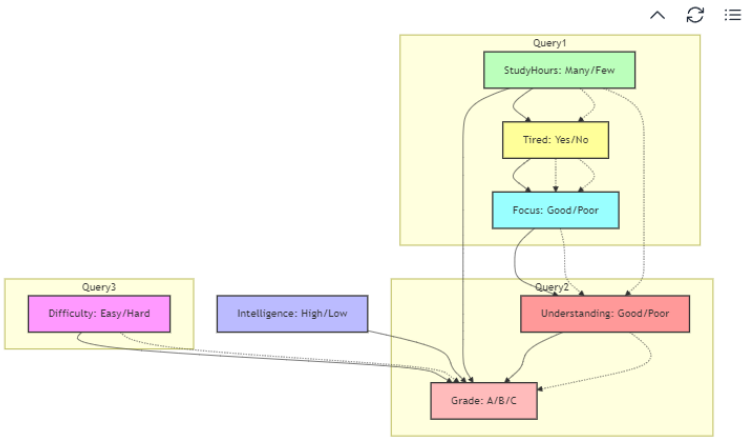
Bayesian Net

M5281054 WANG Binghao

This complete Bayesian network model describes the student's learning status:

1. Node design (7 nodes):

- Difficulty：Easy/Hard
- Intelligence：High/Low
- StudyHours：Many/Few
- Grade：A/B/C
- Tired：Yes/No
- Focus：Good/Poor
- Understanding：Good/Poor

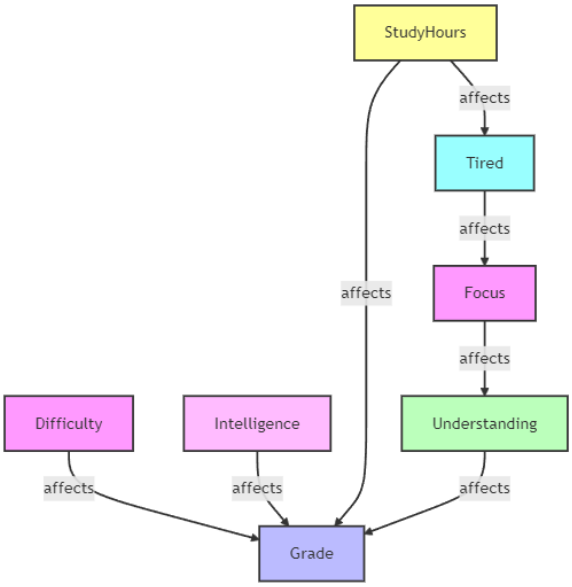


Bayesian network construction diagram

2.Dependencies:

- The exam results (Grade) depend on:
  - Difficulty
  - Intelligence
  - StudyHours
  - Understanding
- Tiredness depends on study hours

- Focus depends on Tiredness
- Understanding depends on focus



Dependency diagram

Each node in a Bayesian network defines a joint probability distribution

Prior probability (root node):

- The sum must equal 1
- Set reasonable values based on actual conditions

```
# Difficulty的先验概率
difficulty_cpt = {
    'Easy': 0.6,
    'Hard': 0.4
}

# Intelligence的先验概率
intelligence_cpt = {
    'High': 0.3,
    'Low': 0.7
}

# StudyHours的先验概率
study_hours_cpt = {
    'Many': 0.4,
    'Few': 0.6
}
```

Prior probability of the root node (no parent node)

Conditional probability (with parent node):

- For each combination of parent node states, the sum of the probabilities of the child nodes is 1

- Reflect the actual dependencies between variables

```
# Tired依赖于StudyHours的条件概率
tired_cpt = {
    ('Many', 'Yes'): 0.8,  # P(Tired=Yes|StudyHours=Many)
    ('Many', 'No'): 0.2,   # P(Tired=No|StudyHours=Many)
    ('Few', 'Yes'): 0.2,   # P(Tired=Yes|StudyHours=Few)
    ('Few', 'No'): 0.8     # P(Tired=No|StudyHours=Few)
}

# Focus依赖于Tired的条件概率
focus_cpt = {
    ('Yes', 'Good'): 0.3,  # P(Focus=Good|Tired=Yes)
    ('Yes', 'Poor'): 0.7,  # P(Focus=Poor|Tired=Yes)
    ('No', 'Good'): 0.9,   # P(Focus=Good|Tired=No)
    ('No', 'Poor'): 0.1    # P(Focus=Poor|Tired=No)
}

# Understanding依赖于Focus的条件概率
understanding_cpt = {
    ('Good', 'Good'): 0.8,  # P(Understanding=Good|Focus=Good)
    ('Good', 'Poor'): 0.2,  # P(Understanding=Poor|Focus=Good)
    ('Poor', 'Good'): 0.3,  # P(Understanding=Good|Focus=Poor)
    ('Poor', 'Poor'): 0.7   # P(Understanding=Poor|Focus=Poor)
}
```

Conditional probability of having a single parent

Processing of multiple parent nodes:

- Need to consider the combination of all parent node states
- The probability distribution of each combination should be reasonable

```
# Grade依赖于Difficulty, Intelligence, StudyHours, Understanding的条件概率
grade_cpt = {
    # 格式: (Difficulty, Intelligence, StudyHours, Understanding, Grade值)
    ('Easy', 'High', 'Many', 'Good', 'A'): 0.9,
    ('Easy', 'High', 'Many', 'Good', 'B'): 0.08,
    ('Easy', 'High', 'Many', 'Good', 'C'): 0.02,

    ('Easy', 'High', 'Many', 'Poor', 'A'): 0.7,
    ('Easy', 'High', 'Many', 'Poor', 'B'): 0.2,
    ('Easy', 'High', 'Many', 'Poor', 'C'): 0.1,

    # ... 其他组合的概率

    ('Hard', 'Low', 'Few', 'Poor', 'A'): 0.1,
    ('Hard', 'Low', 'Few', 'Poor', 'B'): 0.3,
    ('Hard', 'Low', 'Few', 'Poor', 'C'): 0.6
}
```

