Lecture 4

Generative Models for Discrete Data - Part 3

Luigi Freda

ALCOR Lab
DIAG
University of Rome "La Sapienza"

October 6, 2017

Outline

Outline

Generative Classifiers vs Discriminative Classifiers

probabilistic classifier

- we are given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$
- the goal is to compute the **class posterior** $p(y = c|\boldsymbol{x})$ which models the mapping $y = f(\boldsymbol{x})$

generative classifiers

- $p(y = c|\boldsymbol{x})$ is computed starting from the **class-conditional density** $p(\boldsymbol{x}|y = c, \boldsymbol{\theta})$ and the **class prior** $p(y = c|\boldsymbol{\theta})$ given that

$$p(y = c|\boldsymbol{x}, \boldsymbol{\theta}) \propto p(\boldsymbol{x}|y = c, \boldsymbol{\theta})p(y = c|\boldsymbol{\theta}) \qquad (= p(y = c, \boldsymbol{x}|\boldsymbol{\theta}))$$

- this is called a **generative classifier** since it specifies how to generate the feature vector $\boldsymbol{x}$ for each class $y = c$ (by using $p(\boldsymbol{x}|y = c, \boldsymbol{\theta})$)
- the model is usually fit by maximizing the joint log-likelihood, i.e. one computes $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_i \log p(y_i, \boldsymbol{x}_i|\boldsymbol{\theta})$

discriminative classifiers

- the model $p(y = c|\boldsymbol{x})$ is directly fit to the data
- the model is usually fit by maximizing the conditional log-likelihood, i.e. one computes $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_i \log p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})$

Naive Bayes Classifiers

Basic Concepts

a Naive Bayes Classifier (**NBC**) uses a generative approach

- let $\boldsymbol{x} = [x_1, ..., x_D]^T$ be our feature vector with D components[1]
- let $y \in \{1, ..., C\}$ where C is the number of classes
- **assumption**: the D **features** are assumed to be **conditionally independent** given the class label, i.e.

$$p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{j=1}^{D} p(x_j|y = c, \theta_{jc})$$

- this is the simplest approach to specify a class-conditional density
- it is called "naive" since we do not actually expect the features to be conditionally independent, even conditional to the class label $y = c$
- even if the naive assumption is not true, NBC often works well given that the model is quite simple and depends on $O(CD)$ parameters and hence is relatively immune to overfitting

[1] one can have $\boldsymbol{x} \in \mathbb{R}^D$ or $\boldsymbol{x} \in \{1, 2, ..., K\}^D$ or $\boldsymbol{x} \in \{0, 1\}^D$

Outline

the form of the class-conditional density depends on the type of each feature

- if $x_j \in \mathbb{R}$ we can use the Gaussian distribution

$$p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{j=1}^{D} \mathcal{N}(x_j|\mu_{jc}, \sigma_{jc}^2)$$

  where for each class c we specify the mean μ_{jc} of feature j and its variance σ_{jc}

- if $x_j \in \{0, 1\}$ we can use the Bernoulli distribution

$$p(\boldsymbol{x}|y = c, \boldsymbol{\theta}) = \prod_{j=1}^{D} Ber(x_j|\mu_{jc})$$

  where for each class c we specify the probability $\mu_{jc} = p(x_j = 1|y = c)$, i.e. the probability that feature j occurs

Naive Bayes Classifiers

Class-Conditional Distributions

- if $x_j \in \{1, ..., K\}$ we can use the categorical distribution

$$p(\mathbf{x}|y = c, \boldsymbol{\theta}) = \prod_{j=1}^{D} Cat(x_j|\boldsymbol{\mu}_{jc})$$

  where for each class c we specify the histogram

$$\boldsymbol{\mu}_{jc} = [p(x_j = 1|y = c), ..., p(x_j = K|y = c)]$$

- other kind of features can be conceived and we can mix different kind of features

