Overview

• Multinomial Naïve Bayes
  • Model
  • Code

• Gaussian Naïve Bayes
  • Mode
  • Code
A Multinomial Bag of Words

- We are given a collection of documents written in a three word language \{a, b, c\}. All the documents have exactly \(n\) words (each word can be either a, b or c).
- We are given a labeled document collection \{D_1, D_2, \ldots, D_m\}. The label \(y_i\) of document \(D_i\) is 1 or 0, indicating whether \(D_i\) is “good” or “bad”.
- This model uses the multinomial distribution. That is, \(a_i, b_i, c_i\) (resp.) is the number of times word \(a\) (\(b\), \(c\), resp.) appears in document \(D_i\).
- Therefore: \(a_i + b_i + c_i = |D_i| = n\).
- In this generative model, we have:
  \[
P(D_i|y = 1) = \frac{n!}{a_i! b_i! c_i!} \alpha_1^{a_i} \beta_1^{b_i} \gamma_1^{c_i}
\]
  where \(\alpha_1, (\beta_1, \gamma_1\) resp.) is the probability that \(a\) (\(b\), \(c\)) appears in a “good” document.
- Similarly,
  \[
P(D_i|y = 0) = \frac{n!}{a_i! b_i! c_i!} \alpha_0^{a_i} \beta_0^{b_i} \gamma_0^{c_i}
\]
- Unlike the discriminative case, the “game” here is different:
  - We make an assumption on how the data is being generated.
  - (multinomial, with \(\alpha_i, (\beta_i, \gamma_i)\))
  - Now, we observe documents, and estimate these parameters.
  - Once we have the parameters, we can predict the corresponding label.
A Multinomial Bag of Words (2)

- We are given a collection of documents written in a three word language \{a, b, c\}. All the documents have exactly \(n\) words (each word can be either a, b or c).

- We are given a labeled document collection \{D_1, D_2, \ldots, D_m\}. The label \(y_i\) of document \(D_i\) is 1 or 0, indicating whether \(D_i\) is “good” or “bad”.

- The classification problem: given a document \(D\), determine if it is good or bad; that is, determine \(P(y|D)\).

- This can be determined via Bayes rule: \(P(y|D) = P(D|y) \ P(y) / P(D)\)

- But, we need to know the parameters of the model to compute that.
A Multinomial Bag of Words (3)

• How do we estimate the parameters?

• We derive the most likely value of the parameters defined above, by maximizing the log likelihood of the observed data.

• \[ PD = \prod_i P(y_i, D_i) = \prod_i P(D_i \mid y_i) P(y_i) = \]

  • We denote by \( P(y_i=1) = \eta \) the probability that an example is “good” (\( y_i=1 \); otherwise \( y_i=0 \)). Then:

\[
\Pi_i P(y_i, D_i) = \prod_i \left[ \left( \eta \frac{n!}{a_i! b_i! c_i!} \alpha_1^{a_i} \beta_1^{b_i} \gamma_1^{c_i} \right)^{y_i} \left( (1-\eta) \frac{n!}{a_i! b_i! c_i!} \alpha_0^{a_i} \beta_0^{b_i} \gamma_0^{c_i} \right)^{1-y_i} \right]
\]

• We want to maximize it with respect to each of the parameters. We first compute \( \log(PD) \) and then differentiate:

\[
\log(PD) = \sum_i y_i \left[ \log(\eta) + C + a_i \log(\alpha_1) + b_i \log(\beta_1^{-1}) + c_i \log(\gamma_1) + \right.
\]

\[
(1- y_i) \left[ \log(1-\eta) + C' + a_i \log(\alpha_0) + b_i \log(\beta_0^{-1}) + c_i \log(\gamma_0) \right]
\]

\[
\frac{d\log(PD)}{\eta} = \sum_i \left[ \frac{y_i}{\eta} - (1-y_i)/(1-\eta) \right] = 0 \quad \Rightarrow \quad \sum_i (y_i - \eta) = 0 \quad \Rightarrow \quad \eta = \frac{\sum_i y_i}{m}
\]

