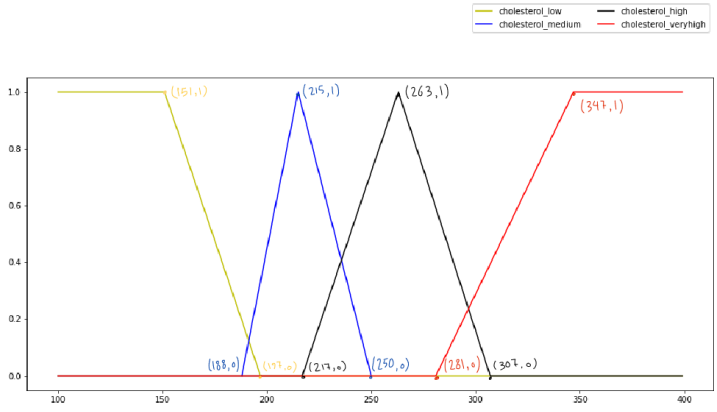
گزارشکار پروژه پیاده سازی سیستم فازی خبره برای تشخیص بیماری قلبی

محدثه سادات اطیابی ۹۷۲۶۰۰۴

در این پروژه ۴ فایل fuzzification.py، inference.py، defuzzification.py و final_result.py را تغییر داده این و در ادامه به بررسی موارد موجود در این فایل ها می پردازیم.

۱) fuzzification.py:

در این فایل باید بر اساس نمودارهای داده شده در صورت پروژه، معادلات خط برای بدست آوردن وضعیت های مختلف هر پارامتر را بدست آوریم. مثلا طبق تصویر زیر، کلسترول دارای ۴ مقدار low, medium, high و very high است. پس در کلاس کلسترول برای هر مقدار یک تابع می نویسیم و بر اساس بازه های داده شده و مقادیر تعلق، معادلات آن را می نویسیم تا مقدار تعلق در هر بازه را بدست آوریم. سپس یک تابع بنام fuzzy_cholesterol می سازیم که مقادیر تعلق بدست آمده را بصورت dictionary در بیاورد و آن را باز گرداند. کد این قسمت بصورت زیر است:



نمودار مقادیر تعلق کلسترول

```python
class fuzzification_cholesterol:
    def __init__(self):
        pass

    def low_cholesterol(self, x):
        if 0 <= x <= 151:
            return 1.0
        elif 151 < x <= 197:
            return (-1.0 / 46) * x + 197 / 46.0
        else:
            return 0.0

    def medium_cholesterol(self, x):
        if 188 <= x <= 215:
            return (1.0 / 27) * x - 188 / 27.0
        elif 215 < x <= 250:
            return (-1.0 / 35) * x + 250 / 35.0
        else:
            return 0.0
```

```
def high_cholesterol(self, x):
    if 217 <= x <= 263:
        return (1.0 / 46) * x - 217 / 46.0
    elif 263 < x <= 307:
        return (-1.0 / 44) * x + 307 / 44.0
    else:
        return 0.0

def veryHigh_cholesterol(self, x):
    if 0 <= x <= 281:
        return 0.0
    elif 281 < x <= 347:
        return (1.0 / 66) * x - 281 / 66.0
    else:
        return 1.0

def fuzzy_cholesterol(self, x):
    return {'low': self.low_cholesterol(x), 'medium': self.medium_cholesterol(x),
            'high': self.high_cholesterol(x), 'very_high': self.veryHigh_cholesterol(x)}
```

به همین ترتیب برای تمام ورودی هایی که نمودار برایشان داریم، عمل می کنیم و توابع آنها را پیاده سازی می کنیم. برای ورودی هایی که دارای مقادیر crisp هستند، با توجه به خروجی هایی که باید داشته باشند، برایشان یک تابع می نویسیم و براساس شرایط مطرح شده مقادیر تعلق آنها را تنظیم می کنیم. به عنوان نمونه تالیوم را در نظر بگیرید که با توجه به اینکه مقدار آن ۳ یا ۶ یا ۷ باشد یکی از مقادیر normal, medium و high را می گیرد و پیاده سازی آن بصورت زیر است:

```
class fuzzification_thallium:
    def __init__(self):
        pass

    def normal_thallium(self, x):
        if x == 3:
            return 1.0
        else:
            return 0.0

    def medium_thallium(self, x):
        if x == 6:
            return 1.0
        else:
            return 0.0

    def high_thallium(self, x):
        if x == 7:
            return 1.0
        else:
            return 0.0

    def fuzzy_thallium(self, x):
        return {'normal': self.normal_thallium(x), 'medium': self.medium_thallium(x), 'high': self.high_thallium(x)}
```

به همین ترتیب برای باقی ورودی هایی که دارای مقادیر crisp هستند، توابع مورد نیاز را پیاده سازی می کنیم. نمودار خروجی را نیز مانند موارد قبلی پیاده سازی می کنیم با این تفاوت که نیازی نیست در انتها یک dictionary برای نتایج درست کنیم.