Outline

Naive Bayes Classifiers

Likelihood

- **probability for single data case**

$$p(\mathbf{x}_i, y_i|\boldsymbol{\theta}) = p(y_i|\boldsymbol{\pi})p(\mathbf{x}_i|y_i, \boldsymbol{\theta}) = (\text{NBC assumption}) = p(y_i|\boldsymbol{\pi}) \prod_j p(x_{ij}|y_i, \boldsymbol{\theta}_j)$$

  where $\boldsymbol{\theta}$ is a compound vector parameter containing $\boldsymbol{\pi}$ and $\boldsymbol{\theta}_j$

- since $y_i \sim \text{Cat}(\boldsymbol{\pi})$

$$p(y_i|\boldsymbol{\pi}) = \prod_c \pi_c^{\mathbb{I}(y_i=c)}$$

- for each class c we allocate a specific set of parameters $\boldsymbol{\theta}_{jc}$

$$p(x_{ij}|y_i, \boldsymbol{\theta}_j) = \prod_c p(x_{ij}|\boldsymbol{\theta}_{jc})^{\mathbb{I}(y_i=c)}$$

- hence

$$p(\mathbf{x}_i, y_i|\boldsymbol{\theta}) = \prod_c \pi_c^{\mathbb{I}(y_i=c)} \prod_j \prod_c p(x_{ij}|\boldsymbol{\theta}_{jc})^{\mathbb{I}(y_i=c)}$$

- the **log-likelihood** is given by

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i, y_i|\boldsymbol{\theta}) = \sum_{i=1}^{N}\sum_{c=1}^{C} \log \pi_c^{\mathbb{I}(y_i=c)} + \sum_{i=1}^{N}\sum_{j=1}^{D}\sum_{c=1}^{C} \log p(x_{ij}|\boldsymbol{\theta}_{jc})^{\mathbb{I}(y_i=c)}$$

$$= \sum_{c=1}^{C} N_c \log \pi_c + \sum_{j=1}^{D}\sum_{c=1}^{C}\sum_{i:y_i=c} \log p(x_{ij}|\boldsymbol{\theta}_{jc})$$

where $N_c \triangleq \sum_i \mathbb{I}(y_i = c)$ and we assumed as usual that the pairs $(\boldsymbol{x}_i, y_i)$ are iid

Outline

1. Naive Bayes Classifiers
   - Basic Concepts
   - Class-Conditional Distributions
   - Likelihood
   - MLE
   - Bayesian Naive Bayes
   - Prior
   - Posterior
   - MAP
   - Posterior Predictive
   - Plug-in Approximation
   - Log-Sum-Exp Trick
   - Posterior Predictive Algorithm
   - Feature Selection

- the **log-likelihood** is

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{j=1}^{D} \sum_{c=1}^{C} \sum_{i:y_i=c} \log p(x_{ij}|\boldsymbol{\theta}_{jc})$$

here we have the sum of two terms, the first concerning $\boldsymbol{\pi} = [\pi_1, ..., \pi_C]$ and the second concerning DC set of parameters $\boldsymbol{\theta}_{jc}$

- in order to compute the MLE we can optimize the two group of parameters $\boldsymbol{\pi}$ and $\boldsymbol{\theta}_{jc}$ separately

- the **log-likelihood** is

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{j=1}^{D} \sum_{c=1}^{C} \sum_{i:y_i=c} \log p(x_{ij}|\boldsymbol{\theta}_{jc})$$

- the first term concerns the labels $y_i \sim \text{Cat}(\boldsymbol{\pi})$, recall how we computed the MLE of the Dirichlet-multinomial model
- the MLE can be computed by optimizing the Lagrangian

$$l(\boldsymbol{\pi}, \lambda) = \sum_{c} N_c \log \pi_c + \lambda\left(1 - \sum_{c} \pi_c\right)$$

where we enforce the constraint $\sum_c \pi_c = 1$

- we impose $\frac{\partial l}{\partial \pi_c} = 0$, $\frac{\partial l}{\partial \lambda} = 0$ and we obtain the MLE estimation

$$\hat{\pi}_c = \frac{N_c}{N}$$

- the **log-likelihood** is

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{j=1}^{D} \sum_{c=1}^{C} \sum_{i:y_i=c} \log p(x_{ij}|\boldsymbol{\theta}_{jc})$$

