DEPAUL UNIVERSITY

DSC478: Programming Machine Learning Applications

Roselyne Tchoua

rtchoua@depaul.edu

School of Computing, CDM, DePaul University

Classification: 3 Step Process

1. **Model construction** (**Learning**):
   - Each record (instance, example) is assumed to belong to a predefined class, as determined by one of the attributes
     - This attribute is called the **target attribute**
     - The values of the target attribute are the **class labels**
   - The set of all instances used for learning the model is called **training set**
2. **Model Evaluation** (**Accuracy**):
   - Estimate accuracy rate of the model based on a **test set**
   - The known labels of test instances are compared with the predicts class from model
   - Test set is independent of training set otherwise over-fitting will occur
3. **Model Use** (**Classification**):
   - The model is used to classify unseen instances (i.e., to predict the class labels for new unclassified instances)
   - Predict the value of an actual attribute

Naïve Bayes: Basic Idea

- For a given new record to be classified, find other records like it (i.e., same values for the predictors)
- What is the prevalent class among those records?
- Assign that class to your new record

Naïve Bayes: Usage

- Categorical variables
- Numerical variable must be binned and converted to categorical
- Can be used with very large data sets
  - Example: Spell check programs assign your misspelled word to an established "class" (i.e., correctly spelled word)

Bayesian Learning

- Bayes' theorem plays a critical role in probabilistic learning and classification
  - Uses prior probability of each class given no information about an item
  - Classification produces a posterior probability distribution over the possible classes given a description of an item
  - The models are incremental in the sense that each training example can incrementally increase or decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data
- Given a data instance *X* with an unknown class label, *H* is the hypothesis that *X* belongs to a specific class *C*
  - The *conditional probability* of hypothesis *H* given *observation X,* Pr*(H|X),* follows the Bayes's theorem:

$$\Pr(H \mid X) = \frac{\Pr(X \mid H)\Pr(H)}{\Pr(X)}$$

- Practical difficulty: requires initial knowledge of many probabilities

Basic Concepts in Probability

- P(*A* | *B*) is the probability of *A* given *B*
- Assumes that *B* is all and only information known.
- Note that: $P(A \wedge B) = P(A \mid B).P(B)$
- Bayes' Rule:
  Direct corollary of above definition

$$P(A \wedge B) = P(A \mid B).P(B)$$
$$P(B \wedge A) = P(B \mid A).P(A)$$
$$\text{but } P(A \wedge B) = P(B \wedge A)$$
$$\therefore P(A \mid B).P(B) = P(B \mid A).P(A)$$
$$\therefore P(A \mid B) = \frac{P(B \mid A).P(A)}{P(B)}$$

- Often written in terms of hypothesis and evidence:

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Basic Concepts in Probability

- *A* and *B* are *independent* iff:

$$P(A \mid B) = P(A)$$

$$P(B \mid A) = P(B)$$

These two constraints are logically equivalent

- Therefore, if *A* and *B* are independent:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)} = P(A)$$

$$P(A \wedge B) = P(A)P(B)$$

Bayesian Classification

- Let set of classes be $\{c_1, c_2, \ldots c_n\}$
- Let E be description of an instance (e.g., vector representation)
- Determine class of E by computing for each class c_i

$$P(c_i \mid E) = \frac{P(c_i)P(E \mid c_i)}{P(E)}$$

- P(E) can be determined since classes are complete and disjoint:

$$\sum_{i=1}^{n} P(c_i \mid E) = \sum_{i=1}^{n} \frac{P(c_i)P(E \mid c_i)}{P(E)} = 1$$

$$P(E) = \sum_{i=1}^{n} P(c_i)P(E \mid c_i)$$

Bayesian Classification

- Need to know:
  - Priors: P(c_i) and Conditionals: P(E | c_i)
- P(c_i) are easily estimated from data.
  - If n_i of the examples in D are in c_i, then P(c_i) = n_i / |D|
- Assume instance is a conjunction of binary features/attributes:

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

$$E = e_1 \wedge e_2 \wedge \cdots \wedge e_m$$

$$E = Outlook = rain \wedge Temp = cool \wedge Humidity = normal \wedge Windy = true$$

