Machine Learning
10-701/15-781, Fall 2011

Generative verses discriminative classifier

Eric Xing
Lecture 3, September 19, 2011

Reading:
© Eric Xing @ CMU, 2006-2011

- Homework 1 out today! Save at least 10 hours for it.
- About project
- Midterm and final
Generative vs. Discriminative Classifiers

- **Goal:** Wish to learn \( f: X \rightarrow Y \), e.g., \( P(Y|X) \)

- **Generative:**
  - Modeling the joint distribution of all data

- **Discriminative:**
  - Modeling only points at the boundary

Learning Generative and Discriminative Classifiers

- **Goal:** Wish to learn \( f: X \rightarrow Y \), e.g., \( P(Y|X) \)

- **Generative classifiers (e.g., Naïve Bayes):**
  - Assume some functional form for \( P(X|Y), P(Y) \)
  - This is a ‘generative’ model of the data!
  - Estimate parameters of \( P(X|Y), P(Y) \) directly from training data
  - Use Bayes rule to calculate \( P(Y|X=x) \)

- **Discriminative classifiers (e.g., logistic regression):**
  - Directly assume some functional form for \( P(Y|X) \)
  - This is a ‘discriminative’ model of the data!
  - Estimate parameters of \( P(Y|X) \) directly from training data
Suppose you know the following

- Class-specific Dist.: $P(X|Y)$

\[
p(X | Y = 1) = p_1(X; \mu_1, \Sigma_1)
\]

\[
p(X | Y = 2) = p_2(X; \mu_2, \Sigma_2)
\]

- Class prior (i.e., "weight"): $P(Y)$

- This is a generative model of the data!

Bayes classifier:

\[
P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}
\]

Optimal classification

- **Theorem:** Bayes classifier is optimal!

  That is

\[
error_{true}(h_{Bayes}) \leq error_{true}(h), \ \forall h(x)
\]

- How to learn a Bayes classifier?

  Recall density estimation. We need to estimate $P(X|y=k)$, and $P(y=k)$ for all $k$
Learning Bayes Classifier

- Training data (discrete case):

\[
\begin{array}{c|cccccc|c}
\text{Sky} & \text{Temp} & \text{Humid} & \text{Wind} & \text{Water} & \text{Forecast} & \text{EnjoySpt} \\
\hline
\text{Sunny} & \text{Warm} & \text{Normal} & \text{Strong} & \text{Warm} & \text{Same} & \text{Yes} \\
\text{Sunny} & \text{Warm} & \text{High} & \text{Strong} & \text{Warm} & \text{Same} & \text{Yes} \\
\text{Rainy} & \text{Cold} & \text{High} & \text{Strong} & \text{Warm} & \text{Change} & \text{No} \\
\text{Sunny} & \text{Warm} & \text{High} & \text{Strong} & \text{Cool} & \text{Change} & \text{Yes} \\
\end{array}
\]

- Learning = estimating \( P(X|Y) \), and \( P(Y) \)

- Classification = using Bayes rule to calculate \( P(Y | X_{\text{new}}) \)

Parameter learning from iid data:
The Maximum Likelihood Est.

- Goal: estimate distribution parameters \( \theta \) from a dataset of \( N \) independent, identically distributed (iid), fully observed, training cases

\[
D = \{x_1, \ldots, x_N\}
\]

- Maximum likelihood estimation (MLE)
  1. One of the most common estimators
  2. With iid and full-observability assumption, write \( L(\theta) \) as the likelihood of the data:

\[
L(\theta) = P(x_1, x_2, \ldots, x_N; \theta)
= P(x_1; \theta)P(x_2; \theta), \ldots, P(x_N; \theta)
= \prod_{n=1}^{N} P(x_n; \theta)
\]

3. pick the setting of parameters most likely to have generated the data we saw:

\[
\theta^* = \arg \max_{\theta} L(\theta) = \arg \max_{\theta} \log L(\theta)
\]
How hard is it to learn the optimal classifier?

