UVA CS 6316
– Fall 2015 Graduate: Machine Learning

Lecture 14: Generative Bayes Classifier

Dr. Yanjun Qi

University of Virginia

Department of
Computer Science

Where are we ? ➔ Five major sections of this course

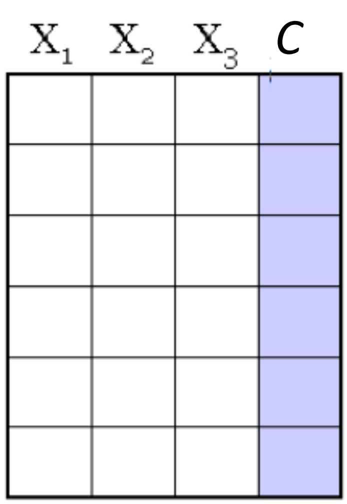
- ❑ Regression (supervised)
- ❑ Classification (supervised)
- ❑ Unsupervised models
- ❑ Learning theory
- ❑ Graphical models

Where are we ? ➔ Three major sections for classification

- We can divide the large variety of classification approaches into roughly three major types

1. Discriminative
   - directly estimate a decision rule/boundary
   - e.g., support vector machine, decision tree
2. Generative:
   - build a generative statistical model
   - e.g., naïve bayes classifier, Bayesian networks
3. Instance based classifiers
   - Use observation directly (no models)
   - e.g. K nearest neighbors

A Dataset for classification



$$f : X \longrightarrow C$$

Output as Discrete Class Label
$C_1, C_2, \ldots, C_L$

$P(C \mid \mathbf{X})$

$(X_1, X_2, \ldots, X_p)$

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

Bayes classifier

- Treat each attribute and class label as random variables.

- Given a sample **x** with attributes (x_1, x_2, … , x_p):
  - Goal is to predict class *C*.
  - Specifically, we want to find the value of C_i that maximizes $p(C_i \mid x_1, x_2, \ldots, x_p)$.

- Generative Bayes classification

$$P(C \mid \mathbf{X}) \propto P(\mathbf{X} \mid C)P(C) = \underline{P(X_1,\cdots,X_p \mid C)}P(C)$$

Difficulty: learning the joint probability $P(X_1,\cdots,X_p \mid C)$

Probabilistic Models of text documents

$W_1 \ldots W_K \in Voc$

$d_1 \vdots d_n$

$$\Pr(D \mid C = c) \quad ?$$

$D = (W_1, W_2, \ldots, W_K)$

Two Previous models

$$\Pr(W_1 = true, W_2 = false \ldots, W_k = true \mid C = c)$$

Multivariate Bernoulli Distribution

$$\Pr(W_1 = n_1, W_2 = n_2, \ldots, W_k = n_k \mid C = c)$$

Multinomial Distribution

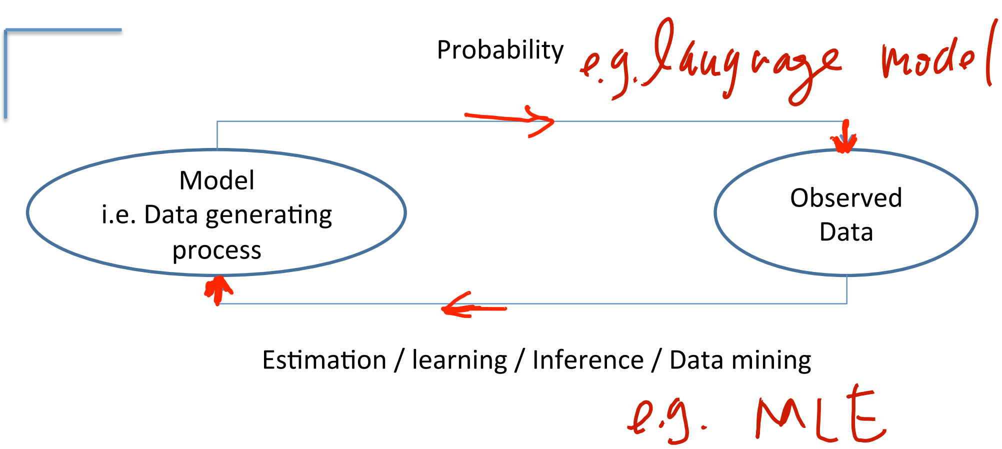
Today : Generative Bayes Classifier

- ✓ Multinomial naïve Bayes classifier as Conditional Stochastic Language Models
  - A unigram Language model approximates how a text document is produced.

$$\Pr(W_1 = n_1, \ldots, W_k = n_k \mid C = c)$$

- ✓ Maximum Likelihood Estimation of parameters
- ✓ Gaussian Naïve Bayes Classifiers

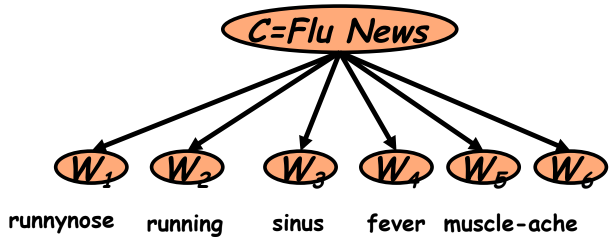
The Big Picture



But how to specify a model?

