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
Discussion: Generative and discriminative classifiers

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

- **Discriminative:**
  - Modeling only points at the boundary
  - How? Regression!

Linear regression

- The data:
  \[ \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)\} \]

- 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 \( p(x,y) \)
The logistic function

\[ g(z) = \frac{1}{1 + e^{-z}} \]
Logistic regression (sigmoid classifier)

- The condition distribution: a Bernoulli
  \[ p(y|x) = \mu(x)^y (1 - \mu(x))^{1-y} \]
  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 \( p(y|x) \) 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 = \( \prod_{i=1}^{N} P(y_i|x_i; \theta) \)

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

\[
l(\theta) = \sum_{i} \ln P(y_i | x_i; \theta) = \sum_{i} y_i \ln \mu(x_i) + (1 - y_i) \ln (1 - \mu(x_i))
\]

\[
= \sum_{i} y_i \ln \frac{\mu(x_i)}{1 - \mu(x_i)} + \ln (1 - \mu(x_i))
\]

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

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

\[ l(\theta) = \ln \prod_{i} P(y_i|\tau_i; \theta) \]
\[ = \sum_{i} (y_i - 1)\theta^T x_i + \ln(1 + e^{-\theta^T x_i}) = \sum_{i} (y_i - 1)\theta^T x_i - \ln \mu(\theta^T x_i) \]

- 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

Overfitting …
How about MAP?

- It is very common to use **regularized** maximum likelihood.

- One common approach is to define priors on $\theta$
  - Normal distribution, zero mean, identity covariance
  - Helps avoid very large weights and overfitting

  \[
P(y|x; \theta) = \mu(\theta^T x)^y (1 - \mu(\theta^T x))^{1-y}
\]

  \[
P(\theta) = \text{Normal}(0, \lambda^{-1} I)
\]

  \[
l'(\theta) = \sum_i (y_i - 1)\theta^T x_i - \ln \mu(\theta^T x_i) - \frac{\lambda}{2} \theta^T \theta
\]

- MAP estimate

---

MLE vs MAP

- Maximum conditional likelihood estimate

  \[
  \theta = \arg \max_\theta \ln \prod_i P(y_i|x_i; \theta)
  \]

  \[
  \theta_j \leftarrow \theta_j + \eta \sum_i (y_i - P(y_i = 0|x_i; \theta)x_i^j
  \]

- Maximum a posteriori estimate

  \[
  \theta = \arg \max_\theta \ln p(\theta) \prod_i P(y_i|x_i; \theta)
  \]

  \[
  \theta_j \leftarrow \theta_j + \eta \{ \sum_i (y_i - P(y_i = 0|x_i; \theta))x_i^j - \lambda \theta_j \}
  \]
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

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

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

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:
  \[
  \begin{aligned}
  \theta^{t+1} &= (X^T R X)^{-1} X^T R z \\
  \text{where} & \quad z = X \theta^t - R^{-1} (u - y) \\
  \text{and} & \quad R_{ii} = u_i (1 - u_i)
  \end{aligned}
  \]
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.

Case study

- Dataset
  - 20 News Groups (20 classes)
  - Download: (http://people.csail.mit.edu/jrennie/20Newsgroups/)
  - 61,118 words, 18,774 documents
  - Class labels descriptions

<table>
<thead>
<tr>
<th>comp.graphics</th>
<th>rec.autos</th>
<th>sci.crypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp.os.ms-windows</td>
<td>rec.motorcycles</td>
<td>sci.electronics</td>
</tr>
<tr>
<td>misc</td>
<td>rec.sport.baseball</td>
<td>sci.med</td>
</tr>
<tr>
<td>comp.sys.mac.hardware</td>
<td>rec.sport.hockey</td>
<td>sci.space</td>
</tr>
<tr>
<td>comp.windows.x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| | talk.politics.misc | talk.religion.misc |
| | talk.politics.guns | alt.atheism |
| | talk.politics.mideast | soc.religion.christian |
Experimental setup

- Parameters
  - Binary feature;
  - Using (stochastic) Gradient descent vs. IRLS to estimate parameters

- Training/Test Sets:
  - Subset of 20ng (binary classes)
  - Use SVD to do dimension reduction (to 100)
  - Random select 50% for training
  - 10 run and report average result

- Convergence Criteria: RMSE in training set

Convergence curves

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

Gaussian Discriminative Analysis

- learning \( f: X \rightarrow Y \), where
  - \( X \) is a vector of real-valued features, \( < X_1...X_m > \)
  - \( Y \) is boolean

- What does that imply about the form of \( P(Y|X) \)?
  - The joint probability of a datum and its label is:
    \[
    p(x_i, y_i^1 = 1 | \mu, \sigma) = p(y_i^1 = 1) \times p(x_i | y_i^1 = 1, \mu, \sigma) \\
    = \pi_i \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2} (x_i - \mu_i)^2 \right\}
    \]

