Bayesian Inference

The purpose of this document is to review belief networks and naive Bayes classifiers.

At the end of lesson 9, Charles introduces the Bayes optimal classifier. Although this is the best performing classification model for a given hypothesis space, data set and a priori knowledge, the Bayes optimal classifier is computationally very costly. This is because the posterior probability $P(h \mid D)$ must be computed for each hypothesis $h \in H$ and combined with the prediction $P(v \mid h)$ before v_{MAP} can be computed.

In lesson 10, Michael discusses Bayesian inference. The end goal of this lesson is to introduce an alternative classification model to the optimal Bayes classifier: the naive Bayes classifier. This model is much more computationally efficient than optimal Bayes classification, and under certain conditions it has performance comparable to neural networks and decision trees[1]. Naive Bayes classifiers represent a special case of classifiers derived from belief networks -- graphical models which represent a set of random variables and their conditional dependencies[2]. In these notes we review belief networks and the special case of naive Bayes classifiers, along with some definitions from probability.

Definitions from Probability:

In this section we recall a few definitions from probability that we will need moving forward. Feel free to skip this section if you are familiar with conditional probability and Bayes' theorem.

[1] Mitchell, Tom M. "Machine learning. 1997." *Burr Ridge, IL: McGraw Hill* 45 (1997).

[2] "Bayesian network-Wikipedia, the free encyclopedia." 2003. 9 May. 2014 <http://en.wikipedia.org/wiki/Bayesian_network>

We say that X is *conditionally independent* of Y given Z if for all values (x_i, y_j, z_k) we have

$$P(X = x_i \mid Y = y_j, Z = z_k) = P(X = x_i \mid Z = z_k).$$

Writing out all definitions, we see that it is equivalent to say that for all values (x_i, y_j, z_k) we have

$$P(X = x_i, Y = y_j \mid Z = z_k) = P(X = x_i \mid Z = z_k)\, P(Y = y_j \mid Z = z_k).$$

We will also recall the following inferencing rules:

The product rule (aka, the chain rule):

$$P(X, Y) = P(X \mid Y)\, P(Y) = P(Y \mid X)\, P(X)$$

It is helpful to note that this rule also has the more general form:

$$P(X_1, \ldots, X_n) = \\ P(X_1 \mid X_2, \ldots, X_n)\, P(X_2 \mid X_3, \ldots, X_n) \cdots P(X_{n-1} \mid X_n)\, P(X_n).$$

Bayes' theorem for a hypothesis h and data set D:

$$P(h \mid D) = \frac{P(D \mid h)\, P(h)}{P(D)}$$

It is useful to note that Bayes' theorem makes some sense heuristically. If we increase the probability of a certain hypothesis $P(h)$, then we would naturally expect an increase in $P(h \mid D)$. Similarly, if some data D is more likely to occur in a world where hypothesis h is true (this probability is $P(D \mid h)$), then we might expect to see h given that we already see the data (this is $P(h \mid D)$). Last, if we increase $P(D)$, the probability that data D is observed independently, then this decreases the support that D provides for any specific hypothesis h. Hence $P(D)$ and $P(h \mid D)$ are inversely proportional.

Marginalization (aka, the theorem of total probability):

If $X_1, \ldots, X_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

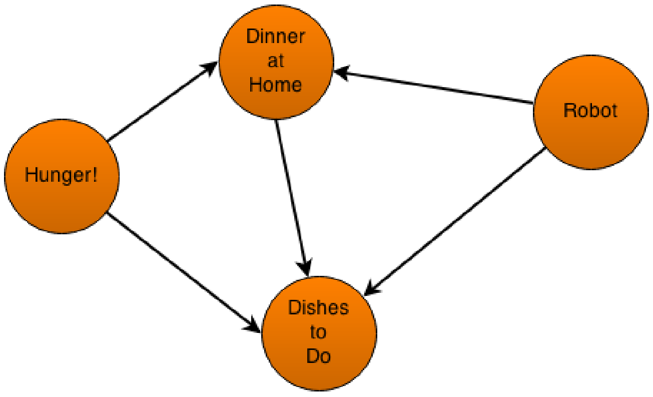
$$P(Y) = \sum_{i=1}^{n} P(Y \mid A_i)\, P(A_i).$$

