Parametric Density Estimation:

Bayesian Estimation.

Naïve Bayes Classifier

Bayesian Parameter Estimation

- Suppose we have some idea of the range where parameters θ should be
  - Shouldn't we formalize such prior knowledge in hopes that it will lead to better parameter estimation?
- Let θ be a random variable with prior distribution $\boldsymbol{P}(\theta)$
  - This is the key difference between ML and Bayesian parameter estimation
  - This key assumption allows us to fully exploit the information provided by the data

Bayesian Parameter Estimation

- θ is a random variable with prior $\boldsymbol{p}(\theta)$
  - Unlike MLE case, $p(x|\theta)$ is a conditional density
- The training data D allow us to convert $p(\theta)$ to a posterior probability density $p(\theta|D)$.
  - After we observe the data D, using Bayes rule we can compute the posterior $p(\theta|D)$
- But θ is not our final goal, our final goal is the unknown $\boldsymbol{p}(\boldsymbol{x})$
- Therefore a better thing to do is to maximize $\boldsymbol{p}(\boldsymbol{x}|\boldsymbol{D})$, this is as close as we can come to the unknown $\boldsymbol{p}(\boldsymbol{x})$!

Bayesian Estimation: Formula for $p(x|D)$

- From the definition of joint distribution:

$$p(\mathbf{x} \mid D) = \int p(\mathbf{x}, \theta \mid D)\, d\theta$$

- Using the definition of conditional probability:

$$p(\mathbf{x} \mid D) = \int p(\mathbf{x} \mid \theta, D)\, p(\theta \mid D)\, d\theta$$

- But $p(\mathbf{x}|\theta, D) = p(\mathbf{x}|\theta)$ since $p(\mathbf{x}|\theta)$ is completely specified by θ

$$p(\mathbf{x} \mid D) = \int \overset{\text{known}}{\boxed{p(\mathbf{x} \mid \theta)}}\, \overset{\text{unknown}}{\boxed{p(\theta \mid D)}}\, d\theta$$

- Using Bayes formula,

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{\int p(D \mid \theta)\, p(\theta)\, d\theta} \qquad p(D \mid \theta) = \prod_{k=1}^{n} p(\mathbf{x}_k \mid \theta)$$

Bayesian Estimation vs. MLE

- So in principle ***p***(***x***|***D***) can be computed
  - In practice, it may be hard to do integration analytically, may have to resort to numerical methods

$$p(x \mid D) = \int p(x \mid \theta) \frac{\prod_{k=1}^{n} p(x_k \mid \theta) p(\theta)}{\int \prod_{k=1}^{n} p(x_k \mid \theta) p(\theta) d\theta} d\theta$$

- Contrast this with the MLE solution which requires differentiation of likelihood to get $p(x \mid \hat{\theta})$
  - Differentiation is easy and can always be done analytically

Bayesian Estimation vs. MLE

$$p(x \mid D) = \int \underbrace{p(x \mid \theta)}_{\text{proposed model with certain } \theta} \overbrace{p(\theta \mid D)}^{\text{support } \theta \text{ receives from the data}} d\theta$$

- The above equation implies that if we are less certain about the exact value of **θ,** we should consider a weighted average of *p(***x***|***θ***)* over the possible values of **θ.**

- Contrast this with the MLE solution which always gives us a single model:

$$p(x \mid \hat{\theta})$$

Bayesian Estimation for Gaussian with unknown μ

- Let $\boldsymbol{p}(\boldsymbol{x}|\ \mu)$ be $\boldsymbol{N}(\mu, \sigma^2)$ that is σ^2 is known, but μ is unknown and needs to be estimated, so $\theta = \mu$

- Assume a prior over μ **:** $p(\mu) \sim N(\mu_0, \sigma_0^2)$

- μ_0 encodes some prior knowledge about the true mean μ , while σ_0^2 measures our prior uncertainty.

