9.1 Binary Classification

Final Tip:

- First Question: Similar to homework
- Second Question: Basic idea, gradient calculation, maximum likelihood calculation

9.1.1 Logistic Function / Sigmoid Function

Given:

- Two classes $C = \{c_1, c_2\}$
- A particular vector $\mathbf{x} \in \mathbb{R}^d$

  Do
- Assign the vector $\mathbf{x}$ to one of the two classes.
  - Namely, to calculate the probability of $\mathbf{x} \in c_i$ for all $c_i \in C$.

From the Bayesian Rule, we derive that

$$
\begin{aligned}
P(c_1|\mathbf{x}) &= \frac{P(\mathbf{x}|c_1)P(c_1)}{P(\mathbf{x})} \\
&= \frac{P(\mathbf{x}|c_1)P(c_1)}{P(\mathbf{x}|c_1)P(c_1) + P(\mathbf{x}|c_2)P(c_2)} \\
&= \frac{1}{1 + \frac{P(\mathbf{x}|c_2)P(c_2)}{P(\mathbf{x}|c_1)P(c_1)}} \\
&= \frac{1}{1 + e^{-\ln\left[\frac{P(\mathbf{x}|c_1)}{P(\mathbf{x}|c_2)}\right] - \ln\left[\frac{P(c_1)}{P(c_2)}\right]}}
\end{aligned}
$$

which can be written in the form of a Logistic function / Sigmoid Function:

$$
P(c_1|\mathbf{x}) = \frac{1}{1 + e^{-\xi}}
$$

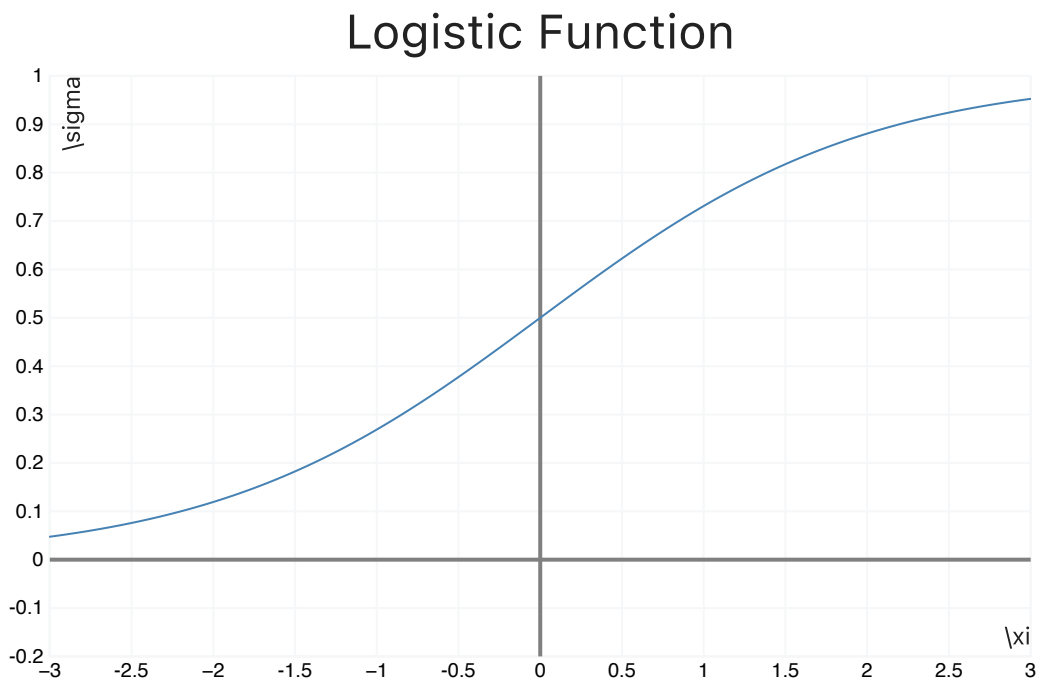
where,

$$
\xi = \ln\left[\frac{P(\mathbf{x}|c_1)}{P(\mathbf{x}|c_2)}\right] + \ln\left[\frac{P(c_1)}{P(c_2)}\right]
$$