Conditional probability of Grade node (multiple parent nodes)

3. Provide three query examples:

- Query 1: Given learning time and attention state, calculate the probability of fatigue
- Query 2: Given fatigue and understanding, calculate the probability of getting an A
- Query 3: Given the course difficulty and study time, calculate the probability of good understanding

```
Query 1: Probability of being tired when studying many hours
P(Tired=Yes | StudyHours=Many) = 0.800
P(Tired=No | StudyHours=Many) = 0.200

Query 2: Probability of understanding given good focus
P(Understanding=Good | Focus=Good) = 0.800
P(Understanding=Poor | Focus=Good) = 0.200

Query 3: Probability of good focus given few study hours
P(Focus=Good | StudyHours=Few) = 0.780
```

4. Calculation process:

```
Query 1: P(Tired | StudyHours=Many, Focus=Good)

Calculation Steps:
1. Using Bayes' Theorem:
P(Tired | StudyHours, Focus) = P(Focus | Tired) * P(Tired | StudyHours) *
P(StudyHours) / P(Focus, StudyHours)

2. Calculate probabilities for both Tired = Yes and Tired = No:

For Tired = Yes:
P(Focus=Good | Tired=Yes) = 0.3
P(Tired=Yes | StudyHours=Many) = 0.8
P(StudyHours=Many) = 0.4

For Tired = No:
P(Focus=Good | Tired=No) = 0.9
P(Tired=No | StudyHours=Many) = 0.2
P(StudyHours=Many) = 0.4

3. Normalize the results
```

```python
def calculate_query1(study_hours='Many', focus='Good'):
    # Prior probabilities
    p_study_hours = 0.4 if study_hours == 'Many' else 0.6

    # Calculate for Tired=Yes
    p_focus_given_tired_yes = 0.3 if focus == 'Good' else 0.7
    p_tired_yes_given_study = 0.8 if study_hours == 'Many' else 0.2
    prob_yes = p_focus_given_tired_yes * p_tired_yes_given_study * p_study_hours

    # Calculate for Tired=No
    p_focus_given_tired_no = 0.9 if focus == 'Good' else 0.1
    p_tired_no_given_study = 0.2 if study_hours == 'Many' else 0.8
    prob_no = p_focus_given_tired_no * p_tired_no_given_study * p_study_hours

    # Normalize
    total = prob_yes + prob_no
    return prob_yes/total, prob_no/total
```

Query 2: P(Grade=A | Tired=Yes, Understanding=Poor)

Calculation Steps:

1. Using Total Probability Formula and Bayes' Theorem:

P(Grade=A | Tired, Understanding) =

Σ(Difficulty, Intelligence, StudyHours) P(Grade=A | Difficulty, Intelligence, StudyHours, Understanding) *

P(Difficulty) * P(Intelligence) * P(StudyHours | Tired)

2. Consider all possible combinations

```python
def calculate_query2(tired='Yes', understanding='Poor'):
    # Initialize probability
    p_grade_a = 0

    # Iterate through all possible combinations
    for difficulty in ['Easy', 'Hard']:
        for intelligence in ['High', 'Low']:
            for study_hours in ['Many', 'Few']:
                # Get individual probabilities
                p_difficulty = 0.6 if difficulty == 'Easy' else 0.4
                p_intelligence = 0.3 if intelligence == 'High' else 0.7

                # Calculate P(StudyHours | Tired)
                if tired == 'Yes':
                    p_study = 0.8 if study_hours == 'Many' else 0.2
                else:
                    p_study = 0.2 if study_hours == 'Many' else 0.8

                # Get P(Grade=A) for this combination
                grade_prob = grade_cpt.get((difficulty, intelligence,
                                            study_hours, understanding, 'A'), 0)

                # Add to total probability
                p_grade_a += grade_prob * p_difficulty * p_intelligence * p_study

    return p_grade_a
```

Query 3: P(Understanding=Good | Difficulty=Easy, StudyHours=Many)

Calculation Steps:

1. Using Chain Rule of Conditional Probability:

P(Understanding | Difficulty, StudyHours) =

Σ(Focus) P(Understanding | Focus) * P(Focus | Tired) * P(Tired | StudyHours)

2. Calculate each conditional probability step by step

```python
def calculate_query3(difficulty='Easy', study_hours='Many'):
    p_understanding_good = 0

    # Calculate for each possible Focus state
    for focus in ['Good', 'Poor']:
        # Calculate P(Focus) through Tired
        p_focus = 0
        for tired in ['Yes', 'No']:
            p_tired = 0.8 if study_hours == 'Many' else 0.2  # P(Tired|StudyHours)
            p_focus_given_tired = (0.3 if tired == 'Yes' else 0.9) if focus == 'Good' else (0.7 if tired == 'Yes' else 0.1)
            p_focus += p_focus_given_tired * p_tired

        # Get P(Understanding|Focus)
        p_understanding_given_focus = 0.8 if focus == 'Good' else 0.3

        # Add to total probability
        p_understanding_good += p_understanding_given_focus * p_focus

    return p_understanding_good
```