Bayesian Estimation for Gaussian with unknown μ

- The posterior distribution is:

$$p(\mu \mid D) \propto p(D \mid \mu) p(\mu)$$

$$= \alpha' \exp\left[-\frac{1}{2}\left(\sum_{k=1}^{n}\left(\frac{x_k-\mu}{\sigma}\right)^2+\left(\frac{\mu-\mu_0}{\sigma_0}\right)^2\right)\right]$$

$$= \alpha'' \exp\left[-\frac{1}{2}\left[\left(\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2}\right)\mu^2-2\left(\frac{1}{\sigma^2}\sum_{k=1}^{n}x_k+\frac{\mu_0}{\sigma_0^2}\right)\mu\right]\right]$$

- Where factors that do not depend on μ have been absorbed into the constants α' and α''
- $p(\mu \mid D)$ is an exponent of a quadratic function of μ i.e. it is a normal density; it remains normal for any number of training samples.
- If we write

$$p(\mu \mid D) = \frac{1}{\sqrt{2\pi}\sigma_n}\exp\left[-\frac{1}{2}\left(\frac{\mu-\mu_n}{\sigma_n}\right)^2\right]; \quad \alpha'' \exp\left[-\frac{1}{2}\left[\left(\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2}\right)\mu^2-2\left(\frac{1}{\sigma^2}\sum_{k=1}^{n}x_k+\frac{\mu_0}{\sigma_0^2}\right)\mu\right]\right]$$

then identifying the coefficients, we get $\displaystyle \frac{1}{\sigma_n^2}=\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2} \qquad \frac{\mu_n}{\sigma_n^2}=\frac{n}{\sigma^2}\hat{\mu}_n+\frac{\mu_0}{\sigma_0^2}$

where $\displaystyle \hat{\mu}_n=\frac{1}{n}\sum_{k=1}^{n}x_k$

Bayesian Estimation for Gaussian with unknown μ

- Solving explicitly for μ_n and σ_n^2 we obtain:

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \right) \hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2 + \sigma^2} \mu_0$$

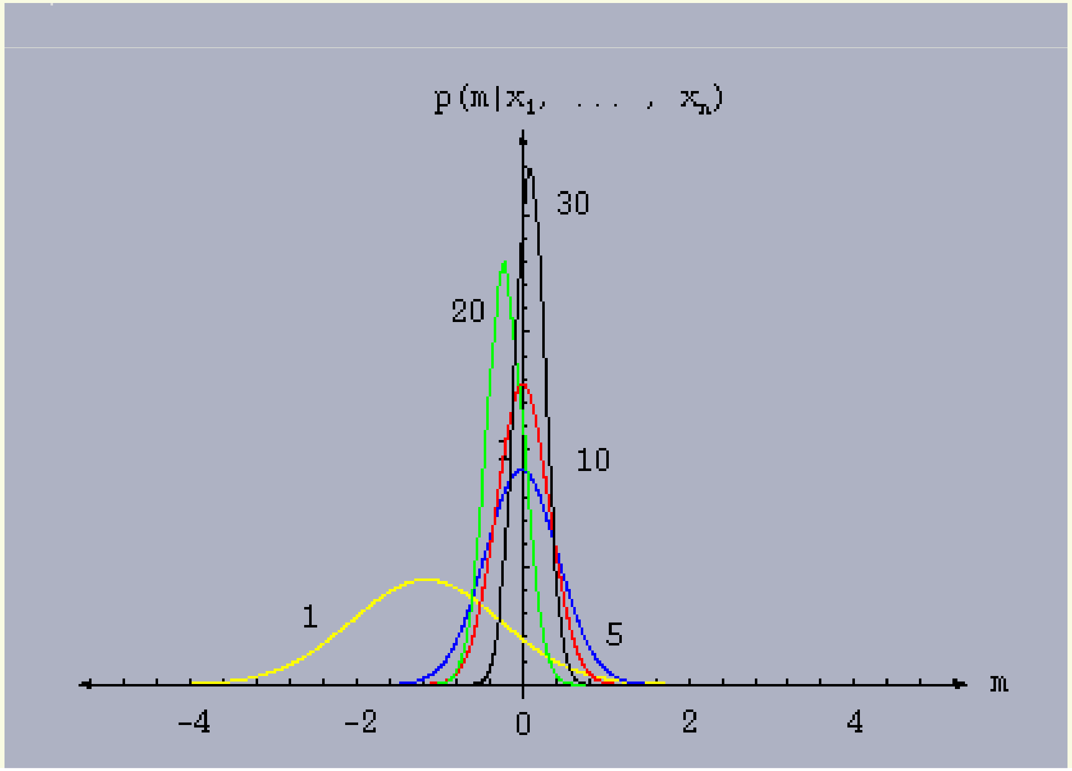
our best guess after observing n samples