- as for the second term optimization, we assume the features x_{ij} are binary, i.e. $x_{ij} \in \{0,1\}$, and $x_{ij}|y = c \sim \text{Ber}(\theta_{jc})$, hence $\boldsymbol{\theta}_{jc} = \theta_{jc} \in [0,1]$
- in this case, we could compute the MLE by using the analysis which was performed with the beta-binomial model
- doing the math again, we have to optimize the function

$$J = \sum_{j=1}^{D} \sum_{c=1}^{C} \sum_{i:y_i=c} \log p(x_{ij}|\theta_{jc}) = \sum_{j=1}^{D} \sum_{c=1}^{C} \sum_{i:y_i=c} \Big(\mathbb{I}(x_{ij} = 1) \log \theta_{jc} + \mathbb{I}(x_{ij} = 0) \log(1-\theta_{jc}) \Big)$$

$$= \sum_{j=1}^{D} \sum_{c=1}^{C} N_{jc} \log \theta_{jc} + \sum_{j=1}^{D} \sum_{c=1}^{C} (N_c - N_{jc}) \log(1 - \theta_{jc})$$

where $N_{jc} \triangleq \sum_i \mathbb{I}(x_{ij} = 1, y_i = c)$ and $N_c \triangleq \sum_i \mathbb{I}(y_i = c)$

- we have to optimize the function

$$J = \sum_{j=1}^{D} \sum_{c=1}^{C} N_{jc} \log \theta_{jc} + \sum_{j=1}^{D} \sum_{c=1}^{C} (N_c - N_{jc}) \log(1 - \theta_{jc})$$

where $N_{jc} \triangleq \sum_i \mathbb{I}(x_{ij} = 1, y_i = c)$ and $N_c \triangleq \sum_i \mathbb{I}(y_i = c)$

- by imposing $\dfrac{\partial J}{\partial \theta_{jc}} = 0$ one obtains the MLE estimate

$$\hat{\theta}_{jc} = \frac{N_{jc}}{N_c}$$

algorithm: MLE fitting a naive Bayes classifier to binary features (i.e. $\boldsymbol{x}_i \in \{0,1\}^D$)

$N_c = 0,\ N_{jc} = 0$;
for $i = 1 : N$ **do**
    $c := y_i$;   `// get the class label of the` i`-th sample`
    $N_c := N_c + 1$;
    for $j = 1 : D$ **do**
        if $x_{ij} = 1$ **then**
            $N_{jc} := N_{jc} + 1$
        end
    end
end
$\hat{\pi}_c = \frac{N_c}{N},\ \hat{\theta}_{jc} = \frac{N_{jc}}{N_c}$;

- see the *naiveBayesFit* script for some Matlab code
- the algorithm takes $O(ND)$ time

Outline

- as we know the MLE estimates can overfit
- recall the black swan paradox and the issue of using empirical fractions N_i/N
- a simple solution to overfitting is to be Bayesian

Outline

- for simplicity we use a factored prior

$$p(\boldsymbol{\theta}) = p(\boldsymbol{\pi}) \prod_{j=1}^{D} \prod_{c=1}^{C} p(\theta_{jc})$$

  where $\boldsymbol{\theta}$ is a compound vector parameter containing $\boldsymbol{\pi}, \theta_{jc}$

- as for the prior of $\boldsymbol{\pi}$ we use

$$p(\boldsymbol{\pi}) = \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})$$

  which is a conjugate prior w.r.t. the multinomial part

- as for the prior of each θ_{jc} we use

$$p(\theta_{jc}) = \text{Beta}(\theta_{jc}|\beta_0, \beta_1)$$

  which is a conjugate prior w.r.t. the binomial part

- we can obtain a uniform prior by setting $\boldsymbol{\alpha} = \mathbf{1}$ and $\beta_0 = \beta_1 = 1$

Outline

The Beta-Binomial Model

Posterior

- factored likelihood

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \log \mathrm{Cat}(y|\boldsymbol{\pi}) + \sum_{j=1}^{D}\sum_{c=1}^{C}\sum_{i:y_i=c} \log \mathrm{Ber}(x_{ij}|\theta_{jc})$$

- factored prior

$$p(\boldsymbol{\theta}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_{j=1}^{D}\prod_{c=1}^{C} \mathrm{Beta}(\theta_{jc}|\beta_0, \beta_1)$$

- factored posterior

$$p(\boldsymbol{\theta}|\mathcal{D}) = p(\boldsymbol{\pi}|\mathcal{D}) \prod_{j=1}^{D}\prod_{c=1}^{C} p(\theta_{jc}|\mathcal{D})$$

$$p(\boldsymbol{\pi}|\mathcal{D}) = \mathrm{Dir}(\boldsymbol{\pi}|N_1 + \alpha_1, ..., N_C + \alpha_C)$$

$$p(\theta_{jc}|\mathcal{D}) = \mathrm{Beta}(\theta_{jc}|N_{jc} + \beta_1, (N_c - N_{jc}) + \beta_0)$$

- to compute the posterior we just updates the empirical counts of the likelihood with the prior counts

Outline

The Beta-Binomial Model

MAP

- factored posterior

$$p(\boldsymbol{\theta}|\mathcal{D}) = p(\boldsymbol{\pi}|\mathcal{D}) \prod_{j=1}^{D} \prod_{c=1}^{C} p(\theta_{jc}|\mathcal{D})$$

$$p(\boldsymbol{\pi}|\mathcal{D}) = \text{Dir}(\boldsymbol{\pi}|N_1 + \alpha_1, ..., N_C + \alpha_C)$$

$$p(\theta_{jc}|\mathcal{D}) = \text{Beta}(\theta_{jc}|N_{jc} + \beta_1, (N_c - N_{jc}) + \beta_0)$$

- MAP estimate of $\boldsymbol{\pi} = [\pi_1, ..., \pi_C]$

$$\hat{\boldsymbol{\pi}} = \arg\max_{\boldsymbol{\pi}} \text{Dir}(\boldsymbol{\pi}|N_1 + \alpha_1, ..., N_C + \alpha_C) \implies \hat{\pi}_c = \frac{N_c + \alpha_c - 1}{N + \alpha_0 - C}$$

- MAP estimate of θ_{jc} for $j \in \{1, ..., D\}$, $c \in \{1, ..., C\}$

$$\hat{\theta}_{jc} = \arg\max_{\theta_{jc}} \text{Beta}(\theta_{jc}|N_{jc} + \beta_1, (N_c - N_{jc}) + \beta_0) \implies \hat{\theta}_{jc} = \frac{N_{jc} + \beta_1 - 1}{N_c + \beta_1 + \beta_0 - 2}$$

algorithm: MAP fitting a naive Bayes classifier to binary features (i.e. $\mathbf{x}_i \in \{0,1\}^D$)

$N_c = 0$, $N_{jc} = 0$;
for $i = 1 : N$ **do**
  $c := y_i$; `// get the class label of the` i`-th sample`
  $N_c := N_c + 1$;
  for $j = 1 : D$ **do**
    if $x_{ij} = 1$ **then**
      $N_{jc} := N_{jc} + 1$
    end
  end
end
$\hat{\pi}_c = \frac{N_c+\alpha_c-1}{N+\alpha_0-C}$, $\hat{\theta}_{jc} = \frac{N_{jc}+\beta_1-1}{N_c+\beta_1+\beta_0-2}$;

Outline

- if we are given a new sample $\boldsymbol{x}$ the **posterior predictive** is

$$p(y = c|\boldsymbol{x}, \mathcal{D}) \propto p(\boldsymbol{x}|y = c, \mathcal{D})p(y = c|\mathcal{D})$$

