# CS229 Problem Sets

## PSet 0

1. **Gradients and Hessians**

(a) $f(x) = \frac{1}{2}x^T Ax + b^T x$, $A \in \mathbb{S}^n$, $b \in \mathbb{R}^n$

$$\Rightarrow \nabla f(x) = Ax + b$$

(b) $f(x) = g(h(x))$, $g : \mathbb{R} \to \mathbb{R}, h : \mathbb{R}^n \to \mathbb{R}$

$$\begin{aligned}(\nabla f(x))_i &= \frac{\partial f(x)}{\partial x_i} \\ &= \frac{\partial g(t)}{\partial t}\bigg|_{t=h(x)} \frac{\partial h(x)}{\partial x_i} \\ &= g'(h(x))\frac{\partial h(x)}{\partial x_i} \\ \Rightarrow \nabla f(x) &= g'(h(x))\nabla h(x)\end{aligned}$$

(c) $f(x) = \frac{1}{2}x^T Ax + b^T x, A \in \mathbb{S}^n, b \in \mathbb{R}^n$

$$\nabla^2 f(x) = A$$

(d) $f(x) = g(a^T x), g : \mathbb{R} \to \mathbb{R}, a \in \mathbb{R}^n$

$$\begin{aligned}\nabla f(x) &= ag'(a^T x) \\ \left(\nabla^2 f(x)\right)_{ij} &= \frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i} g'(a^T x)a_j = a_i a_j g''(a^T x) \\ \Rightarrow \nabla^2 f(x) &= g''(a^T x)\begin{bmatrix} a_1^2 & a_1 a_2 & \dots & a_1 a_n \\ a_2 a_1 & a_2^2 & \dots & a_2 a_n \\ \vdots & \vdots & \ddots & \vdots \\ a_n a_1 & a_n a_2 & \dots & a_n^2 \end{bmatrix} \\ &= aa^T g''(a^T x)\end{aligned}$$

2. **Positive Definite Matrices**

$$A \succ 0 \text{ iff } x^T Ax > 0 \forall x \neq 0$$

(a)
$$A = zz^T$$

Firstly, note that $A$ is symmetric.

$$\begin{aligned} x^T Ax &= x^T zz^T x \\ &= \left(z^T x\right)^T \left(z^T x\right) \\ &= \left\|z^T x\right\|_2^2 > 0 \qquad \forall x \neq 0 \end{aligned}$$

Thus, $A \succ 0$.

(b) $$A = zz^T$$

Outer product. Null space: vectors $x$ such that $Ax = 0$.

$$\begin{aligned} Ax &= 0 \\ \Rightarrow zz^T x &= 0 \\ \Rightarrow (x^T z) z^T &= 0 \end{aligned}$$

Note that the resultant vector is a scalar multiple of $z^T$. A scalar multiple of $z^T$ can be zero only if the scalar factor is zero. (Given that $z$ is non-zero.)

$$\Rightarrow x^T z = 0$$

This is true for all $x$ orthogonal to $z$. Thus null space is the $(n-1)$ dimensional subspace orthogonal to $z$.

Since $\dim(\mathcal{R}(A)) + \dim(\mathcal{N}(A)) = n$, $\text{rank}(A) = 1$. This means there is only one linearly independent column of $A$, i.e. any column can be written as a multiple of any other column. This is evident from the definition of the outer product.

(c) $A \in \mathbb{R}^{n \times n}$ is PSD, $B \in \mathbb{R}^{m \times n}$ is arbitrary. Let $G = BAB^T \in \mathbb{R}^{m \times m}$.

$$\begin{aligned} G &= BAB^T \\ x^T Gx &= x^T BAB^T x \\ &= y^T Ay \end{aligned}$$

for $y = B^T x$. Note that since $A$ is PSD, $y^T Ay \geq 0 \forall y$.

So, $G$ is PSD.

3. **Eigenvectors, Eigenvalues and the spectral theorem**

(a) $$\begin{aligned} A &= T\Lambda T^{-1} \\ \Rightarrow AT &= T\Lambda \end{aligned}$$

Here, $i$th column of the LHS will be $At^{(i)}$. The $i$th column of the RHS will be $\lambda_i t^{(i)}$.

