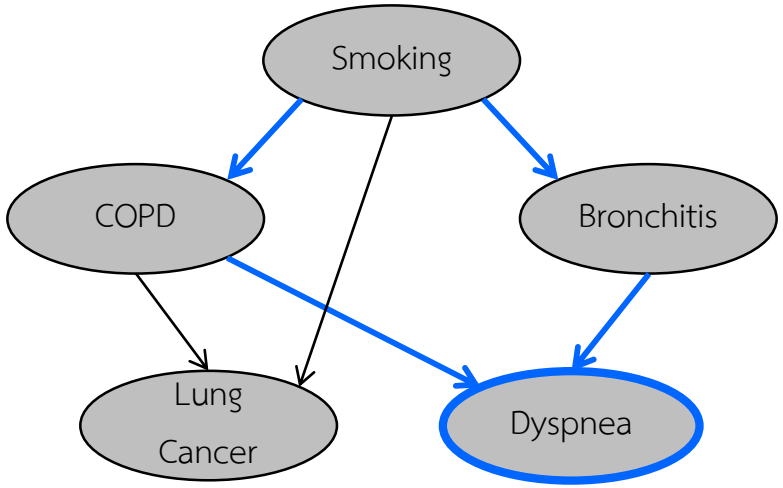
Mahidol University
Faculty of Medicine Ramathibodi Hospital
Section for Clinical Epidemiology and Biostatistics

Bayesian Network

Ratchainant Thammasudjarit, Ph.D.

What is Bayesian Network

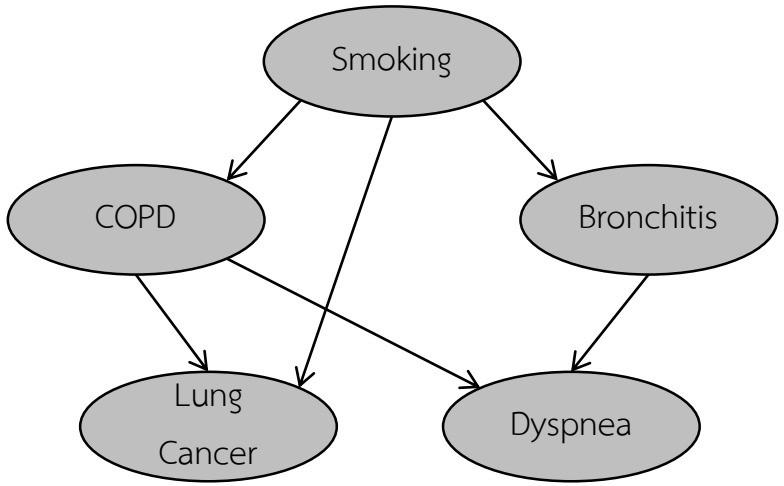
DAG



- Directed Acyclic Graph (D.A.G.)
- Each node is assumed conditional independence of its non-descendants, given its parents
  - P(Dyspnea | COPD, Bronchitis, Smoking)

Applications of Bayesian Network

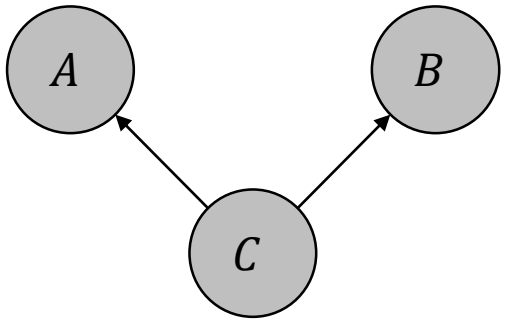
Medical Diagnostic

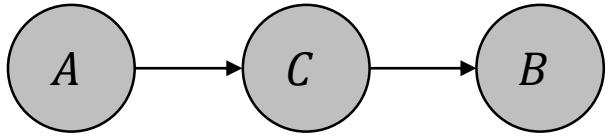


- Bayesian Network allows to query the system with partial observations
  - What is the probability to convert from predm to dm of a male patient with BMI 30
- Network structure
  - Define by expert
  - Learning from data