- with a NBC the class conditional density can factorized as

$$p(\boldsymbol{x}|y = c, \mathcal{D}) = \prod_{j=1}^{D} p(x_j|y = c, \mathcal{D})$$

(since features are assumed to be conditionally independent given the class label)

- combining the two above equations returns

$$p(y = c|\boldsymbol{x}, \mathcal{D}) \propto p(y = c|\mathcal{D}) \prod_{j=1}^{D} p(x_j|y = c, \mathcal{D})$$

- we start from this factorization and we apply the Bayesian procedure

$$p(y = c|\boldsymbol{x}, \mathcal{D}) \propto p(y = c|\mathcal{D}) \prod_{j=1}^{D} p(x_j|y = c, \mathcal{D})$$

- first step, we integrate out the unknown $\boldsymbol{\pi}$ on the first factor

$$p(y = c|\mathcal{D}) = \int p(y = c, \boldsymbol{\pi}|\mathcal{D})d\boldsymbol{\pi} = \int p(y = c|\boldsymbol{\pi}, \mathcal{D})p(\boldsymbol{\pi}|\mathcal{D})d\boldsymbol{\pi} =$$

$$\left(\boldsymbol{\pi} \text{ gives enough information to compute } p(y = c)\right) = \int p(y = c|\boldsymbol{\pi})p(\boldsymbol{\pi}|\mathcal{D})d\boldsymbol{\pi}$$

- second step, we integrate out the unknowns θ_{jc} on each remaining factor

$$p(x_j|y = c, \mathcal{D}) = \int p(x_j, \theta_{jc}|y = c, \mathcal{D})d\theta_{jc} = \int p(x_j|\theta_{jc}, y = c, \mathcal{D})p(\theta_{jc}|y = c, \mathcal{D})d\theta_{jc}$$

$$\left(\text{the new } \boldsymbol{x} \text{ is independent from } \mathcal{D}\right) = \int p(x_j|\theta_{jc}, y = c)p(\theta_{jc}|\mathcal{D})d\theta_{jc}$$

- recollecting everything together returns

$$p(y = c|\mathbf{x}, \mathcal{D}) \propto \left[\int p(y = c|\boldsymbol{\pi}) p(\boldsymbol{\pi}|\mathcal{D}) d\boldsymbol{\pi} \right] \prod_{j=1}^{D} \left[\int p(x_j|\theta_{jc}, y = c) p(\theta_{jc}|\mathcal{D}) d\theta_{jc} \right]$$

and plugging-in the model PDFs/PMFs we adopted

$$p(y = c|\mathbf{x}, \mathcal{D}) \propto \left[\int \text{Cat}(y = c|\boldsymbol{\pi}) \text{Dir}(\boldsymbol{\pi}|N_1 + \alpha_1, ..., N_C + \alpha_C) d\boldsymbol{\pi} \right] \times$$

$$\prod_{j=1}^{D} \left[\int \text{Ber}(x_j|\theta_{jc}, y = c) \text{Beta}(\theta_{jc}|N_{jc} + \beta_1, (N_c - N_{jc}) + \beta_0) d\theta_{jc} \right] =$$

- the first part is a Dirichlet-multinomial model
- the second part is a product of beta-binomial models

- doing the math again for the first part

$$\int \text{Cat}(y = c|\boldsymbol{\pi})\text{Dir}(\boldsymbol{\pi}|N_1 + \alpha_1, ..., N_C + \alpha_C)d\boldsymbol{\pi} =$$

$$\int \pi_c \, \text{Dir}(\boldsymbol{\pi}|N_1 + \alpha_1, ..., N_C + \alpha_C)d\boldsymbol{\pi} = \mathbb{E}[\pi_c|\mathcal{D}] = \frac{N_c + \alpha_c}{N + \alpha_0}$$

where $\alpha_0 = \sum_c \alpha_c$

- this is exactly how we computed the **posterior mean** for the Dirichlet-multinomial model

$$\overline{\pi}_c = \mathbb{E}[\pi_c|\mathcal{D}] = \frac{N_c + \alpha_c}{N + \alpha_0}$$

