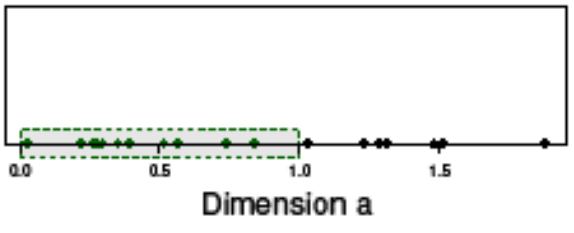
Attributes

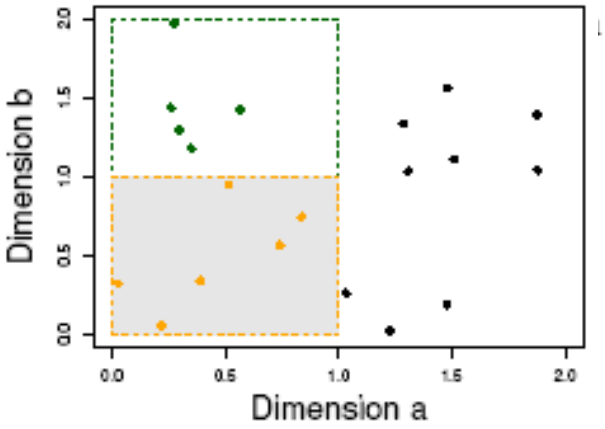
- **Attribute (**or **dimensions, features, variables**): a data field, representing a characteristic or feature of a data object.
  - *E.g., customer _ID, name, address*

- Types:
  - Nominal (aka **categorical**)
  - Binary (aka **binominal**)
    - Symmetric vs Asymmetric
  - Ordinal
  - Continuous

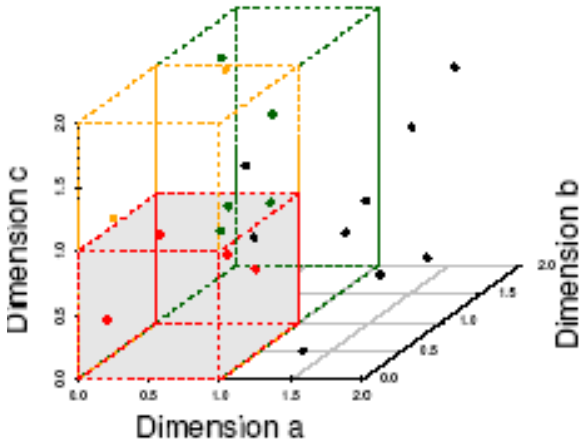
Important Characteristics of Structured Data

- Dimensionality
  - Curse of dimensionality
- Sparsity
  - Only presence counts
- Resolution
  - Patterns depend on the scale
- Distribution
  - Centrality and dispersion





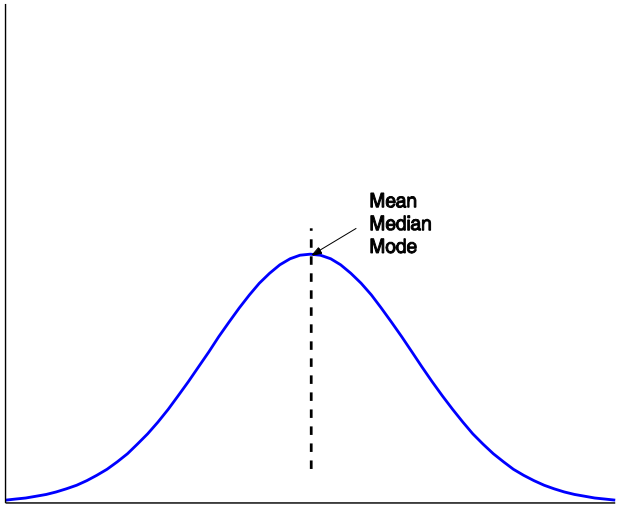
(b) 6 Objects in One Unit Bin

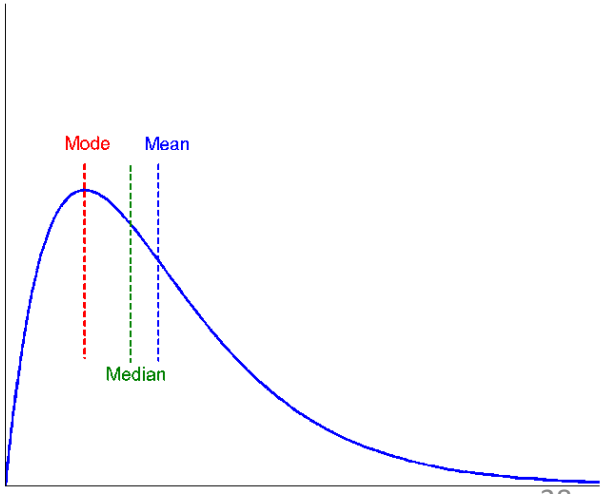


(c) 4 Objects in One Unit Bin

Important Characteristics of Structured Data

- Dimensionality
  - Curse of dimensionality
- Sparsity
  - Only presence counts
- Resolution
  - Patterns depend on the scale
- Distribution
  - Centrality and dispersion





Data Does not Come Prepared

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
    - e.g., do not need both age and birth year
- Data transformation
  - Generalize and/or normalize data
    - e.g. do not need to know specific address of a customer, city or postal code is good enough.

Normalization

- Scaling individual samples to have unit norm.
- Could be L1 norm or L2 norm
  - Very important for vector space model (think about dot-product)

```
from sklearn import preprocessing
import numpy as np
from numpy.linalg import norm

X_train = np.array([[ 1., -1.,  2.],
                    [ 2.,  0.,  0.],
                    [ 0.,  1., -1.]])
X_scaled = preprocessing.normalize(X_train, norm='l2')

print(X_scaled)
print('norm', norm(X_scaled, axis=1))
print('std ', X_scaled.std(axis=0))
print('mean', X_scaled.mean(axis=0))
```

```
[[ 0.40824829 -0.40824829  0.81649658]
 [ 1.          0.          0.        ]
 [ 0.          0.70710678 -0.70710678]]
norm [1. 1. 1.]
std  [0.41053309 0.46075826 0.62254262]
mean [0.46941161 0.09961952 0.03646327]
```

Standardization

- A common requirement
  - Making the feature a Gaussian **with zero mean** and **unit variance**

```
1 from sklearn import preprocessing
2 import numpy as np
3 X_train = np.array([[ 1., -1.,  2.],
4                     [ 2.,  0.,  0.],
5                     [ 0.,  1., -1.]])
6 X_scaled = preprocessing.scale(X_train)
7
8 print(X_scaled)
9 print('mean', X_scaled.mean(axis=0))
10 print('std ', X_scaled.std(axis=0))
```

```
[[ 0.         -1.22474487  1.33630621]
 [ 1.22474487  0.         -0.26726124]
 [-1.22474487  1.22474487 -1.06904497]]
mean [0. 0. 0.]
std  [1. 1. 1.]
```

Categorical Encoding

- Converting categorical attribute (aka nominal) into numeric features (or a one-hot encoding vector)

```
from sklearn import preprocessing
import numpy as np
from numpy.linalg import norm

enc = preprocessing.OrdinalEncoder()
X = [
     ['male', 'from US', 'uses Safari'],
     ['female', 'from Europe', 'uses Firefox']]
enc.fit(X)

enc.transform([
               ['female', 'from US', 'uses Safari']
               ])
```

```
array([[0., 1., 1.]])
```

Continuous Values? Discretization

- Bin-based Discretization

```
1 from sklearn import preprocessing
2 import numpy as np
3 from numpy.linalg import norm
4
5 X = np.array([[ -3., 5., 15 ],
6                [  0., 6., 14 ],
7                [  6., 3., 11 ]])
8
9 est = preprocessing.KBinsDiscretizer(
10     n_bins=[3, 2, 2], encode='ordinal'
11     ).fit(X)
12
13 est.transform(X)
```

```
array([[0., 1., 1.],
       [1., 1., 1.],
       [2., 0., 0.]])
```

Continuous Values? Discretization

- Thresholding numerical features => binominal

```
1 X = [[ 1., -1.,  2.],
2      [ 2.,  0.,  0.],
3      [ 0.,  1., -1.]]
4
5 binarizer = preprocessing.Binarizer(threshold=1.5).fit(X)
6
7 binarizer.transform(X)
```

```
array([[0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 0.]])
```

Missing Values?

