Generative vs. Discriminative Classifiers

- **Generative classifier**, e.g., Naïve Bayes:
  - Assume some functional form for $P(Y | X)$
  - Estimate parameters of $P(Y | X)$ directly from training data
  - Use Bayes rule to calculate $P(Y | X = x)$
  - This is a 'generative' model
  - Indirect computation of $P(Y | X)$ through Bayes rule
  - As a result, can also generate a sample of the data: $P(X) = \sum_y P(y) P(X | y)$

- **Discriminative classifiers**, e.g., Logistic Regression:
  - Assume some functional form for $P(Y | X)$
  - Estimate parameters of $P(Y | X)$ directly from training data
  - This is the 'discriminative' model
  - Directly learn $P(Y | X)$
  - But cannot obtain a sample of the data, because $P(X)$ is not available

For Univariate Linear Regression:

$$h_w(x) = w_1 x + w_0$$

$$\text{Loss}(h_w) = \sum_{j=1}^{n} (y_j - (w_1 x_j + w_0))^2$$

For Finding Minimum Loss:

$$\text{Argmin}_w \text{Loss}(h_w)$$

$$\frac{\partial}{\partial w_0} \text{Loss}(h_w) = 0$$

$$\frac{\partial}{\partial w_1} \text{Loss}(h_w) = 0$$
Unique Solution!  
\[ h_w(x) = w_1 x + w_0 \]
\[ w_1 = \frac{N \sum (x_j y_i) - (\sum x_j)(\sum y_i)}{N \sum (x_j^2) - (\sum x_j)^2} \]
\[ w_0 = \frac{(\sum y_i) - w_1 (\sum x_j)}{N} \]

Could also Solve Iteratively  
\[ \text{Argmin}_w \text{Loss}(h_w) \]
\[ w = \text{any point in weight space} \]
Loop until convergence  
For each \( w_i \) in \( w \) do  
\[ w_i := w_i - \alpha \frac{\partial}{\partial w_i} \text{Loss}(w) \]

Multivariate Linear Regression  
\[ h_w(x_j) = w_0 + \sum w_i x_{j,i} = \sum w_i x_{j,i} = w^T x_j \]
\[ \text{Argmin}_w \text{Loss}(h_w) \]
Unique Solution = \( (x^T x)^{-1} x^T y \)

Problem…. Overfitting: Possible that some dimension that is actually irrelevant appears by chance to be useful.

Overfitting  
Regularize!!  
Penalize high weights (complex hypothesis)  
Minimize cost: Loss + Complexity  
\[ \text{Cost}(h_w) = \sum_j (y_j - \sum_i w_i x_{j,i})^2 + \lambda \sum_i |w_i|^p \]
\( p=1 \): L1 regularization (Lasso)  
\( p=2 \): L2 regularization

Regularization  
\[ w_1 \]
\[ w_2 \]
\[ \text{L1} \]
\[ \text{L2} \]

Back to Classification  
\[ P(\text{edible}|X) = 1 \]
\[ P(\text{edible}|X) = 0 \]
Decision Boundary
Logistic Regression

- Learn $P(Y|X)$ directly!
  - Assume a particular functional form
    - Not differentiable...

- Logistic Function
  - Aka Sigmoid

Logistic Function in n Dimensions

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i} w_i x_i)}$$

Features can be discrete or continuous!

Very convenient!

$$P(Y = 1|X = <X_1,...X_n>) = \frac{1}{1 + \exp(w_0 + \sum_{i} w_i x_i)}$$

P(Y = 0|X = <X_1,...X_n>) = \frac{\exp(w_0 + \sum_{i} w_i x_i)}{1 + \exp(w_0 + \sum_{i} w_i x_i)}

implies

$$P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i} w_i x_i)}{P(Y = 1|X)}$$

implies

$$\ln \frac{P(Y = 0|X)}{P(Y = 1|X)} = w_0 + \sum_{i} w_i x_i$$

Linear classification rule!

