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 $h(x) \propto f(x)$

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!

---

**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):

\[ X \quad Y \]

\[ X = \{ x_1, x_2, \ldots, x_k \} \]

\[ Y = \{ y_1, y_2, \ldots, y_l \} \]

\[ P(x_i; \theta)P(x_j; \theta), \ldots, P(x_k; \theta) \]

\[ \prod_{i=1}^{N} P(x_i; \theta) \]

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

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

Parameter learning from \textit{iid} data: The Maximum Likelihood Est.

- Goal: estimate distribution parameters \( \theta \) from a dataset of \( N \) independent, identically distributed (\textit{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:

\[ L(\theta) = P(x_1, x_2, \ldots, x_N; \theta) \]

\[ = P(x_1; \theta)P(x_2; \theta), \ldots, P(x_N; \theta) \]

\[ = \prod_{i=1}^{N} P(x_i; \theta) \]

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
      $$ P(y_1 \ldots y_k) = \prod_{i=1}^{k} P(y_i | \mathbf{x}) = \prod_{i=1}^{k} \frac{1}{k!} \sum_{i=1}^{k} \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x_j - \mu_i)^2}{2\sigma_j^2}\right) $$
  - Likelihood, $P(X|Y)$:
    - Suppose $X$ is composed of $n$ binary features
      $$ P(x_1 \ldots x_n | y_i) = \prod_{i=1}^{n} P(x_i | y_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x_j - \mu_i)^2}{2\sigma_j^2}\right) $$
- Complex model $\rightarrow$ High variance with limited data!!!
  - Suppose $Y$ is composed of $k$ classes
    - Likelihood, $P(X|Y)$:
      $$ P(x_1 \ldots x_n | y_i) = \prod_{i=1}^{n} P(x_i | y_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(x_j - \mu_i)^2}{2\sigma_j^2}\right) $$

Gaussian Discriminative Analysis

- learning $f: \mathbf{x} \rightarrow Y$, where
  - $\mathbf{x}$ is a vector of real-valued features, $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$
  - $Y$ 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:
    $$ p(x_n, y_n^i = 1 | \mu, \Sigma) = p(y_n^i = 1) \times p(x_n | y_n^i = 1, \mu, \Sigma) $$
    $$ = \pi_i \frac{1}{(2\pi\Sigma)^{n/2}} \exp\left(-\frac{1}{2}(x_n - \mu_i)^T \Sigma^{-1} (x_n - \mu_i)\right) $$
  - Given a datum $x_n$, we predict its label using the conditional probability of the label given the datum:
    $$ p(y_n^i = 1 | x_n, \mu, \Sigma) = \frac{\pi_i}{\sum_{j=1}^{k} \pi_j \frac{1}{(2\pi\Sigma)^{n/2}} \exp\left(-\frac{1}{2}(x_n - \mu_j)^T \Sigma^{-1} (x_n - \mu_j)\right)} $$
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.,

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

- 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

$$\theta_1 \ldots \theta_m$$

$\alpha_{k-1}$

$\alpha_{k}$
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:

  $$y^* = h_{NB}(x) = \arg \max_y P(y) P(x^1, \ldots, x^m | y)$$
  $$= \arg \max_y P(y) \prod \limits_i P(x^i | y)$$

- If assumption holds, NB is optimal classifier!

The Gaussian Discriminative Naïve Bayes Classifier

- When $X$ is multivariate-Gaussian vector:
  - The joint probability of a datum and its label is:
    $$p(x_n, y_n = 1 | \mu, \Sigma) = p(y_n = 1) \times p(x_n | y_n = 1, \mu_n, \Sigma)$$
    $$= \pi_1 \frac{1}{(2\pi)^{d/2}} \exp \left\{ -\frac{1}{2} (x_n - \mu_n)^\top \Sigma^{-1} (x_n - \mu_n) \right\}$$

  - The naïve Bayes simplification
    $$p(x_n, y_n = 1 | \mu, \sigma) = p(y_n = 1) \times \prod \limits_j p(x_n^j | y_n = 1, \mu_n^j, \sigma_n^j)$$
    $$= \pi_1 \prod \limits_j \frac{1}{\sqrt{2\pi} \sigma_n^j} \exp \left\{ -\frac{1}{2} \left( \frac{x_n^j - \mu_n^j}{\sigma_n^j} \right)^2 \right\}$$

  - More generally:
    $$p(x_n, y_n | \eta, \pi) = p(y_n | \pi) \times \prod \limits_j p(x_n^j | y_n, \eta)$$