- Likelihood Ratio: $\ln\left[\frac{P(\mathbf{x}|c_1)}{P(\mathbf{x}|c_2)}\right]$

- Prior Ratio: $\ln\left[\frac{P(c_1)}{P(c_2)}\right]$

Properties of Logistic Functions



Limits

$$\lim_{\xi \to -\infty} \sigma(\xi) = 0$$

$$\lim_{\xi \to \infty} \sigma(\xi) = 1$$

Central Symmetry

$$\sigma(-\xi) = 1 - \sigma(\xi)$$

Derivative

$$\begin{aligned} \frac{d}{d\xi}\sigma(\xi) &= \sigma(\xi)(-\xi) \\ &= \sigma(\xi)\Big(1 - \sigma(\xi)\Big) \end{aligned}$$

Proof.

$$
\begin{aligned}
\frac{d}{d\xi}\sigma(\xi) &= \frac{d}{d\xi}\frac{1}{1+e^{-\xi}} \\
&= (\frac{d}{d(1+e^{-\xi})} \cdot \frac{1}{1+e^{-\xi}}) \cdot \frac{d(1+e^{-\xi})}{d\xi} \\
&= \frac{-1}{(1+e^{-\xi})^2} \cdot (-e^{-\xi}) \\
&= \frac{e^{-\xi}}{(1+e^{-\xi})^2} \\
&= \frac{e^{-\xi}}{(1+e^{-\xi})} \cdot \frac{1}{(1+e^{-\xi})} \\
&= \frac{1}{1+e^{\xi}} \cdot \frac{1}{1+e^{-\xi}} \\
&= \sigma(-\xi) \cdot \sigma(\xi)
\end{aligned}
$$

9.1.2 Multivariate Gaussian Model for $P(\mathbf{x}|c_i)$

Assumed that:

- within each class
- the multi-variate input vector $\mathbf{x}$ follows a Gaussian Distribution with a *common* covariate Σ.

  $$
  P(\mathbf{x}|c_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma^{-1}(\mathbf{x}-\mu_i)}
  $$

  From which we derive that

$$\ln\Big(\frac{P(\mathbf{x}|c_1)}{P(\mathbf{x}|c_2)}\Big) = \ln\Big[\frac{e^{-\frac{1}{2}(\mathbf{x}-\mu_1)^\top\Sigma^{-1}(\mathbf{x}-\mu_1)}}{e^{-\frac{1}{2}(\mathbf{x}-\mu_2)^\top\Sigma^{-1}(\mathbf{x}-\mu_2)}}\Big]$$

$$= \frac{1}{2}(\mathbf{x}-\mu_2)^\top\Sigma^{-1}(\mathbf{x}-\mu_2) - \frac{1}{2}(\mathbf{x}-\mu_1)^\top\Sigma^{-1}(\mathbf{x}-\mu_1)$$

$$= \frac{1}{2}(\mathbf{x}^\top\Sigma^{-1} - \mu_2^\top\Sigma^{-1})(\mathbf{x}-\mu_2) - \frac{1}{2}(\mathbf{x}^\top\Sigma^{-1} - \mu_1^\top\Sigma^{-1})(\mathbf{x}-\mu_1)$$

$$= \frac{1}{2}(\mathbf{x}^\top\Sigma^{-1}\mathbf{x} - \mathbf{x}^\top\Sigma^{-1}\mu_2 - \mu_2^\top\Sigma^{-1}\mathbf{x} + \mu_2^\top\Sigma^{-1}\mu_2)$$
$$- \frac{1}{2}(\mathbf{x}^\top\Sigma^{-1}\mathbf{x} - \mathbf{x}^\top\Sigma^{-1}\mu_1 - \mu_1^\top\Sigma^{-1}\mathbf{x} + \mu_1^\top\Sigma^{-1}\mu_1)$$

$$= \frac{1}{2}\Big[\mathbf{x}^\top\Sigma^{-1}(\mu_1-\mu_2) + (\mu_1^\top-\mu_2^\top)\Sigma^{-1}\mathbf{x} + (\mu_2^\top+\mu_1^\top)\Sigma^{-1}(\mu_2-\mu_1)\Big]$$

$$= (\mu_1-\mu_2)^\top\Sigma^{-1}\mathbf{x} + \frac{1}{2}(\mu_2+\mu_1)^\top\Sigma^{-1}(\mu_2-\mu_1)$$