So,

$$At^{(i)} = \lambda_i t^{(i)}$$

(b) If $U$ is orthogonal, $U^T = U^{-1}$, so $A = U\Lambda U^T = U\Lambda U^{-1}$. From (a), it follows that $Au^{(i)} = \lambda_i u^{(i)}$ and thus $u^{(i)}$ is an eigenvector of $A$.

(c) If $A$ is PSD, then $x^T Ax \geq 0 \forall x$. By spectral theorem,

$$
\begin{aligned}
x^T U \Lambda U^T x \geq 0 \forall x \\
\Rightarrow y^T \Lambda y \geq 0 \forall y \\
\Rightarrow \sum \lambda_i y_i^2 \geq 0 \forall y
\end{aligned}
$$

This is only possible if all $\lambda_i \geq 0$.

# Some Matrix Derivatives

1. $\nabla_A \operatorname{tr} AB = \nabla_A \operatorname{tr} BA = B^T$

   Let $B \in \mathbb{R}^{n\times m}$ and $f(A) : \mathbb{R}^{m\times n} \to \mathbb{R} = \operatorname{tr} AB$. Then, $(i \le m, j \le n)$

$$\begin{aligned}
(\nabla_A \operatorname{tr} AB)_{ij} = \frac{\partial(\operatorname{tr} AB)}{\partial A_{ij}} &= \frac{\partial((AB)_{11} + ... + (AB)_{mm})}{\partial A_{ij}} \\
&= \frac{\partial}{\partial A_{ij}}\left(\sum_p^m (AB)_{pp}\right) \\
&= \frac{\partial}{\partial A_{ij}}\left(\sum_p^m \sum_q^n A_{pq}B_{qp}\right) \\
&= B_{ji} \\
\Rightarrow \nabla_A \operatorname{tr} AB &= B^T
\end{aligned}$$

   ■

2. $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$

   Let $A \in \mathbb{R}^{m\times n}$.

$$\begin{aligned}
(\nabla_{A^T} f(A))_{ij} = \frac{\partial}{\partial (A^T)_{ij}} f(A) &= \frac{\partial}{\partial A_{ji}} f(A) \\
&= (\nabla_A f(A))_{ji} \\
\Rightarrow \nabla_{A^T} f(A) &= (\nabla_A f(A))^T
\end{aligned}$$

   ■

3. $\nabla_A \operatorname{tr} ABA^TC = CAB + C^TAB^T$

   Let $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$. Let $D = ABA^TC \in \mathbb{R}^{m\times m}$

$$(\nabla_A \operatorname{tr} ABA^TC)_{ij} = \frac{\partial}{\partial A_{ij}}(\operatorname{tr} ABA^TC)$$

   Notice that for matrix multiplication,

$$\begin{aligned}
(AB)_{ij} &= \sum_k A_{ik}B_{kj} \\
(ABC)_{ij} &= \sum_{k,l} A_{ik}B_{kl}C_{lj} \\
(ABCD)_{ij} &= \sum_{k,l,m} A_{ik}B_{kl}C_{lm}D_{mj}
\end{aligned}$$

   Then,

$$\begin{aligned}
\operatorname{tr} ABA^TC &= \sum_p \sum_{q,r,s} A_{pq} B_{qr} A^T_{rs} C_{sp} \\
\frac{\partial}{\partial A_{ij}} (\operatorname{tr} ABA^TC) &= \sum_{r,s} B_{jr} A^T_{rs} C_{si} + \sum_{p,q} A_{pq} B_{qj} C_{ip} \\
&= \left(BA^TC\right)_{ji} + (CAB)_{ij} \\
\Rightarrow \nabla_A \operatorname{tr} ABA^TC &= C^TAB^T + CAB
\end{aligned}$$

■

# PSet 1

1. **Linear Classifiers (logistic regressions and GDA)**

   (a) Average empirical loss for logistic regression

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log\big(g\big(\theta^T x^{(i)}\big)\big) + \big(1 - y^{(i)}\big) \log\big(1 - g\big(\theta^T x^{(i)}\big)\big)$$