- Normally marked as NA in most DS framework/libraries
- Quick one-liner to check/summarize missing values:
  - pandas dataframe
  - df.isna().sum()
  - (count number of missing values in for each feature)

- Imputation
  - The act of using a specific algorithm/statistical method attempting to fill in or impute missing values

Imputation Methods

- Simple Imputation
  - The feature mean value.
  - The feature median value.
  - The feature mode value.
  - A constant value.

```
SimpleImputer(missing_values=np.nan, strategy='mean')
```

- Univariate vs. Multivariate
  - Univariate – Consider only the feature under study
  - Multivariate – Consider multiple features or the entire set of features

Multivariate Methods

- Regression-based method:
  - See the missing attribute as a function of the other attributes
  - Build a model to predict the missing value, based on the other attributes.

- Python - from sklearn.impute import IterativeImputer
- R - MICE (Multivariate Imputation by Chained Equations)

- Multiple vs Single Permutation
  - Multiple – Permuted into multiple variants – Then multiple analysis
  - Single – Once (integrated with whatever tuning pipeline you have)

Multivariate Methods

- Data Mining Functions for Missing Value Imputation
  - Nearest neighbor
    - Python – KNNImputer
  - Clustering
    - Univariate Imputation within cluster
  - Association
    - Cover in the second part
  - Decision Tree
    - Univariate Imputation with the same leave/bin
  - Generative Methods
    - De-noising auto-encoder

  …

Boston Housing Got A PROBLEM

https://www.reddit.com/r/boston/comments/6io6eq/boston_real_estate/

DATA MINING

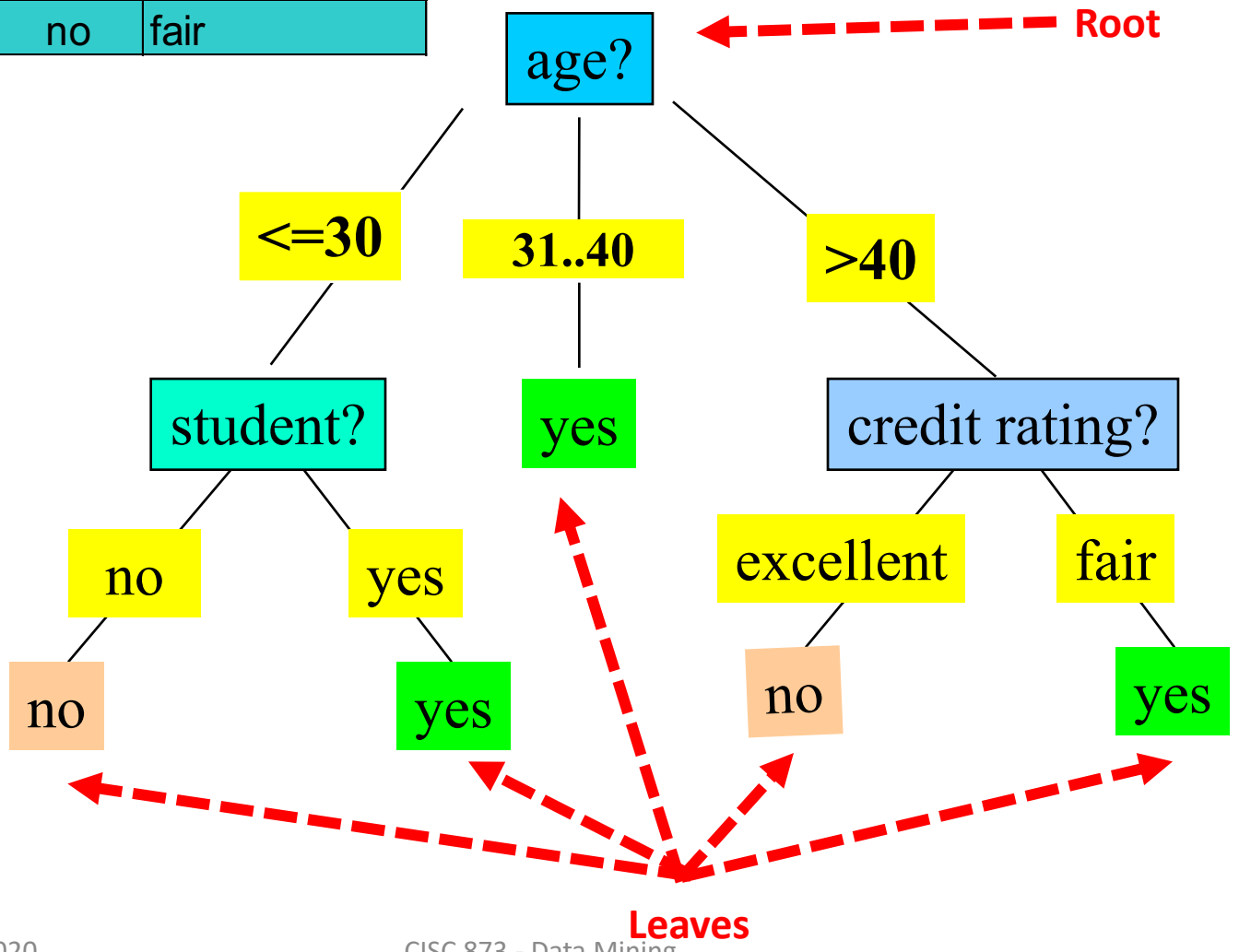
/CISC 873 - Steven Ding
/Week #1/Lecture 3
Tree-based Method