★ Therefore, the exponential ξ can be rewritten as:

$$\xi = \ln\Big(\frac{P(\mathbf{x}|c_1)}{P(\mathbf{x}|c_2)}\Big) + \ln\Big(\frac{P(c_1)}{P(c_2)}\Big)$$

$$= (\mu_1-\mu_2)^\top\Sigma^{-1}\mathbf{x} + \frac{1}{2}(\mu_2+\mu_1)^\top\Sigma^{-1}(\mu_2-\mu_1) + \ln\Big(\frac{P(c_1)}{P(c_2)}\Big)$$

$$= \Big[\Sigma^{-1}(\mu_1-\mu_2)\Big]^\top\mathbf{x} + \Big[\frac{1}{2}(\mu_2+\mu_1)^\top\Sigma^{-1}(\mu_2-\mu_1) + \ln\Big(\frac{P(c_1)}{P(c_2)}\Big)\Big]$$

$$= \mathbf{w}^\top\mathbf{x} + b$$

In conclusion,

$$P(c_1|\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}^\top\mathbf{x}+b)}}$$

where

- $\mathbf{w} = \Sigma^{-1}(\mu_1-\mu_2)$, and
- $b = \frac{1}{2}(\mu_2+\mu_1)^\top\Sigma^{-1}(\mu_2-\mu_1) + \ln\Big[\frac{P(c_1)}{P(c_2)}\Big]$

9.1.3 Maximum Likelihood Formulation

Recall: Bernoulli Distribution

Suppose that a variable Y confronts a Bernoulli Distribution,

- i.e., $Y \sim \text{Bernoulli}(p)$
- where p is the probability of being Success,
  the probabilistic distribution function is

$$P(Y = y) = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{if } y = 0 \end{cases}$$

Convert σ to Probability

We want to predict a binary output $y_n \in \{0, 1\}$ from an input $\mathbf{x}_n$. From above, we know that the logistic regression has the form:

$$y_n = \sigma(\mathbf{w}^\top \mathbf{x}_n) + \epsilon_n$$

where

$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}} = \frac{e^{\xi}}{1 + e^{\xi}}$$

We model input-output by a conditional Bernoulli Distribution:

$$P(y_n = y | \mathbf{x}_n) = \begin{cases} \sigma(\mathbf{w}^\top \mathbf{x}_n) & \text{if } y = 1 \\ 1 - \sigma(\mathbf{w}^\top \mathbf{x}_n) & \text{if } y = 0 \end{cases}$$

Bernoulli Distribution Modelling

Given $\{(\mathbf{x}_n, y_n) | n = 1, \ldots, N\}$, the likelihood is given by

$$\begin{aligned} P(\mathbf{y} | \mathbf{X}, \mathbf{w}) &= \prod_{n=1}^{N} P(y_n | \mathbf{x}_n) \\ &= \prod_{n=1}^{N} P\Big(y_n = 1 | \mathbf{x}_n\Big)^{y_n} \cdot \Big(1 - P(y_n = 1 | \mathbf{x}_n)\Big)^{1 - y_n} \\ &= \prod_{n=1}^{N} \sigma(\mathbf{w}^\top \mathbf{x}_n)^{y_n} \Big(1 - \sigma(\mathbf{w}^\top \mathbf{x}_n)\Big)^{1 - y_n} \end{aligned}$$

The log-likelihood function is thus given by:

$$\mathcal{L}(\mathbf{y}|\mathbf{X},\mathbf{w}) = \log \sum_{n=1}^{N} P(y_n|\mathbf{x}_n)$$

$$= \sum_{n=1}^{N} \Big(y_n \cdot \log\Big[\sigma(\mathbf{w}^\top \mathbf{x}_n)\Big] + (1-y_n) \cdot \log\Big[1-\sigma(\mathbf{w}^\top \mathbf{x})\Big] \Big)$$

We want to maximize this log-likelihood $\mathcal{L}$. However, $\mathcal{L}$ is not a polynomial, the calculation of the maximum of nonlinear function of $\mathbf{w}$ cannot be done in a closed form. That is, it is very costly to directly compute:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$$

Therefore, an iterated re-weighted least squares (IRLS) is then performed, derived from the Newton's method.