   $y^{(i)} \in \{0, 1\}$, $h_\theta(x) = g(\theta^T x)$ and $g(z) = \frac{1}{1+e^{-z}}$

   Hessian of $J(\theta)$:

$$\begin{aligned}
\big(\nabla^2 J(\theta)\big)_{i,j} &= \frac{\partial^2}{\partial\theta_i \partial\theta_j} J(\theta) \\
&= \frac{\partial}{\partial\theta_j}\left(-\frac{1}{m}\sum_{i=1}^{m} \frac{y^{(i)}}{g\big(\theta^T x^{(i)}\big)} g'\big(\theta^T x^{(i)}\big) x_i^{(i)} - \frac{1 - y^{(i)}}{1 - g\big(\theta^T x^{(i)}\big)} g'\big(\theta^T x^{(i)}\big) x_i^{(i)}\right) \\
&= \frac{\partial}{\partial\theta_j}\left(-\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\big(1 - g\big(\theta^T x^{(i)}\big)\big) x_i^{(i)} - \big(1 - y^{(i)}\big) g\big(\theta^T x^{(i)}\big) x_i^{(i)}\right) \\
&= -\frac{1}{m}\frac{\partial}{\partial\theta_j}\sum_{i=1}^{m} x_i^{(i)}\big(y^{(i)} - y^{(i)} g\big(\theta^T x^{(i)}\big) - g\big(\theta^T x^{(i)}\big) + y^{(i)} g\big(\theta^T x^{(i)}\big)\big) \\
&= -\frac{1}{m}\frac{\partial}{\partial\theta_j}\sum_{i=1}^{m} x_i^{(i)}\big(y^{(i)} - g\big(\theta^T x^{(i)}\big)\big) = \frac{1}{m}\sum_{i=1}^{m} x_i^{(i)} x_j^{(i)} g\big(\theta^T x^{(i)}\big)\big(1 - g\big(\theta^T x^{(i)}\big)\big)
\end{aligned}$$

   Here,

$$\begin{aligned}
\nabla_\theta J(\theta) &= \frac{1}{m}\sum_{i=1}^{m}\big(g\big(\theta^T x^{(i)}\big) - y^{(i)}\big) x^{(i)} \\
&= \frac{1}{m}\sum_{i=1}^{m}\left(g\left(\begin{bmatrix} - & \theta^T & - \end{bmatrix}\begin{bmatrix} | \\ x^{(i)} \\ | \end{bmatrix}\right) - y^{(i)}\right)\begin{bmatrix} | \\ x^{(i)} \\ | \end{bmatrix}
\end{aligned}$$

   This can be thought of linear combination of columns $x^{(i)}$. Recall, $Ax$ can be written as linear combination of columns of $A$ with coefficients in $x$:

$$Ax = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1\begin{bmatrix} | \\ a_1 \\ | \end{bmatrix} + x_2\begin{bmatrix} | \\ a_2 \\ | \end{bmatrix} + \cdots + x_n\begin{bmatrix} | \\ a_n \\ | \end{bmatrix}$$

   So, in this case let $X$ be the matrix whose rows are $x^{(i)}$. Our coefficient vector then has components like $g\big(\theta^T x^{(i)}\big) - y^{(i)}$. We can say, that vector is $g(X\theta) - Y$. (Note that $\theta^T x^{(i)} \equiv x^{(i)^T}\theta$ and superposing rows of $x^{(i)}$, we get $X\theta$.)

$$\nabla_\theta J(\theta) = \frac{1}{m} X^T (g(X\theta) - Y)$$

Similarly, the Hessian is of the form

$$H_{jk} = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} x_k^{(i)} g\big(\theta^T x^{(i)}\big)\big(1 - g\big(\theta^T x^{(i)}\big)\big)$$

$$H_k = \frac{1}{m}\sum_{i=1}^{m} x_k^{(i)} g\big(\theta^T x^{(i)}\big)\big(1 - g\big(\theta^T x^{(i)}\big)\big) \begin{bmatrix} | \\ x^{(i)} \\ | \end{bmatrix}$$

$$= \frac{1}{m} X^T (g(X\theta)(1 - g(X\theta))X_k)$$

Here, the notation $1 - g(X\theta)$ means $[1, 1, ..., 1]^T - g(X\theta)$, commonly known as broadcasting.