  - Given a datum \( x_i \), we predict its label using the conditional probability of the label given the datum:
    \[
    p(y_i^1 = 1 | x_i, \mu, \sigma) = \frac{\pi_i}{\sum_{i'} \pi_i'} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp \left\{ -\frac{1}{2\sigma^2} (x_i - \mu_i)^2 \right\}
    \]
Naïve Bayes Classifier

- When X is a multivariate-Gaussian vector:
  
  The joint probability of a datum and its label is:
  \[
p(x_i, y_i | \mu, \Sigma) = p(y_i = 1|x_i) = \frac{1}{(2\pi \sigma_i)^{n/2}} \exp\left(-\frac{1}{2}(x_i - \mu_i)^T \Sigma^{-1} (x_i - \mu_i) \right)
  \]

- The naïve Bayes simplification
  \[
p(x_i, y_i | \mu, \Sigma) = p(y_i = 1|x_i) = \prod_{j} p(x_i^j | \mu_j, \sigma_j)
  \]

- More generally:
  \[
p(x_i, y_i | \eta, \pi) = p(y_i = 1|x_i) \prod_{j} p(x_i^j | y_i, \eta)
  \]

- Where \( p(. | .) \) is an arbitrary conditional (discrete or continuous) 1-D density

---

The predictive distribution

- Understanding the predictive distribution
  \[
p(y_i = 1|x_1, \mu, \Sigma) = \frac{p(y_i = 1|x) \prod_{j} p(x_j | y_i, \mu_j, \sigma_j)}{p(x_1, \ldots, x_m | \mu, \Sigma)}
  \]

- Under naïve Bayes assumption:
  \[
p(y_i = 1|x) = \frac{\pi_i \exp\left(-\frac{1}{2} \sum_j \left( \frac{1}{2\sigma_i^2} (x_i^j - \mu_j)^2 - \log \sigma_i^2 - C \right) \right)}{\sum_i \pi_i \exp\left(-\frac{1}{2} \sum_j \left( \frac{1}{2\sigma_i^2} (x_i^j - \mu_j)^2 - \log \sigma_i^2 - C \right) \right)}
  \]

- For two class (i.e., \( K=2 \)), and when the two classes have the same variance, \( \pi_i \) turns out to be the logistic function
  \[
p(y_i = 1|x) = \frac{1}{1 + \exp\left( \sum_j \frac{1}{2} (x_i^j - \mu_j)^2 + \log \pi_i - \log (1 - \pi_i) \right)}
  \]

© Eric Xing @ CMU, 2006-2008
The decision boundary

- The predictive distribution
  \[ p(y^i = 1 | x_n) = \frac{1}{1 + \exp\left(-\sum_{j=1}^{n} \theta_j x_n^j - \theta_0\right)} = \frac{1}{1 + e^{-x^T \theta}}. \]

- The Bayes decision rule:
  \[ \ln \frac{p(y^i = 1 | x_n)}{p(y^i = 0 | x_n)} = \ln \left( \frac{1 + e^{-x^T \theta}}{1 + e^{x^T \theta}} \right) = \theta^T x_n. \]

- For multiple class (i.e., \(K>2\)), \(\ast\) correspond to a softmax function
  \[ p(y^k = 1 | x_n) = \frac{e^{-\theta_k^T x_n}}{\sum_{j=1}^{K} e^{-\theta_j^T x_n}}. \]

Naive Bayes vs Logistic Regression

- Consider \(Y\) boolean, \(X\) continuous, \(X = X^1 \ldots X^m\)
- Number of parameters to estimate:
  \[ \text{NB}: \]
  \[ \text{LR}: \]

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

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

- Implication: if we want $\epsilon(h_{\text{Dis},m}) \leq \epsilon(h_{\text{Dis},\infty}) + \epsilon_c$ for some small constant $\epsilon_0$, it suffices to pick order $m$ examples

\[\Rightarrow\text{Convergences to its asymptotic classifier, in order } m \text{ examples}\]

- 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 $n = O\left(\frac{1}{\epsilon_1^2} \log(m/\delta)\right)$

- Then with probability at least $1 - \delta$, after $n$ examples:

1. For discrete input, $|\hat{\epsilon}(x_i|y = b) - p(x_i|y = b)| \leq \epsilon_1$ for all $i$ and $b$

2. For continuous inputs, $|\hat{\mu}_{\epsilon|y=b} - \mu_{\epsilon|y=b}| \leq \epsilon_1$ for all $i$ and $b$
Some experiments from UCI data sets

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.
Generalization error (1)

- Versus training size

![Graph showing test prediction error vs. fraction of training set used for training.](Image)

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

Generalization error (2)

- Versus model size

![Graph showing test prediction error vs. number of features.](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
Summary

- 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
  - MLE training: pick W to maximize $P(Y | X; \theta)$
  - MAP training: pick W to maximize $P(\theta | X,Y)$
    - 'regularization'
    - helps reduce overfitting
- Gradient ascent/descent
  - General approach when closed-form solutions unavailable
- Generative vs. Discriminative classifiers
  - Bias vs. variance tradeoff