9.2 Newton's Method

9.2.1 Theoretical Foundations

Key Point 1. Gradient 梯度: Step Direction

Consider a real-valued function $f(\mathbf{x})$, which takes a real-valued vector $\mathbf{x} \in \mathbb{R}^d$ as an input:

$$f(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}$$

The gradient of $f(\mathbf{x})$ is defined by:

$$\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix}$$

Which is the partial derivative of $f(\mathbf{x})$ with respect to all the dimensions of $\mathbf{x}$.

Key Point 2. Hessian Matrix 海森矩阵: Step Size

If $f(\mathbf{x})$ belongs to the class C^2, the Hessian matrix $\mathbf{H}$ is defined as the symmetric matrix with the combination of any two dimensions.

$$
\begin{aligned}
\mathbf{H} &= \nabla^2 f(\mathbf{x}) \\
&= \frac{\partial}{\partial \mathbf{x}} \Big[\nabla f(\mathbf{x})\Big]^\top \\
&= \frac{\partial}{\partial \mathbf{x}} \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} & \frac{\partial f(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f(\mathbf{x})}{\partial x_d} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix}
\end{aligned}
$$

> *The Hessian matrix can not only help us find the extreme points of the function (through the first-order derivative is 0), but also determine whether the point is a minimum, maximum or saddle point by analyzing the curvature of the function near the extreme point. For example, if the Hessian matrix is positive definite, it means that the extreme point is a local minimum.*

Key Point 3. Gradient Descent/Ascent 梯度下降、上升

The gradient descent/ascent learning is a simple first-order iterative method for minimization/maximization.

Gradient Descent: Iterative Minimization

$$
\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta\Big(\frac{\partial \mathcal{J}}{\partial \mathbf{w}}\Big)
$$

Gradient Ascent: Iterative Maximization

$$
\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta\Big(\frac{\partial \mathcal{J}}{\partial \mathbf{w}}\Big)
$$

where the learning rate $\eta > 0$.

The gradient $\frac{\partial \mathcal{J}}{\partial \mathbf{w}}$ gives the direction of the movement of $\mathbf{w}$. The learning rate η gives the step size.

9.2.2 Newton's Method

The Basic idea of Newton's method is to optimize the quadratic (二次的) approximation,

- of the objective function $\mathcal{J}(\mathbf{w})$,

- around the current point $\mathbf{w}^{(k)}$.

The Newton's Method tells us that, for an arbitrary function $\mathcal{J}: \mathbb{R}^D \mapsto \mathbb{R}$, we can iteratively approximate $\frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}} = 0$ by:

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \left[\nabla^2 \mathcal{J}(\mathbf{w}^{\text{old}})\right] \cdot \nabla \mathcal{J}(\mathbf{w}^{\text{old}})$$

Key Point 1. Tayler Series of $\mathcal{J}(\mathbf{w})$

ℹ The objective of using Taylor Series of $\mathcal{J}(\mathbf{w})$ is to:

- Locally approximate the polynomial with Tayler Series at a specific step $\mathbf{w}^{(k)}$.
  The Taylor Series of a function $f(x): \mathbb{R} \mapsto \mathbb{R}$ around a point a yields:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

$$= \sum_{i=0}^{\infty} \frac{f^{(i)}(a)}{i!}(x-a)^i$$

  Now we consider an abstract function $\mathcal{J}(\mathbf{w})$.

