How to Predict a Class with Naive Bayes

Task: *Predict class of a new sample given the training data*

F1	F2	F3	Target
True	Small	False	Class A
False	Medium	False	Class B
True	Small	True	Class B
True	Large	False	Class A

New sample:

False	Medium	True	?

1. **Foreach value v in the new sample calculate the probability of v per class**
    - foreach class X: P(value v | class X)= $\frac{\#\ of\ samples\ with\ value\ v\ in\ Class\ X}{\#of\ samples\ in\ Class\ X}$
    - e.g.: P(F1 = True | Class A) = 2/2 P(F2 = True | Class B) = 1/2
2. **Foreach class calculate the likelihood of the class (given the new sample)**
    - foreach class X: Likelihood(X | E) = $P(F1 = True \mid Class\ X) \times P(F2 = Medium \mid Class\ X) \times \ldots \times$
    - e.g.: Likelihood(Class A | E) =$P(F1 = True \mid Class\ A) \times P(F2 = Medium \mid Class\ A) \times P(F3 = True \mid Class\ A) \times P(Class\ A) = \frac{2+1}{2+1} \times \frac{0+1}{2+1} \times \frac{0+1}{2+1} \times \frac{2}{4} = 0.05555$
    - P(Class A) … Proportion of the Class A in the training data
    - disclaimer: to prevent multiplication with zero a default value of one is added (= Laplace correction)
        - Laplace correction is not needed for the class Probability P(Class X)
3. **Foreach class calculate the probability of the class (given the new sample)**
    - foreach class X: $P(X \mid E) = \frac{Likelihood(X \mid E)}{Likelihood(AllClasses)}$
    - e.g.: $P(Class\ A \mid E) = \frac{Likelihood(Class\ A \mid E)}{Likelihood(Class\ A \mid E) + Likelihood(Class\ B \mid E)}$
4. **Choose the class with highest probability as predicted class of the new sample**

Numerical Values

for numerical values we can calculate mean and sd per class and use the gaussian density function: $\boldsymbol{f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)}}$ to get the probability of the new sample value. (Step 2-4 are the same)

How to Build a Decision Tree

Task: Construct a decision tree from a given Sample Set

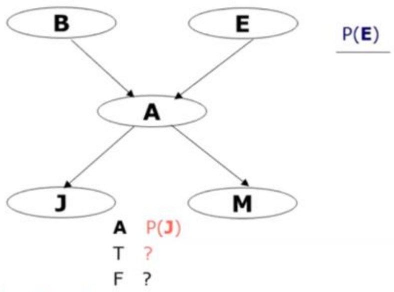
1. **For each attribute A identify possible splits of samples into subspaces**
   a. a possible split is
      i. for categorical values: e.g. one vs. all
      ii. split of continuous variables at values ver
   b. **foreach possible split calculate split score** (Error Rate, Information Gain or Gini Index)
      i. Absolute Error Rate
         1. Absolute number of False Classified Samples in the Subset
         2. Best Split = Split with lowest error rate
      ii. Information gain
         1. Compute the Entropy using the probability of each class
            - Entropy = $H(X) = E\left(I(X)\right) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} \boldsymbol{p(x_i) \log_2 p(x_i)}$
            - n … number of classes
            - $p(x_i)$ … $probability\ of\ picking\ a\ datapoint\ with\ class\ i$
         2. Compute Information gain
            - for subset A and B of the Split:
            - $\boldsymbol{IG(X_A, X_B) = H(X) - p(x_a)H(X_A) - p(x_B)H(X_B)}$
            - H(X) … Entropy of unsplit data
            - $p(x_a)$ … Probability of Subset A = $\frac{\#Samples\ in\ A}{\#\ Samples\ in\ A\ \&\ B}$
            - $H(X_A)$ … Entropy of Subset A (again using all classes)
         3. Best Split = Split with highest information gain
      iii. Gini index
         1. Compute the Gini Index using the probability of each class
            - $Gini\ index = I_G(p) = \boldsymbol{1 - \sum_{i=1}^{|C|} p_i^2}$
            - C … $total\ classes$
            - p_i … $probability\ of\ picking\ a\ datapoint\ with\ class\ i$
         2. Compute Gini index gain
            - $\boldsymbol{GG(X_A, X_B) = I_G(X) - p(x_a)I_G(X_A) - p(x_B)I_G(X_B)}$
            - $I_{G(X)}$ … Gini index of unsplit data
            - $p(x_a)$ … Probability of Subset A = $\frac{\#Samples\ in\ A}{\#\ Samples\ in\ A\ \&\ B}$
            - $I_G(X_A)$ … Gini index of Subset A (again using all classes)
         3. Best Split = Split with highest Gini Gain
   c. choose best split per attribute
   d. choose best split of all attributes
2. **Split samples into subspaces according to the best split**
3. if all subspaces consist of samples of the same classes or max depth is reached stop
4. else start over from 1. with the new subspaces

How to Construct a Bayesian Network

Case A: Structure known → Learn probabilities

1. Calculated Conditioned Probability of an event using the given data
   - e.g. P(J|A) = $\frac{\#\ Samples\ where\ A=true \wedge J=true}{\#\ Samples\ A=true}$
   - add Laplace correction

B	E	A	J	M
t	f	t	t	f
f	f	f	f	t
f	t	f	t	f
...				



Use counts

P(J|A)=(#A = true ∧ J = true) / #(A = true)

Avoid 0 probabilities (Laplace correction):

P(J|A)=(#(A = true ∧ J = true)+**1**)/ (#(A = true) +**2**)

P(E) = (#(E = true)+**1**) / (k+**2**)

k: number of samples

Laplace correction

Case B: Structure not known → Learn Structure and probabilities

Task: find a network that is a good fit to data and has low complexity

➔ *Maximize* $\log(P(D|M)) - \alpha\ \#M$

α *... indicates how important complexity reduction is*

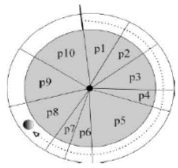
brute force for finding structure would be computational too expensive → use heuristics (local search, simulated annealing etc.)

for instances with Hill Climbing:

1. Construct an initial network
2. Calculate the score of the current Bayesian Network (with learned probabilities)
3. Create network's neighbourhood by modifying the current network
   - small changes to the current network (e.g. add arc, remove arc, reverse arc)
4. Select best of the networks in the neighbourhood as a new current network
5. Go to 3. if stop criteria is not fulfilled

How to do AdaBoost

1. Weight all Samples equally
    - for instances with 1/n
2. Choose a classifier
3. Classify data
    - Option 1: Split data with the current classifier
        - evaluating split with a split score (IG, Gini, Error rate) weighted by the samples
    - Option 2: Choose classifier data from all sample sets where weights are probabilities
        - → samples with higher weights will be over-represented



4. compute total Error of the classification (according to the sample weights)
    - TotalError … weighted sum of wrong predicted samples of this classifier
5. Assign the classifier a weight
    - $\alpha(t) = \frac{1}{2} \times \log\left(\frac{1-TotalError}{TotalError}\right)$
6. Update sample weights D
    - if sample was classified correct: $\boldsymbol{D_{t+1}(i) = D_t(i) \times e^{-\alpha(t)}}$
        - → decrease sample weight
    - if sample was classified incorrect: $\boldsymbol{D_{t+1}(i) = D_t(i) \times e^{\alpha(t)}}$
        - → increase sample weight
    - normalise weights to sum = 1
7. stop if criteria reached or start over from 2.

How to Build a Rule Set with the Covering Algorithm

//todo

1. Fix one class