Also, another subtlety in the last equality is the conversion of $g(\theta^T x)$ to $g(X\theta)$.

$$X\theta = \begin{bmatrix} — & x^{(1)} & — \\ — & x^{(2)} & — \\ & \vdots & \\ — & x^{(m)} & — \end{bmatrix} \theta = \begin{bmatrix} — & \theta^T x^{(1)} & — \\ — & \theta^T x^{(2)} & — \\ & \vdots & \\ — & \theta^T x^{(m)} & — \end{bmatrix}$$

$$g(X\theta) = \begin{bmatrix} — & g\big(\theta^T x^{(1)}\big) & — \\ — & g\big(\theta^T x^{(2)}\big) & — \\ & \vdots & \\ — & g\big(\theta^T x^{(m)}\big) & — \end{bmatrix}$$

$$H_k = \frac{1}{m}\sum_{i=1}^{m} x_k^{(i)} g\big(\theta^T x^{(i)}\big)\big(1 - g\big(\theta^T x^{(i)}\big)\big) \begin{bmatrix} | \\ x^{(i)} \\ | \end{bmatrix}$$

$$= \frac{1}{m} X^T \begin{bmatrix} x_k^{(1)} g\big(\theta^T x^{(1)}\big)\big(1 - g\big(\theta^T x^{(1)}\big)\big) \\ x_k^{(2)} g\big(\theta^T x^{(2)}\big)\big(1 - g\big(\theta^T x^{(2)}\big)\big) \\ \vdots \\ x_k^{(m)} g\big(\theta^T x^{(m)}\big)\big(1 - g\big(\theta^T x^{(m)}\big)\big) \end{bmatrix}$$

$$= \frac{1}{m} X^T (X_k \cdot g(X\theta) \cdot (1 - g(X\theta)))$$

where $\cdot$ represents elementwise multiplication.

$$H = \frac{1}{m} X^T \begin{bmatrix} | & | & & | \\ X_1 \cdot g(X\theta) \cdot (1 - g(X\theta)) & X_2 \cdot g(X\theta) \cdot (1 - g(X\theta)) & \cdots & X_n \cdot g(X\theta) \cdot (1 - g(X\theta)) \\ | & | & & | \end{bmatrix}$$

At this point, the pattern to be identified is the following:

$$H_{jk} = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} x_k^{(i)} g\big(\theta^T x^{(i)}\big)\big(1 - g\big(\theta^T x^{(i)}\big)\big)$$

or,

$$H_{jk} = \frac{1}{m}\sum_{i=1}^{m} w_i x_j^{(i)} x_k^{(i)}$$

This is a weighted sum of outer products. ($i$th feature vector's outer product weighted by $w_i = g\big(\theta^T x^{(i)}\big)\big(1 - g\big(\theta^T x^{(i)}\big)\big)$)

This can be represented by the product

$$H = \frac{1}{m} X^T W X$$

where $W = \text{diag}(w_1, w_2, \cdots, w_m)$.

$$H = \frac{1}{m}\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_m \end{bmatrix} \begin{bmatrix} — & x^{(1)} & — \\ — & x^{(2)} & — \\ & \vdots & \\ — & x^{(m)} & — \end{bmatrix}$$

$$= \frac{1}{m}\begin{bmatrix} | & | & & | \\ w_1 x^{(1)} & w_2 x^{(2)} & \cdots & w_m x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & x^{(1)} & — \\ — & x^{(2)} & — \\ & \vdots & \\ — & x^{(m)} & — \end{bmatrix}$$

$$= \frac{1}{m}\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & w_1 x^{(1)} & — \\ — & w_2 x^{(2)} & — \\ & \vdots & \\ — & w_m x^{(m)} & — \end{bmatrix}$$

The last representation above is the same as the one we arrived at from elementwise operations.

