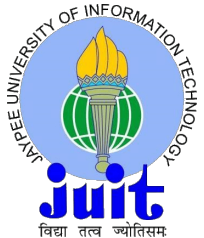

Machine Learning

(Course Code: 18B1WCI634)

Bayesian Learning

Mr. Sandeep Kumar Patel

Assistant Professor

Jaypee University of Information Technology

Waknaghat, Solan, HP-173234

Bayesian Learning

- Probabilistic approach to Machine Learning
- Uses probability distributions
- Goal: Find most probable hypothesis

Example

- Total males = 60
- Male teachers = 12

$$P(Teacher|Male) = \frac{12}{60} = 0.2$$

Bayes Theorem

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

textbfKey Terms

- Prior $P(h)$
- Likelihood $P(D|h)$
- Posterior $P(h|D)$
- Evidence $P(D)$

Example

- Total males = 60
- Male teachers = 12

$$P(Teacher|Male) = \frac{12}{60} = 0.2$$

Step 1: Joint Probability

- Consider two events A and B
- Joint probability:

$$P(A \cap B)$$

- Using conditional probability:

$$P(A \cap B) = P(A)P(B|A)$$

Step 2: Alternative Form

- Joint probability can also be written as:

$$P(A \cap B) = P(B)P(A|B)$$

- Equate both expressions:

$$P(A)P(B|A) = P(B)P(A|B)$$

Step 3: Final Formula

- Rearranging:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

Bayes Theorem

Updates probability using prior knowledge and evidence

Numerical 1: Medical Test

Problem:

- Disease prevalence: $P(D) = 0.01$
- Test accuracy:
  - $P(+|D) = 0.99$ (True Positive)
  - $P(+|\neg D) = 0.05$ (False Positive)
- Find: $P(D|+)$

Numerical 1: Medical Test

Problem:

- Disease prevalence: $P(D) = 0.01$
- Test accuracy:
  - $P(+|D) = 0.99$ (True Positive)
  - $P(+|\neg D) = 0.05$ (False Positive)
- Find: $P(D|+)$

Solution:

$$P(D|+) = \frac{P(+|D)P(D)}{P(+)}$$

$$P(+) = (0.99)(0.01) + (0.05)(0.99) = 0.0099 + 0.0495 = 0.0594$$

$$P(D|+) = \frac{0.99 \times 0.01}{0.0594} = \frac{0.0099}{0.0594} \approx 0.166$$

Answer: $\approx 16.6\%$

Numerical 2: Spam Email

Problem:

- Probability of spam: $P(S) = 0.4$
- Word "Free" appears:
  - $P(F|S) = 0.8$
  - $P(F|\neg S) = 0.2$
- Find: $P(S|F)$

Numerical 2: Spam Email

Problem:

- Probability of spam: $P(S) = 0.4$
- Word "Free" appears:
  - $P(F|S) = 0.8$
  - $P(F|\neg S) = 0.2$
- Find: $P(S|F)$

Solution:

$$P(S|F) = \frac{P(F|S)P(S)}{P(F)}$$

$$P(F) = (0.8)(0.4) + (0.2)(0.6) = 0.32 + 0.12 = 0.44$$

$$P(S|F) = \frac{0.8 \times 0.4}{0.44} = \frac{0.32}{0.44} \approx 0.727$$

Answer: $\approx 72.7\%$

Maximum A Posteriori (MAP) Hypothesis

Goal: Find the hypothesis h that is most probable after observing data D.
Using Bayes Theorem:

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

where:

- $P(h|D)$ = Posterior probability of hypothesis
- $P(D|h)$ = Likelihood of data given hypothesis
- $P(h)$ = Prior probability of hypothesis
- $P(D)$ = Evidence probability

Maximum A Posteriori (MAP) Hypothesis

The MAP hypothesis is:

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

Substituting Bayes theorem:

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

Since $P(D)$ is constant for all hypotheses:

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

Maximum A Posteriori (MAP) Hypothesis

Interpretation:

- Combines prior belief and observed data
- Useful when prior knowledge exists
- Common in Bayesian Learning

Maximum Likelihood (ML) Hypothesis

Maximum Likelihood selects the hypothesis that maximizes the probability of observing the training data.

Starting from MAP:

$$h_{MAP} = \arg\max P(D|h)P(h)$$

Assume all hypotheses are equally likely:

$$P(h_1) = P(h_2) = \cdots = P(h_n)$$

Therefore, prior probabilities become constants.

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

where:

- $P(D|h)$ is the likelihood function, ML ignores prior probabilities

Maximum Likelihood (ML) Hypothesis

Interpretation:

- Chooses hypothesis best fitting observed data
- Widely used in statistical machine learning
- Simpler than MAP because priors are ignored

Relation between MAP and ML

$$\boxed{h_{MAP} = \arg\max P(D|h)P(h)}$$

$$\boxed{h_{ML} = \arg\max P(D|h)}$$

If priors are equal:

$$h_{MAP} = h_{ML}$$

Maximum Likelihood Estimation (MLE)

Suppose we have training data:

$$D = \{x_1, x_2, x_3, \ldots, x_n\}$$

Assume each sample is independently drawn from a probability distribution parameterized by θ.