Naïve Bayesian Classification

- Problem: Too many possible combinations (exponential in *m*) to estimate all P(*E* | c_i)

- If we assume **features/attributes of an instance are independent** given the class (c_i) (*conditionally independent*)

$$P(E \mid c_i) = P(e_1 \wedge e_2 \wedge \cdots \wedge e_m \mid c_i) = \prod_{j=1}^{m} P(e_j \mid c_i)$$

- Therefore, we then only need to know P(e_j | c_i) for each feature and category

Estimating Probabilities

- Normally, probabilities are estimated based on observed frequencies in the training data.
- If D contains n_i examples in class c_i, and n_{ij} of these n_i examples contains feature/attribute e_j, then:

$$P(e_j \mid c_i) = \frac{n_{ij}}{n_i}$$

- If the feature is continuous-valued, $P(e_j|c_i)$ is usually computed based on **Gaussian distribution** with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(e_j|c_i)$ is

$$P(e_j \mid c_i) = g(e_j, \mu_{c_i}, \sigma_{c_i})$$

Smoothing

- Estimating probabilities from small training sets is error-prone:
  - If due only to chance, a rare feature, e_k, is always false in the training data, $\forall c_i : P(e_k \mid c_i) = 0$.
  - If e_k then occurs in a test example, E, the result is that $\forall c_i$: $P(E \mid c_i) = 0$ and $\forall c_i$: $P(c_i \mid E) = 0$
- To account for estimation from small samples, probability estimates are adjusted or smoothed
- Laplace smoothing using an m-estimate assumes that each feature is given a prior probability, p, that is assumed to have been previously observed in a "virtual" sample of size m.

$$P(e_j \mid c_i) = \frac{n_{ij} + mp}{n_i + m}$$

- For binary features, p is simply assumed to be 0.5.

Example

Outlook	Tempreature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

- Here, we have two classes C1="yes" (Positive) and C2="no" (Negative)
- Pr("yes") = instances with "yes" / all instances = 9/14
- If a new instance X had outlook="sunny", then Pr(outlook="sunny" | "yes") = 2/9 (since there are 9 instances with "yes" (or P) of which 2 have outlook="sunny")
- Similarly, for humidity="high", Pr(humidity="high" | "no") = 4/5
- And so on.

Example (continued)

- Now, given the training set, we can compute all the probabilities

Outlook	P	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5
Tempreature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Humidity	P	N
high	3/9	4/5
normal	6/9	1/5
Windy		
true	3/9	3/5
false	6/9	2/5

- Suppose we have new instance X = <sunny, mild, high, true>. How should it be classified?

$X = <$ sunny , mild , high , true $>$

$$\Pr(X \mid \text{“no”}) = 3/5 \cdot 2/5 \cdot 4/5 \cdot 3/5$$

- Similarly:

$$\Pr(X \mid \text{“yes”}) = (2/9 \cdot 4/9 \cdot 3/9 \cdot 3/9)$$

Example (continued)

- To find out to which class X belongs we need to maximize: $\Pr(X \mid C_i).\Pr(C_i)$, for each class C_i (here "yes" and "no")

$$X = \text{<sunny, mild, high, true>}$$

$$\Pr(X \mid \text{"no"}).\Pr(\text{"no"}) = (3/5 \cdot 2/5 \cdot 4/5 \cdot 3/5) \cdot 5/14 = 0.04$$

$$\Pr(X \mid \text{"yes"}).\Pr(\text{"yes"}) = (2/9 \cdot 4/9 \cdot 3/9 \cdot 3/9) \cdot 9/14 = 0.007$$

- To convert these to probabilities, we can normalize by dividing each by the sum of the two:
  - $\Pr(\text{"no"} \mid X) = 0.04 / (0.04 + 0.007) = 0.85$
  - $\Pr(\text{"yes"} \mid X) = 0.007 / (0.04 + 0.007) = 0.15$
- Therefore, the new instance X will be classified as "no".