Build a *generative model* that approximates how data is produced.

Model 1: Multivariate Bernoulli



- **Conditional Independence Assumption:** Features (word presence) are *independent* of each other given the class variable:

Ber

this is naïve

$$\Pr(W_1 = true, W_2 = false, \ldots, W_k = true \mid C = c)$$

$$= P(W_1 = true \mid C) \bullet P(W_2 = false \mid C) \bullet \cdots \bullet P(W_k = true \mid C)$$

- Multivariate Bernoulli model is appropriate for binary feature variables

Review: Bernoulli Distribution e.g. Coin Flips

- You flip a coin
  - Head with probability p
  - Binary random variable
  - Bernoulli trial with success probability p

$$\Pr(W_i = true \mid C = c)$$

Model 2: Multinomial Naïve Bayes
- 'Bag of words' representation of text

word	frequency
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

$$\Pr(W_1 = n_1, \ldots, W_k = n_k \mid C = c)$$

Can be represented as a multinomial distribution.

Words = like colored balls, there are *K* possible type of them (i.e. from a dictionary of K words)

Document = contains N words, each word occurs n_i times (like a bag of N colored balls)

multinomial coefficient, normally can leave out in practical calculations.

why naive ???

WHY is this naïve ???

$$P(W_1 = n_1, \ldots, W_k = n_k \mid N, \theta_1, \ldots, \theta_k) = \boxed{\frac{N!}{n_1! n_2! .. n_k!}} \theta_1^{n_1} \theta_2^{n_2} .. \theta_k^{n_k}$$

Multinomial Naïve Bayes as ➔ a *generative model* that approximates how a text string is produced

- **Stochastic Language Models:**
  - Model *probability* of generating strings (each word in turn following the sequential ordering in the string) in the language (commonly all strings over dictionary ∑).
  - E.g., unigram model

P(s | model(C1))

Model C_1

0.2 the

0.1 a

0.01 boy

0.01 dog

0.03 said

0.02 likes

…

the	boy	likes	the	dog
0.2	0.01	0.02	0.2	0.01

Multiply all five terms

P(s | C_1) = 0.00000008

Adapt From Manning' textCat tutorial

Multinomial Naïve Bayes as Conditional Stochastic Language Models

- Model conditional *probability* of generating any string from two possible models

Model C1	
0.2	the
0.01	boy
0.0001	said
0.0001	likes
0.0001	black
0.0005	dog
0.01	garden

Model C2	
0.2	the
0.0001	boy
0.03	said
0.02	likes
0.1	black
0.01	dog
0.0001	garden

$P(C2) = P(C1)$

the	boy	likes	black	dog
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$p(d|C_2)\, p(C_2)$

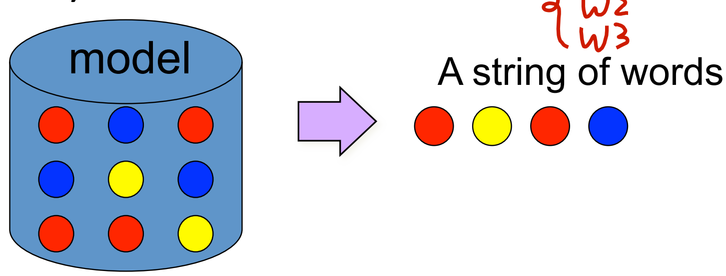
$P(s|C2)\, P(C2) > P(s|C1)\, P(C1)$

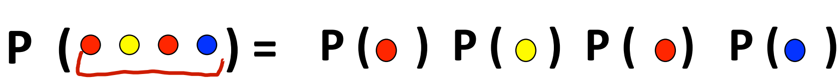
➔ S is more likely to be from class C2

A Physical Metaphor

- Colored balls are randomly drawn from (with replacement)

$K=3$ $\begin{cases} w_1 \\ w_2 \\ w_3 \end{cases}$





Unigram language model ➔ More general:

Generating language string from a probabilistic model

P (🔴 🟡 🔴 🔵)

Chain rule

= P (🔴) P (🟡 | 🔴) P (🔴 | 🔴 🟡) P (🔵 | 🔴 🟡 🔴)

B1   B2 | B1   B3 | B1 B2   B4 | B1 B2 B3

- Unigram Language Models

B1   B2   B3   B4

⇒ P (🔴) P (🟡) P (🔴) P (🔵)

Naive

Easy.
Effective!