Decision Tree Induction: Training Dataset

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

Decision Tree for Classification

age	income	student	credit_rating
<=30	high	no	fair

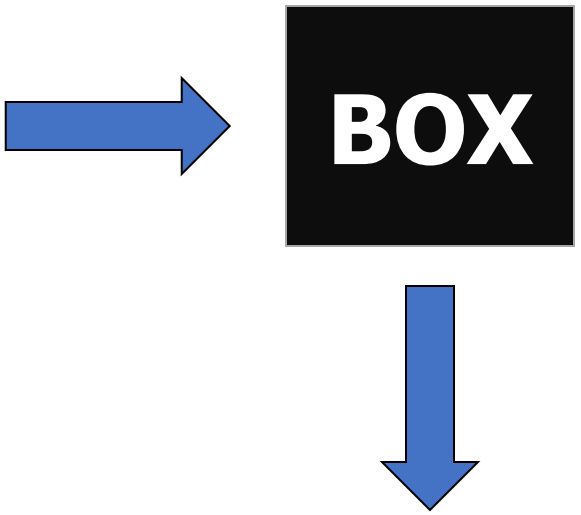


A Magical Black Box

Data table

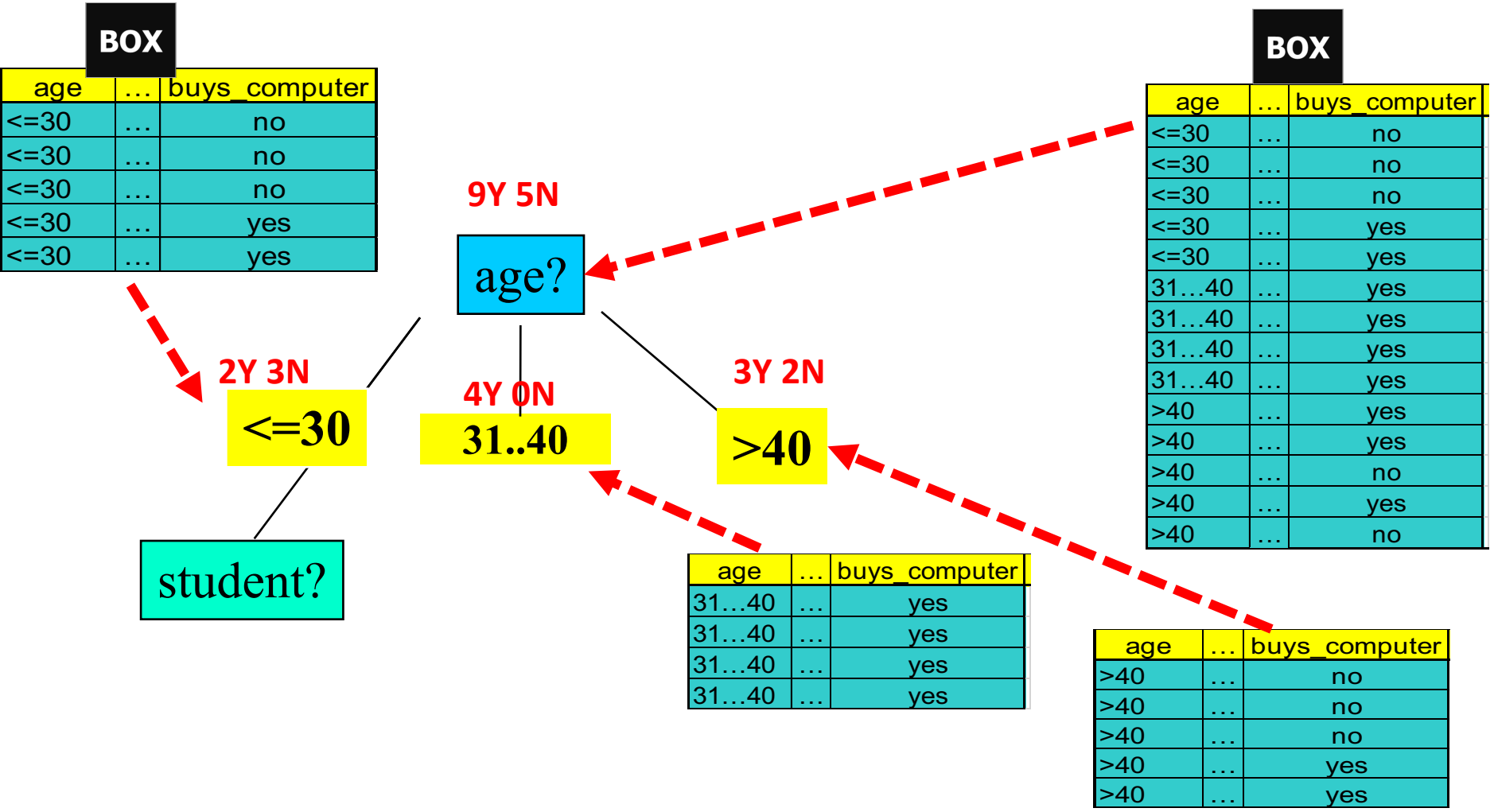
Class Attribute

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

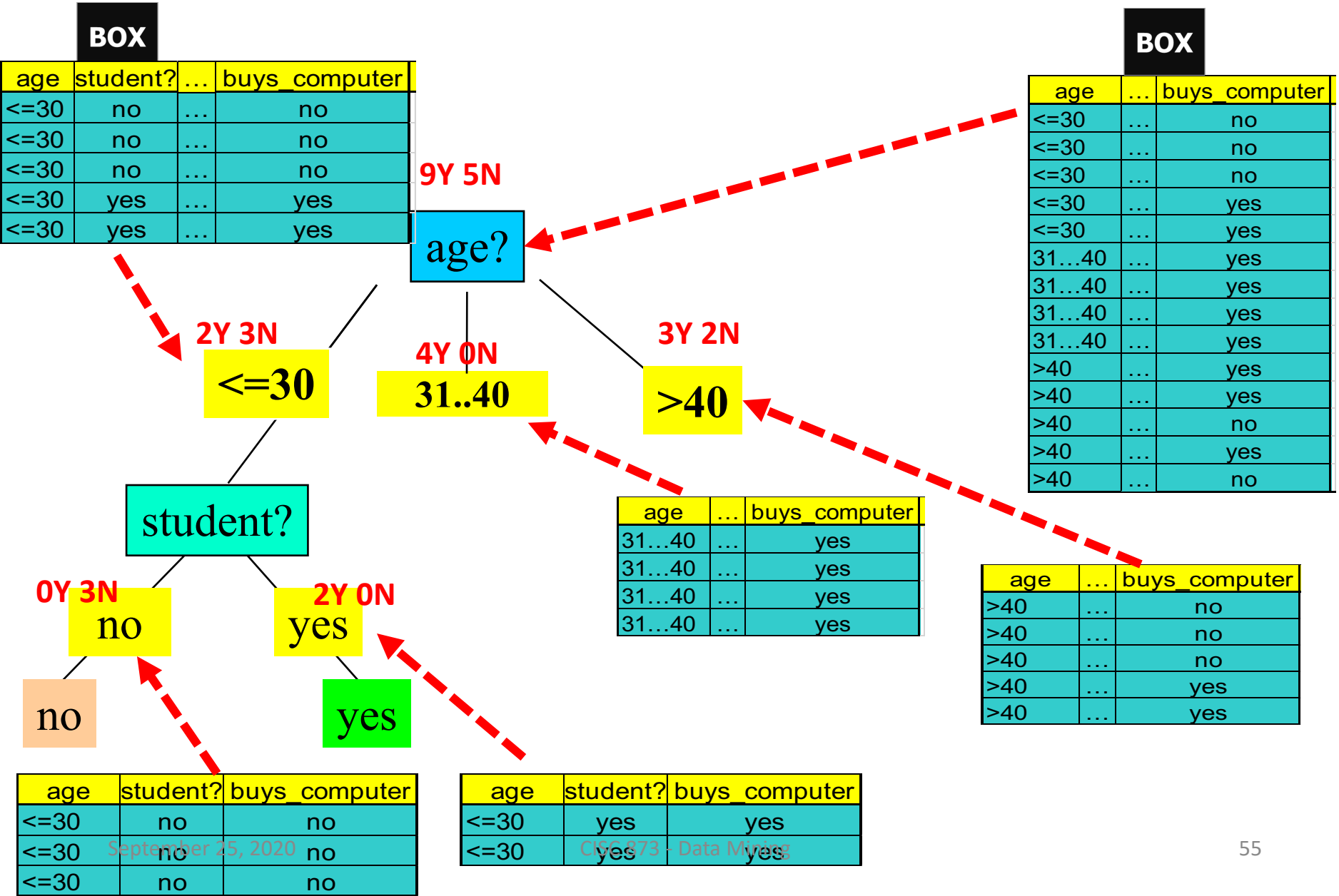


Age is the best attribute to create a node in the tree.