$$\mathcal{J}: \mathbb{R}^D \mapsto \mathbb{R}$$

  The second-order Taylor series expansion of $\mathcal{J}(\mathbf{w})$ at the current $\mathbf{w}^{(k)}$ gives:

$$\mathcal{J}_2(\mathbf{w}) = \mathcal{J}(\mathbf{w}^{(k)}) + \left[\nabla \mathcal{J}(\mathbf{w}^{(k)})\right]^\top (\mathbf{w} - \mathbf{w}^{(k)}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^{(k)})^\top \left[\nabla^2 \mathcal{J}(\mathbf{w}^{(k)})\right](\mathbf{w} - \mathbf{w}^{(k)})$$

Specifically:

- First term: $\mathcal{J}(\mathbf{w}^{(k)})$, is the value of the target function $\mathcal{J}$ with respect to the current $\mathbf{w}^{(k)}$.
- Second term: $\nabla \mathcal{J}(\mathbf{w}^{(k)})$ is the *gradient* of the target function at the current $\mathbf{w}^{(k)}$.
- Third Term: $\nabla^2 \mathcal{J}(\mathbf{w}^{(k)})$ is the *Hessian Matrix* describing the target function's 2nd order derivative at current $\mathbf{w}^{(k)}$.

Key Point 2. Differentiation of 2^{nd} order Taylor Series

Differentiate the above second-order Taylor Series with respect to $\mathbf{w}$, we get the approximation of the gradient function at the point $\mathbf{w}$:

$$\nabla \mathcal{J}(\mathbf{w}) = 0 + \nabla \mathcal{J}(\mathbf{w}^{(k)}) + \nabla^2 \mathcal{J}(\mathbf{w}^{(k)})(\mathbf{w} - \mathbf{w}^{(k)})$$

Remember that $\nabla \mathcal{J}(\mathbf{w}) \in \mathbb{R}^d$. Set the gradient equal to $\mathbb{0}^d$:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = 0$$

$$\implies \quad 0 = \nabla \mathcal{J}(\mathbf{w}^{(k)}) + \nabla^2 \mathcal{J}(\mathbf{w}^{(k)}) \cdot (\mathbf{w} - \mathbf{w}^{(k)})$$

$$\implies \quad 0 = \nabla \mathcal{J}(\mathbf{w}^{(k)}) + \nabla^2 \mathcal{J}(\mathbf{w}^{(k)}) \cdot \mathbf{w} - \nabla^2 \mathcal{J}(\mathbf{w}^{(k)}) \cdot \mathbf{w}^{(k)}$$

$$\implies \quad \nabla^2 \mathcal{J}(\mathbf{w}^{(k)})\mathbf{w} = \nabla^2 \mathcal{J}(\mathbf{w}^{(k)}) \cdot \mathbf{w}^{(k)} - \nabla \mathcal{J}(\mathbf{w})$$

$$\implies \quad \mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \left[\nabla^2 \mathcal{J}(\mathbf{w}^{(k)})\right]^{-1} \cdot \nabla \mathcal{J}(\mathbf{w}^{(k)})$$

> *Note that the learning rate here is the inverse Hessian matrix.*

At this point, we conclude that, for an arbitrary function $\mathcal{J}$ that maps a d-dimensional vector to a scalar.

We can compute the *optimal vector* that produces the *optimum scalar* by:

1. Calculate its gradient at the current step.
2. Calculate its Hessian at current step.
3. Apply Newton's method to get the vector of the next step.

9.3 Logistic Regression Algorithms

- Remark: We have already computed the log-likelihood $\mathcal{L}(\mathbf{y}|\mathbf{X}, \mathbf{w})$ under the given data set $\mathbf{X}, \mathbf{y}$.

Here, $\mathcal{L}$ is an example of the abstract function $\mathcal{J}$ since:

$$\mathcal{L} : \mathbb{R}^d(\text{weight } \mathbf{w}) \mapsto \mathbb{R}(\text{Probability})$$

- We need to find an optimal vector that produces the maximum scalar.
- But calculating $\dfrac{\partial \mathcal{L}}{\partial w}$ is very costly.
- Therefore, we could have a work-around using Newton's method.

