Bayes and Naïve Bayes

Bayes Classifier

- Generative model learns $p(y)$ and $p(\mathbf{x}|y)$
- Prediction is made by

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

  where $p(\mathbf{x}) = \sum_y p(\mathbf{x}, y)$

- This is often referred to as the Bayes Classifier, because of the use of the Bayes rule
- In LDA we assumed that $p(\mathbf{x}|y)$ is Gaussian with different means and shared covariance matrix

Joint Density estimation

- More generally, learning $p(\mathbf{x}|y)$ is a density estimation problem, which is a challenging task
- Consider the case where $\mathbf{x}$ is a d-dimensional binary vector
- Learning the joint distribution of $p(\mathbf{x}|y)$ involves estimating $K * (2^d - 1)$ parameters
  - For a large d, this number is prohibitively large
  - Typically we don't have enough data to estimate the joint distribution accurately
  - It is common to encounter the situation where no training examples have the exact $\mathbf{x} = [u_1, u_2, \ldots. u_d]^T$ value combination then $p(\mathbf{x}|y) = 0$ for all values of y.

Naïve Bayes Assumption

- Assume that each feature is independent from one another given the class label
- **Definition:** x is **conditionally independent** of y given z, if the probability distribution governing x is independent of the value of y, given the value of z

$$\forall i, j, k \; p(x = i | y = j, z = k) = p(x = i | z = k)$$

Often denoted as $p(x|y, z) = p(x|z)$

- For example:

$$p(thunder|raining, lightening) = p(thunder|lightening)$$

- If x, y are conditional independent given z, we have:

$$p(x, y|z) = p(x|z)p(y|z)$$

Naïve Bayes Classifier

- Under Naïve Bayes assumption, we have:

$$p(\mathbf{x}|y) = \prod_{i=1}^{d} p(x_i|y)$$

- Thus, no need to estimate the joint distribution
- Instead, only need to estimate $p(x_i|y)$ for each feature i
- Example: with d binary features and k classes, we reduce the number of parameters from $k\left(2^d - 1\right)$ to kd
  - Significantly reduces overfitting

Case study: spam filtering

- Bag-of-words to describe emails
- Represent an email by a vector whose dimension = the number of words in our “vocabulary”
- Example: Bernoulli feature
  - x_i=1 if the *i*th word is present
  - x_i=0 if the *i*th word is not present

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{array}{l} \text{a} \\ \text{aardvark} \\ \text{aardwolf} \\ \vdots \\ \text{buy} \\ \vdots \\ \text{zygmurgy} \end{array}$$

- The ordering/position of the words do not matter

MLE for Naïve Bayes with Bernoulli Model

Suppose our training set contained N emails, the maximum likelihood estimate of the parameters are :

$P(y=1)=\frac{N_1}{N}$, where N_1 is the number of spam emails

$P(x_i=1\mid y=1)=\frac{N_{i|1}}{N_1}$,

i.e., the fraction of spam emails where x_i appeared

$P(x_i=1\mid y=0)=\frac{N_{i|0}}{N_0}$

i.e., the fraction of the nonspam emails where x_i appeared

Multinomial Model

- Instead of treating the absence/presence of each word in the dictionary as a Bernoulli, we can treat each word in the email as rolling a $|D|$ sided die, where $|D|$ is the total size of the dictionary
- Each of the $|D|$ words has a fixed probability of being selected
- This allows us to take the counts of the words into consideration

MLE for Naïve Bayes with Multinomial model

- The likelihood of observing one email E:

$$p(y) \prod_{i}^{length\ of\ E} p(x_i|y)$$

- MLE estimate for the i-th word in the dictionary:

$$p_{iy} = \frac{\text{total \# of word } i \text{ in class } y \text{ emails}}{\text{total \# of words in class } y \text{ emails}}$$

- Total number of parameters:
  - $k(|D| - 1)$

Problem with MLE

- Suppose you picked up the new word "Mahalanobis" in your class and started using it in your email **x**
- Because "Mahalanobis" (say it's the n+1 th word in the vocabulary) has never appeared in any of the training emails, the probability estimate for this word will be $p_{n+1|1}=0$ and $p_{n+1|0}=0$
- Now P(**x**|*y*) = $\prod_i P(x_i \mid y) = 0$ for both $y=0$ and $y=1$
- Given limited training data, MLE can result in probabilities of 0 or 1. Such extreme probabilities are "too strong" and cause problems.