- How do we represent these? How many parameters?
  - Prior, $P(Y)$:
    - Suppose $Y$ is composed of $k$ classes
  - Likelihood, $P(X|Y)$:
    - Suppose $X$ is composed of $n$ binary features

- Complex model $\rightarrow$ High variance with limited data!!!

---

Gaussian Discriminative Analysis

- learning $f: X \rightarrow Y$, where
  - $X$ is a vector of real-valued features, $X_n = <X_n^1, \ldots, X_n^n>$
  - $Y$ is an indicator vector
- What does that imply about the form of $P(Y|X)$?
  - The joint probability of a datum and its label is:
    $$p(x_n, y_n^k = 1 | \mu, \Sigma) = p(y_n^k = 1) \times p(x_n | y_n^k = 1, \mu, \Sigma)$$
    $$= \pi_k \frac{1}{(2\pi|\Sigma|)^{n/2}} \exp \left\{ \frac{1}{2} (x_n - \mu_k)^T \Sigma^{-1} (x_n - \mu_k) \right\}$$
  - Given a datum $x_n$, we predict its label using the conditional probability of the label given the datum:
    $$p(y_n^k = 1 | x_n, \mu, \Sigma) = \frac{\pi_k \frac{1}{(2\pi|\Sigma|)^{n/2}} \exp \left\{ \frac{1}{2} (x_n - \mu_k)^T \Sigma^{-1} (x_n - \mu_k) \right\}}{\sum_k \pi_k \frac{1}{(2\pi|\Sigma|)^{n/2}} \exp \left\{ \frac{1}{2} (x_n - \mu_k)^T \Sigma^{-1} (x_n - \mu_k) \right\}}$$
Conditional Independence

- 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:

\[(\forall i, j, k) P(X = i | Y = j, Z = k) = P(X = i | Z = k)\]

Which we often write:

\[P(X | Y, Z) = P(X | Z)\]

- e.g.,

\[P(\text{Thunder} | \text{Rain}, \text{Lightning}) = P(\text{Thunder} | \text{Lightning})\]

- Equivalent to:

\[P(X, Y | Z) = P(X | Z)P(Y | Z)\]

The Naïve Bayes assumption

- Naïve Bayes assumption:
  - Features are conditionally independent given class:

  \[P(X_1, X_2 | Y) = P(X_1 | X_2, Y)P(X_2 | Y) = P(X_1 | Y)P(X_2 | Y)\]

  - More generally:

  \[P(X^1 \ldots X^n | Y) = \prod_i P(X^i | Y)\]

- How many parameters now?
  - Suppose X is composed of m binary features
The Naïve Bayes Classifier

- Given:
  - Prior $P(Y)$
  - $m$ conditionally independent features $X$ given the class $Y$
  - For each $X_n$, we have likelihood $P(X_n | Y)$

- Decision rule:
  
  $y^* = h_{NB}(x) = \arg \max_y P(y) \prod_{i} P(x_i | y)$

  $\implies \arg \max_y P(y) \prod_{i} P(x_i | y)$

- If assumption holds, NB is optimal classifier!

The A Gaussian Discriminative Naïve Bayes Classifier

- When $X$ is multivariate-Gaussian vector:
  - The joint probability of a datum and its label is:
    
    $p(x_n, y_n = 1 | \mu, \Sigma) = p(y_n = 1) \times p(x_n | y_n = 1, \mu, \Sigma)$
    
    $= \pi_n \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x_n - \mu)^T \Sigma^{-1} (x_n - \mu) \right)$

  - The naïve Bayes simplification
    
    $p(x_n, y_n = 1 | \mu, \sigma) = p(y_n = 1) \times \prod_{j} p(x_{n,j} | y_n = 1, \mu_j, \sigma_j)$
    
    $= \pi_n \prod_{j} \frac{1}{\sqrt{2\pi \sigma_j}} \exp \left( -\frac{1}{2} \left( \frac{x_{n,j} - \mu_j}{\sigma_j} \right)^2 \right)$