۲) inference.py:

در اینجا باید با استفاده از قوانینی که در فایل rules.fcl آمده است، از داده های ورودی که در مرحله قبل فازی سازی کرده ایم نتیجه گیری و استنتاج داشته باشیم. برای این کار لازم است تا قوانین را برای سیستم تعریف کنیم. در ابتدا برای هرکدام از خروجی ها یعنی sick1, sick2, sick3, sick4, healthy یک آرایه در نظر می گیریم که در ابتدا خالی است. سپس بر اساس قوانین آنها را پر می کنیم. برای تفسیر قوانین، and را با min و or را با max جایگذاری می کنیم. به عنوان مثال قانون اول بصورت زیر است:

```
RULE 1: IF (age IS very_old) AND (chest_pain IS atypical_anginal) THEN health
IS sick_4;
```

پیاده سازی آن در کد بصورت زیر است:

```
output_sick4.append(min(age['very_old'], chest_pain['atypical_anginal']))
```

به همین ترتیب باقی قوانین را هم وارد می کنیم و آرایه ها را پر می کنیم. چون در قوانین مهندسی هستیم، بین همه قوانین OR برقرار است و به همین دلیل در نهایت بین اعضای هر آرایه، max می گیریم. سپس با استفاده از نام خروجی و مقدار بدست آمده از max مرحله قبلی، یک dictionary می سازیم و آن را به عنوان نتیجه این قسمت بر می گردانیم.

۳) defuzzification.py:

در اینجا باید نتایج بدست آمده از مرحله inference که بصورت فازی هستند را به حالت غیرفازی دربیاوریم. برای اینکار از مرکز جرم استفاده می کنیم. چون نمودار ما بصورت پیوسته است، ابتدا باید آن را به بازه های کوچک تبدیل کنیم و سپس با هر بازه بصورت یک عدد گسسته رفتار کنیم و در نهایت فرمول مرکز جرم برای اعداد گسسته را پیاده سازی کنیم. برای اینکار چون بازه خروجی بین ۰ تا ۴ است و می خواهیم از ۱۰۰۰عدد گسسته (گام) استفاده کنیم، ۱۰۰۰ عدد با فاصله های ۰,۰۰۰۴ تولید می کنیم. سپس برای هر کدام از این اعداد، مقدار تعلق روی نمودارهای sick1, sick2, sick3, sick4, healthy را بدست می آوریم.

حال sick1 را در نسر بگسرسد. اگر عدد بدست آمده در مرحله قبل برای sick1 بزرگتر از عدد بدست آمده برای ایم خروجی در مرحله inference باشد، باید مقدار بدست آمده در مرحله inference را به عنوان نتیجه نهایی درنظر بگیریم. اما اگر ایم مقدار کوچکتر باشد، نیازی به این کار نیست. سپس بین مقادیر بدست آمده برای هر خروجی در نقطه مورد نظر، max می گیریم و آن را به عنوان مقدار تعلق حساب می کنیم.

درواقع برای هر نقطه چون می خواهیم انتگرال را بدست آوریم باید ببینیم که سطح زیر نمودار را باید بررسی کنیم یا سطح زیر مقدار تعلق بدست آمده در مرحله استنتاج. با این کار، هر سطحی که پایین تر باشد را برای انتگرال گیری در نظر می گیریم و سپس بین این مقادیر، آنکه دارای بیشترین مقدار است را انتخاب می کنیم.

حال برای محاسبه انتگرال، برای صورت کسر مقدار تعلق بدست آمده برای هر نقطه را در خود نقطه و در مقدار ۰,۰۰۰۴ که طول بازه آن نقطه است ضرب می کنیم و این مقدار را برای نقاط مختلف باهم جمع میکنیم. برای مخرج هم مقدار تعلق را در خود نقطه ضرب می کنیم و در نهایت جمع می زنیم. بعد از بررسی همه نقاط، صورت را بر مخرج تقسیم می کنیم و مرکز جرم را بدست می آوریم.

حال بر اساس بازه هایی که در کانال درس گذاشته شده است، مرکز جرم را بررسی می کنیم و بیماری هایی که فرد به آنها مبتلاست را بر می گردانیم.

۴) final_result.py:

در این فایل ورودی های مورد نیاز را در یک dictionary دریافت می کنیم. سپس ورودی ها را فازی سازی می کنیم، یعنی هرکدام را به تابع مربوطه در کلاس fuzzification می دهیم و نتیجه فازی را دریافت می کنیم. در مرحله بعدی، نتیجه های فازی شده را به تابع inference می دهیم تا خروجی را بدست آوریم. این خروجی فازی است و باید آن را به تابع defuzzification بدهیم تا غیرفازی شود. در نهایت این تابع یک string برمی گرداند که بیماری ها و مقدار مرکز جرم را دربر دارد و ما آن را return می کنیم تا به عنوان نتیجه نهایی نمایش داده شود.

ورودی زیر را در نظر بگیرید:

خروجی آن بصورت زیر است: