Probabilistic Graphical Models using pgmpy

November 25, 2014

```
In [7]: from IPython.display import Image, Math
```

0.0.1 General classification problem in Machine learning

To find the probability of a the class of a new data point given the training data and a new data point i.e $ P(C | x, D) $.

So, if we have the joint Probability distribution over all the variables we can easily marginalize and reduce this probability distribution to find the values for the new data point.

0.0.2 Probabilistic Graphical Models (PGM)

Probabilistic Graphical Model is a way of compactly representing Joint Probability distribution over random variables using the independence conditions of the variables.

0.0.3 How is PGM different than other algorithms?

The distinctive thing about PGM is that it provides a very intuitive and natural approach for modelling complex problems along with maintaining control over the computational costs.

0.0.4 Example

Let's see an example for predicting the price of a house. For simplicity we will consider that the price of the house depends only on Area, Location, Furnishing, Crime Rate and Distance from the airport. And also we will consider that all of these are discrete variables.

Our raw data would look something like this:

```
In [1]: ### GENERATE SOME DATA ###
```

Now let's create a model for the interaction of our variables using intuition:

Let's create this model using pgmpy.

```
In []: from pgmpy.models import BayesianModel
       import pandas as pd
       import numpy as np
       data = pd.DataFrame(raw_data, columns=['A', 'C', 'D', 'L', 'F', 'P'])
       data_train = data[:data.shape[0] * 0.75]
       model = BayesianModel([('F', 'P'), ('A', 'P'), ('L', 'P'), ('C', 'L'), ('D', 'L')])
       model.fit(data_train)
```

0.0.5 What does fit does ?

The fit method adds a Conditional Probability Distribution (CPD) to each of the node in our model

```
In []: model.get_cpds()
       # Show some more code
```

But the data that we have for training might be baised so with pgmpy we also have the option to assign your own Conditional Probability Distributions. Let's say the probability of getting an unfurnished home is equal to getting a furnished house. Let's adjust the values according to this.

```
In []: from pgmpy.factors import TabularCPD
       f_cpd = TabularCPD('F', 2, [[0.5], [0.5]])

       model.remove_cpd('F')
       model.add_cpd(f_cpd)

       model.check_model()
```

Now let's try to do some reasoning on our model to verify if our intuitution for the model was correct or not.

```
In [10]: from pgmpy.Inference import VariableElimination
         model = VariableElimination(model)
         # Returns a probability distribution over variables A and B.
         model.query(variables=['A', 'B'])
```

If you think about prediction about new values from this model, it is basically the same what we have been doing here. We basically ask questions about the probability of some variable giving conditions for other variables. Also we can account for missing values with just leaving it blank.

```
In []: model.predict(data[0.75 * data.shape[0] : data.shape[0]])
```

0.0.6 Why to use PGM rather than simply computing these values from the probaility distribution

The graph structure implies some independence conditions over the variables. The variables can be indirectly connected to each other in the following ways:

This independence due to the structure is responsible for the reduced computational complexity for inference.

For the Joint Probability distribution over all the variables we can write it as:

$$P(A, C, D, L, F, P) = P(A) * P(C|A) * P(D|A, C) * P(L|A, C, D) * P(F|A, C, D, L) * P(P|A, C, D, L, F)$$

But if we apply all the independency conditions that we saw above in this equation we get:

$$P(A, C, D, L, F, P) = P(A) * P(L|C, D) * P(C) * P(D) * P(P|F, A, L) * P(F)$$

For doing inference over this model we can simply eliminate variables or condition this joint probability. Say if we want to calcualate $ P(A) $ we could simply calculate:

$$P(A) = \sum_{C} \sum_{D} \sum_{L} \sum_{F} \sum_{P} P(A, C, D, L, F, P)$$

This algorithm of summing over variables that are not required is known as Variable Elimination.

0.0.7 How to construct model from the data?

Doing inference from the model is really simple. But the tough part is to create the model from the data.

The house price estimation example had very intuitive variables and thus we were able to construct the model very easily. But this is not always the case.

So for constructing the model we use the independence properties implied by the model and by the Joint Probability distribution.

One of the simplest way to construct a model is to find out some independence conditions in the data and according to that we can use that to arrange our variables in the graph structure in such a way to satisfy those independency conditions.

There are many more ways of finding structure in data to properly model them like density estimation etc.