The gradient Ascent Learning has the form

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \eta\left(\frac{\partial \mathcal{L}}{\partial \mathbf{w}}\right)$$

9.3.1 Calculate Gradient

Recall the log-likelihood:

$$\mathcal{L} = \sum_{n=1}^{N} \Big[y_n \log \sigma(\mathbf{w}^\top \mathbf{x}_n) + (1 - y_n) \log \Big(1 - \sigma(\mathbf{w}^\top \mathbf{x}_n) \Big) \Big]$$

Take its first order derivative as the gradient:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \sum_{n=1}^{N} \Big[y_n \frac{\sigma_n'}{\sigma_n} \mathbf{x}_n + (1 - y_n) \frac{-\sigma_n'}{(1 - \sigma_n)} \mathbf{x}_n \Big]$$

Note that using chain rule:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} \log \sigma(\mathbf{w}^\top \mathbf{x}_n) &= \frac{\partial \mathcal{L}}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial (\mathbf{w}^\top \mathbf{x}_n)} \cdot \frac{\partial \mathbf{w}^\top \mathbf{x}_n}{\partial \mathbf{w}} \\ &= \frac{1}{\sigma(\mathbf{w}^\top \mathbf{x}_n)} \cdot \sigma'(\mathbf{w}^\top \mathbf{x}_n) \cdot \mathbf{x}_n \end{aligned}$$

By using the 2nd and 3rd property of the logistic function σ, we obtain:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= \sum_{n=1}^{N} \Big[y_n \frac{\sigma_n(1 - \sigma_n)}{\sigma_n} \mathbf{x}_n + (1 - y_n) \frac{-\sigma_n(1 - \sigma_n)}{1 - \sigma_n} \mathbf{x}_n \Big] \\ &= \sum_{n=1}^{N} \Big(y_n (1 - \sigma_n) \mathbf{x}_n - (1 - y_n) \sigma_n \mathbf{x}_n \Big) \\ &= \sum_{n=1}^{N} \Big[y_n (1 - \sigma_n) - (1 - y_n) \sigma_n \Big] \mathbf{x}_n \\ &= \sum_{n=1}^{N} \Big(y_n - y_n \sigma_n - \sigma_n + y_n \sigma_n \Big) \mathbf{x}_n \\ &= \sum_{n=1}^{N} (y_n - \sigma_n) \mathbf{x}_n \in \mathbb{R}^D \end{aligned}$$

Lastly, it could be concluded that:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \sum_{n=1}^{N} \Big(y_n - \sigma(\mathbf{w}^\top \mathbf{x}_n) \Big) \mathbf{x}_n$$

As discussed before, it is a vector with the same shape of $\mathbf{x}_n$.

★ By now we know that we could update $\mathbf{w}$ by:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta \cdot \sum_{n=1}^{N} \Big[y_n - \sigma\Big(\mathbf{w^{(k)}}^\top \mathbf{x_n} \Big) \Big] \mathbf{x}_n$$

> *记到这里就行了，学习率老师会给所以海森矩阵不用手算。*

Example: Gradient Calculation

Given:

- Historical data:
  - $x_1 = \begin{pmatrix}2\\1\end{pmatrix}, x_2 = \begin{pmatrix}1\\2\end{pmatrix}, x_3 = \begin{pmatrix}3\\3\end{pmatrix}$
  - $y_1 = 1, y_2 = 0, y_3 = 0$
- Probability:
  - $P(\mathbf{x}|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$.
- Initial weight:
  - $\mathbf{w}_0 = \begin{bmatrix}0\\0\end{bmatrix}$.
- Learning rate:
  - $\eta = 0.1$

Do:

- Update weight for one step.