NAÏVE : conditional independent on each position of the string

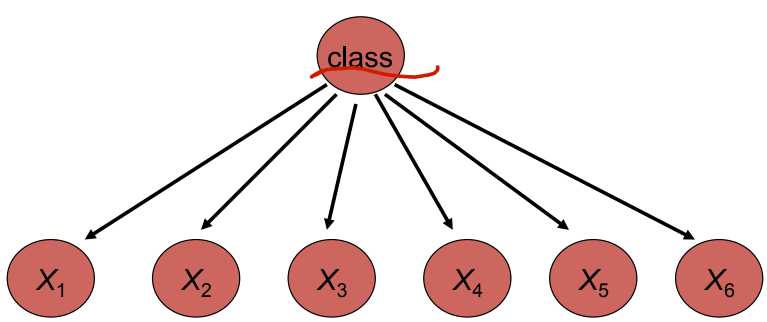
- Also could be bigram (or generally, *n*-gram) Language Models

B1   B2 | B1   B3 | B2   B4 | B3   Bj | Bj-1

P (🔴) P (🟡 | 🔴) P (🔴 | 🟡) P (🔵 | 🔴)

Adapt From Manning' textCat tutorial

Multinomial Naïve Bayes = a class conditional unigram language model



- Think of X_i as the word on the i^{th} position in the document string
- Effectively, the probability of each class is done as a class-specific unigram language model

Adapt From Manning' textCat tutorial

Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.

$= \text{argmax}\, P(C \mid X)$

$$c_{NB} = \underset{c_j \in C}{\text{argmax}}\, P(c_j) \prod_i P(x_i \mid c_j)$$

the boy like the dog

$$= \underset{c_j \in C}{\text{argmax}}\, P(c_j) \underbrace{P(x_1 = \text{"the"} \mid c_j)}_{?} \cdots \underbrace{P(x_n = \text{"the"} \mid c_j)}_{?}$$

- Still too many possibilities
  - Use same parameters for a word across positions
  - Result is bag of words model (over word tokens)

Multinomial Naïve Bayes: Classifying Step

- Positions ← all word positions in current document which contain tokens found in *Vocabulary*
- Return c_{NB}, where

Easy to implement, no need to construct bag-of-words vector explicitly !!!

$P(w_k \mid c_j)$

$$c_{NB} = \underset{c_j \in C}{\text{argmax}}\, P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

the	boy	likes	black	dog
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

P(s|C2) P(C2) > P(s|C1) P(C1)

Equal to, (leaving out of multinomial coefficient)

$$\Pr(W_1 = n_1, \ldots, W_k = n_k \mid C = c_j)$$

Adapt From Manning' textCat tutorial

Unknown Words

- How to handle words in the test corpus that did not occur in the training data, i.e. ***out of vocabulary*** (OOV) words?
- Train a model that includes an explicit symbol for an unknown word (<UNK>).
  - Choose a vocabulary in advance and replace other (i.e. not in vocabulary) words in the training corpus with <UNK>.

Underflow Prevention: log space

- Multiplying lots of probabilities, which are between 0 and 1, can result in floating-point underflow.
- Since log(xy) = log(x) + log(y), it is better to perform all computations *by summing logs of probabilities rather than multiplying probabilities*.
- Class with highest final un-normalized log probability score is still the most probable.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

- Note that model is now just max of sum of weights…

Adapt From Manning' textCat tutorial

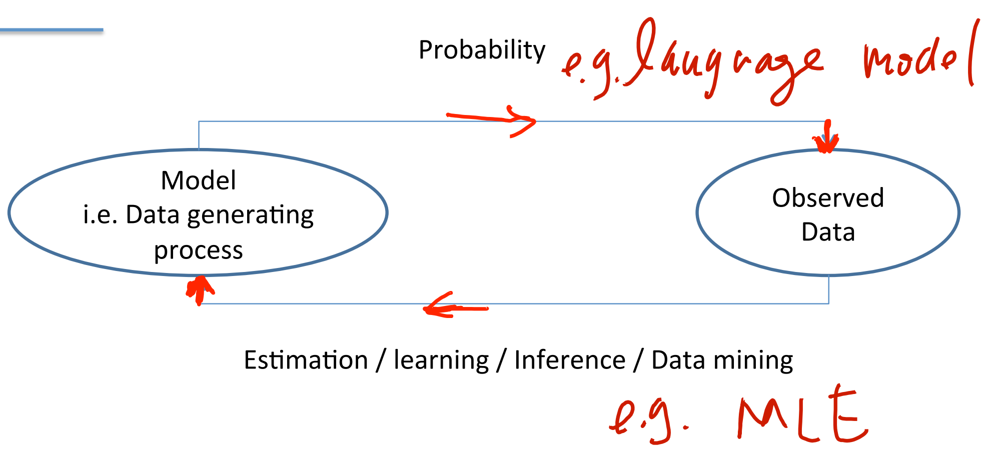
Today : Generative Bayes Classifier

- ✓ Multinomial naïve Bayes classifier as Conditional Stochastic Language Models
  - A unigram Language model approximates how a text document is produced.

$$\Pr(W_1 = n_1, \dots, W_k = n_k \mid C = c)$$

- ✓ Maximum Likelihood Estimation of parameters
- ✓ Gaussian Naïve Bayes Classifiers

The Big Picture



But how to specify a model?

Build a *generative model* that approximates how data is produced.

Generative Model & MLE

- Language model can be seen as a probabilistic automata for generating text strings

??

$$P(W_1 = n_1, \ldots, W_k = n_k \mid N, \theta_1, \ldots, \theta_k) \propto \theta_1^{n_1} \theta_2^{n_2} \ldots \theta_k^{n_k}$$

- Relative frequency estimates can be proven to be ***maximum likelihood estimates*** (MLE) since they maximize the probability that the model *M* will generate the training corpus *T*.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \, P(Train \mid M(\theta))$$

likelihood

MLE

Maximum Likelihood Estimation

A general Statement

training

Consider a sample set $T=(X_1 \ldots X_n)$ which is drawn from a probability distribution P(X|\theta) where \theta are parameters.