Bayesian Parameter Estimation

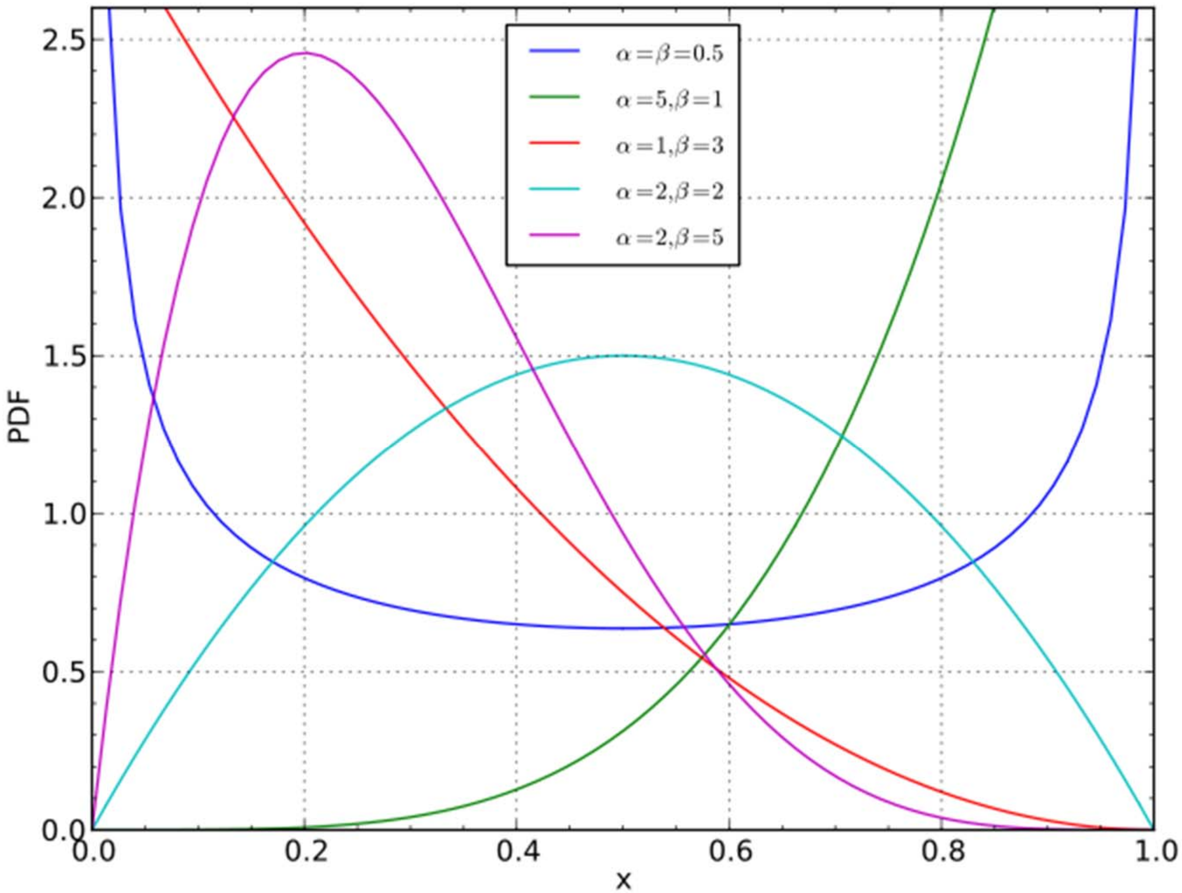
- In the Bayesian framework, we consider the parameters we are trying to estimate to be random variables
- We give them a prior distribution, which is used to capture our prior believe about the parameter
- When the data is sparse and this allows us to fall back to the prior and avoid the issues faced by MLE

Example: Bernoulli

- Given a unfair coin, we want to estimate θ - the probability of head
- We toss the coin n times, and observe n_1 heads
- MLE estimate:$\theta = \frac{n_1}{n}$
- Now let's go Bayesian and assume that θ is a random variable and has a prior distribution
- For reasons that will become clear later, we assume the following prior for θ:

$$\theta \sim Beta(\alpha, \beta), p(\theta; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