The second last one can be written as $(X^T \cdot g(X\theta) \cdot (1 - g(X\theta)))X$ where $g$ $m$-vectors are broadcasted to $n \times m$ ones and elementwise multiplied.

$$z^T H z = z^T X^T W X z = (Xz)^T W (Xz) = y^T W y$$

which is a diagonalised quadratic form.

$$= \sum_{i=1}^{n} w_i y_i^2$$

(Note that Hessian wrt $\theta$ is $\in \mathbb{R}^{n \times n}$ since each feature vector (and $\theta$) is $\in \mathbb{R}^n$)

Notice that $w_i$'s are always positive (since $g(\cdot)$ and $1 - g(\cdot) \geq 0 \forall\, \cdot$). Hence, $H$ is positive semi-definite.

Alternatively, expand the original quadratic form:

$$z^T H z = \frac{1}{m}\sum_{i=1}^{m} g\left(\theta^T x^{(i)}\right)\left(1 - g\left(\theta^T x^{(i)}\right)\right)\sum_{j=1}^{n}\sum_{k=1}^{n} z_j z_k x_j^{(i)} x_k^{(i)}$$
$$= \frac{1}{m}\sum_{i=1}^{m} w_i \sum_{j=1}^{n} z_j x_j^{(i)} \sum_{k=1}^{n} z_k x_k^{(i)} = \frac{1}{m}\sum_{i=1}^{m} w_i \left(z^T x^{(i)}\right)^2 \geq 0$$