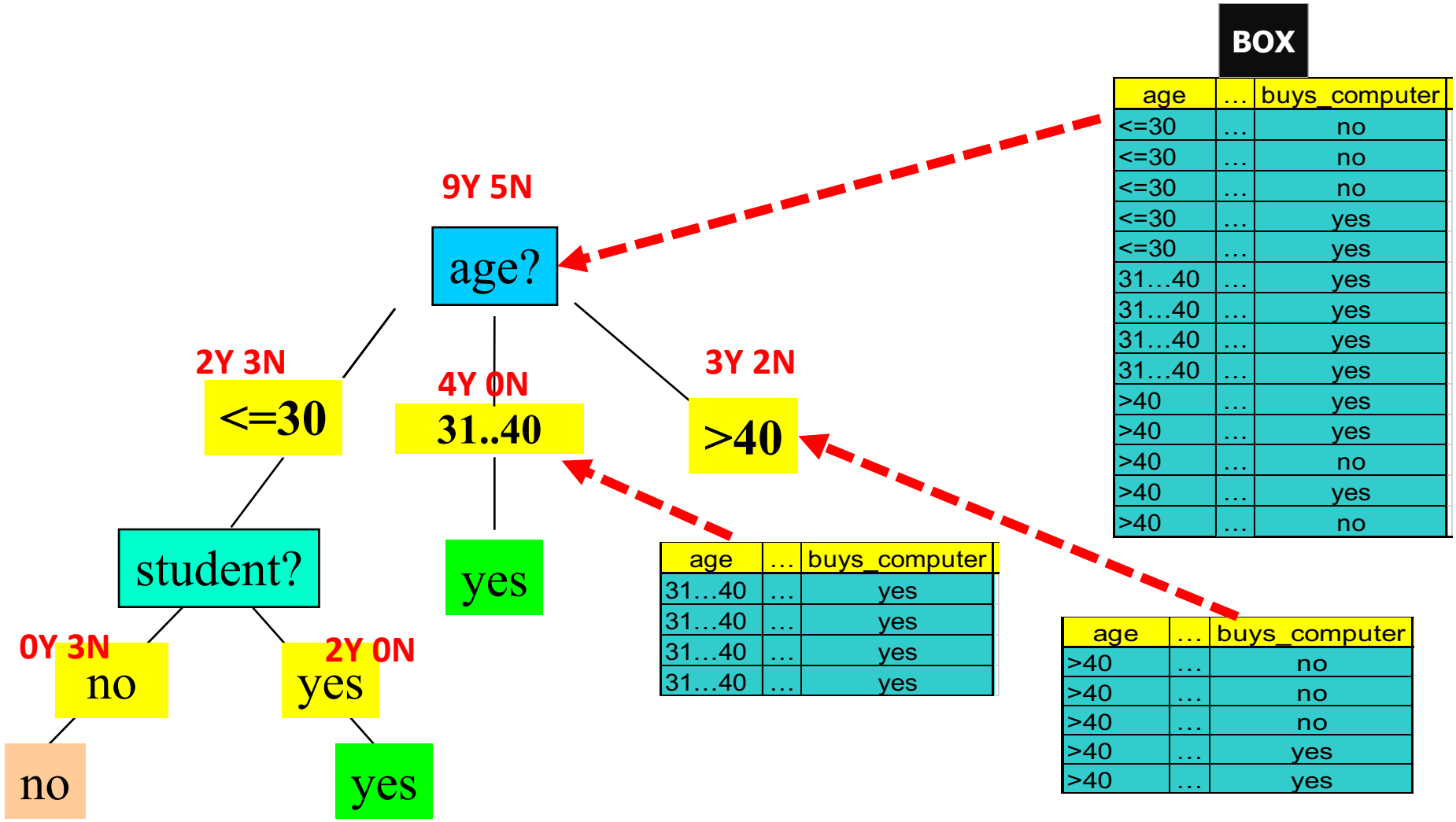
Output: A Decision Tree for "*buys_computer*"



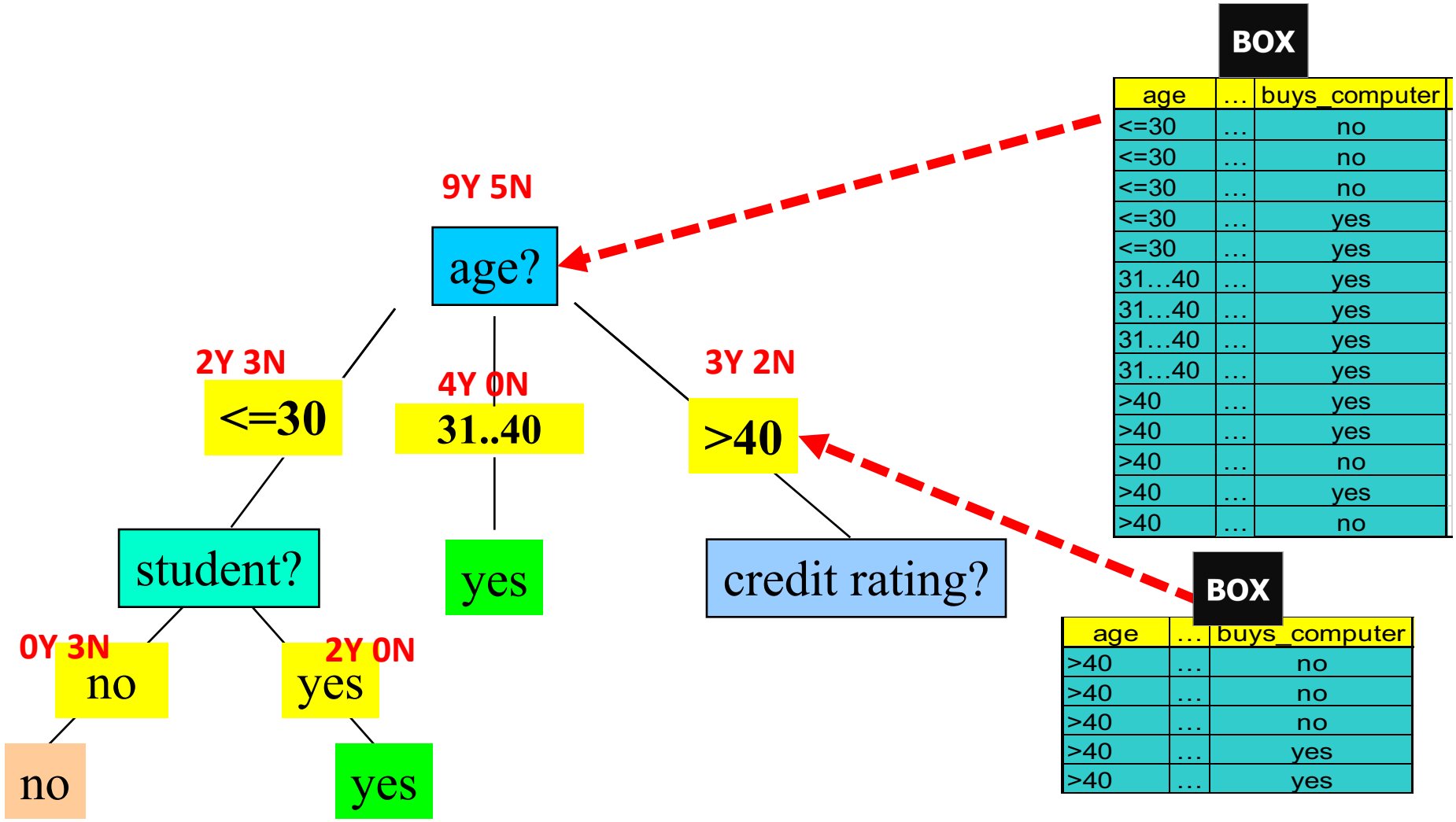
Output: A Decision Tree for "*buys_computer*"



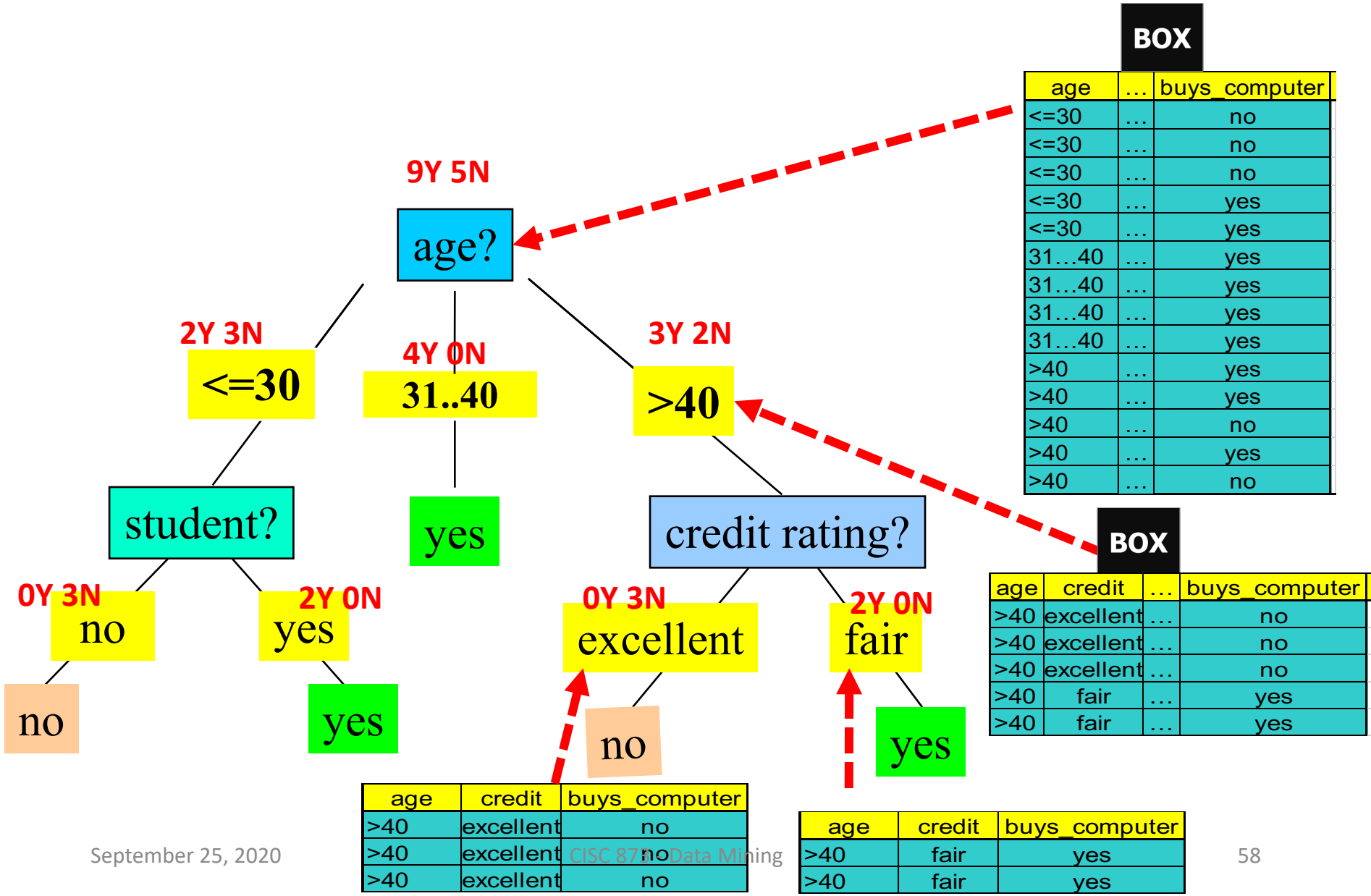
Output: A Decision Tree for "*buys_computer*"