If the Xs are independent with probability density function $P(X_i|$ \theta), the joint probability of the whole set is

$$P(X_1 \ldots X_n \mid \theta) = \prod_{i=1}^{n} P(X_i \mid \theta)$$

this may be maximised with respect to \theta to give the maximum likelihood estimates.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \, P(Train \mid M(\theta)) = \underset{\theta}{\operatorname{argmax}} \, P(X_1 \ldots X_n \mid \theta)$$

The idea is to

- ✓ assume a particular model with unknown parameters, θ
- ✓ we can then define the probability of observing a given event conditional on a particular set of parameters. $P(X_i \mid \theta)$
- ✓ We have observed a set of outcomes in the real world. $x_1, x_2, \cdots, x_n$
- ✓ It is then possible to choose a set of parameters which are most likely to have produced the observed results.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \underbrace{P(X_1 \ldots X_n \mid \theta)}_{\text{likehood}}$$

This is maximum likelihood. In most cases it is both consistent and efficient. It provides a standard to compare other estimation techniques.

$$\log(L(\theta)) = \sum_{i=1}^{n} \underbrace{\log(P(X_i \mid \theta))}_{\text{log likehood}}$$

It is often convenient to work with the Log of the likelihood function.

Review: Binomial Distribution e.g. Coin Flips

- You flip n coins
  - How many heads would you expect
  - Head with probability p
  - Number of heads X: discrete random variable

  Following Binomial distribution with parameters n and p

Defining Likelihood

- Likelihood = p(data | parameter) $= \prod p(x_i|p) = p^x (1-p)^{1-x}$

Bernulli $x_i = \{0,1\}$ $p(x_1, x_2 .. x_n | p)$

➔ e.g., for n independent tosses of coins, with unknown p

Observed data ➔ x heads-up from n trials

function of x_i

PMF: $$f(x_i|p) = p^{x_i}(1-p)^{1-x_i}$$

$$x = \sum_{i=1}^{n} x_i$$

LIKELIHOOD:

$$L(p) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = p^x(1-p)^{n-x}$$

function of p

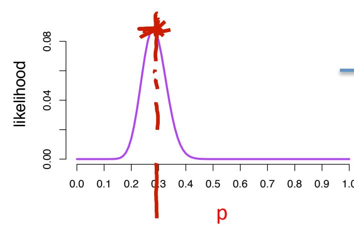
Deriving the Maximum Likelihood Estimate

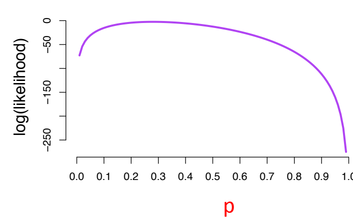
$$\underbrace{L(p)}_{\text{maximize}} = p^x(1-p)^{n-x}$$

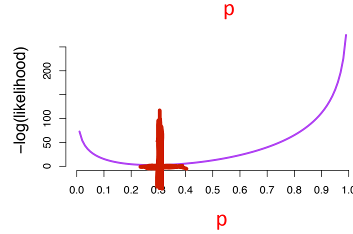
$$\underbrace{\log(L(p)}_{\text{maximize}} = \log\left[p^x(1-p)^{n-x}\right]$$

Minimize the negative log-likelihood

$$l(p) = -\log\left[p^x(1-p)^{n-x}\right]$$







Deriving the Maximum Likelihood Estimate

Minimize the negative log-likelihood

$$l(p) = -\log(L(p)) = -\log\left[p^x(1-p)^{n-x}\right]$$

$$l(p) = -\log(p^x) - \log((1-p)^{n-x})$$

$$l(p) = \quad -x\log(p) - (n-x)\log(1-p)$$

Deriving the Maximum Likelihood Estimate

$$l(p) = \quad -x\log(p) - (n-x)\log(1-p)$$

$$\frac{dl(p)}{dp} = -\frac{x}{p} - \frac{-(n-x)}{1-p} = 0$$

$$0 = -\frac{x}{\hat{p}} + \frac{n-x}{1-\hat{p}}$$

$$0 = \frac{-x(1-\hat{p}) + \hat{p}(n-x)}{\hat{p}(1-\hat{p})}$$

$$0 = -x + \cancel{\hat{p}x} + \hat{p}n - \cancel{\hat{p}x}$$

$$0 = -x + \hat{p}n$$

Minimize the negative log-likelihood

➔ MLE parameter estimation

$$\boxed{\hat{p} = \frac{x}{n}}$$

i.e. Relative frequency of a binary event

Parameter estimation for homework3-Q1

- Multivariate Bernoulli model:

$$\hat{P}(X_w = true \mid c_j) = \text{fraction of documents of topic } c_j \text{ in which word } w \text{ appears}$$

- Multinomial model:

$$\hat{P}(X_i = w \mid c_j) = \text{fraction of times in which word } w \text{ appears across all documents of topic } c_j$$