$$\sigma_n^2 = \frac{\sigma_0^2 \sigma^2}{n\sigma_0^2 + \sigma^2}$$

uncertainty about the guess, decreases monotonically with n

Bayesian Estimation for Gaussian with unknown μ

- Each additional observation decreases our uncertainty about the true value of μ.
- As n increases, $p(\mu | D)$ becomes more and more sharply peaked, approaching a Dirac delta function as n approaches infinity. This behavior is known as Bayesian Learning.



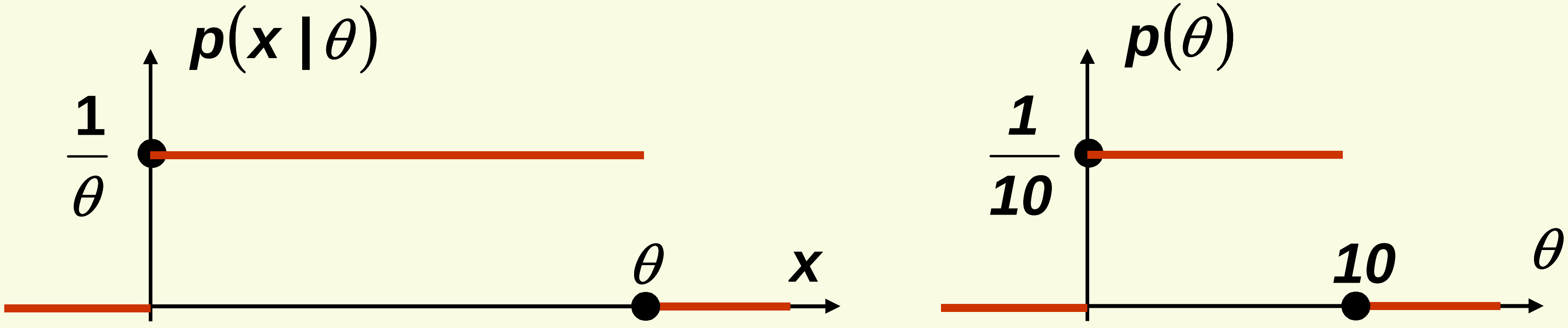
Bayesian Estimation for Gaussian with unknown μ

$$\mu_n = \left(\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\right)\hat{\mu}_n + \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0$$

- In general, μ_n is a linear combination of a sample mean $\hat{\mu}_n$ and a prior μ_0, with coefficients that are non-negative and sum to 1.
- Thus μ_n lies somewhere between $\hat{\mu}_n$ and μ_0 .
- If $\sigma_0 \neq 0$, $\mu_n \to \hat{\mu}_n$ as $n \to \infty$
- If $\sigma_0 = 0$, our a priori certainty that $\mu = \mu_0$ is so strong that no number of observations can change our opinion.
- If a priori guess is very uncertain (σ_0 is large), we take $\mu_n = \hat{\mu}_n$

Bayesian Estimation: Example for U[0,θ]

