Naive Bayes

Nishanth Nair

15 Oct,2019

Parameter Estimation

- **Likelihood**
  - Probability of observed features, given parameters.
- Maximum Liklihood Estimator(MLE):
  - $argmax_{\theta} \sum_j \sum_i logP_{\theta}(F_{ji}|Y_j)$
- With infinite data, MLE is the best estimator
- $P(X|Y) = \frac{P(X,Y)}{P(Y)} \approx \frac{count(X,Y)}{count(Y)}$ (USING the MLE for features)
  - E.g., if we observed [hht]:
    * MLE: P(heads) = 2/3
- **Naive Bayes**
  - Most statistical models do not have such a simple MLE for parameters;
  - They are usually fit by making incremental improvements.
  - Naive Bayes is appealing because the parameters are so easy to estimate.
    * Easy to parallelize.
    * All we need are counts.
    * Very fast compared to training for logistic regression, other models.
- **Smoothing** : Avoid overfitting and zero probabilities
- **Pierre-Simon Laplace**
  - Our belief about a random variable is a function of the evidence we have collected.
  - No matter how many times we observe an event, we should save some small probability for unforeseen events.
  - **Laplace Smoothing**:
    * Idea: Pretend we saw every outcome k more times than we actually did.Larger k more smoothing.
    * if we observed [hht]:
      · MLE: P(heads) = $\frac{2}{3}$
      · LAP1: P(heads) = $\frac{2+1}{3+1h+1t} = \frac{3}{5}$
      · LAP100: P(heads) = $\frac{2+100}{3+100h+100t} = \frac{102}{203}$
    * **Smoothing conditionals**:
      · $LAP_k : P(x|y) = [count(x,y) + k]/[count(y) + k[X]]$

- Performs poorly when [X] is large or [Y] is large

- **Interpolation Smoothing**:
  - Linear interpolation biases P(x|y) toward P(x):
    - $LIN_{\alpha}P(x|y) = \alpha.P(x|y) + (1-\alpha)P(x)$
- There are many varieties of smoothing.
  - All try to compensate for a lack of training data.
  - All try to allow parameters to generalize better to new data.

Confidence Estimation

- If higher confidence values correspond with higher accuracy, the classifier is said to be calibrated.
- Confidence of a classifier is the posterior of the top label: $max_y P(y|x)$
- Naive Bayes tends to be overconfident because of the independence assumption. (It ignores corelation among features)

Classifiers based on Bayes Rule

- Input : X
- Class : Y
- Goal: Determine P(Y=y|X=x). Example: Predict P(Y=spam | X=email)

Learning a full bayesian model

- To learn P(Y|X), estimate joint probability distribution of P(X|Y) and P(Y) from the training data.
- $P(Y = y_i) = \frac{\text{Number of examples with label}}{\text{Total number of examples}}$
- If X has n attributes each taking 2 discrete values and Y has 2 possible classes, then, $P(X_1, X_2...X_n|Y)$ will require $2.(2^n - 1)$ parameters to characterize the probablity distribution. This requires too much data and maynot be possible.

Learning a Naive bayesian model

- Instead of assuming that all of the different permutations have different probabilities, assume features are statistically independent.
- Two events A and B are statistically independent if occurence of one does not affect the occurence of the other.

  - P(A|B) = P(A)
  - P(A & B) = P(A).P(B)

- **Pairwise Independence**:
  - P(A|C) = P(A)
  - P(A,B|C) = P(A|C).P(B|C)
    * P(A,B|C) = P(A|B,C).P(B|C) = P(A|C).P(B|C)

- **Conditional Independence**:
  - P(A|B) = P(A)
  - P(A|B,C) = P(A|C)
  - P(A,B) = P(A).P(B)

- General Conditional Independence: When X contains n attributes that are conditionally independent given Y,

$$P(X_1, X_2, ...X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

Note:

- If X and Y are boolean variables, we need only 2n parameters to define P(X|Y), which is a drastic reduction from $2(2^n - 1)$
- This has high bias. Eventhough its a false assumption, it makes the problem tractable.

