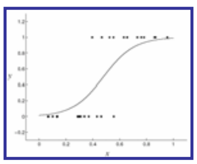
Advanced Machine Learning

Generative verses discriminative classifier

Eric Xing

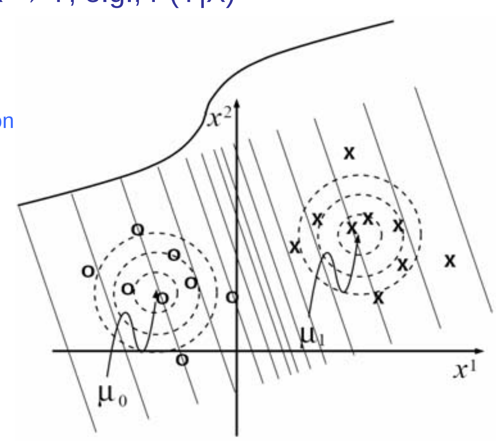
Lecture 4, August 10, 2009



Reading:

Discussion: Generative and discriminative classifiers

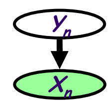
- Goal: Wish to learn f: X → Y, e.g., P(Y|X)
- Generative:
  - Modeling the joint distribution of all data
- Discriminative:
  - Modeling only points at the boundary



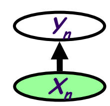
Generative vs. Discriminative Classifiers

- Goal: Wish to learn f: X → 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)

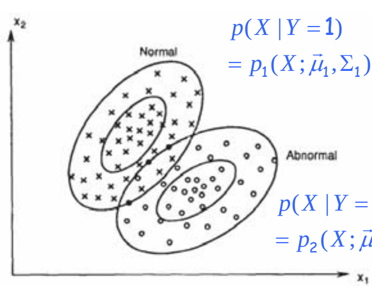


- Discriminative classifiers (e.g., logistic regression)
  - Directly assume some functional form for P(Y|X)
    This is a '***discriminative***' model of the data!
  - Estimate parameters of P(Y|X) directly from training data



Suppose you know the following …

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



$$p(X \mid Y=1) = p_1(X; \vec{\mu}_1, \Sigma_1)$$

$$p(X \mid Y=2) = p_2(X; \vec{\mu}_2, \Sigma_2)$$

Bayes classifier:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

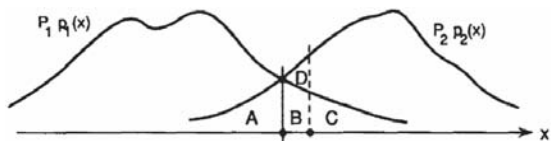
- Class prior (i.e., "weight"): P(Y)

- This is a generative model of the data!

Optimal classification

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

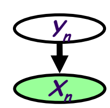
$$error_{true}(h_{Bayes})) \leq error_{true}(h), \;\; \forall h(\mathbf{x})$$



- How to learn a Bayes classifier?
  - Recall density estimation. We need to estimate P(X|y=k), and P(y=k) for all k

Gaussian Discriminative Analysis

- learning f: X → Y, where
  - X is a vector of real-valued features, $\mathbf{X}_n = < X_{n,1} \ldots X_{n,m} >$
  - 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:



$$p(\mathbf{x}_n, y_n^k = 1 \mid \mu, \sigma) = p(y_n^k = 1) \times p(\mathbf{x}_n \mid y_n^k = 1, \mu, \sigma)$$
$$= \pi_k \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ -\tfrac{1}{2\sigma^2} (\mathbf{x}_n - \mu_k)^2 \right\}$$

  - Given a datum $\mathbf{x}_n$, we predict its label using the conditional probability of the label given the datum:

$$p(y_n^k = 1 \mid \mathbf{x}_n, \mu, \sigma) = \frac{\pi_k \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ -\tfrac{1}{2\sigma^2} (\mathbf{x}_n - \mu_k)^2 \right\}}{\sum_{k'} \pi_{k'} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ -\tfrac{1}{2\sigma^2} (\mathbf{x}_n - \mu_{k'})^2 \right\}}$$

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

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

- e.g.,

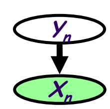
$$P(Thunder | Rain, Lightning) = P(Thunder | Lightning)$$

- Equivalent to:

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

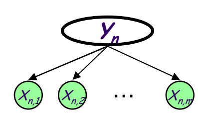
Naïve Bayes Classifier

- When X is multivariate-Gaussian vector:
  - The joint probability of a datum and it label is:



$$p(\mathbf{x}_n, y_n^k = 1 | \vec{\mu}, \Sigma) = p(y_n^k = 1) \times p(\mathbf{x}_n | y_n^k = 1, \vec{\mu}, \Sigma)$$
$$= \pi_k \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\tfrac{1}{2}(\mathbf{x}_n - \vec{\mu}_k)^T \Sigma^{-1} (\mathbf{x}_n - \vec{\mu}_k)\right\}$$

  - The naïve Bayes simplification



$$p(\mathbf{x}_n, y_n^k = 1 | \mu, \sigma) = p(y_n^k = 1) \times \prod_j p(x_{n,j} | y_n^k = 1, \mu_{k,j}, \sigma_{k,j})$$
$$= \pi_k \prod_j \frac{1}{(2\pi\sigma_{k,j}^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma_{k,j}^2}(x_{n,j} - \mu_{k,j})^2\right\}$$

  - More generally: $p(\mathbf{x}_n, y_n | \eta, \pi) = p(y_n | \pi) \times \prod_{j=1}^{m} p(x_{n,j} | y_n, \eta)$
    - Where $p(.\,|\,.)$ is an arbitrary conditional (discrete or continuous) 1-D density

The predictive distribution

- Understanding the predictive distribution

$$p(y_n^k = 1 \mid x_n, \vec{\mu}, \Sigma, \pi) = \frac{p(y_n^k = 1, x_n \mid \vec{\mu}, \Sigma, \pi)}{p(x_n \mid \vec{\mu}, \Sigma)} = \frac{\pi_k N(x_n, \mid \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} N(x_n, \mid \mu_{k'}, \Sigma_{k'})} \quad *$$

- Under naïve Bayes assumption:

$$p(y_n^k = 1 \mid x_n, \vec{\mu}, \Sigma, \pi) = \frac{\pi_k \exp\left\{-\sum_j \left(\frac{1}{2\sigma_{k,j}^2}(x_n^j - \mu_k^j)^2 - \log \sigma_{k,j} - C\right)\right\}}{\sum_{k'} \pi_{k'} \exp\left\{-\sum_j \left(\frac{1}{2\sigma_{k',j}^2}(x_n^j - \mu_{k'}^j)^2 - \log \sigma_{k',j} - C\right)\right\}} \quad **$$

- For two class (i.e., *K*=2), and when the two classes haves the same variance, ** turns out to be a logistic function

$$p(y_n^1 = 1 \mid x_n) = \frac{1}{1 + \frac{\pi_2 \exp\left\{-\sum_j \left(\frac{1}{2\sigma_j^2}(x_n^j - \mu_2^j)^2 - \log \sigma_j - C\right)\right\}}{\pi_1 \exp\left\{-\sum_j \left(\frac{1}{2\sigma_j^2}(x_n^j - \mu_1^j)^2 - \log \sigma_j - C\right)\right\}}} = \frac{1}{1 + \exp\left\{-\sum_j \left(x_n^j \frac{1}{\sigma_j^2}(\mu_1^j - \mu_2^j) + \frac{1}{\sigma_j^2}([\mu_1^j]^2 - [\mu_2^j]^2)\right) + \log \frac{(1-\pi_1)}{\pi_1}\right\}}$$

$$= \frac{1}{1 + e^{-\theta^T x_n}}$$

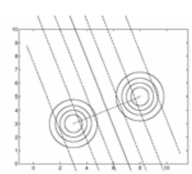
The decision boundary

- The predictive distribution

$$p(y_n^1 = 1 \mid x_n) = \frac{1}{1 + \exp\left\{-\sum_{j=1}^{M} \theta_j x_n^j - \theta_0\right\}} = \frac{1}{1 + e^{-\theta^T x_n}}$$

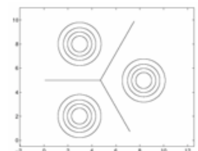
- The Bayes decision rule:

$$\ln \frac{p(y_n^1 = 1 \mid x_n)}{p(y_n^2 = 1 \mid x_n)} = \ln \left(\frac{\frac{1}{1 + e^{-\theta^T x_n}}}{\frac{e^{-\theta^T x_n}}{1 + e^{-\theta^T x_n}}} \right) = \theta^T x_n$$