(b) **Coding Problem**

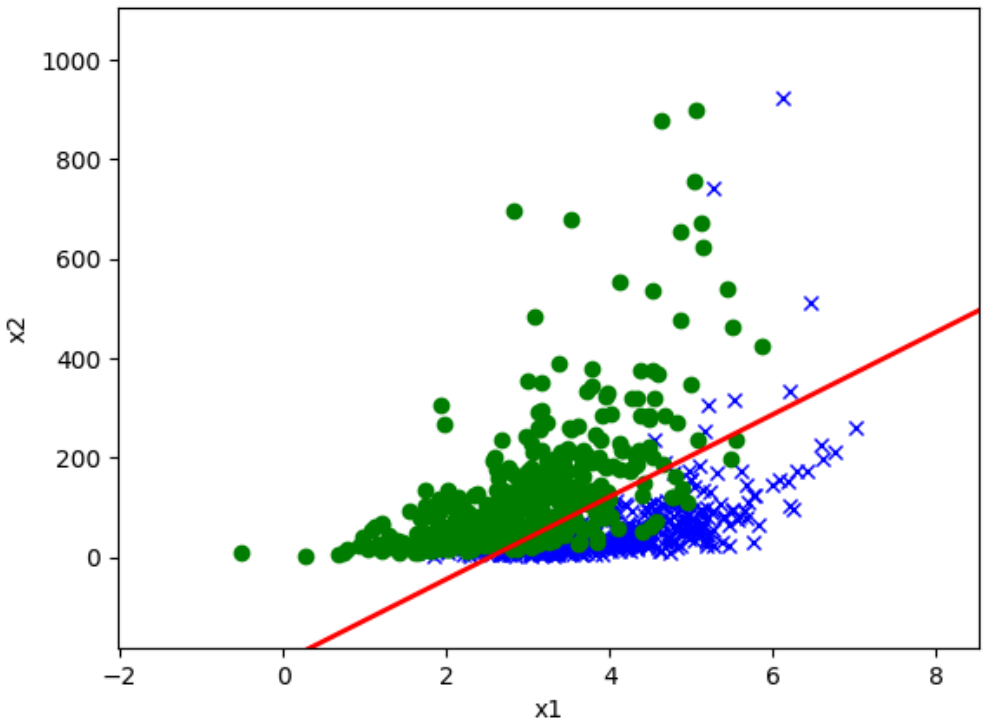
Newton's Method: Update $\theta$ as

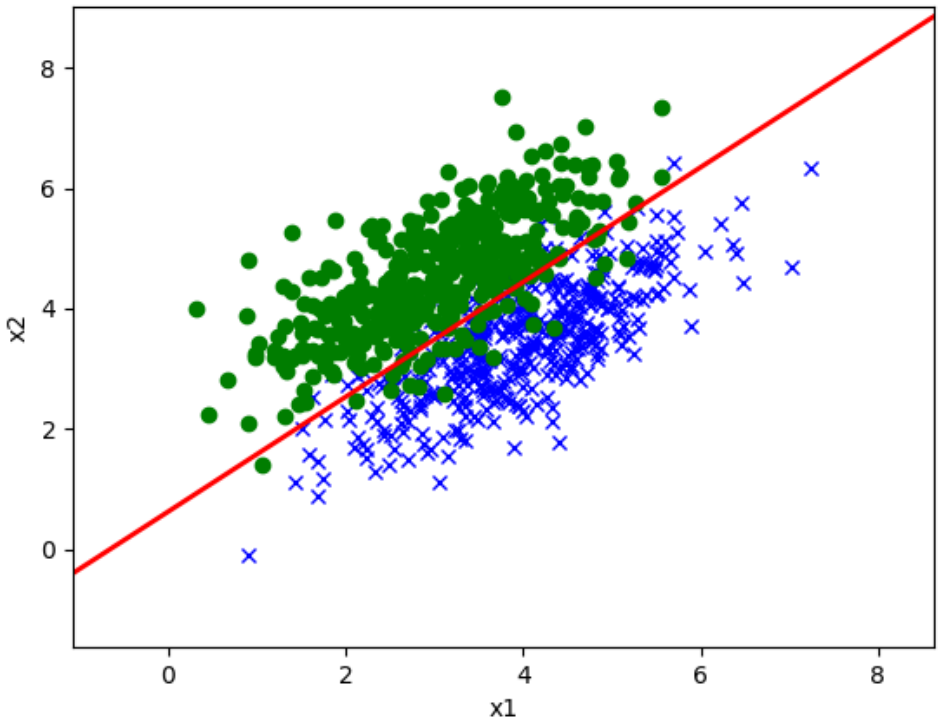
$$\theta := \theta - H^{-1}\nabla_\theta l(\theta)$$

where $l(\theta) = \sum_{i=1}^{m} y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - h\left(x^{(i)}\right)\right)$.

$$H = X^T W X$$
$$\nabla_\theta J(\theta) = X^T\left(h\left(x^{(i)}\right) - y^{(i)}\right)$$

```
class LogisticRegression(LinearModel):
    theta = []

    def fit(self, x: np.ndarray, y: np.ndarray):
        # y is in {0, 1}

        m, n = x.shape
        self.theta = np.zeros(n)

        def h(x):
            return 1 / (1 + np.exp(-x @ self.theta))

        # grad:
        g = x.T @ (h(x) - y)

        # Hessian:
        W = np.diag(h(x) * (1 - h(x)))
        H = x.T @ W @ x

        while True:
            prev = self.theta
            self.theta -= np.dot(np.linalg.inv(H), g)

            if np.linalg.norm(self.theta - prev, ord=1) < 1e-5:
                break

    def predict(self, x):
        return (1 / (1 + np.exp(-x @ self.theta)))
```




(c) Given that

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

$$p(x|y=0) = \frac{1}{(2\pi)^{n/2}\ |\Sigma|^{1/2}} \exp\Big(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\Big)$$

$$p(x|y=1) = \frac{1}{(2\pi)^{n/2}\ |\Sigma|^{1/2}} \exp\Big(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\Big)$$

parametrized by $\phi, \mu_0, \mu_1, \Sigma$

To show:

$$p(y=1|x;\phi,\mu_0,\mu_1,\Sigma) = \frac{1}{1+\exp(-(\theta^T x + \theta_0))}$$

From Bayes rule,

$$\begin{aligned} p(y=1|x) &= \frac{p(x|y=1)p(y=1)}{p(x)} \\ &= \frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1) + p(x|y=0)p(y=0)} \\ &= \frac{\exp\Big(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\Big) \times \phi}{\exp\Big(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\Big) \times \phi + \exp\Big(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\Big) \times (1-\phi)} \\ &= \frac{1}{1+\exp\Big(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0) + \frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\Big) \times \frac{1-\phi}{\phi}} \end{aligned}$$

Let negative of value within the exp be $M$.

