Naive Bayes

Vibhav Gogate
The University of Texas at Dallas

Supervised Learning: Revisited

- **Given:** Dataset D defined over features $\mathbf{X}$ and desired output variable Y (also called the class variable).
- **Assumption:** There is an unknown function f such that $f(\mathbf{x}) = y$ where $\mathbf{x}$ denotes an assignment of values to all features $\mathbf{X}$ and y denotes an assignment of value to the class variable Y.
- **To do:** Find h using D such that h is the best approximation of f according to some performance measure.
- **Later on:** Use h to find y given $\mathbf{x}$.

A Fully Bayesian Approach

- (Recall: Bayes rule.) Given $\mathbf{x}$, for each $Y = c_i$ compute

$$P(Y = c_i|\mathbf{x}) = \frac{P(\mathbf{x}|Y = c_i)P(Y = c_i)}{P(\mathbf{x})}$$

- Assign to $\mathbf{x}$, the class with the highest probability, namely

$$\text{Class of } \mathbf{x} = \arg\max_{c_i} P(Y = c_i|\mathbf{x})$$

- We do not need to compute the *normalization constant* $P(\mathbf{x})$, namely

$$\text{Class of } \mathbf{x} = \arg\max_{c_i} P(y = c_i|\mathbf{x}) = \arg\max_{c_i} P(\mathbf{x}|Y = c_i)P(Y = c_i)$$

Fully Bayesian Approach as presented is impractical

- Need to estimate (and store) the unconditional distribution $P(y)$ and the conditional joint distribution $P(\mathbf{x}|y = y_i)$ from data D
- Compare:
  - Number of parameters for $P(Y)$ assuming m classes.
    Linear in m.
  - Number of parameters for $P(\mathbf{X} = \mathbf{x}|Y = c_i)$ assuming m classes and n Boolean features. (Namely $\mathbf{x}$ is a $0/1$ vector of size n).
    Exponential in n.

Naive Bayes (Representation)

- Make the following conditional independence assumption: All features are conditionally independent of each other given the class variable.

$$P(\mathbf{x}|Y = c_i) = \prod_{j=1}^{n} P(x_j|Y = c_i)$$

  where $\mathbf{x} = (x_1, \ldots, x_n)$ denotes the assignment of values to all features $X_j \in \mathbf{X}$ such that feature X_j is assigned the value x_j.
- Number of parameters is now linear in m and n. Why?
- Naive Bayes Model Description
  - Class Priors: $P(Y)$.
  - n Conditional Distributions, one associated with each feature X_j: $P(X_j|Y = c_i)$

Maximum Likelihood Estimate (Learning Algorithm)

- Estimate the (conditional) probability tables $P(Y)$ and $P(X_j|Y)$.
- Let $D = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(d)}, y^{(d)})\}$ denote the dataset having d examples. Then the log-likelihood of the data is

$$\log \prod_{k=1}^{d} P(Y = y^{(k)}) \prod_{j=1}^{n} P(x_j^{(k)}|Y = y^{(k)})$$

- Taking derivatives with respect to each parameter and setting them to zero, we get:

$$\text{Estimate of } P(Y = c_i) = \frac{\#(Y = c_i)}{d}$$

$$\text{Estimate of } P(X_j = x_j|Y = c_i) = \frac{\#(Y = c_i, X_j = x_j)}{\#(Y = c_i)}$$

How to classify a test example?

Given a test example $\mathbf{x}$, the class of $\mathbf{x}$

$$
\begin{aligned}
&= \arg\max_{c_i} P(y = c_i|\mathbf{x}) = \arg\max_{c_i} P(Y = c_i)P(\mathbf{x}|Y = c_i) \\
&= \arg\max_{c_i} P(Y = c_i) \prod_{j=1}^{n} P(X_j = x_j|Y = c_i) \\
&= \arg\max_{c_i} \log \left\{ P(Y = c_i) \prod_{j=1}^{n} P(X_j = x_j|Y = c_i) \right\} \\
&= \arg\max_{c_i} \left\{ \log P(Y = c_i) + \sum_{j=1}^{n} \log P(X_j = x_j|Y = c_i) \right\}
\end{aligned}
$$

After estimating, store parameters in log-space: Why?
We are multiplying lots of small numbers. Danger of underflow!
(e.g., on your computer, $0.5^{300} = 4.91 \times 10^{-91}$, $0.5^{3000} = 0$)

