Naïve Bayes

CS 760@UW-Madison

Goals for the lecture

- understand the concepts
  - generative/discriminative models
  - examples of the two approaches
  - MLE (Maximum Likelihood Estimation)
  - Naïve Bayes
    - Naïve Bayes assumption
    - model 1: Bernoulli Naïve Bayes
    - model 2: Multinomial Naïve Bayes
    - model 3: Gaussian Naïve Bayes
    - model 4: Multiclass Naïve Bayes

Review: supervised learning

problem setting

- set of possible instances: X
- unknown *target function* (concept): $f : X \rightarrow Y$
- set of *hypotheses* (hypothesis class): $H = \{h \mid h : X \rightarrow Y\}$

given

- *training set* of instances of unknown target function f

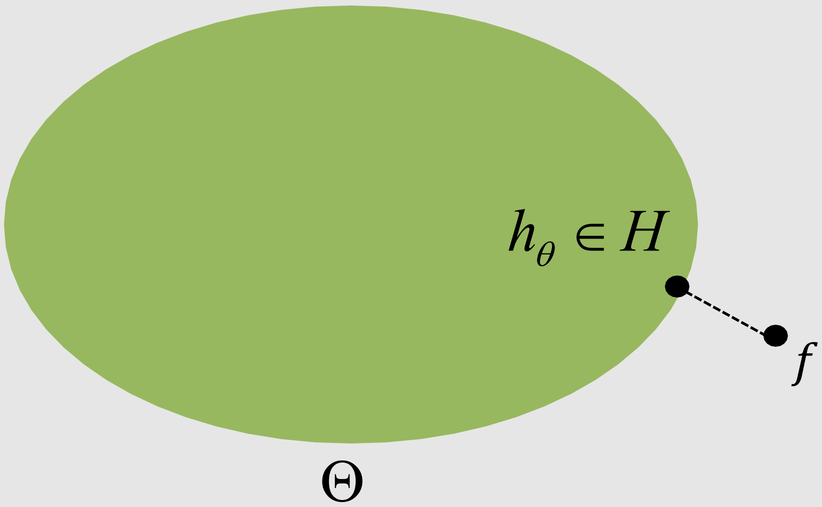
$$\left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right) \ \dots \ \left(\mathbf{x}^{(m)}, y^{(m)}\right)$$

output

- hypothesis $h \in H$ that best approximates target function

Parametric hypothesis class

- hypothesis $h \in H$ is indexed by (fixed dimensional) parameter $\theta \in \Theta$
- learning: find the θ such that $h_\theta \in H$ best approximate the target



- different from nonparametric approaches like decision trees and nearest neighbor
- advantages: various hypothesis class; easier to use math/optimization

Discriminative approaches

- hypothesis $h \in H$ directly predicts the label y given the features x

$$y = h(x) \text{ or more generally, } p(y \mid x) = h(x)$$

- then define a loss function $L(h)$ and find hypothesis with min. loss
  - A special case is a probabilistic model, finding MLE or MAP

- example: linear regression

$$h_\theta(x) = \langle x, \theta \rangle$$

$$L(h_\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Generative approaches

- hypothesis $h \in H$ specifies a **generative probabilistic story** for how the full data (x,y) was created

$$h(x, y) = p(x, y)$$

- then pick a hypothesis by maximum likelihood estimation (MLE) or Maximum A Posteriori (MAP)

- example: roll a weighted die
- weights for each side (θ) define how the data are generated
- use MLE on the training data to learn θ

Comments on discriminative/generative

- Orthogonal to the parametric / nonparametric divide
  - nonparametric Bayesian: a large subfield of ML

- when discriminative/generative is likely to be better? Discussed in later lecture

- typical discriminative: linear regression, logistic regression, SVM, many neural networks (not all!), …
- typical generative: Naïve Bayes, Bayesian Networks, …

MLE and MAP

MLE vs. MAP

Suppose we have data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$

$$\boldsymbol{\theta}^{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \prod_{i=1}^{N} p(\mathbf{x}^{(i)} | \boldsymbol{\theta})$$

Maximum Likelihood Estimate (MLE)

Background: MLE

Example: MLE of Exponential Distribution

- pdf of Exponential(λ): $f(x) = \lambda e^{-\lambda x}$
- Suppose $X_i \sim$ Exponential(λ) for $1 \leq i \leq N$.
- Find MLE for data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$
- First write down log-likelihood of sample.
- Compute first derivative, set to zero, solve for λ.
- Compute second derivative and check that it is concave down at λ^{MLE}.