  - More generally:
    
    $p(x_n, y_n | \eta, \pi) = p(y_n | \pi) \times \prod_{j} p(x_{n,j} | y_n, \eta)$

    - Where $p(\cdot | \cdot)$ is an arbitrary conditional (discrete or continuous) 1-D density
The predictive distribution

- Understanding the predictive distribution

\[ p(y^k_a = 1 \mid x_a, \mu, \Sigma) = \frac{p(y^k_a = 1, x_a \mid \mu, \Sigma)}{p(x_a \mid \mu, \Sigma)} = \frac{\pi_k N(x_a \mid \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} N(x_a \mid \mu_{k'}, \Sigma_{k'})} \]

- Under naïve Bayes assumption:

\[
\begin{aligned}
p(y^1_a = 1 \mid x_a, \mu, \Sigma, \pi) &= \pi_k \exp \left\{ \sum \left( \frac{1}{2} \left( \frac{x^2_a - \mu^2_k}{\sigma^2_k} \right) - \log \sigma_k - C \right) \right\} \\
&= \frac{1}{1 + e^{-\theta^T x_a}}
\end{aligned}
\]

- For two class (i.e., \( K=2 \)), \( \pi \) correspond to a softmax function

\[
\begin{aligned}
p(y_1^1 = 1 \mid x_a) &= \frac{1}{1 + \exp \left( -\sum \left( \frac{x^2_a - \mu^2_k}{\sigma^2_k} \right) - \log \sigma_k - C \right)} \\
&= \frac{1}{1 + \exp \left( \sum \left( \frac{x^2_a - \mu^2_k}{\sigma^2_k} \right) - \log \sigma_k - C \right)}
\end{aligned}
\]

The decision boundary

- The predictive distribution

\[
p(y^k_a = 1 \mid x_a) = \frac{1}{1 + \exp \left\{ -\sum \theta^T x_a - \theta \right\}} = \frac{1}{1 + e^{-\theta^T x_a}}
\]

- The Bayes decision rule:

\[
\ln \frac{p(y_1^1 = 1 \mid x_a)}{p(y_0^1 = 1 \mid x_a)} = \ln \left( \frac{1 + e^{-\theta^T x_a}}{e^{-\theta^T x_a}} \right) = \theta^T x_a
\]

- For multiple class (i.e., \( K>2 \)), \( \pi \) correspond to a softmax function

\[
p(y^k_a = 1 \mid x_a) = \frac{e^{-\theta^T x_a}}{\sum e^{-\theta^T x_a}}
\]
Summary:
The Naïve Bayes Algorithm

- Train Naïve Bayes (examples)
  - for each value $y_k$
  - estimate $\pi_k \equiv P(Y = y_k)$
  - for each value $x_i$ of each attribute $X_i$
  - estimate $\theta_{ijk} \equiv P(X_i = x_{ij} | Y = y_k)$

- Classify ($X_{\text{new}}$)
  \[
  Y^{\text{new}} \leftarrow \arg \max_{y_k} P(Y = y_k) \prod_j P(X_j = x_{ij} | Y = y_k)
  \]

Generative vs. Discriminative Classifiers

- Goal: Wish to learn $f: X \rightarrow Y$, e.g., $P(Y|X)$

- Generative classifiers (e.g., Naïve Bayes):
  - Assume some functional form for $P(X|Y)$, $P(Y)$
    This is a 'generative' model of the data!
  - Estimate parameters of $P(X|Y)$, $P(Y)$ directly from training data
  - Use Bayes rule to calculate $P(Y|X=x)$

- Discriminative classifiers:
  - Directly assume some functional form for $P(Y|X)$
    This is a 'discriminative' model of the data!
  - Estimate parameters of $P(Y|X)$ directly from training data
Logistic regression (sigmoid classifier)