- Let X be U[***0,*** θ]. Recall $\boldsymbol{p}(\boldsymbol{x}|\theta)$=1/$\theta$ inside [***0,*** θ], else 0

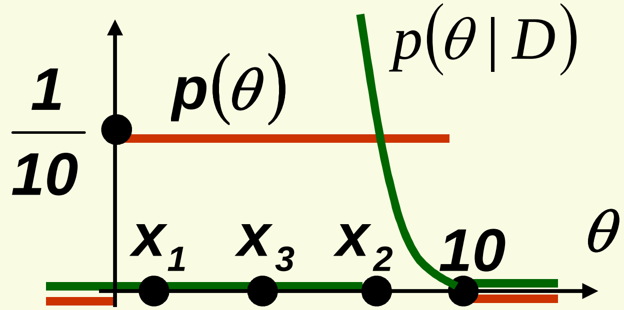


- Suppose we assume a U[0,10] prior on θ
  - good prior to use if we just know the range of θ but don't know anything else

Bayesian Estimation: Example for U[0,θ]

- We need to compute $p(x \mid D) = \int p(x \mid \theta) p(\theta \mid D) d\theta$
  - using $p(\theta \mid D) = \dfrac{p(D \mid \theta) p(\theta)}{\int p(D \mid \theta) p(\theta) d\theta}$ and $p(D \mid \theta) = \prod_{k=1}^{n} p(x_k \mid \theta)$
- When computing MLE of θ, we had

$$p(D \mid \theta) = \begin{cases} \dfrac{1}{\theta^n} & \text{for } \theta \geq \max\{x_1, \ldots, x_n\} \\ 0 & \text{otherwise} \end{cases}$$



- Thus

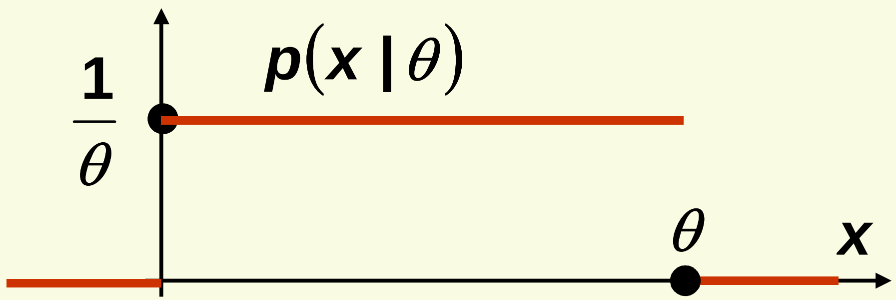
$$p(\theta \mid D) = \begin{cases} c\dfrac{1}{\theta^n} & \text{for } \max\{x_1, \ldots, x_n\} \leq \theta \leq 10 \\ 0 & \text{otherwise} \end{cases}$$

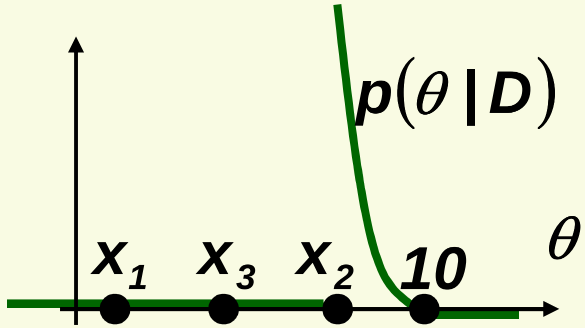
- where c is the normalizing constant, i.e. $c = \dfrac{1}{\displaystyle\int_{\max\{x_1, \ldots, x_n\}}^{10} \frac{d\theta}{\theta^n}}$

Bayesian Estimation: Example for $U[0,\theta]$

- We need to compute $p(x \mid D) = \int p(x \mid \theta) p(\theta \mid D) d\theta$

$$p(\theta \mid D) = \begin{cases} c\dfrac{1}{\theta^n} & \text{for } \max\{x_1,\dots,x_n\} \le \theta \le 10 \\ 0 & \text{otherwise} \end{cases}$$