Conditional Independence

Definition



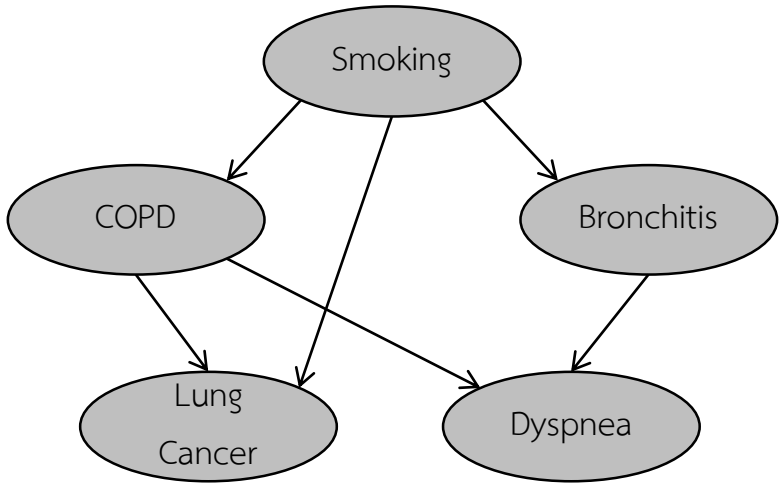


- Two events A and B are conditionally independent given an event C with $P(C) > 0$ if and only if

$$P(A \cap B|C) = P(A|C)P(B|C)$$

D-Separation

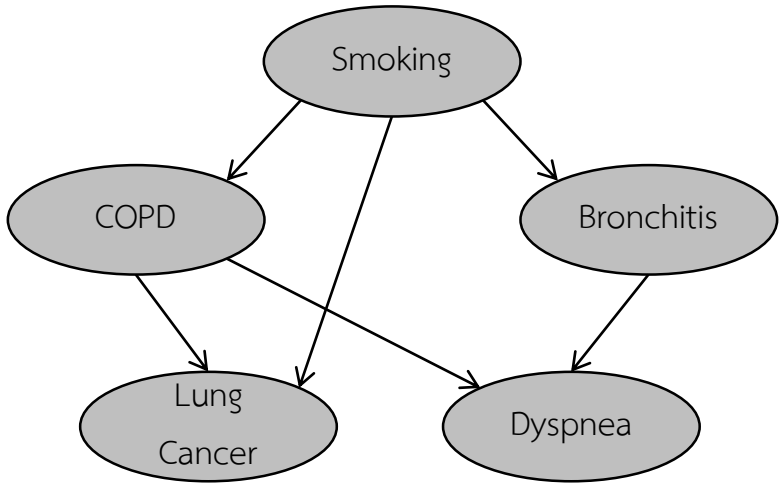
Conditional Independence for set of nodes



- D-separation determines whether a set of nodes X is independent of another set of nodes Y, given a set of evidence nodes E
- E makes d-separation on X and Y if and only if every undirected path from a node x in X to a node y in Y is blocked given E