Background: MLE

Example: MLE of Exponential Distribution

- First write down log-likelihood of sample.

$$\ell(\lambda) = \sum_{i=1}^{N} \log f(x^{(i)}) \tag{1}$$

$$= \sum_{i=1}^{N} \log(\lambda \exp(-\lambda x^{(i)})) \tag{2}$$

$$= \sum_{i=1}^{N} \log(\lambda) + -\lambda x^{(i)} \tag{3}$$

$$= N \log(\lambda) - \lambda \sum_{i=1}^{N} x^{(i)} \tag{4}$$

Background: MLE

Example: MLE of Exponential Distribution

- Compute first derivative, set to zero, solve for λ.

$$\frac{d\ell(\lambda)}{d\lambda} = \frac{d}{d\lambda} N \log(\lambda) - \lambda \sum_{i=1}^{N} x^{(i)} \qquad (1)$$

$$= \frac{N}{\lambda} - \sum_{i=1}^{N} x^{(i)} = 0 \qquad (2)$$

$$\boxed{\Rightarrow \lambda^{\text{MLE}} = \frac{N}{\sum_{i=1}^{N} x^{(i)}}} \qquad (3)$$

MLE vs. MAP

Suppose we have data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$

Maximum Likelihood Estimate (MLE)

$$\boldsymbol{\theta}^{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\boldsymbol{\theta})$$

Maximum *a posteriori* (MAP) estimate

$$\boldsymbol{\theta}^{\text{MAP}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\boldsymbol{\theta})\underbrace{p(\boldsymbol{\theta})}_{\text{Prior}}$$

Naïve Bayes

Model 0: Not-so-naïve Model?

Generative Story:

1. Flip a weighted coin (Y)
2. If heads, roll the **yellow** many sided die to sample a document vector (X) from the Spam distribution
3. If tails, roll the **blue** many sided die to sample a document vector (X) from the Not-Spam distribution

$$P(X_1, \ldots, X_K, Y) = P(X_1, \ldots, X_K | Y) P(Y)$$

This model is computationally naïve!

Model 0: Not-so-naïve Model?

Generative Story:

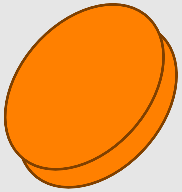
1. Flip a weighted coin (Y)
2. If heads, sample a document ID (X) from the Spam distribution
3. If tails, sample a document ID (X) from the Not-Spam distribution

$$P(X, Y) = P(X|Y)P(Y)$$

This model is computationally naïve!

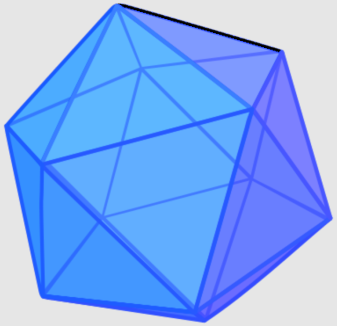
Model 0: Not-so-naïve Model?

Flip weighted coin



If HEADS, roll yellow die

If TAILS, roll blue die



y	x_1	x_2	x_3	…	x_K
0	1	0	1	…	1
1	0	1	0	…	1
1	1	1	1	…	1
0	0	0	1	…	1
0	1	0	1	…	0
1	1	0	1	…	0

Each side of the die is labeled with a document vector (e.g. [1,0,1,…,1])

Naïve Bayes Assumption

Conditional independence of features:

$$P(X_1, \ldots, X_K, Y) = \boxed{P(X_1, \ldots, X_K | Y)} P(Y)$$

$$= \boxed{\left(\prod_{k=1}^{K} P(X_k | Y) \right)} P(Y)$$

Estimating a joint from conditional probabilities

C	P(C)
0	0.33
1	0.67

$$P(A,B|C) = P(A|C) * P(B|C)$$

$$\forall a, bc : P(A=a \wedge B=b|C=c) = P(A=a|C=c) * P(B=b|C=c)$$

A	C	P(A|C)
0	0	0.2
0	1	0.5
1	0	0.8
1	1	0.5

B	C	P(B|C)
0	0	0.1
0	1	0.9
1	0	0.9
1	1	0.1

A	B	C	P(A,B,C)
0	0	0	…
0	0	1	…
0	1	0	…
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Estimating a joint from conditional probabilities