Belief networks:

As described in the introduction, belief networks (a.k.a. Bayes(ian) net(work)s, probabilistic directed acyclic graphical models) are graphical models that describe the probability distribution of a set of variables in terms of the variables' conditional dependencies. So, in particular, given a set of random variables $Y_1, Y_2, \ldots, Y_n$, a network is called belief network if the joint probability distribution of the n-tuple $(Y_1 \ldots Y_n)$ can be written as

$$P(Y_1, \ldots, Y_n) = \prod_{i=1}^{n} P(Y_i \mid Parents(Y_i)).$$

We will explain what variables are included in the set $Parents(Y_i)$ below. In fact, let's start with an example illustrating this idea, and we can nail down the definitions as we go. Let's suppose we are given the following variables: "you are hungry!", "you own a robot that can cook and clean", "you will be eating dinner at home", and "you will have dishes to do":



In the network above, each node represents a variable, and arrows from different nodes represent conditional independence assumptions related to the variables. More on this below. For brevity, we will be referring to these variables as

$$Robot = Y_1, \quad HomeDinner = Y_2,$$
$$Dishes = Y_3, \quad Hungry! = Y_4.$$

Here, we say Y_i is a *descendant* of Y_j if there exists a directed path from Y_i to Y_j. For belief networks, we define the *Parents* of a variable to be the the variable's immediate predecessors in the network. So, for example,

$$Descendants(Robot) = (Dishes, Dinner)$$
$$Parents(Dishes) = (Robot, HomeDinner, Hunger!)$$

For each node in the network, we are given a conditional probability table describing the probability distribution of the corresponding variable, given the variable's parents. The belief network shows that for an assignment of values

$$(Y_1 = v_1, \ldots, Y_4 = v_4),$$

the joint probability can be written as

$$P(v_1, \ldots, v_4) = \prod_{i=1}^{n} P(v_i \mid Parents(Y_i)).$$

Note here that

$$P(v_i \mid Parents(Y_i))$$

is the information provided by the table for each node. Thus, in our case, given the belief network pictured above, we could determine the probability that you are hungry!, will eat dinner at home, not have to do dishes *and* own an awesome dish-cleaning-dinner-making robot,

$$P(Robot = 1, HomeDinner = 1, Dishes = 0, Hungry! = 1),$$

with the following four values from the belief network tables:

$$P(Robot = 1),$$
$$P(HomeDinner = 1 \mid Robot = 1, Hungry! = 1),$$
$$P(Dishes = 0 \mid Robot = 1, HomeDinner = 1, Hungry! = 1)$$
$$P(Hungry! = 1).$$

Ok, great! We have shown how you can use a belief net to compute joint probabilities, but what if you are given a set of random variables? Can you create a belief net that corresponds to these variables?

It turns out you can. We demonstrate how to do this with the variables

X_1, X_2, X_3 and X_4. Using the chain rule we can write:

$$P(X_1, X_2, X_3, X_4) = \\ P(X_4 \mid X_3, X_2, X_1)\, P(X_3 \mid X_2, X_1)\, P(X_2 \mid X_1)\, P(X_1).$$

For each conditional probability $P(X_i \mid X_{i-1}, X_{i-2}, \ldots, X_1)$ on the right side of the equation, we can choose the smallest subset of $X_i's$, which will be the $Parents(X_i)$, such that

$$P(X_i \mid X_{i-1}, X_{i-2}, \ldots, X_1) = P(X_i \mid Parents(X_i)).$$

For example, it may be the case for $P(X_4 \mid X_3, X_2, X_1)$ that the smallest subset gives