D-Separation

Conditional Independence for set of nodes

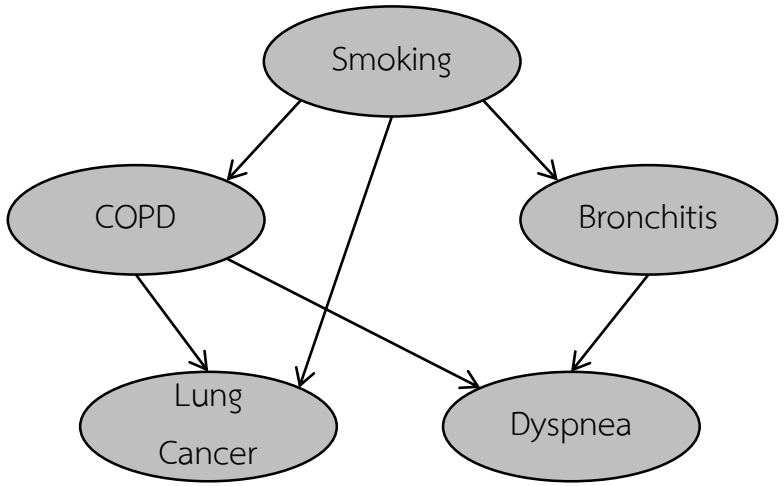


- Given this network and event E
- E = {}
- E = {COPD}

- Which sets of nodes are d-separated?

D-Separation

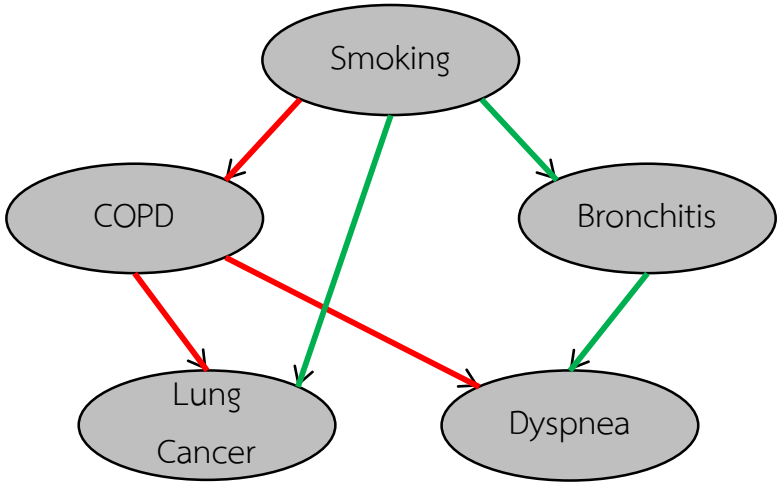
Conditional Independence for set of nodes



- E = {}
  - None of sets are blocked by E

D-Separation

Conditional Independence for set of nodes



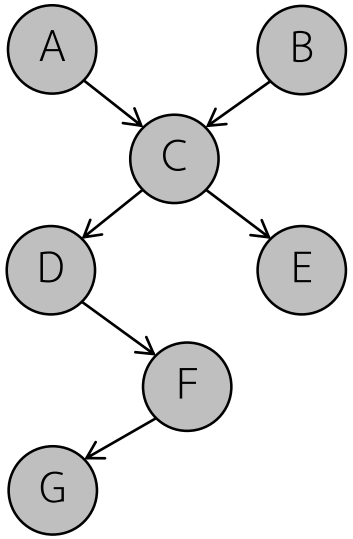
- Bronchitis to Lung Cancer given E = {COPD}
- COPD does not d-separate Bronchitis and Lung Cancer

Green lines are the undirected paths unblocked paths by E

Red lines are the undirected paths blocked paths by E

D-Separate in Complex Network

Algorithm



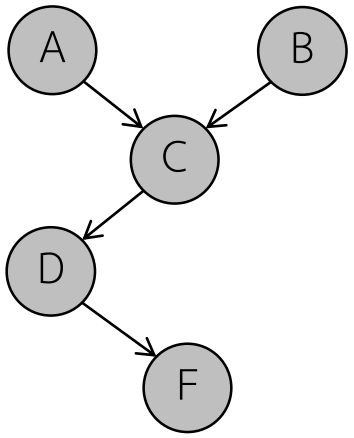
- Are A and B are d-separated given D and F?