  – Can create a mega-document for topic *j* by concatenating all documents on this topic
  – Use frequency of *w* in mega-document

Adapt From Manning' textCat tutorial

Deriving the Maximum Likelihood Estimate for multinomial distribution

train T documents

LIKELIHOOD: $\arg\max_{\theta_1,..,\theta_k} P(d_1,\dots,d_T \mid \theta_1,..,\theta_k)$, $C=c_j$

$w_1, \dots, w_K$

function of θ $= \arg\max_{\theta_1,..,\theta_k} \prod_{t=1}^{T} P(d_t \mid \theta_1,..,\theta_k)$

$$= \arg\max_{\theta_1,..,\theta_k} \prod_{t=1}^{T} \cancel{\frac{N_{dt}!}{n_{1,d_t}!\,n_{2,d_t}!\dots n_{k,d_t}!}}\,\theta_1^{n_{1,d_t}}\theta_2^{n_{2,d_t}}..\theta_k^{n_{k,d_t}}$$

$$s.t. \sum_{i=1}^{k} \theta_i = 1$$

$$\underbrace{= \arg\max_{\theta_1,..,\theta_k} \prod_{t=1}^{T} \theta_1^{n_{1,d_t}}\theta_2^{n_{2,d_t}}..\theta_k^{n_{k,d_t}}}_{}$$

data likihood

Deriving the Maximum Likelihood Estimate for multinomial distribution

$$\arg\max_{\theta_1,..,\theta_k} \log(L(\theta))$$

Constrained optimization

$$s.t. \sum_{i=1}^{k} \theta_i = 1$$

$$= \arg\max_{\theta_1,..,\theta_k} \log(\prod_{t=1}^{T} \theta_1^{n_{1,d_t}} \theta_2^{n_{2,d_t}} .. \theta_k^{n_{k,d_t}})$$

$$= \arg\max_{\theta_1,..,\theta_k} \sum_{t=1,...T} n_{1,d_t} \log(\theta_1) + \sum_{t=1,...T} n_{2,d_t} \log(\theta_2) + ... + \sum_{t=1,...T} n_{k,d_t} \log(\theta_k)$$

Constrained optimization MLE estimator

$$\theta_i = \frac{\sum_{t=1,...T} n_{i,d_t}}{\sum_{t=1,...T} n_{1,d_t} + \sum_{t=1,...T} n_{2,d_t} + ... + \sum_{t=1,...T} n_{k,d_t}} = \frac{\sum_{t=1,...T} n_{i,d_t}}{\sum_{t=1,...T} N_{d_t}}$$

for c_j

How optimize ? See Handout - EXTRA

➔ i.e. We can create a mega-document by concatenating all documents d_1 to d_T

➔Use relative frequency of *w* in mega-document

Naïve Bayes: Learning Algorithm for parameter estimation with MLE

- From training corpus, extract *Vocabulary*
- Calculate required $P(c_j)$ and $P(w_k \mid c_j)$ terms
  - For each c_j in C do
    - $docs_j$ ← subset of documents for which the target class is c_j

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- $Text_j$ ← is length n and is a single document containing all $docs_j$
- for each word w_k in *Vocabulary*
  - n_k ← number of occurrences of w_k in $Text_j$; n is length of $Text_j$
  - $P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$ $\quad e.g., \alpha = 1$

Relative frequency of word *w_k* appears across all documents of class c_j

Naive Bayes: Time Complexity

|D| num. doc
|V| dict size
|C| class size

- Training Time: $O(|D|L_d + |C||V|)$
  where L_d is the average length of a document in D.
  - Assumes V and all D_i, n_i, and n_{ij} pre-computed in $O(|D|L_d)$ time during one pass through all of the data.
  - Generally just $O(|D|L_d)$ since usually $|C||V| < |D|L_d$
- Test Time: $O(|C| L_t)$
  where L_t is the average length of a test document.
  - Very efficient overall, linearly proportional to the time needed to just read in all the data.
  - Plus, robust in practice

Adapt From Manning' textCat tutorial

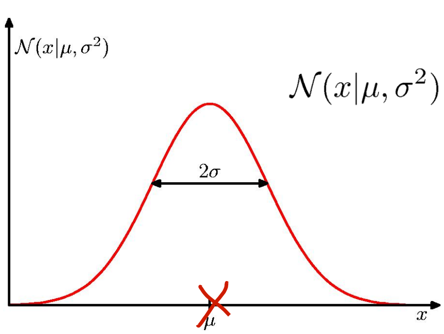
Naive Bayes is Not So Naive

- Naïve Bayes: First and Second place in KDD-CUP 97 competition, among 16 (then) state of the art algorithms
  Goal: Financial services industry direct mail response prediction model: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.
- Robust to Irrelevant Features
  Irrelevant Features cancel each other without affecting results
  Instead Decision Trees can heavily suffer from this.
- Very good in domains with many <u>equally important</u> features
  Decision Trees suffer from *fragmentation* in such cases – especially if little data
- A good dependable baseline for text classification (but not the best)!
- Optimal if the Independence Assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- Very Fast: Learning with one pass of counting over the data; testing linear in the number of attributes, and document collection size
- Low Storage requirements

