Logistic regression

\[ Y \in \mathbb{S}_0(1, \epsilon), \quad X \in \mathbb{R}^d \]

Then \( p(y=1 \mid x) = \sigma(w^T x) \)

\[ \sigma(z) = \frac{1}{1 + \exp(-z)} \]

Logistic regression model

\[ p(y=1 \mid x) = \sigma(w^T x) \]
\[ p(y=0 \mid x) = 1 - \sigma(w^T x) = \sigma(-w^T x) \]

\[ p(y=x) = \sigma(w^T x)^y \sigma(-w^T x)^{1-y} \]

\( Y \mid X \) is a Bernoulli \( \sigma(w^T x) \)

\[ p(y) = \sigma(w^T x)^y \sigma(-w^T x)^{1-y} \]

Given data \( (x_i, y_i)_{i=1}^n \)

Maximum conditional likelihood:

\[ L(w) = \sum_{i=1}^n \log p(y_i \mid x_i; w) = \sum_{i=1}^n \left[ y_i \log \sigma(w^T x_i) + (1-y_i) \log \sigma(-w^T x_i) \right] \]

\( \sigma'(z) = \sigma(z) \sigma(-z) \)

\[ \nabla \sigma(w^T x_i) = x_i \left[ \sigma(w^T x_i) \sigma(-w^T x_i) \right] \quad \forall i \in \{1, 2, \ldots, n\} \]

\[ \nabla L(w) = \sum_{i=1}^n x_i \left[ y_i \frac{\sigma(w^T x_i) \sigma(-w^T x_i)}{\sigma(w^T x_i) + \sigma(-w^T x_i)} - (1-y_i) \frac{\sigma(w^T x_i) \sigma(-w^T x_i)}{\sigma(w^T x_i) + \sigma(-w^T x_i)} \right] \]

\[ \nabla L(w) = \sum_{i=1}^n x_i \left[ y_i \left[ \frac{\sigma(w^T x_i) \sigma(-w^T x_i)}{\sigma(w^T x_i) + \sigma(-w^T x_i)} \right] - \sigma(w^T x_i) \right] \]

\[ \nabla L(w) = 0 \iff \text{need to solve transcendental eq.} \]

\[ \text{because} \quad \frac{1}{\text{exp}(w^T x)} \left( \ldots \right) = 0 \]

Contrast with least squares when get linear in \( w \)

\[ \nabla L(w) = 0 \implies \text{need to solve transcendental eq.} \]

\[ \text{because} \quad \frac{1}{\text{exp}(w^T x)} \left( \ldots \right) = 0 \]
Brief recap on optimization:

want to minimize \( \min_{w \in \mathbb{R}^d} f(w) \)

1) Gradient descent:
\( w_{t+1} = w_t - \gamma_t \nabla f(w_t) \)

Step-size rules:
- Constant step-size: \( \gamma_t = \frac{1}{L} \)
  \( L \) is the Lipschitz constant of \( \nabla f \)
- Decreasing step-size rule: \( \gamma_t = \frac{1}{t} \)
  usually want \( \sum \frac{1}{t} = \infty \), \( \sum \frac{1}{t^2} < \infty \)
  choose \( \gamma_t \) by line-search; \( \min_{\gamma \in \mathbb{R}} f(w_t + \gamma \delta_t) \)
- Costly in general → cannot avoid approximate search
  e.g. Armijo line search

2) Newton's method (2nd order method)

Motivation: minimize quadratic approximation:
\[ f(w) = f(w_0) + \nabla f(w_0)^T (w - w_0) + \frac{1}{2} (w - w_0)^T H(w_0) (w - w_0) \]

want
\[ \nabla Q(w) = 0 \]
\[ Q(w) = \frac{1}{2} (w - w_0)^T H(w_0) (w - w_0) \]

Thus this is
\[ w_{t+1} = w_t - H(w_t)^{-1} \nabla f(w_t) \]
Newton's update

Damped Newton:

You add a step-size
\[ w_{t+1} = w_t - \gamma_t H(w_t)^{-1} \nabla f(w_t) \]

Why Newton's: much faster convergence in # of iterations vs gradient.
Newton for logistic regression: IRLS:

Recall: \( \nabla L(w) = \sum_{i=1}^{n} x_i [y_i - \sigma(w^T x_i)] \)

\( H(L(w)) = -\sum_{i=1}^{n} x_i x_i^T \sigma'(w^T x_i) \sigma(-w^T x_i) \)

\( v^T H v = -\sum_{i=1}^{n} (v^T x_i)^2 \sigma'(\theta) \sigma(-\theta) \)

\( v^T H v \leq 0 \forall v \)

ie. \( H \preceq 0 \)

Let \( \mu_i \triangleq \sigma(w^T x_i) \) (\( \in [0,1] \))

Let's look at minimizing the negative log-likelihood

then Hessian = \( \sum_{i=1}^{n} x_i x_i^T [\mu_i (1-\mu_i)] \)

Gradient = \( \sum_{i=1}^{n} x_i [\mu_i - y_i] = x^T (\mu - y) \)

\( = x^T (D x) \) where \( D_{ij} = \mu_i (1-\mu_i) \) depends on \( \omega \)

Newton's update: \( \omega_{t+1} = \omega_t - (x^T x)^{-1} x^T (\mu - y) \)

\( = (x^T x)^{-1} [x^T D x \omega_t + x^T (y - \mu)] \)

\( = (x^T x)^{-1} [x^T D z_t] \)

\( z_t \triangleq x \omega_t + D^T (y - \mu) \)

\( (z_t) \triangleq \frac{z_t - x \omega_t}{\mu_t (1-\mu_t)} \)

This is solution to weighted least squares problem

\( \min_w \| D^T (z - x \omega) \|^2 \)

Compare with Gaussian rate model \( \leq \frac{1}{2} \sum_{i=1}^{n} (y_i - z_i)^2 \quad \text{for least squares} \)

Note

Newton's method for logistic regression = iteratively reweighted least squares
Newton's method for logistic regression = iterative reweighted least squares

log data logistic regression

cannot do $O(d^2)$, $O(d^3)$ operations

=> first order methods

if $n$ is huge, you cannot use batch method

instead you can use SGD: stochastic gradient descent

$$\nabla f \approx \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\theta_n)$$

SGD: $\theta_{t+1} = \theta_t - \eta \nabla f_i(\theta_n)$  
(much cheaper update but slow convergence)

batch gradient $\theta_{t+1} = \theta_t - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\theta_n)$  
(expensive update; fast convergence)

SA (Stochastic average gradient):

log error of pluggable
time

Kernels

recall for least square, gradient

log regression

$k_x$ - regression

you always have $W_t = \sum_{i=1}^{n} x_i^T \alpha_i$ 

log least square:

$W_t = (X^T X + \lambda I)^{-1} Y$ 

Gram matrix $K$

$k_{ij} = \alpha_i^T x_j$

for prediction:

$W x = \sum_{i=1}^{n} \alpha_i (x_i^T x) $

Suppose map $x \rightarrow \phi(x)$

$W^T q(x) = \sum_{i=1}^{n} \alpha_i \phi(x_i)^T \phi(x)$

"Kernel trick": you can work implicitly in high dimensional space $\phi(x)$, only using $K(\cdot, \cdot)$ kernel

$\phi(x) \rightarrow y \rightarrow (x,y) \rightarrow (x^T \phi(y), y)$
\[ \langle \hat{z}' \rangle = (x' y')^2 + 2(x x')(y y') + y^2 y'^2 \]
\[ = (x x' + y y')^2 \]
\[ = (\langle z, z' \rangle)^2 \]