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta \cdot \sum_n (y_n - \sigma(\mathbf{w}^\top \mathbf{x}_n))\mathbf{x}_n$$

$$\mathbf{w}_1 = \begin{bmatrix}0\\0\end{bmatrix} + 0.1 \cdot \left[(1-0.5)\begin{pmatrix}2\\1\end{pmatrix} + (0-0.5)\begin{pmatrix}1\\2\end{pmatrix} + (0-0.5)\begin{pmatrix}3\\3\end{pmatrix}\right]$$

$$= \begin{bmatrix}0\\0\end{bmatrix} + 0.1 \cdot \begin{bmatrix}1-0.5-1.5\\0.5-1-1.5\end{bmatrix}$$

$$= \begin{bmatrix}0\\0\end{bmatrix} + 0.1 \cdot \begin{bmatrix}-1\\-2\end{bmatrix}$$

$$= \begin{bmatrix}-0.1\\-0.2\end{bmatrix}$$

9.3.2 Calculate Hessian

Calculate the Hessian:

$$\mathbf{H} = \nabla^2 \mathcal{L}$$

Differentiate every term in the gradient:

$$
\begin{aligned}
\mathbf{H} &= \nabla^2 \mathcal{L} \\
&= \frac{\partial}{\partial \mathbf{w}} \Big[\sum_{n=1}^{N} (y_n - \sigma_n) \mathbf{x}_n \Big]^\top \\
&= \frac{\partial}{\partial \mathbf{w}} \sum_{n=1}^{N} (y_n - \sigma_n) \mathbf{x}_n^\top \\
&= \sum_{n=1}^{N} \frac{\partial}{\partial \mathbf{w}} (y_n - \sigma_n) \mathbf{x}_n^\top \\
&= \sum_{n=1}^{N} -\sigma_n' \mathbf{x}_n^\top \\
&= \sum_{n=1}^{N} -\sigma_n (1 - \sigma_n) \mathbf{x}_n \mathbf{x}_n^\top
\end{aligned}
$$

In general:

$$
\nabla^2 \mathcal{L} = \sum_{n=1}^{N} -\sigma_n (1 - \sigma_n) \mathbf{x}_n \mathbf{x}_n^\top
$$

9.3.3 Objective Function

Notice that the original Log-Likelihood:

$$
\mathcal{L}(\mathbf{y} | \mathbf{X}, \mathbf{w}) = \log \sum_{n=1}^{n} P(y_n | \mathbf{x}_n)
$$

is *negative* since the probability $P(y_n | \mathbf{x}_n)$ is lower than 1.

Therefore, we set the objective function $\mathcal{J}(\mathbf{w})$ to be the negative log-likelihood:

$$
\mathcal{J}(\mathbf{w}) = -\mathcal{L}(\mathbf{w}) = -\sum_{n=1}^{N} \Big[y_n \log \Big(\sigma(\mathbf{w}^\top \mathbf{x}_n) \Big) + (1 - y_n) \log \Big(1 - \sigma(\mathbf{w}^\top \mathbf{x}_n) \Big) \Big]
$$

Therefore,

- The gradient: $\nabla \mathcal{J}(\mathbf{w}) = -\sum_{n=1}^{N} (y_n - \sigma_n) \mathbf{x}_n$
- The Hessian: $\nabla^2 \mathcal{J}(\mathbf{w}) = \sum_{n=1}^{N} \sigma_n (1 - \sigma_n) \mathbf{x}_n \mathbf{x_n}^\top$

The optimization problem went from:

- Maximizing $\mathcal{L}$

- to minimizing $-\mathcal{L}$

Therefore, we use gradient descent:

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \eta \cdot \frac{\partial(-\mathcal{L})}{\partial \mathbf{w}}$$

The *update* part of the Newton's method $\eta\left(\frac{\partial \mathcal{L}}{\partial \mathbf{w}}\right)$ has the form:

$$\Delta \mathbf{w} = -\Big[\sum_{n}^{N} \sigma_n(1-\sigma_n)\mathbf{x}_n\mathbf{x}_n^\top\Big]^{-1}\Big[-\sum_{n=1}^{N}(y_n-\sigma_n)\mathbf{x}_n\Big]$$