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

- Understanding the predictive distribution
  \[ p(y^k_a = 1 | x_a, \bar{\mu}, \Sigma, \pi) = \frac{p(y^k_a = 1, x_a | \bar{\mu}, \Sigma, \pi)}{p(x_a | \bar{\mu}, \Sigma)} = \frac{\pi_k N(x_a | \mu_k, \Sigma_k)}{\sum_k \pi_k N(x_a | \mu_k, \Sigma_k)} * \]

- Under naïve Bayes assumption:
  \[ p(y^k_a = 1 | x_a, \bar{\mu}, \Sigma, \pi) = \pi_k \exp \left( \sum_{j=1}^{n} \left( \frac{x_{aj} - \mu_{kj}}{\sigma_{kj}} \right)^2 - \log \sigma_{kj} - C \right) \]
  \[ \sum_k \pi_k \exp \left( \sum_{j=1}^{n} \left( \frac{x_{aj} - \mu_{kj}}{\sigma_{kj}} \right)^2 - \log \sigma_{kj} - C \right) \]

- For two class (i.e., \( K=2 \)), and when the two classes have the same variance, * turns out to be a logistic function
  \[ p(y^2_a = 1 | x_a) = \frac{1}{1 + e^{-\theta^T x_a}} \]
  \[ \theta = \theta(\bar{\mu}, \bar{\sigma}, \pi) \]

The decision boundary

- The predictive distribution
  \[ p(y^k_a = 1 | x_a) = \frac{1}{1 + \exp \left( -\sum_{l=1}^{K} \theta_{kl} x_{al} \right)} = \frac{1}{1 + e^{-\theta^T x_a}} \]

- The Bayes decision rule:
  \[ \ln \frac{p(y^k_a = 1 | x_a)}{p(y^k_a = 1 | x_a)} = \ln \left( \frac{1 + e^{-\theta^T x_a}}{1 + e^{-\theta^T x_a}} \right) = \theta^T x_a = \theta \]

- For multiple class (i.e., \( K>2 \)), * correspond to a softmax function
  \[ p(y^k_a = 1 | x_a) = \frac{e^{-\theta_{k}^T x_a}}{\sum_l e^{-\theta_{l}^T x_a}} \]
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 $x_{ij}$ of each attribute $X_i$
  - estimate $\theta_{ijk} \equiv P(X_i^j = x_{ij}|Y = y_k)$

- **Classify ($X_{\text{new}}$)**

\[
Y_{\text{new}} \leftarrow \arg\max_{y_k} \pi_k \prod_{i} \theta_{ijk}
\]

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 \textit{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 \textit{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 \( \theta = <\theta_0, \theta_1, \ldots, \theta_m> \) to maximize the conditional likelihood of training data

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

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

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

\[ \theta = \arg \max_{\theta} \ln \prod_{i} P(y_i \mid x_i; \theta) \]
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:
  \[ P(y|x) = \mu(x)^y (1 - \mu(x))^{1-y} \]

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

Maximizing Conditional Log Likelihood

- The objective:
  \[ p(x, y) \]

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

- 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:
  \[ \mu = \frac{1}{1 + e^{-\theta}} \]
  \[ \frac{d\mu}{dt} = \mu(1 - \mu) \]

- The gradient:
  \[ \frac{\partial l(\theta)}{\partial \theta_i} = (y_i - \mu_i \chi_i - \frac{1}{\mu_i} \chi_i) = (y_i - \hat{y}_i \chi_i - (1 - \hat{y}_i) \chi_i) \]

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

$$H = \nabla_{\theta} \nabla_{\theta}l(\theta) = \sum u_i (1 - u_i) x_i x_i^T = X^T R X$$

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

Iterative reweighed least squares (IRLS)

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

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

which can also derived from Newton-Raphson

- Now for logistic regression:

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

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

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

$$\quad = (X^T R X)^{-1} X^T R z$$
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 $O(Nd^3)$ 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 $X^i$ 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!

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

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**: $P(X_i=x_i|Y=y) = P(X^k=x^k|Y=y)$
  - “Bag of words” model – order of words on the page ignored
  - Sounds really silly, but often works very well!

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

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

$$h_{NB}(x_{\text{new}}) = \arg \max_y P(y) \prod_{i=1}^{\text{LengthDoc}} P(x_{\text{new}}^i|y)$$
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

![Accuracy vs. Training Ratio](image_url)
Results: Multiple Classes

Accuracy

Training Ratio

5-out-of-20 classes

10-out-of-20 classes

All 20 classes

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](image)

- 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:

\[
p(y|x) = \frac{\pi_y \exp \left( \sum \frac{1}{2\sigma_j} (x_j - \mu_j)^2 - \log \sigma_j - C \right)}{\sum \pi_y \exp \left( \sum \frac{1}{2\sigma_j} (x_j - \mu_j)^2 - \log \sigma_j - C \right)}
\]