$$P(A|BDF) = P(A|DF)$$
$$P(B|ADF) = P(B|DF)$$

- Solution requires 4 steps

D-Separate in Complex Network

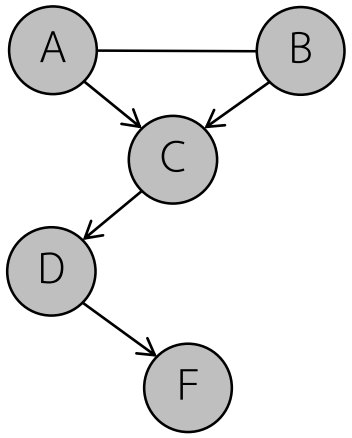
Algorithm



- Step 1 (Ancestral): Draw ancestral graph of all variables mentioned in probability expression

D-Separate in Complex Network

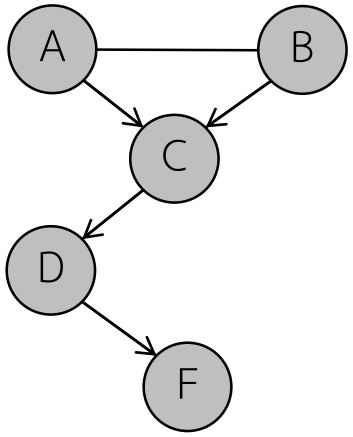
Algorithm



- Step 2 (Moralize): For each pair of variables with common child, draw an undirected edge between them

D-Separate in Complex Network

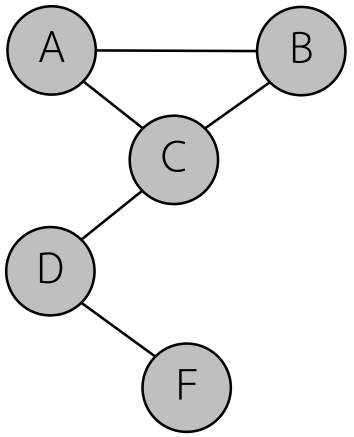
Algorithm



- Step 3 (Disorient): Replace directed edges with undirected edges

D-Separate in Complex Network

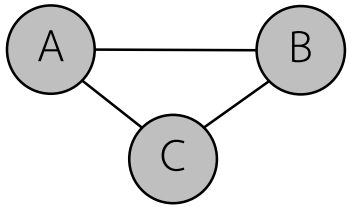
Algorithm



- Step 4 (Delete givens): Delete the givens and their edges

D-Separate in Complex Network

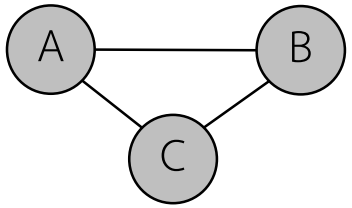
Algorithm



- Interpretation
  - If the variables are disconnected in this graph, they are guaranteed to be independent
  - If the variables are connected in this graph, they are not guaranteed to be independent
  - If one or both of the variables are missing (because they were givens, and were therefore deleted), they are independent