$$\begin{aligned} M &= \frac{1}{2}\Big((x-\mu_0)^T\Sigma^{-1}(x-\mu_0) - (x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\Big) - \log\left(\frac{1-\phi}{\phi}\right) \\ &= \frac{1}{2}(x^T\Sigma^{-1}x - \mu_0^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_0 + \mu_0^T\Sigma^{-1}\mu_0 - x^T\Sigma^{-1}x + \mu_1^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_1 - \mu_1^T\Sigma^{-1}\mu_1) \\ &\quad - \log\left(\frac{1-\phi}{\phi}\right) \\ &= \frac{1}{2}(\mu_1^T - \mu_0^T)\Sigma^{-1}x + \frac{1}{2}x^T\Sigma^{-1}(\mu_1-\mu_0) + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_1^T\Sigma^T\mu_1 - \log\left(\frac{1-\phi}{\phi}\right) \\ &= (\Sigma^{-1}(\mu_1-\mu_0))^T x + \frac{1}{2}(\mu_0+\mu_1)^T\Sigma^{-1}(\mu_0-\mu_1) - \log\left(\frac{1-\phi}{\phi}\right) \end{aligned}$$

Comparing this quantity to $\theta^T x + \theta_0$, we can see that

$$\theta = \Sigma^{-1}(\mu_1 - \mu_0)$$

$$\theta_0 = \frac{1}{2}(\mu_0+\mu_1)^T\Sigma^{-1}(\mu_0-\mu_1) - \log\left(\frac{1-\phi}{\phi}\right)$$

Thus, GDA results in a classifier having a linear decision boundary (since $p(y = 1|x)$ follows a sigmoid).

(d)
$$l(\phi, \mu_0, \mu_1, \Sigma) = \sum_{i=1}^{m} \log p\big(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma\big)$$
$$= \sum_{i=1}^{m} \log p\big(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma\big)p\big(y^{(i)}; \phi\big)$$

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

$$p(x|y = 0) = \frac{1}{(2\pi)^{n/2}\ |\Sigma|^{1/2}} \exp\Big(-\frac{1}{2}(x - \mu_0)^T \Sigma^{-1}(x - \mu_0)\Big)$$

$$p(x|y = 1) = \frac{1}{(2\pi)^{n/2}\ |\Sigma|^{1/2}} \exp\Big(-\frac{1}{2}(x - \mu_1)^T \Sigma^{-1}(x - \mu_1)\Big)$$

To maximise $l(\phi, \mu_0, \mu_1, \Sigma)$, make $\frac{\partial l}{\partial \phi} = 0$, $\frac{\partial l}{\partial \Sigma} = 0$, $\frac{\partial l}{\partial \mu_0} = 0$, $\frac{\partial l}{\partial \mu_1} = 0$.

$$\frac{\partial l}{\partial \phi} = \sum_{i=1}^{m} \frac{\partial}{\partial \phi} \log p\big(y^{(i)}; \phi\big)$$
$$= \sum_{i=1}^{m} \frac{1}{p(y^{(i)}; \phi)}(-1)^{y^{(i)}+1} = 0$$
$$\Rightarrow \sum_{i=1}^{m} 1\big\{y^{(i)} = 1\big\}\frac{1}{\phi} - \sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\}\frac{1}{1 - \phi} = 0$$
$$\Rightarrow \frac{\phi}{1 - \phi} = \frac{\sum_{i=1}^{m} 1\big\{y^{(i)} = 1\big\}}{\sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\}}$$
$$\Rightarrow \phi = \frac{1}{m} \sum_{i=1}^{m} 1\big\{y^{(i)} = 1\big\}$$

$$\frac{\partial l}{\partial \mu_0} = \sum_{i=1}^{m} \frac{\partial}{\partial \mu_0} \log p\big(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma\big)$$
$$= \sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\}\frac{\partial}{\partial \mu_0} \log \frac{1}{\sqrt{2\pi\Sigma}} \exp\left(-\frac{1}{2}\frac{\big(x^{(i)} - \mu_0\big)^2}{\Sigma}\right)$$
$$= \sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\}\Sigma^{-1}\big(x^{(i)} - \mu_0\big) = 0$$
$$\Rightarrow \mu_0 = \frac{\sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\}x^{(i)}}{\sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\}}$$