- doing the math again for the second part

$$\int \text{Ber}(x_j|\theta_{jc}, y = c)\text{Beta}(\theta_{jc}|N_{jc} + \beta_1, (N_c - N_{jc}) + \beta_0)d\theta_{jc} =$$

$$= \int \theta_{jc}^{\mathbb{I}(x_j=1)}(1 - \theta_{jc})^{\mathbb{I}(x_j=0)}\text{Beta}(\theta_{jc}|N_{jc} + \beta_1, (N_c - N_{jc}) + \beta_0)d\theta_{jc} =$$

$$= (\overline{\theta}_{jc})^{\mathbb{I}(x_j=1)}(1 - \overline{\theta}_{jc})^{\mathbb{I}(x_j=0)}$$

where

$$\overline{\theta}_{jc} = \mathbb{E}[\theta_{jc}|\mathcal{D}] = \frac{N_{jc} + \beta_1}{N_c + \beta_0 + \beta_1}$$

- in the above equations we first worked on $x_j = 1$ and then on $x_j = 0$
- this is exactly how we computed the **posterior mean** for the beta-binomial model

- the final **posterior predictive** is

$$p(y = c|\mathbf{x}, \mathcal{D}) \propto \overline{\pi}_c \prod_{j=1}^{D} (\overline{\theta}_{jc})^{\mathbb{I}(x_j=1)} (1 - \overline{\theta}_{jc})^{\mathbb{I}(x_j=0)}$$

with the **posterior means**

$$\overline{\theta}_{jc} = \mathbb{E}[\theta_{jc}|\mathcal{D}] = \frac{N_{jc} + \beta_1}{N_c + \beta_0 + \beta_1}$$

and

$$\overline{\pi}_c = \mathbb{E}[\pi_c|\mathcal{D}] = \frac{N_c + \alpha_c}{N + \alpha_0}$$

Outline

- we can approximate the posterior with a single point, i.e. $p(\boldsymbol{\theta}|\mathcal{D}) \approx \delta_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\theta})$ where $\hat{\boldsymbol{\theta}}$ can be the MAP or the MLE
- we obtain in this case a **plug-in approximation**

$$p(y = c|\mathbf{x}, \mathcal{D}) \propto \hat{\pi}_c \prod_{j=1}^{D} (\hat{\theta}_{jc})^{\mathbb{I}(x_j=1)} (1 - \hat{\theta}_{jc})^{\mathbb{I}(x_j=0)}$$

- the plug-in approximation is obviously more prone to overfitting

Outline

- the posterior predictive has the following form

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

- $p(\mathbf{x}|y=c)$ is often a very small number, especially if $\mathbf{x}$ is a high-dimensional vector, since we have to enforce $\sum_{\mathbf{x}'} p(\mathbf{x}'|y=c) = 1$
- this entails that a naive implementation of the posterior predictive can fail due to **numerical underflow**
- the obvious solution is to use logs

$$\log p(y=c|\mathbf{x}) = \log p(\mathbf{x}|y=c) + \log p(y=c) - \log p(\mathbf{x})$$

and if we define $b_c \triangleq \log p(\mathbf{x}|y=c) + \log p(y=c)$, one has

$$\log p(y=c|\mathbf{x}) = b_c - \log \left[\sum_{c'} e^{b_{c'}} \right]$$

- with $b_c \triangleq \log p(\boldsymbol{x}|y=c) + \log p(y=c)$ we have

$$\log p(y=c|\boldsymbol{x}) = b_c - \log \left[\sum_{c'} e^{b_{c'}} \right]$$

- now we have the problem that computing $e^{b_{c'}}$ can cause an overflow[2]
- we can use the **log-sum-exp trick** in order to avoid this problem

$$\log \left[\sum_c e^{b_c} \right] = \log \left[\left(\sum_c e^{b_c - B} \right) e^B \right] = \log \left[\sum_c e^{b_c - B} \right] + B$$

  where $B \triangleq \max_c b_c$
- with this trick the biggest term $e^{b_c - B}$ equals zero