$$P(X_4 \mid X_3, X_2, X_1) = P(X_4 \mid X_3, X_2),$$

so we will denote

$$Parents(X_i) = (X_3, X_2).$$

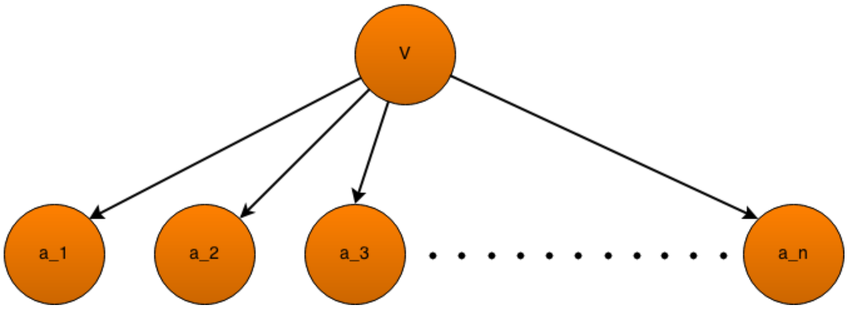
Then to create the belief net, we just need to create a graph with incoming edges from each variable in $Parents(X_i)$ to the variable X_i, and a conditional probability table for $Parents(X_i)$.

Naive Bayes Classifiers:

Naive Bayes classifiers are classifiers that represent a special case of the belief nets covered above, but with stronger independence assumptions. In particular, let's suppose we are given a classification variable V, and some attribute variables $a_1, \ldots, a_n$. The example of this seen in the lectures is the email spam-filtering example where

$$V = spam,$$
$$a_1 = viagra, \quad a_2 = prince, \quad a_3 = Udacity.$$

For our classifier to be a naive Bayes classifier, we make the (naive) assumption that every attribute variable is conditionally independent of every other attribute variable. Graphically, in terms of the belief nets described above, this is going to look like:



For the classification variable V , as always, we would like to find the most probable target value v_{map}, given the values for our attributes. We can write the expression for v_{map} and then use Bayes theorem to manipulate the expression as follows:

$$v_{map} = argmax_{v_j \in V} P(v_j \mid a_1, a_2, \ldots, a_n)$$

$$= argmax_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n \mid v_j)P(v_j)}{P(a_1, a_2, \ldots, a_n)}$$

$$= argmax_{v_j \in V} P(a_1, a_2, \ldots, a_n \mid v_j)P(v_j).$$

Next we would like to simplify the last expression. In what follows below, we use the general product rule for the first step, and then our naive conditional independence assumption for the second step:

$$P(a_1, a_2, \ldots, a_n \mid v_j)$$

$$= P(a_1 \mid a_2, \ldots, a_n, v_j)P(a_2 \mid a_3, \ldots, a_n, v_j) \cdots P(a_n \mid v_j)$$

$$= P(a_1 \mid v_j)P(a_2 \mid v_j) \cdots P(a_n \mid v_j) .$$

Substituting this equality into the formula for v_{map}, and writing the product more compactly we have the following expression:

$$v_{map} = argmax_{v_j \in V} P(v_j) \prod_{i=1}^{n} P(a_i \mid v_j).$$

One great computational advantage of this formula is the small number of terms that must be estimated to compute v_{map}. More precisely, for each classification category v_j that V can take, we must estimate the n values $P(a_i \mid v_j)$. Thus, the total number of terms to be estimated is just the number of attributes n multiplied by the number of distinct values v_j that V can take.

A second computational advantage of the formula is the fact that each of the terms to be estimated is a one-dimensional probability, which can be estimated with a smaller data set than the joint probability. By contrast, a direct estimate of the joint probability $P(\ a_1,\ a_2, \ldots,\ a_n \mid v_j\ )$ suffers from the "curse of dimensionality" [3]. Recall from lesson four of the lectures, the curse of dimensionality occurs when the amount of data needed to develop an acceptable (i.e. non-overfitted) classifier grows exponentially with the number of features.

Advantages and Disadvantages of Naive Bayes Classifiers:

As mentioned above, the naive Bayes classifier is very efficient for training, in terms of the total number of computations needed. Also, as mentioned in the introduction, naive Bayes performs well on many different training tasks. However, because of the strong conditional independence assumption placed on the attributes in the model, there are situations where naive Bayes is not appropriate.

To get an idea of why this might be the case, consider the task of using naive Bayes to learn XOR. So here, our attributes will be the inputs $X_1$and X_2, and we also have the classification variable $V = (X_1\, XOR\ X_2)$. Naive Bayes will consider each of the attributes independently and will be unable to accurately predict v_{map} given input values $X_1 = x_1$ and $X_2 = x_2$.

[3] Mitchell, Tom M. "Machine learning. 1997." *Burr Ridge, IL: McGraw Hill* 45 (1997).