Namely,

$$\Delta \mathbf{w} = \left(\mathbf{X}\, S\, \mathbf{X}^\top\right)^{-1} Sb$$

where:

$$\mathbf{X} = \begin{bmatrix} - & \mathbf{x}_1^\top & - \\ - & \mathbf{x}_2^\top & - \\ & \vdots & \\ - & \mathbf{x}_d^\top & - \end{bmatrix}$$

$$S = \begin{bmatrix} \sigma_1(1-\sigma_1) & 0 & \cdots & 0 \\ 0 & \sigma_2(1-\sigma_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \sigma_n(1-\sigma_n) \end{bmatrix}$$

$$b = \begin{bmatrix} \frac{y_1-\sigma_1}{\sigma_1(1-\sigma_1)} \\ \frac{y_2-\sigma_2}{\sigma_2(1-\sigma_2)} \\ \vdots \\ \frac{y_N-\sigma_N}{\sigma_N(1-\sigma_N)} \end{bmatrix}$$

9.3.4 Recap: IRLS Algorithm

Input

- $\{(\mathbf{x}_n, y_n)|n = 1, 2, \cdots, N\}$

Do

1. Initialize $\mathbf{w} = 0$ and $w_0 = \log \frac{\bar{\mathbf{y}}}{1-\bar{\mathbf{y}}}$

2. Repeat until convergence:
    1. for $n = 1, 2, \cdots, N$ do:
    1. Compute $\sigma_n = \sigma(\mathbf{w}^\top \mathbf{x}_n + w_0)$
    2. Compute $s_n = \sigma_n(1 - \sigma_n)$
    3. Compute $b_n = \frac{y_n - \sigma_n}{s_n}$
    2. Construct $S = diag(s_{1:N})$
    3. Update $\mathbf{w} = (XSX^\top)Sb$

    Output

- $\mathbf{w}$

9.4 Multi-Class Extension

9.4.1 Model IO

In the multi-class classification essence, we will give each class its own weight. Suppose that we have M classes, the weights are:

$$\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M\}$$

Therefore, given an input $\mathbf{x}_n$, it will generate a score for each class:

$$\text{scores} = \{\mathbf{w}_1^\top \mathbf{x}_n, \mathbf{w}_2^\top \mathbf{x}_n, \ldots, \mathbf{w}_M^\top \mathbf{x}_n\} \subset \mathbb{R}$$

These individual scores $\theta_k = \mathbf{w}_k^\top \mathbf{x}_n$ are called "Logits". We softmax these logits to make it sum to 1:

$$p(y_n = k|\mathbf{x}_n) = \text{softmax}(\mathbf{w}_k^\top \mathbf{x}_n) = \frac{e^{\mathbf{w}^\top \mathbf{x}_n}}{\sum_{j=1}^{M} e^{\mathbf{w}_j^\top \mathbf{x}_n}}$$

9.4.2 Likelihood Function

....

CISC3023 Assignment 3

1. Question 1

Given historical data as below. If current weights $\mathbf{w}$ for the logistic regression model (in which $P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$) is $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Update $\mathbf{w}$ for one step according to the gradient ascent. The learning rate is $\eta = 0.1$.

y	x_1	x_2	index
1	2	1	1
0	1	2	2
0	3	3	3

Answer:
Use Gradient Ascend:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta \cdot \sum_{n=1}^{3} \Big[y_n - \sigma\Big((\mathbf{w}^{(k)})^\top \mathbf{x}_n \Big) \Big] \mathbf{x}_n$$

Summation term:

$$\begin{aligned}
\mathbf{w}_1 &= 0.1 \times \left[\Big(1 - \sigma(0)\Big) \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \Big(0 - \sigma(0)\Big) \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \Big(0 - \sigma(0)\Big) \begin{bmatrix} 3 \\ 3 \end{bmatrix} \right] \\
&= 0.1 \times \left[\frac{1}{2} \begin{bmatrix} 2 \\ 1 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 3 \\ 3 \end{bmatrix} \right] \\
&= 0.1 \times \begin{bmatrix} 1 - \frac{1}{2} - \frac{3}{2} \\ \frac{1}{2} - 1 - \frac{3}{2} \end{bmatrix} \\
&= \begin{bmatrix} -0.1 \\ -0.2 \end{bmatrix}
\end{aligned}$$

That is, the updated one step of $\mathbf{w}$ is:

$$\mathbf{w}_1 = \begin{bmatrix} -0.1 \\ -0.2 \end{bmatrix}$$