C	P(C)
0	0.33
1	0.67

A	C	P(A|C)
0	0	0.2
0	1	0.5
1	0	0.8
1		

B	C	P(B|C)
0	0	0.1
0	1	0.9
1	0	0.9
1	1	0.1

D	C	P(D|C)
0	0	0.1
0	1	0.1
1	0	0.9
1	1	0.1

A	B	D	C	P(A,B,D,C)
0	0	0	0	…
0	0	1	0	…
0	1	0	0	…
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	0	
0	0	0	1	
0	0	1	0	
…	…	..	…	…

Assuming conditional independence, the conditional probabilities encode the **same information** as the joint table.

They are very convenient for estimating

$P(X_1,\dots,X_n|Y)=P(X_1|Y)*\dots*P(X_n|Y)$

They are almost as good for computing

$$P(Y|X_1,\dots,X_n)=\frac{P(X_{1,}\dots,X_n|Y)P(Y)}{P(X_1,\dots,X_n)}$$

$$\forall \mathbf{x},y: P(Y=y|X_1,\dots,X_n=\mathbf{x})=\frac{P(X_{1,}\dots,X_n=\mathbf{x}|Y)P(Y=y)}{P(X_1,\dots,X_n=\mathbf{x})}$$

Generic Naïve Bayes Model

Support: Depends on the choice of **event model,** $P(X_k|Y)$

Model: Product of **prior** and the event model

$$P(\mathbf{X}, Y) = P(Y) \prod_{k=1}^{K} P(X_k|Y)$$

Training: Find the **class-conditional** MLE parameters

For $P(Y)$, we find the MLE using all the data. For each $P(X_k|Y)$ we condition on the data with the corresponding

Classification: Find the class that maximizes the posterior

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, p(y|\mathbf{x})$$

Generic Naïve Bayes Model

Classification:

$$\hat{y} = \operatorname*{argmax}_y p(y|\mathbf{x}) \quad \text{(posterior)}$$

$$= \operatorname*{argmax}_y \frac{p(\mathbf{x}|y)p(y)}{p(x)} \quad \text{(by Bayes' rule)}$$

$$= \operatorname*{argmax}_y p(\mathbf{x}|y)p(y)$$

Various Naïve Bayes Models

Model 1: Bernoulli Naïve Bayes

Support: Binary vectors of length K

$$\mathbf{x} \in \{0,1\}^K$$

Generative Story:

$$Y \sim \text{Bernoulli}(\phi)$$

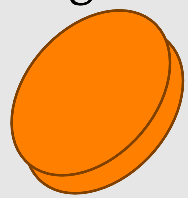
$$X_k \sim \text{Bernoulli}(\theta_{k,Y}) \;\forall k \in \{1,\ldots,K\}$$

Model:

$$\begin{aligned} p_{\phi,\boldsymbol{\theta}}(\boldsymbol{x}, y) &= p_{\phi,\boldsymbol{\theta}}(x_1, \ldots, x_K, y) \\ &= p_\phi(y) \prod_{k=1}^{K} p_{\boldsymbol{\theta}_k}(x_k | y) \\ &= (\phi)^y (1-\phi)^{(1-y)} \prod_{k=1}^{K} (\theta_{k,y})^{x_k} (1-\theta_{k,y})^{(1-x_k)} \end{aligned}$$

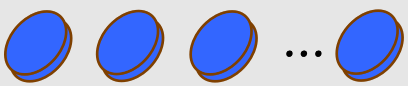
Model 1: Bernoulli Naïve Bayes

Flip weighted coin



If HEADS, flip each yellow coin

If TAILS, flip each blue coin



y	x_1	x_2	x_3	…	x_K
0	1	0	1	…	1
1	0	1	0	…	1
1	1	1	1	…	1
0	0	0	1	…	1
0	1	0	1	…	0
1	1	0	1	…	0

Each red coin corresponds to an x_k

We can **generate** data in this fashion. Though in practice we never would since our data is **given.**

Instead, this provides an explanation of **how** the data was generated (albeit a terrible one).