Subtleties of Naive Bayes

- Often the conditional independence assumption is violated in practice. Still works surprisingly well!
  - One possible reason: Only need the probability of the correct class to be the largest. For example, in two-way classification, we just need to figure out the correct side of 0.5 and not the actual probability (0.51 is the same as 0.99).
- What if you never see a training instance $(X_j = a, Y = c_i)$ in discrete Naive Bayes?
  Estimate of $P(X_j = a|Y = c_i) = 0/(\text{positive number}) = 0$.
  Solution
  - Use Beta priors or their generalization called Dirichlet priors. Replace the MLE by MAP. Also called Laplace smoothing in literature (see the next slide)

Laplace Smoothing: Fixing the zero estimate problem

- Pretend you saw every outcome $k_{i,j}$ extra times

  MAP Estimate of $P(X_j = x_j | Y = c_i) \propto \#(Y = c_i, X_j = x_j) + k_{i,j}$

- $k_{i,j}$ is the strength of the prior (our prior knowledge)
- Whats Laplace with $k_{i,j} = 0$? (Same as MLE!)
- Usually use the same k for all conditionals, namely it does not depend on i and j. We call this k-laplace smoothing.
  Very Popular: 1-Laplace smoothing where you pretend that you saw every outcome once.

Naive Bayes: Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Class Priors: P(PlayTennis)

Conditionals: P(Outlook|PlayTennis), P(Humidity|PlayTennis), P(Temperature|PlayTennis) and P(Wind|PlayTennis).

Gaussian Naive Bayes

What if the features $X_i \in \mathbf{X}$ are continuous?

Model description:

- Class Priors: $P(Y)$, conditional probability table as before
- $P(X_j = x_j | Y = c_i)$ is given by $\mathcal{N}(\mu_{i,j}, \sigma^2_{i,j})$, a normal distribution with mean $\mu_{i,j}$ and variance $\sigma^2_{i,j}$.

Maximum Likelihood estimates:

$$\text{Estimate of } P(Y = c_i) = \frac{\#(Y = c_i)}{d}$$

$$\text{Estimate of } \mu_{i,j} = \widehat{\mu_{i,j}} = \frac{\sum_{k=1}^{d} x_j^{(k)} \delta(y^{(k)} = c_i)}{\sum_{k=1}^{d} \delta(y^{(k)} = c_i)}$$

$$\text{Estimate of } \sigma^2_{i,j} = \widehat{\sigma^2_{i,j}} = \frac{\sum_{k=1}^{d} \left(x_j^{(k)} - \widehat{\mu_{i,j}}\right)^2 \delta(y^{(k)} = c_i)}{\left(\sum_{k=1}^{d} \delta(y^{(k)} = c_i)\right) - 1}$$

Gaussian Naive Bayes: Special Cases

Sometimes Assume Variance

- is independent of the class (indexed by i in our notation)
- independent of the features (indexed by j in our notation)
- Both

Why do this? More data implies better estimates and Prior knowledge.

How will the MLE/MAP estimates change?

Example: When variance is independent of the class. Namely, when $\sigma^2_{a,j} = \sigma^2_{b,j} = \sigma^2_j$ for all values of a, b.

$$\text{Estimate of } \sigma_j^2 = \widehat{\sigma_j^2} = \frac{\sum_{k=1}^{d} \left(x_j^{(k)} - \widehat{\mu_j} \right)^2}{d-1} \text{ where } \widehat{\mu_j} = \frac{\sum_{k=1}^{d} x_j^{(k)}}{d}$$

Fully Bayesian Approach Revisited

Description of the approach:

- Given **x**, for each $Y = c_i$ compute the conditional probability

$$P(Y = c_i|\mathbf{x}) \propto P(\mathbf{x}|Y = c_i)P(Y = c_i)$$

- Assign to **x**, the class with the highest conditional probability.

A possible view of Naive Bayes:

- Naive Bayes is just one of the many available options for solving the problem of estimating and storing $P(\mathbf{x}|Y = c_i)$. However, it makes **strong assumptions**.

In general, we can solve the problem as follows:

- Use a **compact representation** for $P(\mathbf{x}|Y = c_i)$.
- Develop a fast algorithm that accurately learns the parameters of the chosen representation.

Naive Bayes for Text Classification

- Given labeled documents (by a class value), induce a function f that maps a document to a class value such that a selected performance measure is optimized.
- Feature Engineering: What features to use so that we can convert documents to our assumed data representation (matrix of [examples] $\times$ [features,class]).
- Some options:
  - $(Word, location)$ are our features. (**Sequence matters**)
  - Whether a word appears in a document or not. Position in document does not matter.
  - The number of times a word appears in a document. Again, position in document does not matter. This is called **Bag of Words**.

Feature Engineering: Case 1

Case 1: ($Word, location$) are our features.