Y=0 if the RHS>0

Logistic Regression

- Learn $P(Y|X)$ directly!
  - Assume a particular functional form
  - Logistic Function
    - Aka Sigmoid

Understanding Sigmoid

$$g(w_0 + \sum_{i} w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_{i} w_i x_i}}$$

$w_0 = -2$, $w_1 = -1$

$w_0 = 0$, $w_1 = -0.5$

Likelihood vs. Conditional Likelihood

Generative (Naive Bayes) maximizes Data likelihood

$$\ln P(D|w) = \sum_{j=1}^{N} \ln P(x_j^j, y_j^j|w)$$

$$= \sum_{j=1}^{N} \ln P(x_j^j|w) + \sum_{j=1}^{N} \ln P(y_j^j|w)$$

Discriminative (Logistic Regr.) maximizes Conditional Data Likelihood

$$\ln P(D_Y|D_X, w) = \sum_{j=1}^{N} \ln P(y_j^j|X_j^j, w)$$

Classification models can’t compute $P(y|w)$!

Or... “They don’t waste effort learning $P(X)$”

Focus only on $P(Y|X)$ - all that matters for classification
Maximizing Conditional Log Likelihood

\[ l(w) = \ln P(Y = y | x, w) = \ln \prod_j P(y_j | x^j, w) \]

\[ = \sum_j y_j w_0 + \sum_i w_i x_i^j - \ln (1 + \exp (w_0 + \sum_i w_i x_i^j)) \]

Bad news: no closed-form solution to maximize \( l(w) \)
Good news: \( l(w) \) is concave function of \( w \)!
No local minima
Concave functions easy to optimize

Maximizing Conditional Log Likelihood: Gradient ascent

\[ l(w) = \ln \prod_j P(y_j | x^j, w) \]
\[ = \sum_j y_j w_0 + \sum_i w_i x_i^j - \ln (1 + \exp (w_0 + \sum_i w_i x_i^j)) \]
\[ \frac{\partial l(w)}{\partial w_i} = \sum_j \left[ y_j x_i^j - \frac{\exp (w_0 + \sum_i w_i x_i^j)}{1 + \exp (w_0 + \sum_i w_i x_i^j)} \right] \]
\[ = \sum_j \left[ y_j x_i^j - \frac{\exp (w_0 + \sum_i w_i x_i^j)}{1 + \exp (w_0 + \sum_i w_i x_i^j)} \right] \]
\[ = \sum_j \left[ y_j - \frac{\exp (w_0 + \sum_i w_i x_i^j)}{1 + \exp (w_0 + \sum_i w_i x_i^j)} \right] \]

\[ \frac{\partial l(w)}{\partial w_0} = \sum_j \left[ y_j - P(Y_j = 1 | x^j, w) \right] \]

Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave!

Gradient:
\[ \nabla_w l(w) = \frac{\partial l(w)}{\partial w_0}, \ldots, \frac{\partial l(w)}{\partial w_i} \]

Update rule:
\[ \Delta w = \eta \nabla_w l(w) \]
\[ w_i^{(t+1)} = w_i^{(t)} + \eta \frac{\partial l(w)}{\partial w_i} \]

- Gradient ascent is simplest of optimization approaches

Gradient Ascent for LR

Gradient ascent algorithm: (learning rate \( \eta > 0 \))
do:
\[ w_0^{(t+1)} = w_0^{(t)} + \eta \sum_j [y_j - P(Y_j = 1 | x^j, w)] \]
For \( t=1 \) to \( n \) (iterate over weights)
\[ w_i^{(t+1)} = w_i^{(t)} + \eta \sum_j [y_j - P(Y_j = 1 | x^j, w)] \]
until "change" < \( \varepsilon \)

That’s all MCLE. How about MCAP?

\[ p(w | Y, X) \propto P(Y | X, w) p(w) \]

- One common approach is to define priors on \( w \)
  - Normal distribution, zero mean, identity covariance
  - "Pushes" parameters towards zero

- Regularization
  - Helps avoid very large weights and overfitting

- MAP estimate:
\[ w^* = \arg \max_w \prod_{i=1}^N p(y_i | x_i^j, w) \]

\[ p(w) = \prod_i \frac{1}{\sqrt{2\pi} \sigma^2} e^{-\frac{w_i^2}{2\sigma^2}} \]
M_CAP as Regularization

\[ w^* = \arg \max_w \ln \left( \prod_{i=1}^{N} P(y^i | x^i, w) \right) - \frac{1}{2} \sum_{j=1}^{k} w_j^2 \]