Model 1: Bernoulli Naïve Bayes

Support: Binary vectors of length K

$$\mathbf{x} \in \{0,1\}^K$$

Generative Story:

$$Y \sim \text{Bernoulli}(\phi)$$

$$X_k \sim \text{Bernoulli}(\theta_{k,Y}) \; \forall k \in \{1, \ldots, K\}$$

Same as Generic Naïve Bayes

Model: $p_{\phi,\boldsymbol{\theta}}(\boldsymbol{x}, y) = (\phi)^y (1-\phi)^{(1-y)} \prod$

Classification: Find the class that maximizes the posterior

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, p(y|\mathbf{x})$$

Generic Naïve Bayes Model

Recall...

Classification:

$$\hat{y} = \operatorname*{argmax}_y p(y|\mathbf{x}) \quad \text{(posterior)}$$

$$= \operatorname*{argmax}_y \frac{p(\mathbf{x}|y)p(y)}{p(x)} \quad \text{(by Bayes' rule)}$$

$$= \operatorname*{argmax}_y p(\mathbf{x}|y)p(y)$$

Model 1: Bernoulli Naïve Bayes

Training: Find the **class-conditional** MLE parameters

For *P(Y)*, we find the MLE using all the data. For each *P(X_k|Y)* we condition on the data with the corresponding class.

$$\phi = \frac{\sum_{i=1}^{N} \mathbb{I}(y^{(i)} = 1)}{N}$$

$$\theta_{k,0} = \frac{\sum_{i=1}^{N} \mathbb{I}(y^{(i)} = 0 \wedge x_k^{(i)} = 1)}{\sum_{i=1}^{N} \mathbb{I}(y^{(i)} = 0)}$$

$$\theta_{k,1} = \frac{\sum_{i=1}^{N} \mathbb{I}(y^{(i)} = 1 \wedge x_k^{(i)} = 1)}{\sum_{i=1}^{N} \mathbb{I}(y^{(i)} = 1)}$$

$$\forall k \in \{1, \ldots, K\}$$

Model 2: Multinomial Naïve Bayes

Support: Integer vector (word IDs)

$\mathbf{x} = [x_1, x_2, \ldots, x_M]$ where $x_m \in \{1, \ldots, K\}$ a word id.

Generative Story:

for $i \in \{1, \ldots, N\}$**:**

$y^{(i)} \sim$ Bernoulli(ϕ)

for $j \in \{1, \ldots, M_i\}$**:** (Assume $M_i = M$ for all i)

$x_j^{(i)} \sim$ Multinomial$(\boldsymbol{\theta}_{y^{(i)}}, 1)$

Model:

$$p_{\phi,\boldsymbol{\theta}}(\boldsymbol{x}, y) = p_\phi(y) \prod_{k=1}^{K} p_{\boldsymbol{\theta}_k}(x_k | y)$$

$$= (\phi)^y (1-\phi)^{(1-y)} \prod_{j=1}^{M_i} \theta_{y, x_j}$$

Model 3: Gaussian Naïve Bayes

Support:

$$\mathbf{x} \in \mathbb{R}^K$$

Model: Product of **prior** and the event model

$$p(\boldsymbol{x}, y) = p(x_1, \ldots, x_K, y)$$
$$= p(y) \prod_{k=1}^{K} p(x_k | y)$$

Gaussian Naive Bayes assumes that $p(x_k|y)$ is given by a Normal distribution.

Model 4: Multiclass Naïve Bayes

Model:

The only change is that we permit y to range over C classes.

$$p(\boldsymbol{x}, y) = p(x_1, \ldots, x_K, y)$$

$$= p(y) \prod_{k=1}^{K} p(x_k | y)$$

Now, $y \sim$ Multinomial$(\phi, 1)$ and we have a separate conditional distribution $p(x_k|y)$ for each of the C classes.

THANK YOU

Some of the slides in these lectures have been adapted/borrowed from materials developed by Yingyu Liang, Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Matt Gormley, Elad Hazan, Tom Dietterich, and Pedro Domingos.