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

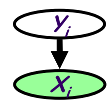
$$p(y_n^k = 1 \mid x_n) = \frac{e^{-\theta_k^T x_n}}{\sum_j e^{-\theta_j^T x_n}}$$



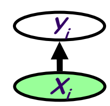
Generative vs. Discriminative Classifiers

- Goal: Wish to learn f: X → 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)



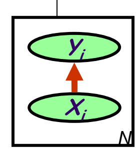
- Discriminative classifiers:
  - Directly assume some functional form for P(Y|X)
    This is a '***discriminative***' model of the data!
  - Estimate parameters of P(Y|X) directly from training data



Linear Regression

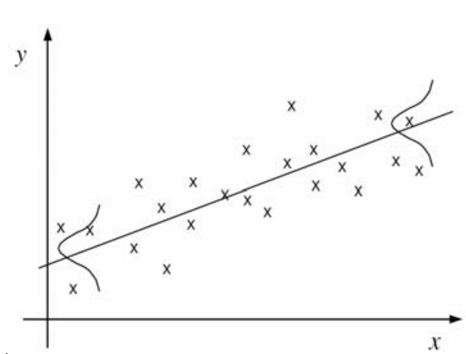
- The data:

$$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \cdots, (x_N, y_N)\}$$



- Both nodes are observed:
  - *X* is an input vector
  - *Y* is a response vector
    (we first consider y as a generic continuous response vector, then we consider the special case of classification where y is a discrete indicator)

- A regression scheme can be used to model $p(y|x)$ directly, rather than $p(x,y)$



Linear Regression

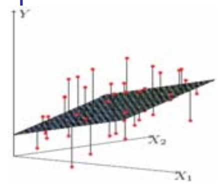
- Assume that Y (target) is a linear function of X (features):
  - e.g.:
$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$
  - let's assume a vacuous "feature" X^0=1 (this is the intercept term, why?), and define the feature vector to be:
$$\mathbf{x} = [1, x_1, x_2]$$
  - then we have the following general representation of the linear function:
$$\hat{y} = \mathbf{x}^T \theta$$
- Our goal is to pick the optimal θ . How!
  - We seek θ that minimize the following **cost function**:
$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (\hat{y}_i(\vec{x}_i) - y_i)^2$$

The Least-Mean-Square (LMS) method

- Consider a **gradient descent** algorithm:
$$\theta_j^{t+1} = \theta_j^{t} - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \Big|_t$$
- Now we have the following descent rule:
$$\theta_j^{t+1} = \theta_j^{t} + \alpha \sum_{i=1}^{n} (y_i - \bar{\mathbf{x}}_i^T \theta^t) x_i^j$$
- For a single training point, we have:
$$\theta_j^{t+1} = \theta_j^{t} + \alpha (y_i - \bar{\mathbf{x}}_i^T \theta^t) x_i^j$$
  - This is known as the LMS update rule, or the Widrow-Hoff learning rule
  - This is actually a "**stochastic**", "**coordinate**" descent algorithm
  - This can be used as a **on-line** algorithm

Probabilistic Interpretation of LMS

- 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:

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

- By independence assumption:

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

Probabilistic Interpretation of LMS, cont.

- Hence the log-likelihood is:

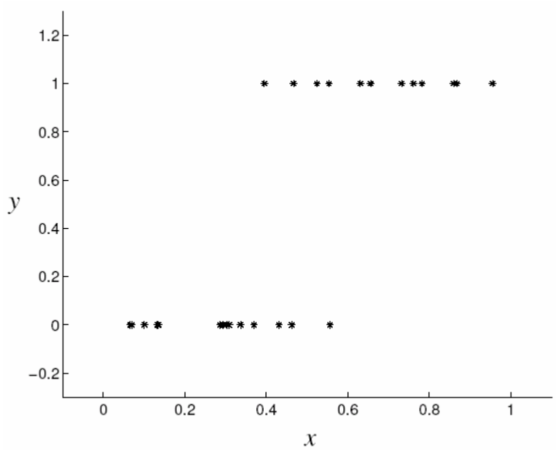
$$l(\theta) = n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta^T \mathbf{x}_i)^2$$

- Do you recognize the last term?