- We have 2 cases:

1. case $x < \max\{x_1, x_2, \dots, x_n\}$

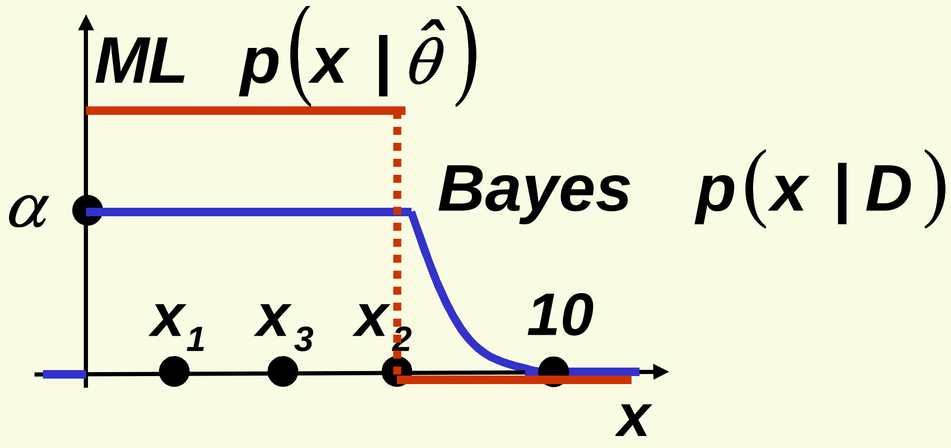
$$p(x \mid D) = \int_{\max\{x_1,\dots x_n\}}^{10} c\frac{1}{\theta^{n+1}} d\theta = \boxed{\alpha}$$

constant independent of x

2. case $x > \max\{x_1, x_2, \dots, x_n\}$

$$p(x \mid D) = \int_x^{10} c\frac{1}{\theta^{n+1}} d\theta = \frac{c}{-n\theta^n}\Big|_x^{10} = \boxed{\frac{c}{nx^n}} - \frac{c}{n10^n}$$

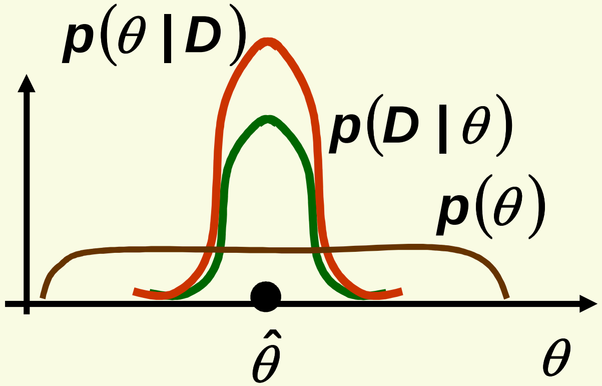
Bayesian Estimation: Example for $U[0,\theta]$



- Note that even after x >max $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$, Bayes density is not zero, which makes sense
- curious fact: Bayes density is not uniform, i.e. does not have the functional form that we have assumed!

ML vs. Bayesian Estimation with Broad Prior

- Suppose $p(\theta)$ is flat and broad (close to uniform prior)
- $p(\theta|D)$ tends to sharpen if there is a lot of data



- Thus $p(D|\theta) \propto p(\theta|D)p(\theta)$ will have the same sharp peak as $p(\theta|D)$
- But by definition, peak of $p(D|\theta)$ is the ML estimate $\hat{\theta}$
- The integral is dominated by the peak:

$$p(x \mid D) = \int p(x \mid \theta)p(\theta \mid D)d\theta \approx p\left(x \mid \hat{\theta}\right)\int p(\theta \mid D)d\theta = p\left(x \mid \hat{\theta}\right)$$

- Thus as n goes to infinity, Bayesian estimate will approach the density corresponding to the MLE!