- The condition distribution: a Bernoulli
  \[ p(y \mid x) = \mu(x)^y (1 - \mu(x))^{1-y} \]
  where \( \mu \) is a logistic function
  \[ \mu(x) = \frac{1}{1 + e^{-\theta^T x}} \]

- We can use the brute-force gradient method as in LR

- But we can also apply generic laws by observing the \( p(y \mid x) \) is an exponential family function, more specifically, a generalized linear model (see future lectures …)

Training Logistic Regression: MCLE

- Estimate parameters \( \theta = \langle \theta_0, \theta_1, \ldots, \theta_m \rangle \) to maximize the conditional likelihood of training data

- Training data \( \mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\} \)

- Data likelihood \( = \prod_{i=1}^{N} P(x_i, y_i; \theta) \)

- Data conditional likelihood \( = \prod_{i=1}^{N} P(x_i \mid y_i; \theta) \)

\[ \theta = \arg \max_{\theta} \ln \prod_{i} P(y_i \mid x_i; \theta) \]
Expressing Conditional Log Likelihood

\[ l(\theta) \equiv \ln \prod_i P(y_i|x_i; \theta) = \sum_i \ln P(y_i|x_i; \theta) \]

- Recall the logistic function:
  \[ \mu = \frac{1}{1 + e^{-\theta^T x}} \]

and conditional likelihood:
\[ P(y|x) = \mu(x)^y (1 - \mu(x))^{1-y} \]

\[ l(\theta) = \sum_i \ln P(y_i|x_i; \theta) = \sum_i y_i \ln \mu(x_i) + (1 - y_i) \ln(1 - \mu(x_i)) \]
\[ = \sum_i y_i \ln \frac{\mu(x_i)}{1 - \mu(x_i)} + \ln(1 - \mu(x_i)) \]
\[ = \sum_i y_i \theta^T x_i - \theta^T x_i + \ln(1 + e^{-\theta^T x_i}) \]
\[ = \sum_i (y_i - 1) \theta^T x_i + \ln(1 + e^{-\theta^T x_i}) \]

Maximizing Conditional Log Likelihood

- The objective:
  \[ l(\theta) = \ln \prod_i P(y_i|x_i; \theta) \]
  \[ = \sum_i (y_i - 1) \theta^T x_i + \ln(1 + e^{-\theta^T x_i}) \]

- Good news: \( l(\theta) \) is concave function of \( \theta \)

- Bad news: no closed-form solution to maximize \( l(\theta) \)
Gradient Ascent

\[ l(\theta) = \ln \prod_i P(y_i|x_i; \theta) \]
\[ = \sum_i (y_i - 1)\theta^T x_i + \ln(1 + e^{-\theta^T x_i}) = \sum_i (y_i - 1)\theta^T x_i - \ln \mu(\theta^T x_i) \]

- Property of sigmoid function:
  \[ \mu = \frac{1}{1 + e^{-t}} \]
  \[ \frac{d\mu}{dt} = \mu(1 - \mu) \]

- The gradient:
  \[ \frac{\partial l(\theta)}{\partial \theta_j} = \]

The gradient ascent algorithm iterate until change < $\varepsilon$
For all $i$,
\[ \theta_j \leftarrow \theta_j + \eta \sum_i (y_i - P(y_i = 0|x_i; \theta))x_i^j \]
repeat

The Newton’s method

- Finding a zero of a function

\[ \theta^{t+1} := \theta^t - \frac{f(\theta^t)}{f'(\theta^t)} \]
The Newton’s method (con’d)

- To maximize the conditional likelihood \( l(\theta) \):

\[
l(\theta) = \sum_i (y_i - 1)\theta^T x_i + \ln(1 + e^{-\theta^T x_i})
\]

since \( l \) is convex, we need to find \( \theta^* \) where \( l'(\theta^*)=0 \!