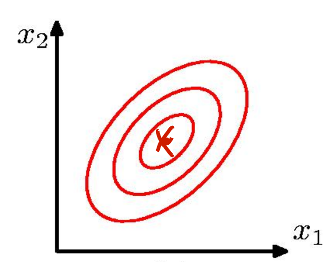
Today : Generative vs. Discriminative

- ✓ Multinomial naïve Bayes classifier as Stochastic Language Models
  - ✓ a unigram Language model approximates how a text document is produced.
- ✓ Maximum Likelihood Estimation of parameters
- ✓ Gaussian Naïve Bayes Classifiers
  - Gaussian distribution
  - Gaussian NBC
  - LDA, QDA

The Gaussian Distribution



$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$



$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

Covariance Matrix

Mean

Courtesy: http://research.microsoft.com/~cmbishop/PRML/index.htm

Multivariate Gaussian Distribution

- A multivariate Gaussian model: **x** ~ N(μ, \Sigma) where

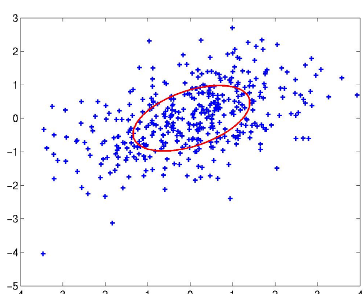
Here μ is the mean vector and \Sigma is the covariance matrix,
if p=2
$\mu = \{\mu_1, \mu_2\}$

\Sigma =

var(x_1)	cov(x_1,x_2)
cov(x_1,x_2)	var(x_2)

- The covariance matrix captures linear dependencies among the variables

MLE Estimation for Multivariate Gaussian

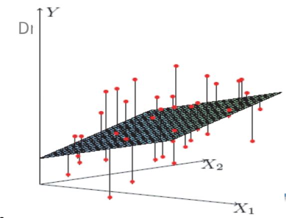


- We can fit statistical models by maximizing the probability / likelihood of generating the observed samples:
L(x_1, … ,x_n | \theta) = p(x_1 | \theta) … p(x_n | \theta)
(the samples are assumed to be IID)

- In the Gaussian case, we simply set the mean and the variance to the sample mean and the sample variance:

$$\overline{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \overline{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{\mu})^2$$

Probabilistic Interpretation of Linear Regression



- Let us assume that the target variable and the inputs are related by the equation:

$$y_i = \theta^T \mathbf{x}_i + \varepsilon_i$$

where ε is an error term of unmodeled effects or random noise

- Now assume that ε follows a Gaussian $N(0,\sigma)$, then we have:

μ

$$p(y_i \mid x_i;\theta) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right)$$

- By IID assumption:

other prob. models.

$$L(\theta) = \prod_{i=1}^{n} p(y_i \mid x_i;\theta) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\sum_{i=1}^{n}(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right)$$

Probabilistic Interpretation of Linear Regression (cont.)

- Hence the log-likelihood is:

$$l(\theta) = n\log\frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{n}(y_i - \theta^T \mathbf{x}_i)^2$$

- Do you recognize the last term?

Yes it is:
$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$

- Thus under independence assumption, residual square error (RRS) is equivalent to MLE of ϑ !

Today : Generative vs. Discriminative

- ✓ Multinomial naïve Bayes classifier as Stochastic Language Models
  - ✓ a unigram Language model approximates how a text document is produced.
- ✓ Maximum Likelihood Estimation of parameters
- ✓ Gaussian Naïve Bayes Classifiers
  - Gaussian distribution
  - Gaussian NBC
  - LDA, QDA

Gaussian Naïve Bayes Classifier

$$\underset{C}{\operatorname{argmax}} P(C \mid X) = \underset{C}{\operatorname{argmax}} P(X, C) = \underset{C}{\operatorname{argmax}} P(X \mid C) P(C)$$

Naïve Bayes Classifier

$$P(X \mid C) = P(X_1, X_2, \cdots, X_p \mid C)$$
$$= P(X_1 \mid X_2, \cdots, X_p, C) P(X_2, \cdots, X_p \mid C)$$
$$= P(X_1 \mid C) P(X_2, \cdots, X_p \mid C)$$
$$= P(X_1 \mid C) P(X_2 \mid C) \cdots P(X_p \mid C)$$

$$\hat{P}(X_j \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2} \right)$$

μ_{ji} : mean (avearage) of attribute values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of attribute values X_j of examples for which $C = c_i$

Gaussian Naïve Bayes Classifier

- Continuous-valued Input Attributes
  - Conditional probability modeled with the normal distribution

$$\hat{P}(X_j \mid C=c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(X_j-\mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

μ_{ji} : mean (avearage) of attribute values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of attribute values X_j of examples for which $C = c_i$

- Learning Phase: for $\mathbf{X} = (X_1, \cdots, X_p)$, $C = c_1, \cdots, c_L$
  Output: $p \times L$ normal distributions and $P(C = c_i)$ $i = 1, \cdots, L$
- Test Phase: for $\mathbf{X}' = (X'_1, \cdots, X'_p)$
  - Calculate conditional probabilities with all the normal distributions
  - Apply the MAP rule to make a decision

Naïve Gaussian means ?

