Naïve Bayes Classification

Bayes Theorem for Classification

- Bayes theorem can be used to classify instances

$$\mathcal{D} = \{(\boldsymbol{X}_i, y_i) | i = 1,2, \dots, n\}$$

- Data of n instances, m classes ***X*** is vector of features, y_i is output class: $y_i \in (Y: \{y_1, \dots, y_j, \dots, y_m\})$

$$P(h|X) = \frac{P(X|h) \cdot P(h)}{P(X)}$$ $h: y = y_j$ outcome to predict

- We want *h* that maximises $P(h|X)$, maximum a posteriori hypothesis (MAP)

$$h_{MAP} = \arg\max_{h \in H} P(h|X) = \arg\max_{h \in H} \frac{P(X|h) \cdot P(h)}{P(X)} = \arg\max_{h \in H} P(X|h) \cdot P(h)$$

Bayes Theorem for Classification cont.

If we know or assume that all hypotheses *h* are equally probable, then we get the maximum likelihood hypothesis (ML)

$$h_{ML} = \arg\max_{h \in H} P(X|h)$$

In order to see how Bayes theorem can be used for classification, let's break down and simplify the formula. Notice that *P(X)* is independent of h, so it was omitted in MAP and ML formulas.

P(h) (prior)for each hypothesis (each class) can be calculated as $\frac{n_h}{n}$, where n_h is number of instances of training set classified into class y_j, and n is the number of instances in training set (which we use to estimate the probabilities)

From Bayes Theorem to Naïve Bayes

Now let's consider $P(X|h)$. In general, $P(X|h)$ is calculated using chain rule: $P(x_1 \wedge x_2 \wedge \cdots \wedge x_m) = \prod_{i=1}^{m} P(x_i | \cap_{j=1}^{i-1} x_j)$.

In order to calculate this, we would need all possible combinations of X, which we do not have. In Naïve Bayes, we assume that x_i occur independently, so the formula is simplified:

$$P(x_1 \wedge x_2 \wedge \cdots \wedge x_m) = P(x_1) * P(x_2) * \cdots * P(x_m)$$

Therefore:

$$P(D = (x_1, x_2, \cdots, x_m)|h) = P(x_1|h) \cdot \cdots \cdot P(x_m|h) = \prod_{x_i \in D} P(x_i|h)$$

Although the independence assumption is often wrong, the Bayes classifier yields good results.

How to Classify Using Naïve Bayes

1. Using training dataset:
   a) For each class, calculate *P(h)* as $\frac{n_h}{n}$, where *n* is number of instances, n_h is number of instances in class *y* for hypothesis *h*
   b) For each feature in training set, calculate $P(x_j|h)$ as $\frac{n_j}{n_h}$, where n_h is number of instances in class *h*, n_j is number of instances in class *h* with feature *j*
   c) Store the above calculations for later

2. Given test instance with features : $\{x_1, \ldots, x_j, \ldots\}$ and class y_k:
   a) Retrieve all *P(h)* and all $P(x_j|h)$ for these features
   b) Calculate $P(X|h) \cdot P(h) = P(h) * P(x_1|h) * \cdots * P(x_j|h) * \cdots$ for all features *x* in the test instance, and all hypotheses *h*, corresponding to all classes. So we will have a vector of *m* values, one for each class
   c) Take arg max of that vector, as the predicted class

Using Naïve Bayes: Example

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

We have Play Tennis data on the left

And we have new instance: $\langle Outlk = sun, Temp = cool, Humid = high, Wind = true\rangle$

Notice that we only need P(h), and counts for 4 attributes, so:

P(h=yes) = 9/14, P(h=yes) = 5/14

P(sun | y=2/9, P(cool | y)=3/9, P(high | y)=3/9, P(true | y)=3/9

P(sun | n=3/5, P(cool | n)=1/5, P(high | n)=4/5, P(true | n)=3/5

Therefore

arg max(9/14*2/9*3/9*3/9*3/9,
5/14*3/5*1/5*4/5*3/5) =

arg max(y:0.0053,n:0,0206)=no

Using Naïve Bayes: Laplace Smoothing

But what if a test case has an attribute that does not occur in every class? Then the probability value for that attribute is 0, so all is 0!

In this case use Laplace smoothing:

$$P(x|y) = \frac{n_{yx}}{n_y} \Longrightarrow \hat{P}(x|y) = \frac{\gamma + n_{yx}}{\gamma \cdot |dom(X)| + n_y}$$

n_y	no. of instances from training in class y
n_{yx}	no. of instances from class y with value x for attribute X
$\lvert dom(X)\rvert$	no. of distinct values in X,

γ is usually taken as 1.

So, for example, P(cool | yes)=(1+3)/(1*3+9) = 4/12. Note that we need to apply Laplace correction to that feature in all classes.