- So we can perform the following iteration:

\[
\theta^{t+1} := \theta^t + \frac{l'(\theta^t)}{l''(\theta^t)}
\]

The Newton-Raphson method

- In LR the \( \theta \) is vector-valued, thus we need the following generalization:

\[
\theta^{t+1} := \theta^t + H^{-1} \nabla_{\theta} l(\theta^t)
\]

- \( \nabla \) is the gradient operator over the function

- \( H \) is known as the Hessian of the function
The Newton-Raphson method

- In LR the $\theta$ is vector-valued, thus we need the following generalization:

$$\theta^{t+1} := \theta^t + H^{-1}\nabla_\theta l(\theta^t)$$

- $\nabla$ is the gradient operator over the function

$$\nabla_\theta l(\theta) = \sum_i (y_i - u_i)x_i = X^T(y - u)$$

- $H$ is known as the Hessian of the function

$$H = \nabla_\theta \nabla_\theta l(\theta) = \sum_i u_i(1 - u_i)x_ix_i^T = X^TRX$$

where $R_{ii} = u_i(1 - u_i)$

Iterative reweighed least squares (IRLS)

- Recall in the least square est. in linear regression, we have:

$$\theta = (X^TX)^{-1}X^Ty$$

which can also derived from Newton-Raphson

- Now for logistic regression:

$$\theta^{t+1} = \theta^t + H^{-1}\nabla_\theta l(\theta^t)$$

$$= \theta^t - (X^TRX)^{-1}X^T(u - y)$$

$$= (X^TRX)^{-1}\{X^TRX\theta^t - X^T(u - y)\}$$

$$= (X^TRX)^{-1}X^TRz$$
IRLS

- Recall in the least square est. in linear regression, we have:

$$\theta = (X^T X)^{-1} X^T y$$

which can also derived from Newton-Raphson

- Now for logistic regression:

$$\theta^{t+1} = (X^T R X)^{-1} X^T R z$$

where  
$$z = X \theta^t - R^{-1} (u - y)$$

and  
$$R_{ii} = u_i (1 - u_i)$$

Convergence curves

Legend:  
- X-axis: Iteration #; Y-axis: error
- In each figure, red for IRLS and blue for gradient descent
Logistic regression: practical issues

- NR (IRLS) takes $O(N + d^3)$ per iteration, where $N$ = number of training cases and $d$ = dimension of input $x$, but converge in fewer iterations.

- Quasi-Newton methods, that approximate the Hessian, work faster.

- Conjugate gradient takes $O(Nd)$ per iteration, and usually works best in practice.

- Stochastic gradient descent can also be used if $N$ is large c.f. perceptron rule.

Case Study: Text classification

- Classify e-mails
  - $Y = \{\text{Spam, NotSpam}\}$

- Classify news articles
  - $Y = \{\text{what is the topic of the article?}\}$

- Classify webpages
  - $Y = \{\text{Student, professor, project, …}\}$

- What about the features $X$?
  - The text!
Features $X$ are entire document – $X^i$ for $i^{th}$ word in article

Bag of words model

- Typical additional assumption – **Position in document doesn’t matter**: $P(X^i=x^i|Y=y) = P(X^k=x^i|Y=y)$
  - "Bag of words" model – order of words on the page ignored
  - Sounds really silly, but often works very well!

$$P(y) \prod_{i=1}^{\text{LengthDoc}} P(x^i|y) \quad \text{or} \quad P(y) \prod_{k=1}^{\text{LengthVol}} P(w^k|y)$$

When the lecture is over, remember to wake up the person sitting next to you in the lecture room.
Bag of words model

- Typical additional assumption – **Position in document doesn’t matter**: $P(X^i=x^i|Y=y) = P(X^k=x^i|Y=y)$
  - “Bag of words” model – order of words on the page ignored
  - Sounds really silly, but often works very well!

\[
P(y) \frac{\text{LengthDoc}}{\prod_{i=1}^{\text{LengthDoc}}} P(x^i|y) \quad \text{or} \quad P(y) \frac{\text{LengthVol}}{\prod_{k=1}^{\text{LengthVol}}} P(w^k|y)
\]

in is lecture lecture next over person remember room sitting the the the to to up wake when you