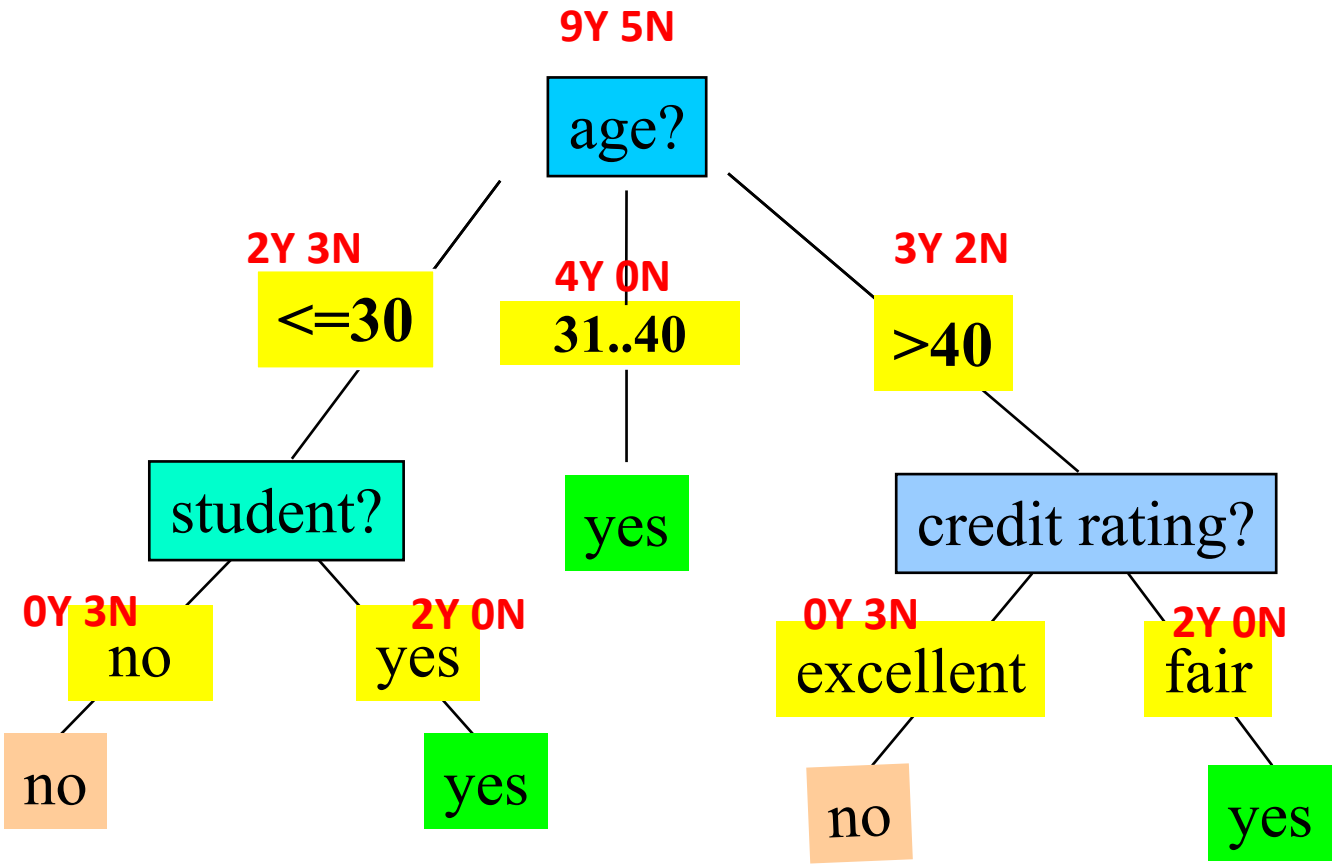
Output: A Decision Tree for "*buys_computer*"



Output: A Decision Tree for "*buys_computer*"



Output: A Decision Tree for "*buys_computer*"



Algorithm for Decision Tree Induction

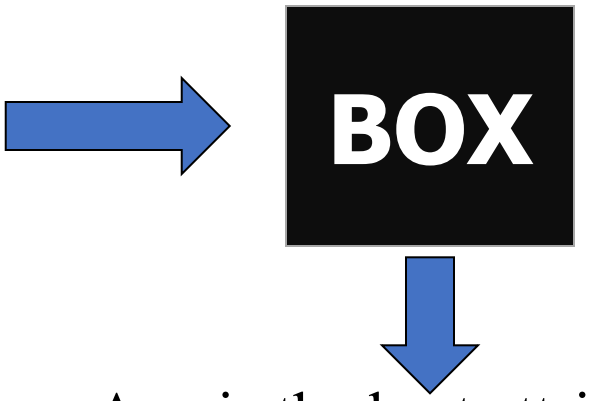
- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Training examples are partitioned recursively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

What is in the box?

Data table

Class Attribute

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 is the best attribute to create a node in the tree.

The chosen attribute should carry **more information than** the others w.r.t. the **class attribute**.

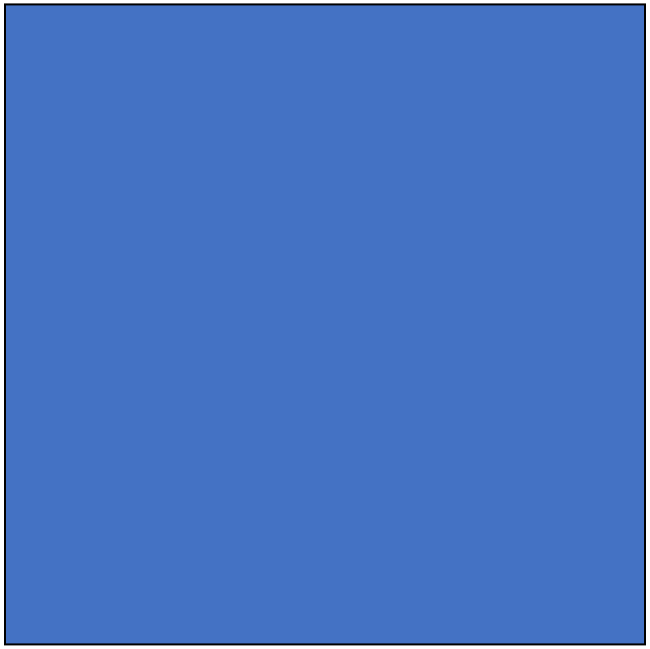
Then how can we measure information?