Beta distribution



- $\alpha = \beta = 1$: uniform
- $\alpha, \beta < 1$: U-shape
- $\alpha, \beta > 1$: uni-model
- $\alpha = \beta$: symmetric

Posterior distribution of θ

$$p(\theta|\mathrm{D}) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

$$p(\theta|\mathrm{D}) \propto p(D|\theta)p(\theta)$$

$$p(\theta|\mathrm{D}) \propto \theta^{n_1}(1-\theta)^{n_0}\frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

$$p(\theta|\mathrm{D}) \propto \frac{1}{B(\alpha,\beta)}\theta^{n_1+\alpha-1}(1-\theta)^{n_0+\beta-1}$$

$$p(\theta|\mathrm{D}) = \frac{1}{B(n_1+\alpha,n_0+\beta)}\theta^{n_1+\alpha-1}(1-\theta)^{n_0+\beta-1}$$

Noting that the posterior has exactly the same form as the prior. This is not a coincidence, it is due to careful selection of the prior distribution – conjugate prior

Maximum *a-Posterior* (MAP)

- A full Bayesian treatment will not care about estimating the parameter
- Instead, it will use the posterior distribution to make predictions for the target variable
- But if we do need to have a concrete estimate of the parameter
- We can use Maximum A Posterior estimation, that is

$$\theta_{map} = \arg\max_{\theta} p(\theta|D)$$

MAP for Bernoulli

$$p(\theta|\mathrm{D}) = \frac{1}{B(n_1+\alpha, n_0+\beta)} \theta^{n_1+\alpha-1}(1-\theta)^{n_0+\beta-1}$$

- Mode: the maximum point for $Beta(\alpha, \beta)$ is

$$\frac{\alpha-1}{\alpha+\beta-2}$$

- In this case we have:

$$\theta_{MAP} = \frac{n_1+\alpha-1}{n+\alpha+\beta-2}$$

- Let $\alpha = 2, \beta = 2$, we have $\theta_{MAP} = \frac{n_1+1}{n+2}$
- Comparing the MLE, it is like adding some fake coin tosses (one head, one tail) into the observed data – this is also called sometimes as Laplacian smoothing

Laplace Smoothing

- Suppose we estimate a probability P(z) and we have n_0 examples where $z = 0$ and n_1 examples where $z = 1$ MLE estimate is

$$P(z = 1) = \frac{n_1}{n_0 + n_1}$$

- Laplace Smoothing. Add 1 to the numerator and 2 to the denominator

$$P(z=1) = \frac{n_1 + 1}{n_0 + n_1 + 2}$$

If we don't observe any examples, we expect P(z=1) = 0.5, but our belief is weak (equivalent to seeing one example of each outcome).

MAP estimation for Multinomials

- The conjugate prior for multinomial is called Dirichlet distribution
- For K outcomes, a Dirichlet distribution has K parameters, each serves a similar purpose as Beta distribution's parameters
  - Acting as fake observation(s) for the corresponding output, the count depends on the value of the parameter
- Laplace smoothing for this case:

$$P(z=k)=\frac{n_k+1}{n+K}$$

MAP for Naïve Bayes Spam Filter

- When estimating $p(x_i|y=1)$ and $p(x_i|y=0)$
  - Bernoulli case:

$$\underbrace{P(x_i = 1 \mid y = 0)}_{\textbf{MLE}} = \frac{N_{i|0}}{N_0} \Rightarrow \underbrace{P(x_i = 1 \mid y = 0)}_{\textbf{MAP}} = \frac{N_{i|0} + 1}{N_0 + 2}$$

  - Multinomial case:

MLE

$$p(w_i|y=0) = \frac{\text{total \# of word } i \text{ in ns emails}}{\text{total \# of words in ns emails}}$$

MAP

$$p(w_i|y=0) = \frac{\text{total \# of word } i \text{ in ns emails} + 1}{\text{total \# of words in ns emails} + |\text{V}|}$$

where $|V|$ is the size of the vocabulary

- When encounter a new word that has not appeared in training set, now the probabilities do not go to zero

Naïve Bayes Summary

- Generative classifier
  - learn P($\mathbf{x}|y$) and P(y)
  - Use Bayes rule to compute P($y|\mathbf{x}$) for classification
- Assumes conditional independence between features given class labels
  - Greatly reduces the numbers of parameters to learn
- MAP estimation (or Laplace smoothing) is necessary to avoid overfitting and extreme probability values