- Too many features! Let us say that our documents have maximum length l and our vocabulary size[1] is v, then we will have $l \times v$ features.
  - Realistic value for l: 10 thousand words
  - Realistic value for v: 3 thousand
  - We have 30 million features!
- Most likely value for each feature will be "False" (namely P(feature=True) $\approx$ 0) and thus we will end up using a large number of uninformative features.
- We typically need large amount of data to estimate probabilities which are close to 0 or 1.

[1] all unique words in our training set or words in oxford dictionary plus non-standard words like "lol" and "omg"

Feature Engineering: Case 2

Case 2: Whether a word appears in a document or not. Position in document does not matter.

- Number of features is manageable. Equal to v, the size of the vocabulary.
- Problem: Consider two documents.
  - Document 1 has just one sentence. "I love fishing."
  - Document 2 has the above sentence repeated 1000 times.
  - Both documents will have the same feature values.
- Will work for tasks in which presence of a word is as informative as the number of times a word appears in a document.
- Another Good news: Our Naive Bayes model which uses Bernoulli random variables will work without any modifications.

Feature Engineering: Case 3

Case 3: The number of times a word appears in a document. Again, position in document does not matter. This is called **Bag of Words**.

- Number of features is manageable. Equal to v, the size of the vocabulary.
- *Compare with Case 2.* In Case 2, each feature can take only two values. Here each feature can take l values where l is the maximum length of the document.

We have to be careful when using the conventional Naive Bayes model we have studied so far. Same issue as case 1.

- We need to estimate $O(vlm)$ parameters.
- $P(X_j = i|y) \approx 0$ for large i where $i \in \{0, \ldots, l\}$.

Multinomial Naive Bayes Model: Bag of Words

A better option is the multinomial Naive Bayes model:

$$P(Y = c_i)P(D_k|Y = c_i) \propto P(Y = c_i)\prod_{j=1}^{v}(p_{i,j})^{x_{j,k}}$$

where D_k is the document, $\{X_i, \ldots, X_v\}$ is the set of words in our vocabulary, $p_{i,j}$ is a parameter we want to estimate from data for each word X_j and class c_i, $x_{j,k}$ is the number of times the word X_j appears in D_k and $\sum_{j=1}^{v} p_{i,j} = 1$

Multinomial distribution. General form

$$P(\mathbf{x}) \propto \prod_{j=1}^{v}(p_j)^{x_j}$$

where x_j is the number of times feature X_j appears, $\mathbf{x} = (x_1, \ldots, x_v)$ and p_j is the parameter associated with X_j such that $\forall j\ p_j > 0$ and $\sum_{j=1}^{v} p_j = 1$.

Multinomial Naive Bayes Model: MAP Estimates

$$\text{Estimate of } P(Y = c_i) = \frac{d_{c_i}}{d}$$

where d_{c_i} denotes the number of documents having class c_i and d is the number of documents in the training set.

$$\text{Estimate of } p_{i,j} = \frac{\#(X_j, Y = c_i) + 1}{v + \sum_{t=1}^{v} \#(X_t, Y = c_i)}$$

where $\#(X_j, Y = c_i)$ denotes the number of times the word X_j appears in all documents of class c_i. We are using 1-Laplace smoothing.

Multinomial Naive Bayes: Test Document

Given a test document D_k

- Convert the document D_k to a bag of words representation, namely compute the counts $x_{j,k}$ for each word X_j in the vocabulary.
- Compute the following weight for each class c_i

$$\text{weight of } c_i = P(Y = c_i) \prod_{j=1}^{v} (p_{i,j})^{x_{j,k}}$$

- Return the class having the largest weight.

Multinomial Naive Bayes: Example (Credit: Dan Jurafsky)

► **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$\begin{aligned}
\hat{P}(\text{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{Tokyo}|c) = \hat{P}(\text{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{Chinese}|\bar{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{Tokyo}|\bar{c}) = \hat{P}(\text{Japan}|\bar{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

$$\begin{aligned}
\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003. \\
\hat{P}(\bar{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.
\end{aligned}$$

Generative versus Discriminative Learning

- Naive Bayes is a generative model because you can generate "new data" from it.
  We can use the following algorithm:
  - $y \leftarrow$ Sample a value from $P(Y)$
  - **For** $j = 1$ to n **do**
    - $x_j \leftarrow$ Sample a value from $P(X_j|Y = y)$
  - **Return** $(x_1, \ldots, x_n, y)$
- It solves the classification problem, namely computes $P(Y = y|\mathbf{x})$ using the Bayes rule.

Why not directly learn $P(Y|\mathbf{x})$ from data?

- Classifiers that directly learn $P(Y = y|\mathbf{x})$ from data are called discriminative learners.
- They cannot generate new data because they do not have access to $P(\mathbf{x})$.
- **Next up:** Discriminative Classifiers.