NB with Bag of Words for text classification

- **Learning phase:**
  - Prior $P(Y)$
    - Count how many documents you have from each topic (+ prior)
  - $P(X|Y)$
    - For each topic, count how many times you saw word in documents of this topic (+ prior)

- **Test phase:**
  - For each document $x_{\text{new}}$
    - Use naïve Bayes decision rule

\[
h_{NB}(x_{\text{new}}) = \arg\max_y P(y) \frac{\text{LengthDoc}}{\prod_{i=1}^{\text{LengthDoc}}} P(x^i_{\text{new}}|y)
\]

© Eric Xing @ CMU, 2006-2011
Back to our 20 NG Case study

- **Dataset**
  - 20 News Groups (20 classes)
  - 61,118 words, 18,774 documents

- **Experiment:**
  - Solve only a two-class subset: 1 vs 2.
  - 1768 instances, 61188 features.
  - Use dimensionality reduction on the data (SVD).
  - Use 90% as training set, 10% as test set.
  - Test prediction error used as accuracy measure.

\[
\text{Accuracy} = \frac{\sum (\text{predict} - \text{true label})}{\text{# of test samples}}
\]

Results: Binary Classes

![Graph showing accuracy vs. training ratio for different classes](image)
Results: Multiple Classes

Accuracy

- 5-out-of-20 classes
- 10-out-of-20 classes
- All 20 classes

Training Ratio

NB vs. LR

- Versus training size

- 30 features.
- A fixed test set
- Training set varied from 10% to 100% of the training set
**NB vs. LR**

- Versus model size

![Graph showing prediction error vs. number of features used for training and test sets for NB and Log Reg.]  
- Number of dimensions of the data varied from 5 to 50 in steps of 5  
- The features were chosen in decreasing order of their singular values  
- 90% versus 10% split on training and test

**Generative vs. Discriminative Classifiers**

- Goal: Wish to learn $f: X \rightarrow Y$, e.g., $P(Y|X)$

- Generative classifiers (e.g., Naïve Bayes):
  - Assume some functional form for $P(X|Y)$, $P(Y)$  
    This is a *'generative'* model of the data!  
  - Estimate parameters of $P(X|Y)$, $P(Y)$ directly from training data  
  - Use Bayes rule to calculate $P(Y|X=x)$

- Discriminative classifiers:
  - Directly assume some functional form for $P(Y|X)$  
    This is a *'discriminative'* model of the data!  
  - Estimate parameters of $P(Y|X)$ directly from training data
Naïve Bayes vs Logistic Regression

- Consider Y boolean, X continuous, X=<X^1 ... X^m>
- Number of parameters to estimate:

\[
\begin{align*}
\text{NB: } p(y|x) &= \frac{\pi_y \exp \left( -\sum_j \left( \frac{1}{2\sigma_{y,j}^2} (x_j - \mu_{y,j})^2 - \log \sigma_{y,j} - C \right) \right)}{\sum_{y'} \pi_{y'} \exp \left( -\sum_j \left( \frac{1}{2\sigma_{y',j}^2} (x_j - \mu_{y',j})^2 - \log \sigma_{y',j} - C \right) \right)} \quad ** \\
\text{LR: } \mu(x) &= \frac{1}{1 + e^{-\theta^T x}}
\end{align*}
\]

- Estimation method:
  - NB parameter estimates are uncoupled
  - LR parameter estimates are coupled

Asymptotic comparison (# training examples \(\to\) infinity)

- when model assumptions correct
  - NB, LR produce identical classifiers

- when model assumptions incorrect
  - LR is less biased – does not assume conditional independence
  - therefore expected to outperform NB
Naïve Bayes vs Logistic Regression

- Non-asymptotic analysis (see [Ng & Jordan, 2002])

- Convergence rate of parameter estimates – how many training examples needed to assure good estimates?
  