ML vs. Bayesian Estimation

- **Number of training data**
  - The two methods are equivalent assuming infinite number of training data (and prior distributions that do not exclude the true solution).
  - For small training data sets, they give different results in most cases.
- **Computational complexity**
  - ML uses differential calculus or gradient search for maximizing the likelihood.
  - Bayesian estimation requires complex multidimensional integration techniques.

ML vs. Bayesian Estimation

- Solution complexity
  - Easier to interpret ML solutions (i.e., must be of the assumed parametric form).
  - A Bayesian estimation solution might not be of the parametric form assumed. Hard to interpret, returns weighted average of models.
- Prior distribution
  - If the prior distribution $p(\theta)$ is uniform, Bayesian estimation solutions are equivalent to ML solutions.

Naïve Bayes Classifier

Unbiased Learning of Bayes Classifiers is Impractical

- Learn Bayes classifier by estimating $P(X|Y)$ and $P(Y)$.
- Assume Y is boolean and X is a vector of *n* boolean attributes. In this case, we need to estimate a set of parameters $\theta_{ij} \equiv P\left(X = x_i \mid Y = y_j\right)$

  i takes on 2^n possible values; j takes on 2 possible values.
- How many parameters?
  - For any particular value y_j, and the 2^n possible values of x_i, we need compute 2^n-1 independent parameters.
  - Given the two possible values for Y, we must estimate a total of $2(2^n-1)$ such parameters.

Complex model →High variance with limited data!!!

Conditional Independence

- 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

$$(\forall i, j, k)\ P(X = x_i \mid Y = y_j, Z = z_k) = P(X = x_i \mid Z = z_k)$$

- Example:

$$P(\text{Thunder} \mid \text{Rain}, \text{Lighting}) = P(\text{Thunder} \mid \text{Lighting})$$

Note that in general Thunder is not independent of Rain, but it is given Lighting.

- Equivalent to:

$$P(X, Y \mid Z) = P(X \mid Y, Z) P(Y \mid Z) = P(X \mid Z) P(Y \mid Z)$$

Derivation of Naive Bayes Algorithm

- Naive Bayes algorithm assumes that the attributes $X_1,\ldots,X_n$ are all conditionally independent of one another, given Y. This dramatically simplifies
  - the representation of $P(X|Y)$
  - estimating $P(X|Y)$ from the training data.
- Consider $X=(X_1,X_2)$

$$P(X\mid Y)=P(X_1,X_2\mid Y)=P(X_1\mid Y)P(X_2\mid Y)$$

- For X containing n attributes

$$P(X\mid Y)=\prod_{i=1}^{n}P(X_i\mid Y)$$

Given the boolean X and Y, now we need only 2*n* parameters to define $P(X|Y)$, which is dramatic reduction compared to the $2(2^n-1)$ parameters if we make no conditional independence assumption.

The Naïve Bayes Classifier

- Given:
  - Prior $P(Y)$
  - *n* conditionally independent features X, given the class Y
  - For each X_i, we have likelihood $P(X_i|Y)$
- The probability that Y will take on its k_{th} possible value, is

$$P(Y = y_k \mid X) = \frac{P(Y = y_k)\prod_i P(X_i \mid Y = y_k)}{\sum_j P(Y = y_k)\prod_i P(X_i \mid Y = y_k)}$$

- The Decision rule:

$$y^* = \arg\max_{y_k} P(Y = y_k)\prod_i P(X_i \mid Y = y_k)$$

If assumption holds, NB is optimal classifier!

Naïve Bayes for the discrete inputs

- Given, n attributes X_i each taking on J possible discrete values and Y a discrete variable taking on K possible values.
- MLE for Likelihood $P(X_i = x_{ij} \mid Y = y_k)$ given a set of training examples *D:*

$$\hat{P}(X_i = x_{ij} \mid Y = y_k) = \frac{\# D\{X_i = x_{ij} \wedge Y = y_k\}}{\# D\{Y = y_k\}}$$

where the *#D{x}* operator returns the number of elements in the set *D* that satisfy property *x*.