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

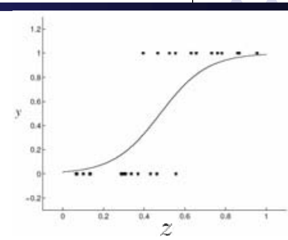
- Thus under independence assumption, LMS is equivalent to MLE of θ !

Classification and logistic regression



The logistic function

$$g(z) = \frac{1}{1 + e^{-z}}$$



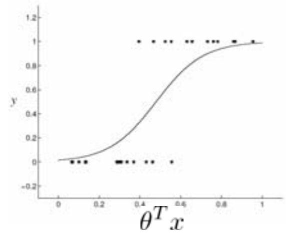
Logistic regression (sigmoid classifier)

- The condition distribution: a Bernoulli

$$p(y \mid x) = \mu(x)^y (1 - \mu(x))^{1-y}$$

where μ is a logistic function

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



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

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

Training Logistic Regression: MCLE

- Estimate parameters θ=<θ_0, θ_1, ... θ_m> 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 = $\prod_{i=1}^{N} P(x_i | y_i; \theta)$

$$\theta = \arg\max_{\theta} \ln \prod_i P(y_i | x_i; \theta)$$

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: $P(y|x) = \mu(x)^y(1-\mu(x))^{1-y}$

$$\begin{aligned} l(\theta) = \sum_i \ln P(y_i|x_i;\theta) &= \sum_i y_i \ln u(x_i) + (1-y_i)\ln(1-\mu(x_i)) \\ &= \sum_i y_i \ln \frac{u(x_i)}{1-\mu(x_i)} + \ln(1-\mu(x_i)) \\ &= \sum_i y_i\theta^T x_i - \theta^T x_i + \ln(1+e^{-\theta^T x_i}) \\ &= \sum_i (y_i - 1)\theta^T x_i + \ln(1+e^{-\theta^T x_i}) \end{aligned}$$

Maximizing Conditional Log Likelihood

- The objective:

$$\begin{aligned} l(\theta) &= \ln \prod_i P(y_i|x_i;\theta) \\ &= \sum_i (y_i - 1)\theta^t x_i + \ln(1+e^{-\theta^T x_i}) \end{aligned}$$

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

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

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 θ^* 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 θ is vector-valued, thus we need the following generalization:

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

- ∇ is the gradient operator over the function
- H is known as the Hessian of the function

The Newton-Raphson method

- In LR the θ is vector-valued, thus we need the following generalization:

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

- ∇ is the gradient operator over the function

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

- H is known as the Hessian of the function

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

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

Iterative reweighed least squares (IRLS)

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

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

which can also derived from Newton-Raphson

- Now for logistic regression:

$$\begin{aligned}\theta^{t+1} &= \theta^t + H^{-1}\nabla_{\theta^t} l(\theta^t) \\ &= \theta^t - (\mathbf{X}^T\mathbf{R}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{u}-\mathbf{y}) \\ &= (\mathbf{X}^T\mathbf{R}\mathbf{X})^{-1}\{\mathbf{X}^T\mathbf{R}\mathbf{X}\theta^t - \mathbf{X}^T(\mathbf{u}-\mathbf{y}) \\ &= (\mathbf{X}^T\mathbf{R}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{R}\mathbf{z}\end{aligned}$$

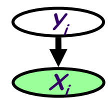
Logistic regression: practical issues

- 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:

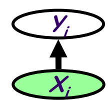
Generative vs. Discriminative Classifiers

- Goal: Wish to learn f: X → 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:

NB:

$$p(y\,|\,\mathbf{x}) = \frac{\pi_k \exp\left\{-\sum_j \left(\frac{1}{2\sigma_{k,j}^2}(x_j-\mu_{k,j})^2 - \log\sigma_{k,j} - C\right)\right\}}{\sum_{k'}\pi_{k'} \exp\left\{-\sum_j \left(\frac{1}{2\sigma_{k',j}^2}(x_j-\mu_{k',j})^2 - \log\sigma_{k',j} - C\right)\right\}} \quad **$$

LR:

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

- 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:

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

- Implication: if we want $\epsilon(h_{Dis,m}) \leq \epsilon(h_{Dis,\infty}) + \epsilon_0$ for some small constant ε_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 ε_1, δ>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-δ, after n examples:

  1. For discrete input, $|\hat{p}(x_i|y=b) - p(x_i|y=b)| \leq \epsilon_1$, $|\hat{p}(y=b) - p(y=b)| \leq \epsilon_1$ for all i and b
  2. For continuous inputs, $|\hat{\mu}_{i|y=b} - \mu_{i|y=b}| \leq \epsilon_1$, $|\hat{\sigma}^2_{i|y=b} - \sigma^2_{i|y=b}| \leq \epsilon_1$ for all i and b

Some experiments from UCI data sets

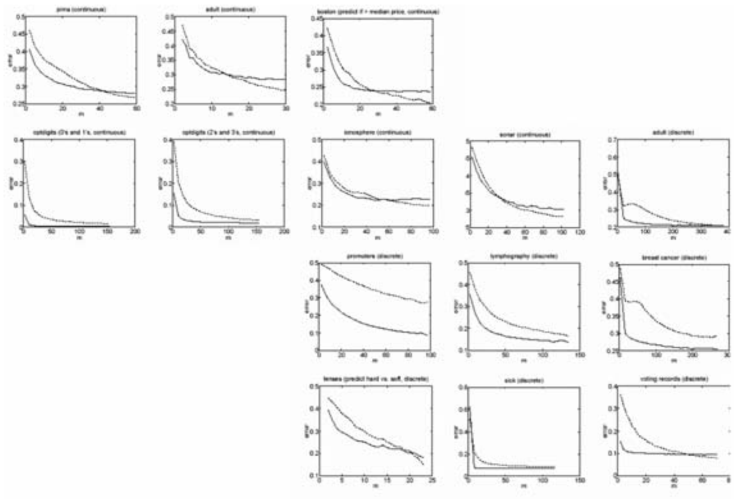
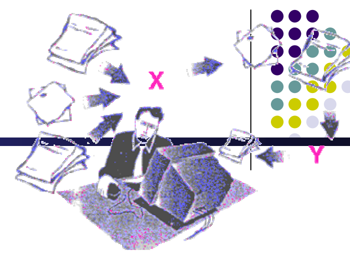


Figure 1: Results of 15 experiments on datasets from the UCI Machine Learnin repository. Plots are of generalization error vs. m (averaged over 1000 randon train/test splits). Dashed line is logistic regression; solid line is naive Bayes.

Case study



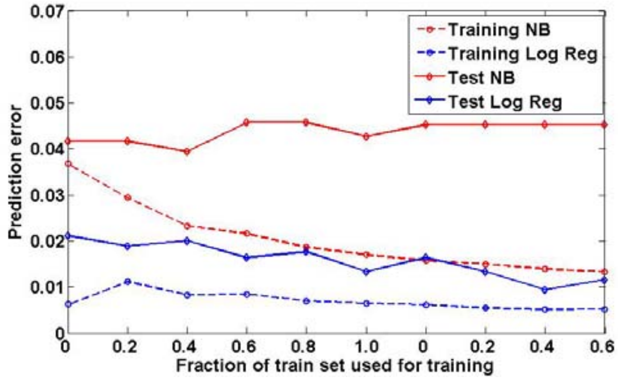
- Dataset
  - 20 News Groups (20 classes)
  - 61,118 words, 18,774 documents

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

- 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.

Generalization error (1)

- Versus training size

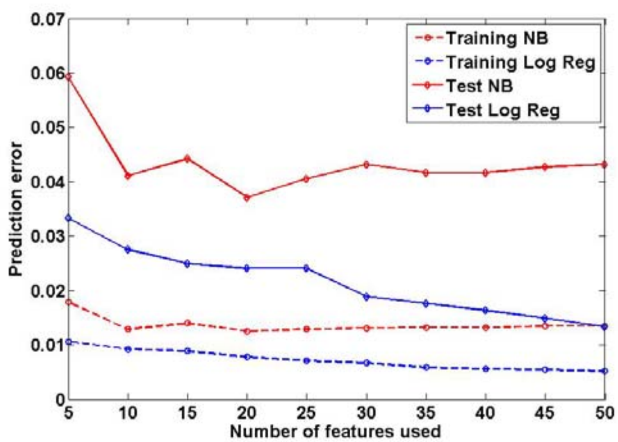


- 30 features.
- A fixed test set
- Training set varied from 10% to 100% of the training set

 37

Generalization error (2)

- Versus model size



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

 38

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