$O(kp^2+kp)$

Σ_k

Not Naïve

$$P(X_1, X_2, \cdots, X_p \mid C) =$$

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

p-dim random vector

Naïve

$$P(X_1, X_2, \cdots, X_p \mid C = c_j) = P(X_1 \mid C)P(X_2 \mid C) \cdots P(X_p \mid C)$$

$$= \prod_i \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(X_j-\mu_{ji})^2}{2\sigma_{ji}^2}\right)$$

$O(kp+kp)$

Diagonal Matrix $\Sigma\_c_k = \Lambda\_c_k$

Each class' covariance matrix is diagonal

$$\underset{k}{\operatorname{argmax}} P(C\_k \mid X) = \underset{k}{\operatorname{argmax}} P(X,C) = \underset{k}{\operatorname{argmax}} P(X \mid C)P(C)$$

Generative Bayes Classifier

Task → Representation → Score Function → Search/Optimization → Models, Parameters

classification → Prob. models p(X|C) → EPE with 0-1 loss ➔ likelihood → Many options → Prob. Models' Parameter

$$P(X_1, \cdots, X_p \mid C)$$

Gaussian Naive

$$\hat{P}(X_j \mid C = c_k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp\left(-\frac{(X_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$$

Bernoulli Naïve

$$p(W_i = true \mid c_k) = p_{i,k}$$

Multinomial

$$P(W_1 = n_1, \ldots, W_v = n_v \mid c_k) = \frac{N!}{n_{1k}!\, n_{2k}! \ldots n_{vk}!} \theta_{1k}^{n_{1k}} \theta_{2k}^{n_{2k}} \ldots \theta_{vk}^{n_{vk}}$$

Today : Generative vs. Discriminative

- ✓ Multinomial naïve Bayes classifier as Stochastic Language Models
  - ✓ a unigram Language model approximates how a text document is produced.
- ✓ Maximum Likelihood Estimation of parameters
- ✓ Gaussian Naïve Bayes Classifiers
  - Gaussian distribution
  - Gaussian NBC
  - Not-naïve Gaussian BC ➔ LDA, QDA

(1) covariance matrix are the same across classes

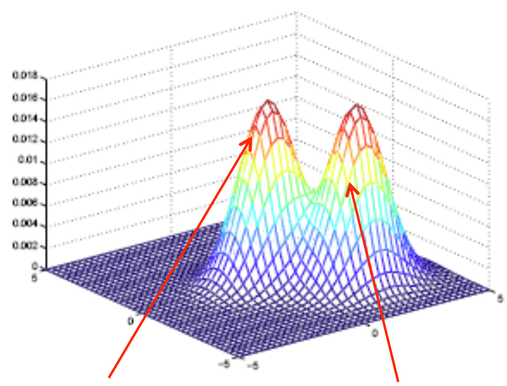
→ LDA (Linear Discriminant Analysis)

$O(p^2+kp)$

Linear Discriminant Analysis : $\Sigma_k = \Sigma, \ \forall k$

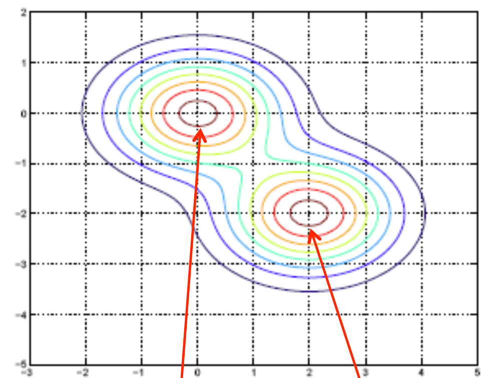
Each class' covariance matrix is the same

The Gaussian Distribution are shifted versions of each other



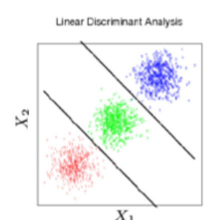
Class *k*

Class *l*



Class *k*

Class *l*



Optimal Classification

$$\arg\max_k P(C\_k \mid X) = \arg\max_k P(X, C) = \arg\max_k P(X \mid C) P(C)$$

$$= \arg\max_k \Big[\cancel{-\log((2\pi)^{p/2}|\Sigma|^{1/2})} - \frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \Big]$$

$\log P(C_k)$

$$= \arg\max_k \boxed{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k)}$$

Linear Discriminant Function for LDA

- Note

$$-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k - \frac{1}{2}x^T\Sigma^{-1}x$$

$$\underset{k}{\operatorname{argmax}} P(C_k \mid X) = \underset{k}{\operatorname{argmax}} P(X, C_k) = \underset{k}{\operatorname{argmax}} P(X \mid C_k) P(C_k)$$

$$= \underset{k}{\operatorname{argmax}} \log\{\underbrace{P(X \mid C_k)}\underbrace{P(C_k)}\}$$

$$= \underset{k}{\operatorname{argmax}} \log p(X \mid C_k) + \log \underbrace{P(C_k)} \Rightarrow \pi_k$$