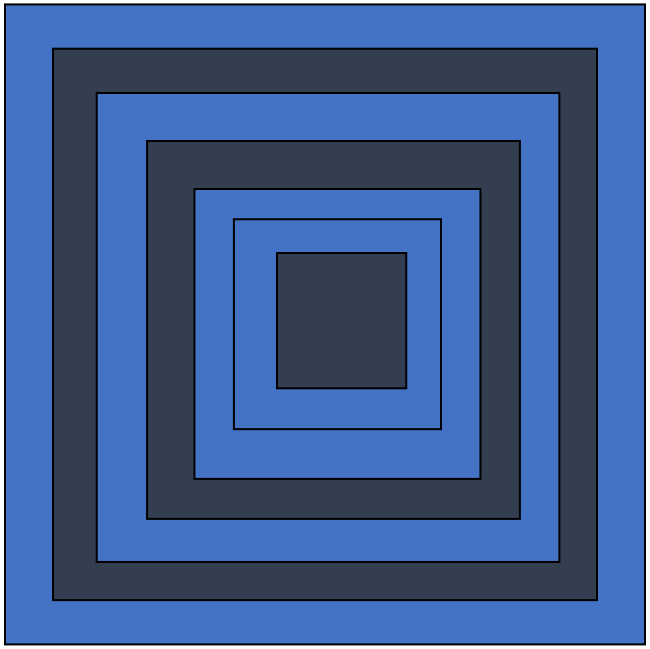
Entropy – a measure of disorder

Which one has higher entropy?

Source: http://www.organizingfabulously.com/blog/2015/03/02/messy-vs-tidy-desks

Which one has higher entropy?





Which one has higher entropy?

buys_computer
yes
no
yes
yes
no
no
yes
yes
yes
no
no
no
yes
yes

buys_computer
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes

Which one has higher entropy?

buys_computer
yes
no
yes
yes
no
no
yes
yes
yes
no
no
no
yes
yes

buys_computer
no
no
no
yes
yes
yes
yes
yes
yes
yes
yes
no
yes
no

We need a quantitative measure of information (i.e. disorder).

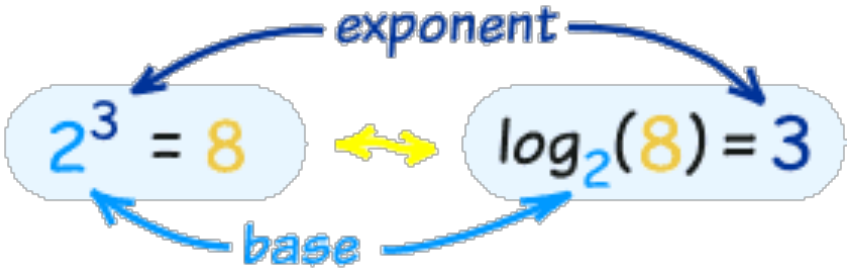
Logarithm

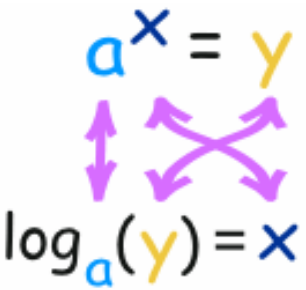
How many of *one number* do we multiply to get *another number?*

Example: How many 2s do we multiply to get 8? $2^{?} = 8$

Answer: **2 × 2 × 2 = 8**, so we needed to multiply 3 of the **2**s to get **8**

So the logarithm to base 2 of 8 is 3





$$10^{?} = 10000$$

$$\log_{10}(10000) = 4 = \log(10000)$$

$$\log_2(32) = \frac{\log(32)}{\log(2)} = 5$$

$$log_x(y) = \frac{log_{10}(y)}{log_{10}(x)} = \frac{log\,(y)}{log(x)}$$

$$log(1) = 0$$

$log(0)$ non-existed

Entropy – a measure of disorder (impurity in class attribute)

Information needed (i.e. entropy) to classify a random record in D

m is all the unique values for the *class attribute*

The probability of seeing *i*

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Loop for each unique value *i*

Binominal class attribute: '*Yes*' or '*No*'

Let:

- *x denotes the count of 'Yes'*
- *y denotes the count of 'No'*

$$p('yes') = \frac{x}{x+y} \qquad p('no') = \frac{y}{x+y}$$

$$Info(D) = -[p('yes')\ log_2(p('yes')) + p('no')\ log_2(p('no'))]$$

$$Info(D) = I(x,y) = -\frac{x}{x+y}\ log_2\left(\frac{x}{x+y}\right) - \frac{y}{x+y} log_2\left(\frac{y}{x+y}\right)$$

Entropy – a measure of disorder (impurity in class attribute)

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info(D) = I(x, y)$$
$$= -\frac{x}{x+y} log_2\left(\frac{x}{x+y}\right) - \frac{y}{x+y} log_2\left(\frac{y}{x+y}\right)$$

buys_computer
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes
yes

buys_computer
no
no
no
no
no
no
no
no
no
no
no
no
no
no

$$x = 14, y = 0$$

$$Info(D) = I(14,0) = -\frac{14}{14}\log_2(\frac{14}{14}) - \frac{0}{14}\log_2(\frac{0}{14}) = -1 \times 0 - 0 \times 0 = 0$$

Entropy – a measure of disorder (impurity in class attribute)

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info(D) = I(x,y)$$
$$= -\frac{x}{x+y} log_2\left(\frac{x}{x+y}\right) - \frac{y}{x+y} log_2\left(\frac{y}{x+y}\right)$$

$$Info(D) = I(7,7) = -\frac{7}{14}\log_2(\frac{7}{14}) - \frac{7}{14}\log_2(\frac{7}{14})$$
$$= (-0.5 \times \log_2 0.5) - (0.5 \times \log_2 0.5) = (-0.5 \times -1) - (0.5 \times -1) = 1$$

buys_computer
yes
no
yes
yes
no
no
yes
yes
yes
no
no
no
yes
no

buys_computer
no
no
no
yes
yes
yes
yes
yes
yes
yes
yes
no
yes
no

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14})$$

$$= (-0.643 \times \log_2 0.643) - (0.357 \times \log_2 0.357)$$
$$= (-0.643 \times -0.637) - (0.357 \times -1.485)$$
$$= 0.410 - (-0.531) = 0.940$$

Conditional Entropy (Weighted Average)

age	…	buys_computer
<=30	…	no
<=30	…	no
<=30	…	no
<=30	…	yes
<=30	…	yes

age	…	buys_computer
31…40	…	yes
31…40	…	yes
31…40	…	yes
31…40	…	yes

age	…	buys_computer
>40	…	no
>40	…	no
>40	…	no
>40	…	yes
>40	…	yes

v is all the unique values for the conditional attribute/feature *A*

Size of the table of value *j*

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

Information for the table of value *j*

Loop through each unique value for the conditional attribute value *j*

Total size of all tables

Conditional Entropy (Weighted Average)

age	…	buys_computer
<=30	…	no
<=30	…	no
<=30	…	no
<=30	…	yes
<=30	…	yes

age	…	buys_computer
31…40	…	yes
31…40	…	yes
31…40	…	yes
31…40	…	yes

age	…	buys_computer
>40	…	no
>40	…	no
>40	…	no
>40	…	yes
>40	…	yes

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

age	yes	no	total	I(yes,no)
<=30	2	3	5	I(2,3)=0.971
31…40	4	0	4	I(4,0)=0
>40	3	2	5	I(2,3)=0.971

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

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

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

Information Gain

age	…	buys_computer
<=30	…	no
<=30	…	no
<=30	…	no
<=30	…	yes
<=30	…	yes

age	…	buys_computer
31…40	…	yes
31…40	…	yes
31…40	…	yes
31…40	…	yes

age	…	buys_computer
>40	…	no
>40	…	no
>40	…	no
>40	…	yes
>40	…	yes

Entropy (information need) before splitting

Entropy (information needed) after splitting with attribute *A*

$$Gain(A) = Info(D) - Info_A(D)$$

The amount of reduced entropy if we split on attribute *A*.

i.e. the amount of information we can gain from attribute age w.r.t. buys_computers.

age	…	buys_computer
<=30	…	no
<=30	…	no
<=30	…	no
<=30	…	yes
<=30	…	yes
31…40	…	yes
31…40	…	yes
31…40	…	yes
31…40	…	yes
>40	…	yes
>40	…	yes
>40	…	no
>40	…	yes
>40	…	no