[2] since $b_{c'}$ can be a big number

Outline

- the computed posterior predictive is

$$p(y = c|\boldsymbol{x}, \mathcal{D}) \propto \overline{\pi}_c \prod_{j=1}^{D} (\overline{\theta}_{jc})^{\mathbb{I}(x_j=1)} (1 - \overline{\theta}_{jc})^{\mathbb{I}(x_j=0)}$$

- if we apply the log we obtain

$$\log p(y = c|\boldsymbol{x}, \mathcal{D}) \propto \log \overline{\pi}_c + \sum_{j=1}^{D} \mathbb{I}(x_j = 1) \log(\overline{\theta}_{jc}) + \mathbb{I}(x_j = 0) \log(1 - \overline{\theta}_{jc})$$

- the above log-posterior is the basis for the next algorithm

algorithm: predicting with a naive Bayes classifier for binary features (i.e. $\mathbf{x}_i \in \{0,1\}^D$)

for $c = 1 : C$ **do**
    $L_c := \log \hat{\pi}_c$;
    for $j = 1 : D$ **do**
        if $x_j = 1$ **then**
            $L_c := L_c + \log \hat{\theta}_{jc}$
        else
            $L_c := L_c + \log(1 - \hat{\theta}_{jc})$
        end
    end
    $p_c := \exp(L_c - \text{logsumexp}(L_{1:C}))$;   `// compute` $p(y = c|\mathbf{x}, \mathcal{D})$
end
$\hat{y} := \arg\max_c \; p_c$;

- the above algorithm computes $\hat{y} = \arg\max_c \; p(y = c|\mathbf{x}, \mathcal{D})$
- the used parameter estimate $\hat{\boldsymbol{\theta}}$ can be obviously best replaced with the posterior mean $\overline{\boldsymbol{\theta}}$ as shown in the computation of the full posterior predictive

Outline

Feature Selection

By using Mutual Information

- an NBC is commonly used to fit a joint distribution over potentially many features
- the NBC fitting algorithm is $O(ND)$ where N is the dataset size and D is the size of $\boldsymbol{x}$
- problems: D can be very high and NBC may suffer from overfitting
- a common approach to reduce these problems is to perform **feature selection**:
  1. evaluate the relevance of each feature
  2. hold only the K most relevant features (K is chosen based on some **tradeoff accuracy-complexity**)

- correlation is a very limited measure of dependence; revise the slides about correlation and independence (lecture 3 part 2)
- a more general approach is to determine how similar is a joint distribution $p(X, Y)$ to $p(X)p(Y)$ (recall the definition $X \perp Y$)
- **mutual information** (MI)

$$\mathbb{I}[X; Y] \triangleq \mathbb{KL}[p(X, Y)||p(X)p(Y)] = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

- one has $\mathbb{I}[X; Y] \geq 0$ with equality **iff** $p(X, Y) = p(X)p(Y)$

- we want to measure the relevance between feature X_j and the class label Y

$$\mathbb{I}[X_j; Y] = \sum_{x_j} \sum_{y} p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)}$$

- for an NBC classifier with binary features one has (**homework** ex 3.21)

$$I_j \triangleq \mathbb{I}[X_j; Y] = \sum_{c} \left[\theta_{jc}\pi_c \log \frac{\theta_{jc}}{\theta_j} + (1 - \theta_{jc})\pi_c \log \frac{1 - \theta_{jc}}{1 - \theta_c} \right]$$

  where the following quantities are computed by the NBC fitting algorithm: $\pi_c = p(y = c)$, $\theta_{jc} = p(x_j = 1|y = c)$ and $\theta_j = p(x_j = 1) = \sum_c \pi_c \theta_{jc}$

- the top K features with the highest I_j can then be selected and used

Credits

- Kevin Murphy's book