- MLE for the prior

$$\hat{P}(Y = y_k) = \frac{\# D\{Y = y_k\}}{|D|}$$

number of elements in the training set *D*

NB Example

- Given, training data X Y

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Classify the following novel instance :

(Outlook=sunny, Temp=cool,Humidity=high,Wind=strong)

NB Example

$$y_{NB} = \arg\max_{y_j=\{yes,no\}} P(y_j)P(Outlook = sunny \mid y_j)P(Temp = cool \mid y_j)$$

$$P(Humidity = high \mid y_j)P(Wind = strong \mid y_j)$$

Priors:

$$P(PlayTennis = yes) = 9/14 = 0.64$$

$$P(PlayTennis = no) = 5/14 = 0.36$$

Conditional Probabilities, e.g. Wind = strong:

$$P(Wind = strong \mid PlayTennis = yes) = 3/9 = 0.33$$

$$P(Wind = strong \mid PlayTennis = no) = 3/5 = 0.6$$

...

$$P(yes)P(sunny \mid yes)P(cool \mid yes)P(high \mid yes)P(strong \mid yes) = 0.0053$$

$$\boxed{P(no)P(sunny \mid no)P(cool \mid no)P(high \mid no)P(strong \mid no) = 0.60}$$

Subtleties of NB classifier 1 –Violating the NB assumption

- Usually, features are not conditionally independent.
- Nonetheless, NB often performs well, even when assumption is violated
  - [Domingos& Pazzani'96] discuss some conditions for good performance

Subtleties of NB classifier 2 – Insufficient training data

- What if you never see a training instance where $X_1=a$ when $Y=b$?
  - $P(X_1=a \mid Y=b) = 0$
- Thus, no matter what the values $X_2,\ldots,X_n$ take:

$$P(Y=b \mid X_1=a,X_2,\ldots,X_n) = 0$$

- Solution?

Subtleties of NB classifier 2 – Insufficient training data

- To avoid this, use a "smoothed" estimate
  - effectively adds in a number of additional "hallucinated" examples
  - assumes these hallucinated examples are spread evenly over the possible values of X_i.
- This smoothed estimate is given by

$$\hat{P}(X_i = x_{ij} \mid Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\} + l}{\#D\{Y = y_k\} + lJ}$$

$$\hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\} + l}{|D| + lK}$$

The number of hallucinated examples

l determines the strength of the smoothing
If $l=1$ called Laplace smoothing

Naive Bayes for Continuous Inputs

- When the X_i are continuous we must choose some other way to represent the distributions $P(X_i|Y)$.
- One common approach is to assume that for each possible discrete value y_k of Y, the distribution of each continuous X_i is Gaussian.
- In order to train such a Naïve Bayes classifier we must estimate the mean and standard deviation of each of these Gaussians

Naive Bayes for Continuous Inputs

- MLE for means

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta\left(Y^j = y_k\right)} \sum_j X_i^j \delta\left(Y^j = y_k\right)$$

  - where j refers to the jth training example, and where $\delta(Y=y_k)$ is 1 if $Y = y_k$ and 0 otherwise.
  - Note the role of δ is to select only those training examples for which $Y = y_k$.

- MLE for standard deviation

$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta\left(Y^j = y_k\right)} \sum_j \left(X_i^j - \hat{\mu}_{ik}\right)^2 \delta\left(Y^j = y_k\right)$$

Learning Classify Text

- Applications:
  - Learn which news article are of interest
  - Learn to classify web pages by topic.
- Naïve Bayes is among most effective algorithms

- Target concept *Interesting?: Document*->{+,-}

1 Represent each document by vector of words

  - one attribute per word position in document

2 Learning: Use training examples to estimate

  - P(+)
  - P(-)
  - P(doc|+)
  - P(doc|-)

Text Classification-Example:

Text

Text Classification, or the task of automatically assigning semantic categories to natural language text, has become one of the key methods for organizing online information. Since hand-coding classification rules is costly or even impractical, most modern approaches employ machine learning techniques to automatically learn text classifiers from examples.