The likelihood function is:

$$L(\theta) = P(D|\theta)$$

Because samples are independent:

$$L(\theta) = \prod_{i=1}^{n} P(x_i|\theta)$$

Maximum Likelihood Estimation (MLE)

The Maximum Likelihood Estimate chooses parameter θ that maximizes the likelihood:

$$\theta_{ML} = \arg\max_{\theta} L(\theta)$$

$$\theta_{ML} = \arg\max_{\theta} \prod_{i=1}^{n} P(x_i|\theta)$$

Taking logarithm for simplicity:

$$\log L(\theta) = \sum_{i=1}^{n} \log P(x_i|\theta)$$

Maximum Likelihood Estimation (MLE)

Thus,

$$\theta_{ML} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(x_i|\theta)$$

Goal of MLE:

- Find parameters that make observed data most probable
- Widely used in regression, classification, and probabilistic models

Least Squares Estimation

Consider linear regression model:

$$y_i = wx_i + b + \epsilon_i$$

where:

- w = slope
- b = intercept
- ϵ_i = error term

Predicted output:

$$\hat{y}_i = wx_i + b$$

Error for each sample:

$$e_i = y_i - \hat{y}_i$$

Least Squares Estimation

Least Squares minimizes sum of squared errors:

$$J(w, b) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Substituting prediction equation:

$$J(w, b) = \sum_{i=1}^{n} (y_i - (wx_i + b))^2$$

Optimal parameters are:

$$(w^*, b^*) = \arg\min_{w,b} \sum_{i=1}^{n} (y_i - (wx_i + b))^2$$

Relation Between MLE and Least Squares

Assume errors are Gaussian distributed:

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Then probability density is:

$$P(y_i|x_i, w, b) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - (wx_i + b))^2}{2\sigma^2}\right)$$

Likelihood for all samples:

$$P(D|w, b) = \prod_{i=1}^{n} P(y_i|x_i, w, b)$$

Relation Between MLE and Least Squares

Taking log likelihood:

$$\log P(D|w,b) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - (wx_i + b))^2 + C$$

where C is constant.

Maximizing log likelihood is equivalent to minimizing:

$$\sum_{i=1}^{n}(y_i - (wx_i + b))^2$$

Hence,

MLE with Gaussian noise $\equiv$ Least Squares Estimation

Naive Bayes Learning Algorithm

Naive Bayes is a probabilistic classification algorithm based on Bayes Theorem.

For a hypothesis (class) C and feature vector X:

$$P(C|X) = \frac{P(X|C)\,P(C)}{P(X)}$$

where:

- $P(C|X)$ = Posterior probability
- $P(X|C)$ = Likelihood
- $P(C)$ = Prior probability
- $P(X)$ = Evidence

Naive Bayes Learning Algorithm

Goal:

$$C_{MAP} = \arg\max_{C} P(C|X)$$

Using Bayes rule:

$$C_{MAP} = \arg\max_{C} \frac{P(X|C)P(C)}{P(X)}$$

Since $P(X)$ is constant:

$$C_{MAP} = \arg\max_{C} P(X|C)P(C)$$

Naive Independence Assumption

Suppose feature vector:

$$X = (x_1, x_2, x_3, \ldots, x_n)$$

The likelihood becomes:

$$P(X|C) = P(x_1, x_2, \ldots, x_n|C)$$

Direct computation is expensive.

Naive Bayes assumes all features are conditionally independent:

$$P(X|C) = \prod_{i=1}^{n} P(x_i|C)$$

Naive Independence Assumption

Thus:

$$P(C|X) \propto P(C) \prod_{i=1}^{n} P(x_i|C)$$

Classification rule:

$$C_{NB} = \arg\max_{C} P(C) \prod_{i=1}^{n} P(x_i|C)$$

Why "Naive"?

- Assumes features are independent
- Simplifies probability computation
- Works surprisingly well in practice

Naive Bayes Learning Algorithm Steps

Training Phase

1. Compute prior probability for each class:

$$P(C_k) = \frac{\text{Number of samples in class } C_k}{\text{Total samples}}$$

2. Compute conditional probabilities:

$$P(x_i|C_k)$$

from training data.

Naive Bayes Learning Algorithm Steps

Prediction Phase

For a new sample:

$$X = (x_1, x_2, \ldots, x_n)$$

Compute score for every class:

$$Score(C_k) = P(C_k) \prod_{i=1}^{n} P(x_i | C_k)$$

Predict class with maximum score:

$$\hat{C} = \arg\max_{C_k} Score(C_k)$$

Example of Naive Bayes Classification

Suppose two classes:

$$C_1 = \text{Spam}$$

$$C_2 = \text{Not Spam}$$

Given email features:

$$X = (\text{free}, \text{offer}, \text{win})$$

Compute:

$$P(C_1|X) \propto P(C_1)P(\text{free}|C_1)P(\text{offer}|C_1)P(\text{win}|C_1)$$

Similarly:

$$P(C_2|X) \propto P(C_2)P(\text{free}|C_2)P(\text{offer}|C_2)P(\text{win}|C_2)$$

Example of Naive Bayes Classification

Decision Rule:

Choose class with larger posterior probability

Naive Assumption

- Features are independent
- Example: Color, Shape, Taste

Applications

- Spam filtering
- Sentiment analysis
- Document classification
- Medical diagnosis