- Add log \( p(w) \) to objective:
  \[ \ln p(w) \propto -\frac{1}{2} \sum_{j=1}^{k} w_j^2 \]
  - Quadratic penalty: drives weights towards zero
  - Adds a negative linear term to the gradients

Penalizes high weights, like we did in linear regression

Naïve Bayes vs. Logistic Regression

<table>
<thead>
<tr>
<th>Generative</th>
<th>Discriminative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assume functional form for</td>
<td>Assume functional form for</td>
</tr>
<tr>
<td>( P(X</td>
<td>Y) \</td>
</tr>
<tr>
<td>( Y )</td>
<td>( Y )</td>
</tr>
<tr>
<td>Gaussian NB for cont features</td>
<td>Est params from training data</td>
</tr>
<tr>
<td>Bayes rule to calc. ( P(Y</td>
<td>X=x) )</td>
</tr>
<tr>
<td>( P(Y</td>
<td>X=x) \neq P(X</td>
</tr>
<tr>
<td>Indirect computation</td>
<td>Can’t generate data sample</td>
</tr>
</tbody>
</table>

Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

- Generative vs. Discriminative classifiers
- Asymptotic comparison
  (# training examples \( \rightarrow \) infinity)
  - when model correct
    - GNB (with class independent variances) and LR produce identical classifiers
  - when model incorrect
    - LR is less biased – does not assume conditional independence
      - therefore LR expected to outperform GNB

Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

- Generative vs. Discriminative classifiers
- Non-asymptotic analysis
  - convergence rate of parameter estimates, \( n = \# \) of attributes in \( X \)
  - Size of training data to get close to infinite data solution
  - Naïve Bayes needs \( O(\log n) \) samples
  - Logistic Regression needs \( O(n) \) samples

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

MCLE vs. MCAP

- Maximum conditional likelihood estimate
  \[ w^* = \arg \max_w \ln \left( \prod_{i=1}^{N} P(y^i | x^i, w) \right) \]
  \[ = \left( w^{(t)} + \eta \right) - \lambda w^{(t)} + \sum_{j=1}^{k} x_j^i [y^j - P(Y^j = 1 | x^i, w)] \]

- Maximum conditional a posteriori estimate
  \[ w^* = \arg \max_w \ln \left( \prod_{i=1}^{N} P(y^i | x^i, w) \right) \]
  \[ = \left( w^{(t+1)} + \eta \right) - \lambda w^{(t+1)} + \sum_{j=1}^{k} x_j^i [y^j - P(Y^j = 1 | x^i, w)] \]
What you should know about Logistic Regression (LR)

- Gaussian Naive Bayes with class-independent variances representationally equivalent to LR
  - Solution differs because of objective (loss) function
- In general, NB and LR make different assumptions
  - NB: Features independent given class \( P(X|Y) \)
  - LR: Functional form of \( P(Y|X) \), no assumption on \( P(X|Y) \)
- LR is a linear classifier
  - Decision rule is a hyperplane
- LR optimized by conditional likelihood
  - No closed-form solution
  - Concave \! global optimum with gradient ascent
  - Maximum conditional a posteriori corresponds to regularization
- Convergence rates
  - GNB (usually) needs less data
  - LR (usually) gets to better solutions in the limit

LR: MCAP Algorithm

- Given Data matrix of size \( m \times (n+2) \) (\( n \) attributes and \( m \) examples)
  - \( Data[i][n+1] \) gives the class for example \( i \)
  - \( Data[i][0] \) is the dummy threshold attribute always set to 1.
- Arrays \( Pr[0..m-1] \) and \( w[0..n] \) initialized to random values
- Until convergence do
  - For each example \( i \)
    - Compute \( Pr[i] = Pr(class=1|Data[i],w) \)
    - Array \( dw[0..n] \) initialized to zero
  - For \( j=0 \) to \( m-1 \) // Go over all the training examples
    - \( dw[i] = dw[i] + Data[i][n+1] * (Data[i][j] - Pr[i]) \)
  - For \( i=0 \) to \( n \)
    - \( w[i] = w[i] + \eta (dw[i] - \lambda w[i]) \)