The text contains 48 words

Text Representation

(a_1=‘text’,a_2=‘classification’,…. a_{48}=‘examples’)

The representation contains 48 attributes

Note: Text size may vary, but it will not cause a problem

NB conditional independence Assumption

$$P(doc \mid y_j) = \prod_{i=1}^{length(doc)} \boxed{P(a_i = w_k \mid y_j)}$$

probability that word in position i is w_k, given y_j

Indicates the kth word in English vocabulary

The NB assumption is that the word probabilities for one text position are independent of the words in other positions, given the document classification y_j

Clearly not true: The probability of word "learning" may be greater if the preceding word is "machine"

Necessary, without it the number of probability terms is prohibitive

Performs remarkably well despite the incorrectness of the assumption

Text Classification-Example:

Text

Text Classification, or the task of automatically assigning semantic categories to natural language text, has become one of the key methods for organizing online information. Since hand-coding classification rules is costly or even impractical, most modern approaches employ machine learning techniques to automatically learn text classifiers from examples.

Text Representation

(a_1=‘text’,a_2=‘classification’,….
a_{48}=‘examples’)

The text contains 48 words

The representation contains 48 attributes

Classification:

$$y^* = \arg\max_{y_j \in \{+,-\}} P(y_j)P(a_1 = 'text' \mid y_j)\ldots P(a_{48} = 'examples' \mid y_j)$$

$$= \arg\max_{y_j \in \{+,-\}} P(y_j)\prod_i P(a_i = w_k \mid y_j)$$

Estimating Likelihood

- Is problematic because we need to estimate it for each combination of text position, English word, and target value: 48*50,000*2≈5 million such terms.
- Assumption that reduced the number of terms – Bag of Words Model
  - The probability of encountering a specific word w_k is independent of the specific word position.

$$P(a_i = w_k \mid y_j) = P(a_m = w_k \mid y_j), \quad \forall i, m$$

  - Instead of estimating $P(a_1 = w_k \mid y_j),\ P(a_k = w_k \mid y_j), \ldots$ we estimate a single term $P(w_k \mid y_j)$
  - Now we have 50,000*2 distinct terms.

Estimating Likelihood

- The estimate for the likelihood is

$$P(w_k \mid y_j) = \frac{n_k + 1}{n + |Vocabulary|}$$

n -the total number of word positions in all training examples whose target value is y_j

n_k -the number times word w_k is found among these n word positions.

$|Vocabulary|$ -the total number of distinct words found within the training data.

Learn_Naive_Bayes_Text*(Examples,V)*

1. *collect all words and other tokens that occur in $Examples$*

- $Vocabulary \leftarrow$ all distinct words and other tokens in $Examples$

2. *calculate the required* $P(y_j)$ *and* $P(w_k \mid y_j)$

- For each target value y_j in V do
  - $docs_j \leftarrow$ subset of *Examples for which the target value is* y_j
  - $P(y_j) \leftarrow \dfrac{|docs_j|}{|Examples|}$
  - $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$
  - $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
  - For each word w_j in the $Vocabulary$
    * $n_k \leftarrow$ number of times word w_k occurs in $Text_j$
    * $P(w_k \mid y_j) \leftarrow \dfrac{n_k + 1}{n + |Vocabulary|}$

Classify_Naive_Bayes_Text(Doc)

- $positions \leftarrow$ all word positions in Doc that contain tokens found in $Vocabulary$
- Return $y^* = \underset{y_j \in \{+,-\}}{\arg\max}\, P(v_j) \prod_{i \in positions} P(a_i \mid v_j)$