  NB order $\log m$ (where $m =$ # of attributes in $X$)
  
  LR order $m$

- NB converges more quickly to its (perhaps less helpful) asymptotic estimates

---

Rate of convergence: logistic regression

- Let $h_{\text{Dis},m}$ be logistic regression trained on $n$ examples in $m$ dimensions. Then with high probability:

  $$
  \epsilon(h_{\text{Dis},n}) \leq \epsilon(h_{\text{Dis},\infty}) + O\left(\sqrt{\frac{m \log n}{n}}\right)
  $$

- Implication: if we want $\epsilon(h_{\text{Dis},m}) \leq \epsilon(h_{\text{Dis},\infty}) + \epsilon_0$

  for some small constant $\epsilon_0$, it suffices to pick order $m$ examples

  $\rightarrow$ Convergences to its asymptotic classifier, in order $m$ examples

- Result follows from Vapnik’s structural risk bound, plus fact that the “VC Dimension” of an $m$-dimensional linear separators is $m$
Rate of convergence: naïve Bayes parameters

- Let any $\epsilon_1, \delta > 0$, and any $n \geq 0$ be fixed. Assume that for some fixed $\rho_0 > 0$, we have that $\rho_0 \leq p(y = T) \leq 1 - \rho_0$

- Let $n = O((1/\epsilon_1^2) \log(m/\delta))$

- Then with probability at least $1-\delta$, after $n$ examples:
  1. For discrete input, $|\hat{p}(x_i | y = b) - p(x_i | y = b)| \leq \epsilon_1$ for all $i$ and $b$
     $|\hat{p}(y = b) - p(y = b)| \leq \epsilon_1$
  2. For continuous inputs, $|\hat{\mu}_{i|y=b} - \mu_{i|y=b}| \leq \epsilon_1$ for all $i$ and $b$
     $|\hat{\sigma}^2_{i|y=b} - \sigma^2_{i|y=b}| \leq \epsilon_1$

Some experiments from UCI data sets

Figure 1: Results of 15 experiments on dataset from the UCI Machine Learning repository. For all set of experiments, solid line is learned over 2000 clean data, dashed and dotted lines represent the noisy cases
Summary

- Naïve Bayes classifier
  - What’s the assumption
  - Why we use it
  - How do we learn it

- Logistic regression
  - Functional form follows from Naïve Bayes assumptions
  - For Gaussian Naïve Bayes assuming variance
  - For discrete-valued Naïve Bayes too
  - But training procedure picks parameters without the conditional independence assumption

- Gradient ascent/descent
  - General approach when closed-form solutions unavailable

- Generative vs. Discriminative classifiers
  - Bias vs. variance tradeoff

Appendix
Subtleties of NB classifier 1 – Violating the NB assumption

- Often the $X_i$ are not really conditionally independent

- We use Naïve Bayes in many cases anyway, and it often works pretty well
  - Often the right classification, even when not the right probability (see [Domingos&Pazzani, 1996])
  - But the resulting probabilities $P(Y|X_{new})$ are biased toward 1 or 0 (why?)

Subtleties of NB classifier 2 – Insufficient training data

- What if you never see a training instance where $w^{1000}>0$ when $Y=b$?
  - e.g., $Y=${SpamEmail or not}, $w^{1000} = \{’pill’, ’enhancement’, ’Rolex’, ’enlarge’ \ldots\}$
  - $P(\text{enlargement}>0 \mid Y=T) = 0$

- Thus, no matter what the values $w_1,\ldots,w_n/’enlargement’$ take:
  - $P(Y=T \mid w^1, w^2, \ldots, \text{enlargement}, \ldots, w^k) = 0$

- What now???
Learning NB: parameter estimation

- Maximum Likelihood Estimate (MLE):
  choose $\theta$ that maximizes probability of observed data $\mathcal{D}$
  \[ \hat{\theta} = \arg \max_{\theta} P(\mathcal{D} | \theta) \]