Information Gain

age	yes	no	total	I(yes,no)
Any	9	5	14	0.94
<=30	2	3	5	0.971
31…40	4	0	4	0
>40	3	2		0.971

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info(D) = I(x,y) = -\frac{x}{x+y}\log_2\left(\frac{x}{x+y}\right) - \frac{y}{x+y}\log_2\left(\frac{y}{x+y}\right)$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$

$$= 0.410 - (-0.531) = 0.940$$

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

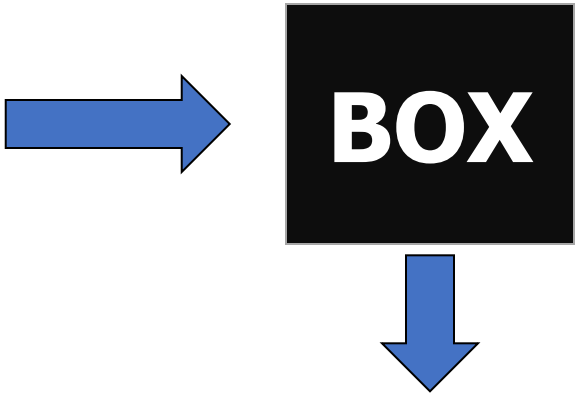
$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

What is in the box?

Data table

Class Attribute

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



- Calculate information gain for each feature.
- Pick the feature that has highest information gain.

A Decision Tree for "*buys_computer*" (1st level)

$$I(x,y) = -\frac{x}{x+y}\log_2(\frac{x}{x+y}) - \frac{y}{x+y}\log_2(\frac{y}{x+y})$$

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

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

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

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

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

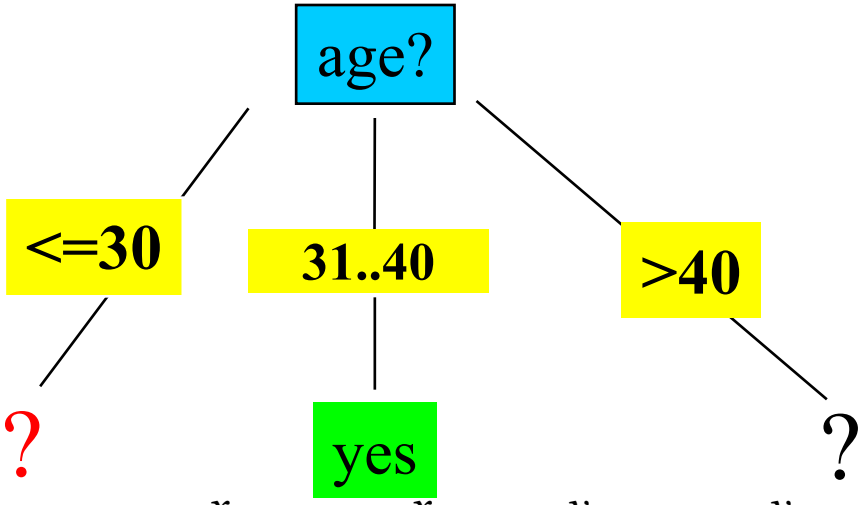
age	yes	no	I(yes,no)
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

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

A Decision Tree for "*buys_computer*" (2nd level)



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

$$I(x,y) = -\frac{x}{x+y}\log_2(\frac{x}{x+y}) - \frac{y}{x+y}\log_2(\frac{y}{x+y})$$

$$Info(D[age <= 30])$$

$$= I(2,3) = -\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5}) = 0.971$$

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

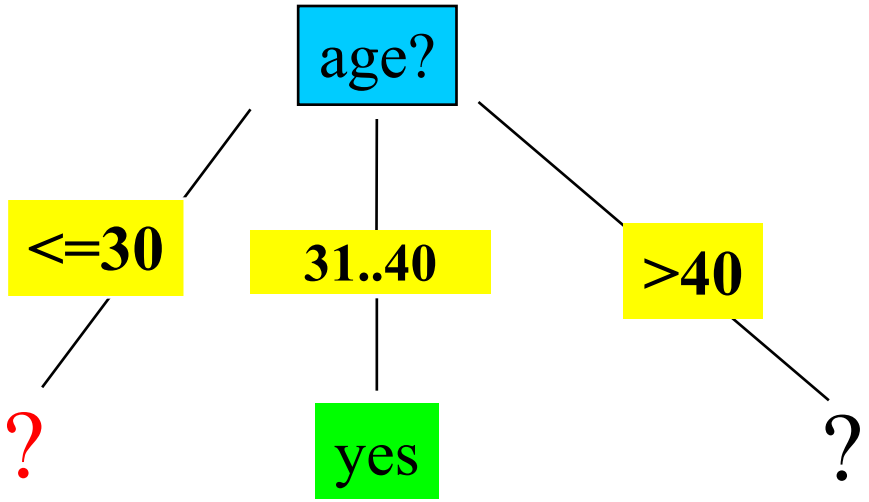
$$Info_{income}(D[age <= 30])$$

$$= \frac{1}{5} I(1,0) + \frac{2}{5} I(1,1) + \frac{2}{5} I(0,2)$$

$$= 0 + 0.4 + 0 = 0.4$$

income	yes	no	I(yes,no)
low	1	0	0
medium	1	1	1
high	0	2	0

A Decision Tree for "*buys_computer*" (2nd level)



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

$$Info_{student}(D[age <= 30]) = 0$$

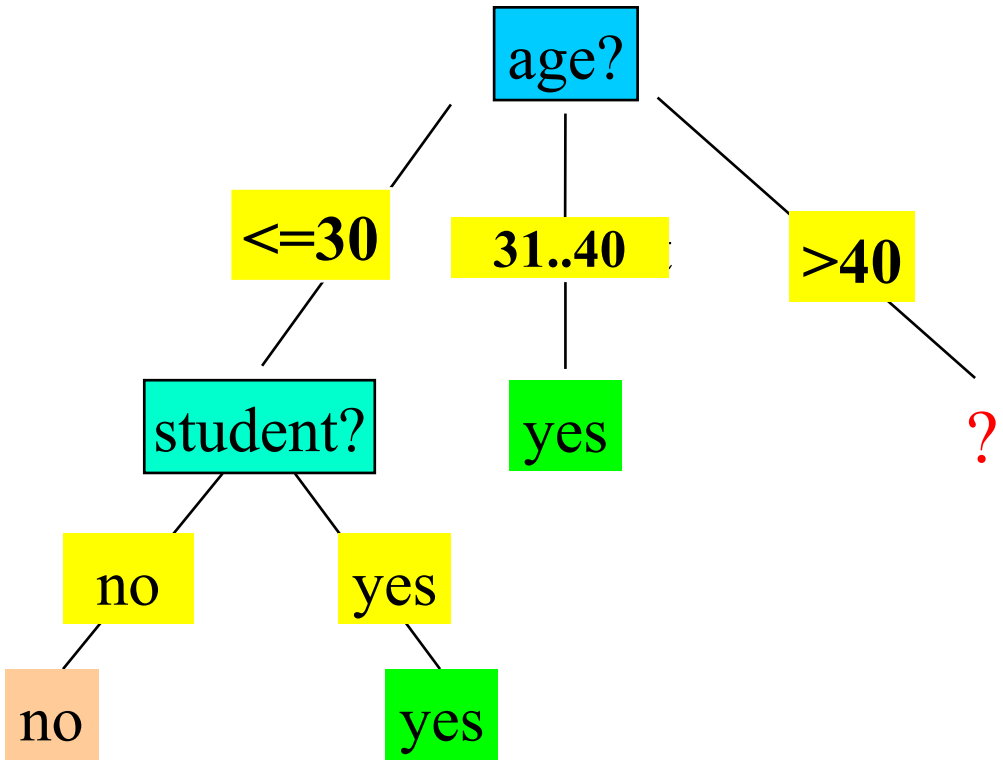
$$Info_{credit\_rating}(D[age <= 30])$$

$$= \frac{3}{5} I(1,2) + \frac{2}{5} I(1,1) = \frac{3}{5} \times 0.918 + \frac{2}{5} \times 1 = 0.951$$

student	yes	no	I(yes,no)
yes	2	0	0
no	0	3	0

credit_rating	yes	no	I(yes,no)
fair	1	2	0.918
excellen	1	1	1

A Decision Tree for "*buys_computer*" (2nd level)