** NB:

\[
\mu(x) = \frac{1}{1 + e^{-\theta^T x}}
\]

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

Naïve Bayes vs Logistic Regression

- Asymptotic comparison (# training examples → 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 $h_{Dis,m}$ be logistic regression trained on $n$ examples in $m$ dimensions. Then with high probability:

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

- Implication: if we want $\epsilon(h_{Dis,m}) \leq \epsilon(h_{Dis,\infty}) + \epsilon_0$ for some small constant $\epsilon_0$, it suffices to pick order $m$ examples.
  - 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 $\varepsilon, \delta > 0$, 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 $n = O((1/\varepsilon_1^2) \log(m/\delta))$

- Then with probability at least $1-\delta$, after $n$ examples:
  
  1. For discrete input, for all $i$ and $b$
     \[ |\tilde{p}(x_i | y = b) - p(x_i | y = b)| \leq \varepsilon_1 \]
     \[ |\tilde{p}(y = b) - p(y = b)| \leq \varepsilon_1 \]

  2. For continuous inputs, for all $i$ and $b$
     \[ |\tilde{\mu}_{i | y = b} - \mu_{i | y = b}| \leq \varepsilon_1 \]
     \[ |\tilde{\sigma}_{i | y = b}^2 - \sigma_{i | y = b}^2| \leq \varepsilon_1 \]

Some experiments from UCI data sets

![Image of some experiments](https://example.com/image.png)
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

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

- Generative vs. Discriminative classifiers
  - Bias vs. variance tradeoff
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., $Y=\{\text{SpamEmail or not}\}$, $w^{'pill'}=\{'pill', \text{‘enhancement’, ‘Rolex’, ‘enlarge’ }…\}$
  
  - $P(\text{enlargement}>0 \mid Y=T) = 0$

- 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 $\mathcal{D}$
  \[ \hat{\theta} = \text{arg max}_{\theta} P(\mathcal{D}|\theta) \]

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

- 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):
  \[ \hat{\theta} = \text{arg max}_{\theta} \frac{P(\mathcal{D}|\theta)p(\theta)}{P(\mathcal{D})} \]

- Given prior:
  - Consider binary feature
  - $\theta$ is a Bernoulli rate
  \[ P(\theta, \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1} \]
  - $\beta = \text{Count}(X=a) \leftarrow$ number of examples where $X=a$
  \[ P(\theta | \mathcal{D}) = \frac{\theta^{\hat{\beta}+\alpha-1}(1-\theta)^{\hat{\beta}+\beta-1}}{\hat{\beta}!/(\hat{\beta}_{+\gamma}/\hat{\beta}_{+} + \hat{\beta}_{+}\hat{\beta}_{+})} \sim B(\hat{\beta}_{+\gamma}/\hat{\beta}_{+} + \hat{\beta}_{+}\hat{\beta}_{+}) \]
Bayesian learning for NB parameters – a.k.a. smoothing

- Posterior distribution of \( \theta \)
  - Bernoulli: \( P(\theta | D) = \frac{\theta^{\gamma_Y+\alpha_Y-1}(1-\theta)^{\gamma_N+\alpha_N-1}}{H(\gamma_Y+\alpha_Y, \gamma_N+\alpha_N)} \sim \text{Beta}(\gamma_Y+\alpha_Y, \gamma_N+\alpha_N) \)
  - Multinomial: \( P(\theta | T) = \frac{\prod_{i=1}^{K} \theta_i^{\gamma_{iY}+\alpha_i-1}}{H(\alpha_1, \ldots, \alpha_K)} \sim \text{Dirichlet}(\gamma_1+\alpha_1, \ldots, \gamma_K+\alpha_K) \)

- 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 \( \gamma_b = \text{Count}(Y=b) \)

- Prior
  \[
  Q(X|Y) \propto \text{Multinomial}(\alpha_{i1}, \ldots, \alpha_{iK}) \text{ or Multinomial}(\alpha/K) \\
  Q(Y) \propto \text{Multinomial}(\tau_{i1}, \ldots, \tau_{iM}) \text{ or Multinomial}(\tau/M)
  \]
  \( m \) “virtual” examples

- MAP estimate
  \[
  \hat{\pi}_k = \arg \max_{\pi_k} \prod_k P(Y = y_k; \pi_k) P(\pi_k | \bar{T}) = ? \\
  \hat{\theta}_{ijk} = \arg \max_{\theta_{ijk}} \prod_i P(X^i = x_{ij}; Y = y_k; \theta_{ijk}) P(\theta_{ijk} | \bar{a}_{ijk}) = ?
  \]
  - Now, even if you never observe a feature/class, posterior probability never zero