Decision Boundary points

$$\log \frac{P(C_k \mid X)}{P(C_l \mid X)} = 0 = \log \frac{P(X \mid C_k)}{P(X \mid C_l)} + \log \frac{\pi_k}{\pi_l}$$

$$= \log P(X \mid C_k) - \log P(X \mid C_l) + \log \frac{\pi_k}{\pi_l}$$

Define **Linear Discriminant Function**

$$\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \sum^{-1} (x - \mu_k) + log\pi_k$$

$$\delta_k(x) = \delta_l(x)$$

➔ The Decision Boundary Between class *k* and *l*, $\{x : \delta_k(x) = \delta_l(x)\}$, is linear

$$\log \frac{P(C_k \mid X)}{P(C_l \mid X)} = \log \frac{P(X \mid C_k)}{P(X \mid C_l)} + \log \frac{P(C_k)}{P(C_l)}$$

$$w^T X + b = 0$$

$$= \log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k + \mu_\ell)^T \Sigma^{-1}(\mu_k - \mu_\ell) + x^T \Sigma^{-1}(\mu_k - \mu_\ell), \quad (4.9)$$

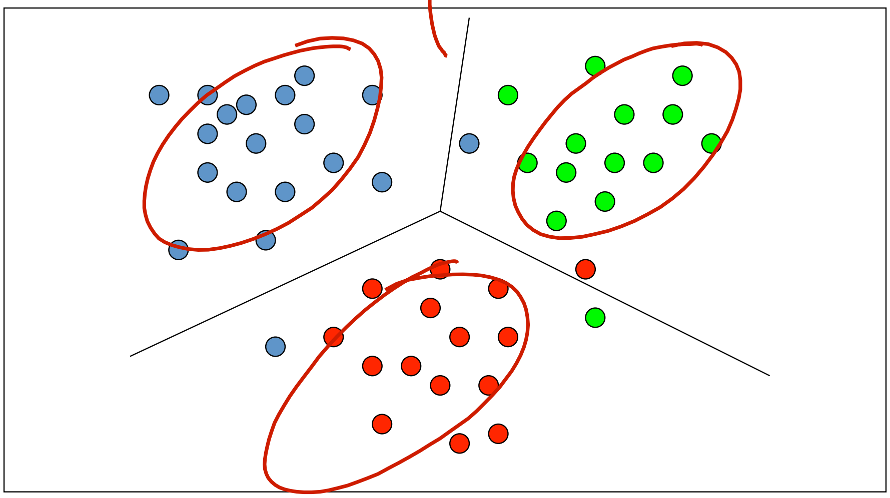
$$= 0$$

Equals to zero

Boundary points X : when P(c_k|X) == P(c_l|X), the left linear equation ==0, a linear line / plane

Visualization (three classes)

linear



(2) If covariance matrix are not same e.g. ➔ QDA (Quadratic Discriminant Analysis)

$O(KP^2 + KP)$

- Estimate the covariance matrix Σ_k separately for each class k, $k = 1, 2, ..., K$.
- *Quadratic discriminant function*:

Σ

$$\delta_k(x) = \underbrace{-\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k}\ .$$

- Classification rule:

$$\hat{G}(x) = \arg\max_k \delta_k(x)\ .$$

$\log \frac{P(c_k|x)}{P(c_l|x)} = 0$

- Decision boundaries are quadratic equations in x.
- QDA fits the data better than LDA, but has more parameters to estimate.

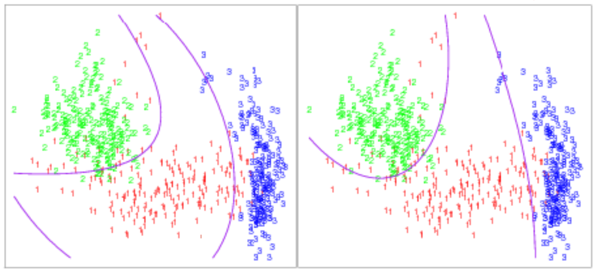
LDA on Expanded Basis

- Expand input space to include X_1X_2, X_1^2, and X_2^2.
- Input is five dimensional: $X = (X_1, X_2, X_1X_2, X_1^2, X_2^2)$.

$\Phi(x) \Leftarrow$ LDA

LDA with $\Phi(x)$

QDA



LDA with quadratic basis Versus QDA

Figure 4.6: *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $x_1, x_2, x_{12}, x_1^2, x_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.*

Dr. Yanjun Qi / UVA CS 6316 / f15

(3) Regularized Discriminant Analysis

- A compromise between LDA and QDA.
- Shrink the separate covariances of QDA toward a common covariance as in LDA.
- Regularized covariance matrices:

$$\hat{\Sigma}_k(\alpha) = \alpha\hat{\Sigma}_k + (1-\alpha)\hat{\Sigma}\ .$$

- The quadratic discriminant function $\delta_k(x)$ is defined using the shrunken covariance matrices $\hat{\Sigma}_k(\alpha)$.
- The parameter α controls the complexity of the model.

References

- Prof. Andrew Moore’s review tutorial
- Prof. Ke Chen NB slides
- Prof. Carlos Guestrin recitation slides
- Prof. Raymond J. Mooney and Jimmy Lin’s slides about language model