D-Separate in Complex Network

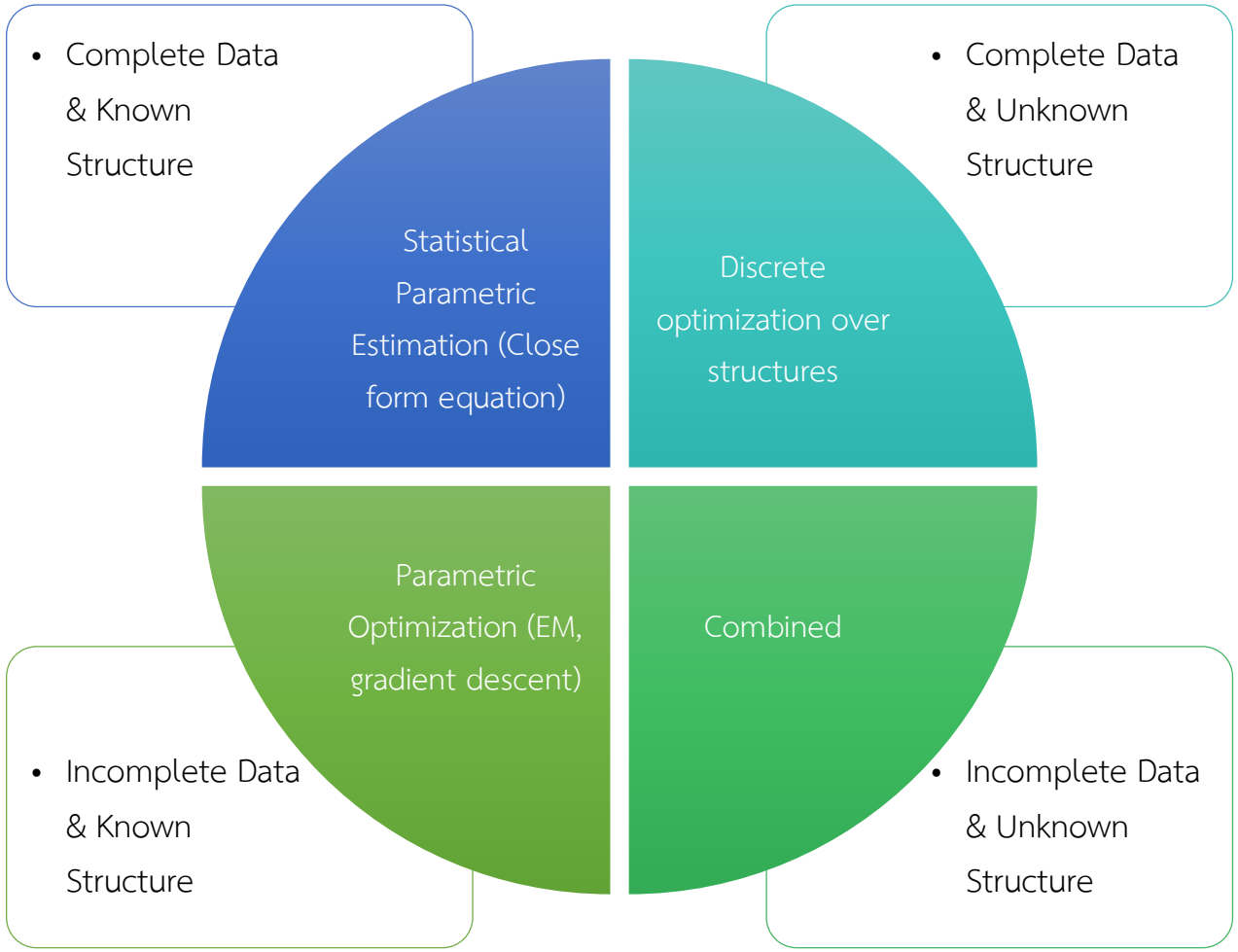
Algorithm



- Conclusions
- A and B are connected, they are not required to be d-separated given D and F

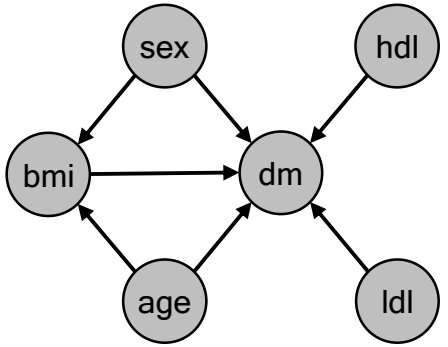
Bayesian Network Learning

Choices



Your First Bayesian Network

Data




```
import pandas as pd
import predm_utils
```

```
df = pd.read_csv("./data/simpredm.csv", usecols=["age", "hdl", "ldl", "bmi", "sex", "dm"])
```

```
df_os = predm_utils.universal_smote(df=df,
                                    target="dm",
                                    cat_vars=["sex"])
```

```
no downsampling
yes oversampling
```

```
df_os["dm"] = df_os["dm"].map({"yes": 1, "no": 0})
df_os["sex"] = df_os["sex"].map({"Male": 1, "Female": 0})
df_os["age"] = df_os["age"].apply(lambda x: predm_utils.age_discretize(x))
df_os["bmi"] = df_os["bmi"].apply(lambda x: predm_utils.bmi_discretize(x))
df_os["hdl"] = df_os.apply(lambda x: predm_utils.hdl_discretize(x["hdl"], x["sex"]), axis=1)
df_os["ldl"] = df_os.apply(lambda x: predm_utils.ldl_discretize(x["ldl"], x["sex"]), axis=1)
```

