Recitation #9
CIS 519

CIS 519 TA Team

Overview

- Multinomial Naïve Bayes
  - Model
  - Code
- Gaussian Naïve Bayes
  - Mode
  - Code

How do we model it?

Our eventual goal will be: Given a document, predict whether it's "good" or "bad"

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 multinominal 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:

$$P(D_i|y = 1) = n!/(a_i!\, b_i!\, c_i!)\ \alpha_1^{a_i}\, \beta_1^{b_i}\, \gamma_1^{c_i}$$

where α_1 (β_1, γ_1 resp.) is the probability that a (b , c) appears in a "good" document.

- Similarly, $P(D_i|y = 0) = n!/(a_i!\, b_i!\, c_i!)\ \alpha_0^{a_i}\, \beta_0^{b_i}\, \gamma_0^{c_i}$

Unlike the discriminative case, the "game" here is different:

- We make an assumption on how the data is being generated.
  - (multinomial, with α_i (β_i, γ_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 $\{D_1, D_2 \dots, D_m\}$. 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

Notice that this is an important trick to write down the joint probability without knowing what the outcome of the experiment is. The ith expression evaluates to $p(D_i, y_i)$
(Could be written as a sum with multiplicative y_i but less convenient)

- How do we estimate the paramete
- We derive the most likely value of th rs defined above, by maximizing the log likelihood of the observed data.
- PD = $\Pi_i P(y_i, D_i) = \Pi_i P(D_i | y_i) P(y_i) =$
  - We denote by $P(y_i=1) = \eta'$ the probability at an example is "good" ($y_i=1$; otherwise $y_i=0$). Then:

Labeled data, assuming that the examples are independent

- $\Pi_i P(y, D_i) = \Pi_i [(\eta' \; n!/(a_i! \, b_i! \, c_i!) \, \alpha_1^{a_i} \beta_1^{b_i} \gamma_1^{c_i})^{y_i} \cdot ((1 - \eta') \; n!/(a_i! \, b_i! \, c_i!) \, \alpha_0^{a_i} \beta_0^{b_i} \gamma_0^{c_i})^{1-y_i}]$
- We want to maximize it with respect to each of the parameters. We first compute log (PD) and then differentiate:
- $\log(PD) = \sum_i y_i \; [\log(\eta') + C + a_i \log(\alpha_1) + b_i \log(\beta_1) + c_i \log(\gamma_1) +$
  $(1 - y_i) [\log(1 - \eta') + C' + a_i \log(\alpha_0) + b_i \log(\beta_0) + c_i \log(\gamma_0)]$

Makes sense?

- $d\log PD/\eta' = \sum_i [y_i / \eta' - (1-y_i)/(1-\eta')] = 0 \Rightarrow \sum_i (y_i - \eta') = 0 \Rightarrow \eta' = \sum_i y_i / m$
- The same can be done for the other 6 parameters. However, notice that they are not independent: $\alpha_0 + \beta_0 + \gamma_0 = \alpha_1 + \beta_1 + \gamma_1 = 1$ 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

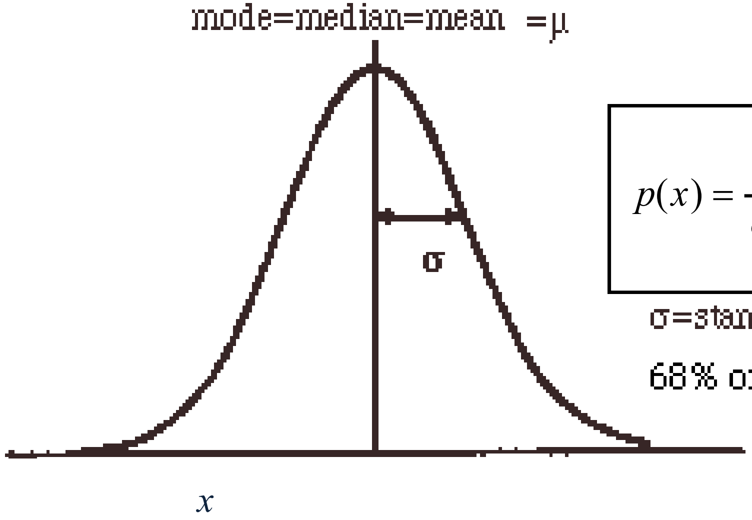
The Gaussian Probability Distribution

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

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

μ = mean of distribution
σ^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)**



$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad gaussian$$

σ=standard deviation
68% of area within ±1σ

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, ¹, ¾²).
f(x) behaves as a probability distribution in the sense that 8 x, f(x) ¸ 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**

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

X_1	X_2	X_3	Y
2	3	1	1
-1.2	2	.4	1
2	0.3	0	0
2.2	1.1	0	1

Naïve Bayes: Continuous Features

- **$P(X_i|Y)$ is Gaussian**

- **Training: estimate mean and standard deviation**

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

X_1	X_2	X_3	Y
2	3	1	1
-1.2	2	.4	1
2	0.3	0	0
2.2	1.1	0	1

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

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

- http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
- http://scikit-learn.org/stable/modules/naive_bayes.html
- http://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html