• The same can be done for the other 6 parameters. However, notice that they are not independent: \( \alpha_0 + \beta_0^{-1} + \gamma_0 = \alpha_1 + \beta_1^{-1} + \gamma_1 = 1 \) and also \( a_i + b_i + c_i = |D_i| = n \).
Code

• >>> import numpy as np
• >>> X = np.random.randint(5, size=(6, 100))
• >>> y = np.array([1, 2, 3, 4, 5, 6])
• >>> from sklearn.naive_bayes import MultinomialNB
• >>> clf = MultinomialNB()
• >>> clf.fit(X, y)
• MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
• >>> print(clf.predict(X[2:3]))
• [3]
Naïve Bayes: Continuous Features

- $X_i$ can be continuous
- We can still use

\[
P(X_1, \ldots, X_n | Y) = \prod_i P(X_i | Y)
\]

- And

\[
P(Y = y | X_1, \ldots, X_n) = \frac{P(Y = y) \prod_i P(X_i | Y = y)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}
\]

- Naïve Bayes classifier:

\[
Y = \arg \max_y P(Y = y) \prod_i P(X_i | Y = y)
\]

- Assumption: $P(X_i | Y)$ has a Gaussian distribution
The Gaussian Probability Distribution

- Gaussian probability distribution also called *normal* distribution.
- It is a continuous distribution with pdf:

\[
p(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
\]

- \( \mu = \) mean of distribution
- \( \sigma^2 = \) variance of distribution
- \( x \) is a continuous variable \((-\infty \leq x \leq \infty)\)
- Probability of \( x \) being in the range \([a, b]\) cannot be evaluated analytically (has to be looked up in a table)

- \( p(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \) gaussian
- \( \sigma = \) standard deviation
- 68% of area within \( \pm 1\sigma \)
Naïve Bayes: Continuous Features

• $P(X_i|Y)$ is Gaussian

• Training: estimate mean and standard deviation

$$\mu_i = E[X_i|Y = y]$$
$$\sigma_i^2 = E[(X_i - \mu_i)^2|Y = y]$$

Note that the following slides abuse notation significantly. Since $P(x) = 0$ for continues distributions, we think of $P(X=x|Y=y)$, not as a classic probability distribution, but just as a function $f(x) = N(x, \sigma^2)$. $f(x)$ behaves as a probability distribution in the sense that $\forall x, f(x) \geq 0$ and the values add up to 1. Also, note that $f(x)$ satisfies Bayes Rule, that is, it is true that:

$$f_Y(y|X = x) = f_X(x|Y = y) f_Y(y)/f_X(x)$$
Naïve Bayes: Continuous Features

- $P(X_i|Y)$ is Gaussian

- Training: estimate mean and standard deviation

$$
\mu_i = E[X_i|Y = y]
$$
$$
\sigma_i^2 = E[(X_i - \mu_i)^2|Y = y]
$$

<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1.2</td>
<td>2</td>
<td>.4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.2</td>
<td>1.1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Naïve Bayes: Continuous Features

• $P(X_i|Y)$ is Gaussian

• Training: estimate mean and standard deviation

\[ \mu_i = E[X_i|Y = y] \]
\[ \sigma_i^2 = E[(X_i - \mu_i)^2|Y = y] \]

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1.2</td>
<td>2</td>
<td>0.4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.2</td>
<td>1.1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\mu_1 = E[X_1|Y = 1] = \frac{2 + (-1.2) + 2.2}{3} = 1 \\
\sigma_1^2 = E[(X_1 - \mu_1)|Y = 1] = \frac{(2-1)^2 + (-1.2-1)^2 + (2.2-1)^2}{3} = 2.43
\]
Code

• >>> 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
Reference