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

$$Info_{income}(D[age <= 30]) = 0.4$$

$$Info_{student}(D[age <= 30]) = 0$$

$$Info_{credit\_rating}(D[age <= 30]) = 0.951$$

$$Gain(income) = 0.971 - 0.4 = 0.571$$

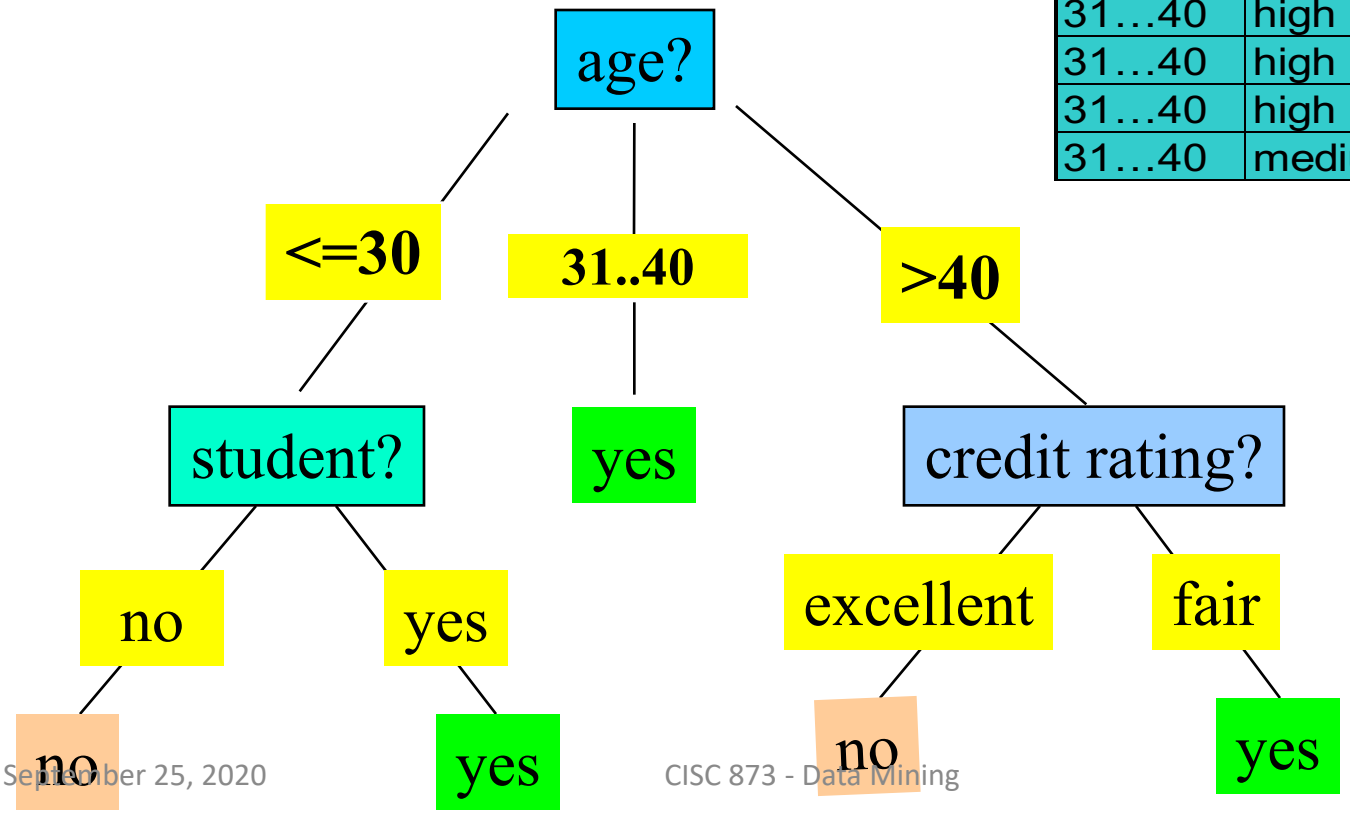
$$Gain(student) = 0.971 - 0 = 0.971$$

$$Gain(credit\_rating) = 0.971 - 0.951 = 0.02$$

Overfitting and Tree Pruning

- 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

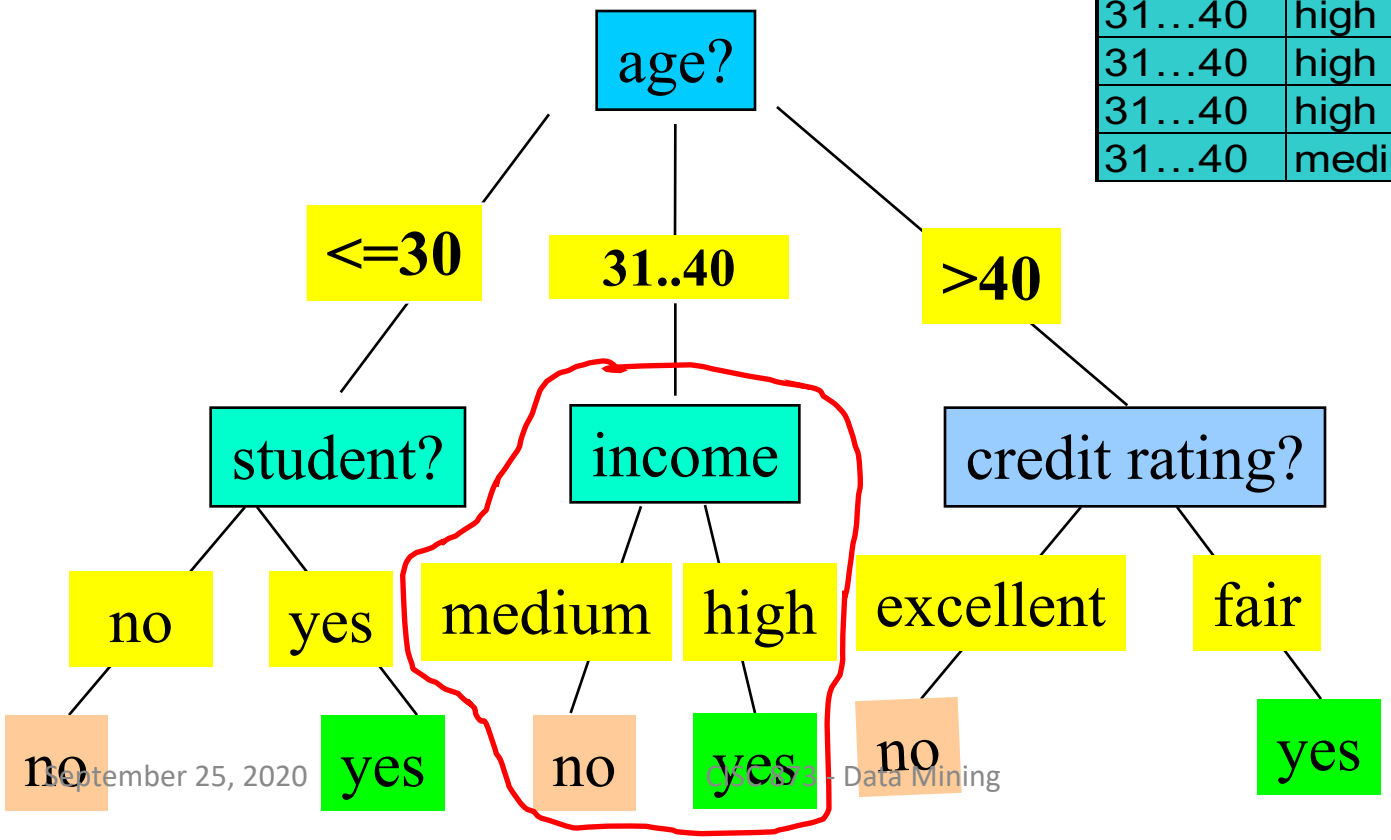
age	income	class
31…40	high	yes
31…40	high	yes
31…40	high	yes
31…40	medium	yes



Overfitting and Tree Pruning

- 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

age	income	class
31…40	high	yes
31…40	high	yes
31…40	high	yes
31…40	medium	no



Overfitting and Tree Pruning

- 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"