Naive Bayes Algorithm

This is a classification algorithm based on Baye's rule that assumes $X_1, X_2...X_n$ are all conditionally independent of one another given Y.The value of this assumption is that it drasstically simplifies the representation of P(X|Y), and the problem of estimating it from the training data.

$$P(Y = y_k|X_1, X_2...X_n) = \frac{P(Y = y_k). \prod_i P(X_i|Y = y_k)}{\sum_j P(Y = y_j). \prod_i P(X_i|Y = y_j)}$$

Class Inference, Y = $argmax_{y_k} P(Y = y_k). \prod_i P(X_i|Y = y_k)$,

where argmax implies, value of Y_k that maximizes the expresssion

- Multiplying lots of probabilities, which are between 0 and 1, can result in floating-point underflow.
- Since log(xy) = log(x) + log(y), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

Class Inference, Y = $argmax_{y_k} \log P(Y = y_k) + \sum_i \log P(X_i|Y = y_k)$,

where argmax implies, value of Y_k that maximizes the expresssion

Examples

Algorithm for document classification

Given a number of documents, each classified y_k, classify a new document as one of y_k.

1. From training data, extract the vocabulary.
2. Compute, P(Y)
   - For each class, y_k:
     - $P(Y = y_k) = \frac{\text{Number of docs classified as } y_k}{\text{total number of docs in training}}$
     - build $text_k$ that contains all the docs in class y_k
3. Compute P(X|Y)
   - For each word, X_i in the vocabulary:
     - n_{ik} : Number of occurences of x_i in $text_k$
     - n : Total tokens in $text_k$
     - α : Smoothing parameter
       * $P(X = x_i|Y = y_k) = \frac{n_{ik}}{n+\alpha.|vocabSize|}$
4. Class Inference:
   - Class, Y = $argmax_{y_k} \log P(Y = y_k) + \sum_i \log P(X_i|Y = y_k)$

Spam Classification (Review naiveBayesModel.pdf)

- Input (X): e-mail
- Output (Y): [spam, ham]
- Data: collection of labeled e-mail
- Goal: predict labels of new e-mails
  - P(Y=spam|e-mail) = ?
- P(spam | features)
  - Words,URLs,Sender etc..
- **Naive Bayes**
  - Single Feature:
    * P(Y|X) ~ P(Y|F) (Reduce to 1 feature)

    * Bayes's Rule: $P(Y|F) = \frac{P(Y).P(F|Y)}{P(F)} = \frac{P(Y).P(F|Y)}{\sum_y P(Y).P(F|Y)}$ (Marginalize)

  - Multiple features (General Naive Bayes):

    * P(Y|X) ~ P(Y|F1,F2…Fn)

    *

$$\begin{aligned}
P(Y|F_1, F_2 ... F_n) &= \frac{P(Y, F_1 ... F_n)}{P(F_1 ... F_n)} \\
&= \frac{P(Y).P(F_1 ... F_n|Y)}{P(F_1 ... F_n)} \approx \frac{P(Y).P(F_1|Y).P(F_2|Y)...P(F_n|Y)}{P(F_1 ... F_n)} \\
&= \frac{P(Y).\prod_i P(F_i|Y)}{\sum_y P(Y).\prod_i P(F_i|Y)}
\end{aligned}$$

    * **Key Assumption** for approximation: Features are independent!!

- Naive Bayes for Spam classification

  - W_i is the word at position i
  - $P(Y|X) \approx P(Y).P(W_1|Y).P(W_2|Y).....P(W_n|Y)$
  - "**Bag of Words**" Assumption (BOW): In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.
    * Keeps the number of parameters manageable.
    * Here, each position has the same distribution: P(W|Y)

- **Generative story of Naive bayes**

  - A generative model learns the joint probability distribution P(X,Y) while a discriminative model learns the conditional probability distribution P(Y|X) i.e the posterior distribution.
  - $P(Y|X) \approx P(Y).P(W_1|Y).P(W_2|Y).....P(W_n|Y)$
  - To Generate a document,
    * Pick a class spam/ham according to P(Y).
    * Repeat until you have enough words: Pick a word according to P(W|Y).