- Maximum a Posteriori (MAP) estimate:
  choose $\theta$ that is most probable given prior probability and the data
  \[ \hat{\theta} = \arg \max_{\theta} p(\theta | \mathcal{D}) = \arg \max_{\theta} \frac{P(\mathcal{D} | \theta) p(\theta)}{P(\mathcal{D})} \]

- Bayesian estimate:
  \[ \hat{\theta} = \int \theta p(\theta | \mathcal{D}) d\theta \]

MAP for the parameters of NB

Discrete features:

- Maximum a Posteriori (MAP) estimate: (MAP's):
  \[ \hat{\theta} = \arg \max_{\theta} \frac{P(\mathcal{D} | \theta) P(\theta)}{P(\mathcal{D})} \]

- Given prior:
  - Consider binary feature
  - $\theta$ is a Bernoulli rate
  \[ P(\theta | \alpha_r, \alpha_p) = \frac{\Gamma(\alpha_r + \alpha_p)}{\Gamma(\alpha_r) \Gamma(\alpha_p)} \theta^{\alpha_r-1} (1-\theta)^{\alpha_p-1} \]

- Let $I_a = \text{Count}(X=a) \leftarrow$ number of examples where $X=a$
\[ P(\theta | \mathcal{D}) = \frac{\theta^{I_H + \alpha_r - 1} (1-\theta)^{I_T + \alpha_p - 1}}{\Gamma(\alpha_r + \alpha_p)} \sim \text{Beta}(\beta_H + \alpha_H, \beta_T + \alpha_T) \]
Bayesian learning for NB parameters – a.k.a. smoothing

- **Posterior distribution of** $\theta$
  - **Bernoulli**
    \[
    P(\theta | D) = \frac{\theta^{y} (1-\theta)^{N-y}}{I(\beta\theta + \alpha, \beta(N-y) + \alpha)} \sim \text{Beta}(\beta\theta + \alpha, \beta(N-y) + \alpha)
    \]
  - **Multinomial**
    \[
    P(\theta | D) = \frac{\prod_{b} \theta_{b}^{y_{b}} (1-\theta_{b})^{N-b_{b}}}{I(\beta\theta_{b} + \alpha_{b}, \beta(N-y_{b}) + \alpha_{b})} \sim \text{Dirichlet}(\beta\theta_{b} + \alpha_{b}, \beta(N-y_{b}) + \alpha_{b})
    \]

- **MAP estimate**:
  \[
  \hat{\theta} = \arg \max_{\theta} P(\theta | D) = 
  \]

- Beta prior equivalent to extra thumbtack flips
- As $N \to \infty$, prior is “forgotten”
- But, for small sample size, prior is important!

MAP for the parameters of NB

- **Dataset of** $N$ **examples**
  - Let $\beta_{iab} = \text{Count}(X_i=a, Y=b)$ ← number of examples where $X_i=a$ and $Y=b$
  - Let $\gamma_{i} = \text{Count}(Y=b)$

- **Prior**
  - $Q(X|Y) \propto \text{Multinomial}(a_1, ..., a_K)$ or $\text{Multinomial}(\alpha/K)$
  - $Q(Y) \propto \text{Multinomial}(t_1, ..., t_M)$ or $\text{Multinomial}(\tau/M)$
    - $m$ “virtual” examples

- **MAP estimate**
  \[
  \hat{\pi}_k = \arg \max_{\pi_k} \prod_{k} P(Y = y_k; \pi_k) P(\pi_k | \hat{\theta}) = ?
  \]
  \[
  \hat{\theta}_{ijk} = \arg \max_{\theta_{ijk}} \prod_{j} P(X^i = x_{ij}; \theta_{ijk}) P(\theta_{ijk} | \hat{\alpha}_{ijk}) = ?
  \]

- Now, even if you never observe a feature/class, posterior probability never zero