- Read data from the file simpredm.csv
- Since data is highly imbalanced, apply oversampling

Your First Bayesian Network

Data Splitting

- Use sklearn

```
X, y = predm_utils.init(df=df_os, target="dm")
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, stratify=y)
```

Your First Bayesian Network

Reconstruct numpy array to dataframe

- pgmpy requires dataframe as input

```
import numpy as np
```

```
trn = pd.DataFrame(np.hstack((X_train, y_train.reshape(len(y_train), 1))), columns=["age", "hdl", "ldl", "bmi", "sex", "dm"])
tst = pd.DataFrame(np.hstack((X_test, y_test.reshape(len(y_test), 1))), columns=["age", "hdl", "ldl", "bmi", "sex", "dm"])

for col in trn.columns:
    trn[col] = trn[col].astype(int)
    tst[col] = tst[col].astype(int)
```

Your First Bayesian Network

Learning

- Use pgmpy

```
from pgmpy.models import BayesianModel
```

```
bayes = BayesianModel([("age", "bmi"), ("age", "dm"), ("sex", "bmi"), ("sex", "dm"),
                       ("bmi", "dm"), ("hdl", "dm"), ("ldl", "dm")])
bayes.fit(trn)
```

Your First Bayesian Network

Evaluation

```
prob_dm = bayes.predict_probability(tst.drop("dm", axis=1)).values[:, 1]
```

```
yhat = bayes.predict(tst.drop("dm", axis=1)).values
```

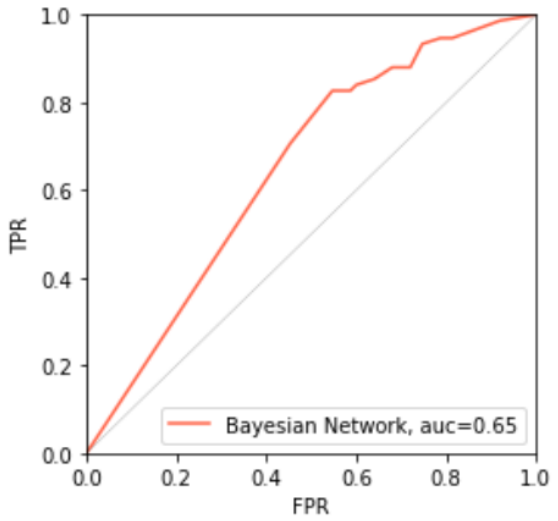
```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

import matplotlib.pyplot as plt
```

```
roc = roc_curve(y_test, prob_dm, pos_label=1)
auc = round(roc_auc_score(y_test, prob_dm), 2)
```

```
fig, ax = plt.subplots(figsize=(4, 4))
ax.plot(roc[0], roc[1], color="tomato", label="Bayesian Network, auc="+str(auc))
ax.set_xlabel("FPR")
ax.set_ylabel("TPR")
ax.set_xlim(left=0, right=1)
ax.set_ylim(bottom=0, top=1)
ax.plot(ax.get_ylim(), ax.get_xlim(), color="gray", linewidth=0.3)
plt.legend(loc=4)
```

```
<matplotlib.legend.Legend at 0x1b517242780>
```



Your First Bayesian Network

Prediction

```
unseen = pd.DataFrame({"age": [0, 1, 2], "hdl": [0, 0, 1], "ldl": [1, 0, 1], "bmi": [2, 1, 0], "sex": [1, 1, 0]})
```

```
bayes.predict(unseen)
```

	dm
0	0
1	0
2	0