Spam Example

Training Data

	t1	t2	t3	t4	t5	Spam
D1	1	1	0	1	0	no
D2	0	1	1	0	0	no
D3	1	0	1	0	1	yes
D4	1	1	1	1	0	yes
D5	0	1	0	1	0	yes
D6	0	0	0	1	1	no
D7	0	1	0	0	0	yes
D8	1	1	0	1	0	yes
D9	0	0	1	1	1	no
D10	1	0	1	0	1	yes

Term	P(t|no)	P(t|yes)
t1	1/4	4/6
t2	2/4	4/6
t3	2/4	3/6
t4	3/4	3/6
t5	2/4	2/6

P(no) = 0.4
P(yes) = 0.6

New email *x* containing t1, t4, t5 ➔ ***x*** **= <1, 0, 0, 1, 1>**

Should it be classified as spam = "yes" or spam = "no"?
Need to find P(yes | *x*) and P(no | *x*) …

Spam Example

New email *x* containing t1, t4, t5

***x* = <1, 0, 0, 1, 1>**

Term	P(t|no)	P(t|yes)
t1	1/4	4/6
t2	2/4	4/6
t3	2/4	3/6
t4	3/4	3/6
t5	2/4	2/6

P(no) = 0.4
P(yes) = 0.6

P(yes | *x*) = [4/6 * (1-4/6) * (1-3/6) * 3/6 * 2/6] * P(yes) / P(*x*)
= [0.67 * 0.33 * 0.5 * 0.5 * 0.33] * 0.6 / P(*x*) = 0.11 / P(*x*)

P(no | *x*) = [1/4 * (1-2/4) * (1-2/4) * 3/4 * 2/4] * P(no) / P(*x*)
= [0.25 * 0.5 * 0.5 * 0.75 * 0.5] * 0.4 / P(*x*) = 0.019 / P(*x*)

To get actual probabilities need to normalize: note that P(yes | x) + P(no | x) must be 1

0.11 / P(x) + 0.019 / P(x) = 1 ➔ P(x) = 0.11 + 0.019 = 0.129

So:

P(yes | *x*) = 0.11 / 0.129 = 0.853

P(no | *x*) = 0.019 / 0.129 = 0.147

Naïve Bayes (Pros)

- Easy to compute and often works well for rough estimates
- Also fast with low storage requirements
- Hence, commonly used and works quite well in many domains – despite assumption of independence
- Also, relatively straight forward to understand and explain
- Performs well in case of categorical input variables compared to numerical variables. For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption)
- Does well in domains with many equally important features (more like KNN)
  - Decision Trees suffer from fragmentation in such cases – especially if little data
- Optimal if the independence assumptions hold: If assumed independence is correct and you need less training data

Naïve Bayes (Cons)

- Often a good first try or baseline ("punching bag") for fancier algorithms
- In real life, it is almost impossible that we get a set of predictors which are completely independent.
- Normal distribution assumption
- Statistical assumptions are more restrictive for numerical values and regression
- Known as a bad estimator: Probability ranking are more accurate than the actual probability estimates (predicting the class but not with a probability)
  - Probability outputs are not to be taken too seriously (what you get with predict_proba in scikit-learn)
- Naive Bayes is a high-bias model and has no variance to minimize - employs a very simple (linear) hypothesis function, the function it uses to model data.

Applications

- Real time Prediction: NB is an **eager learning classifier**, and it is fast.
- Text classification/ Spam Filtering/ Sentiment Analysis (positive or negative customer reviews): due to better result in multi class problems and independence rule, instances aren't related and can be only one of N classes)
- Recommendation System: Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System used to filter unseen information and predict whether a user would like a given resource or not
- Use when
  - Text Classification
  - When dataset is huge, but you have small training set

Improving NB

- If continuous features do not have normal distribution, use transformation to convert it in normal distribution.
- If test data set has zero frequency issue, apply smoothing techniques “Laplace Correction” to predict the class of test data set.
- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.
- Naive Bayes classifiers has limited options for parameter tuning like alpha=1 for smoothing. Worth spending time in pre-processing of data and the feature selection.

NB is Scikit-Learn

The parameters σ_y and μ_y are estimated using maximum likelihood.

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(iris.data, iris.target).predict(iris.data)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (iris.data.shape[0],(iris.target != y_pred).sum()))
Number of mislabeled points out of a total 150 points : 6
```