Similarly

$$\mu_1 = \frac{\sum_{i=1}^{m} 1\big\{y^{(i)} = 1\big\}x^{(i)}}{\sum_{i=1}^{m} 1\big\{y^{(i)} = 1\big\}}$$

$$\frac{\partial l}{\partial \Sigma} = \sum_{i=1}^{m} \frac{\partial}{\partial \Sigma} \log p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma)$$
$$= \sum_{i=1}^{m} \frac{\partial}{\partial \Sigma} \left( -\log\left((2\pi)^{n/2} \ |\Sigma|^{1/2}\right) - \frac{1}{2}(x^{(i)} - \mu_y)^T \Sigma^{-1} (x^{(i)} - \mu_y) \right)$$

where $\mu_y = \begin{cases} \mu_0 & y=0 \\ \mu_1 & y=1 \end{cases} = 1\{y=0\}\mu_0 + 1\{y=1\}\mu_1$

$$= \sum_{i=1}^{m} -\frac{1}{2\Sigma} - \frac{1}{2}(x^{(i)} - \mu_y)^2 \frac{\partial}{\partial \Sigma} \frac{1}{\Sigma} = \sum_{i=1}^{m} -\frac{1}{2\Sigma} + \frac{(x^{(i)} - \mu_y)^2}{2\Sigma^2} = 0$$

$$\Rightarrow m\Sigma = \sum_{i=1}^{m} (x^{(i)} - \mu_y)^2$$

$$\Rightarrow \Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu_y)^2$$

$$\Rightarrow \Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T$$

(e) **Coding Problem**

See (c). We can write

$$p(y=1|x) = \frac{1}{1 + \exp(-(\theta^T x + \theta_0))}$$
$$= \frac{1}{1 + \exp(-\theta^T x)}$$

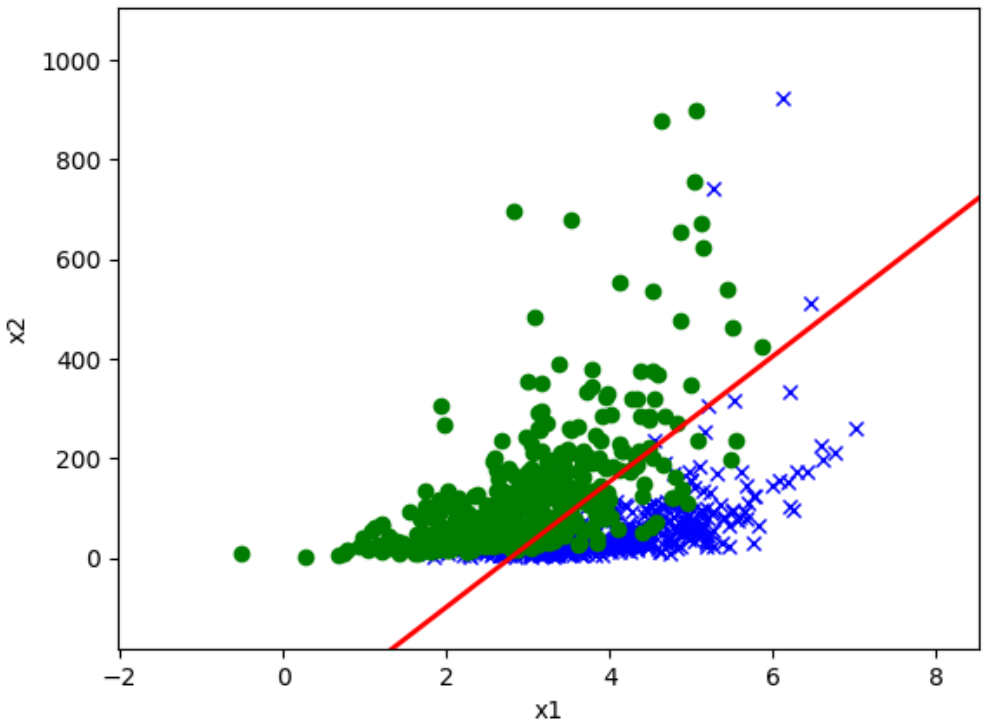
(redefine $x$ to be $n+1$-dimensional with $x_0 = 1$ and $\theta_0$)

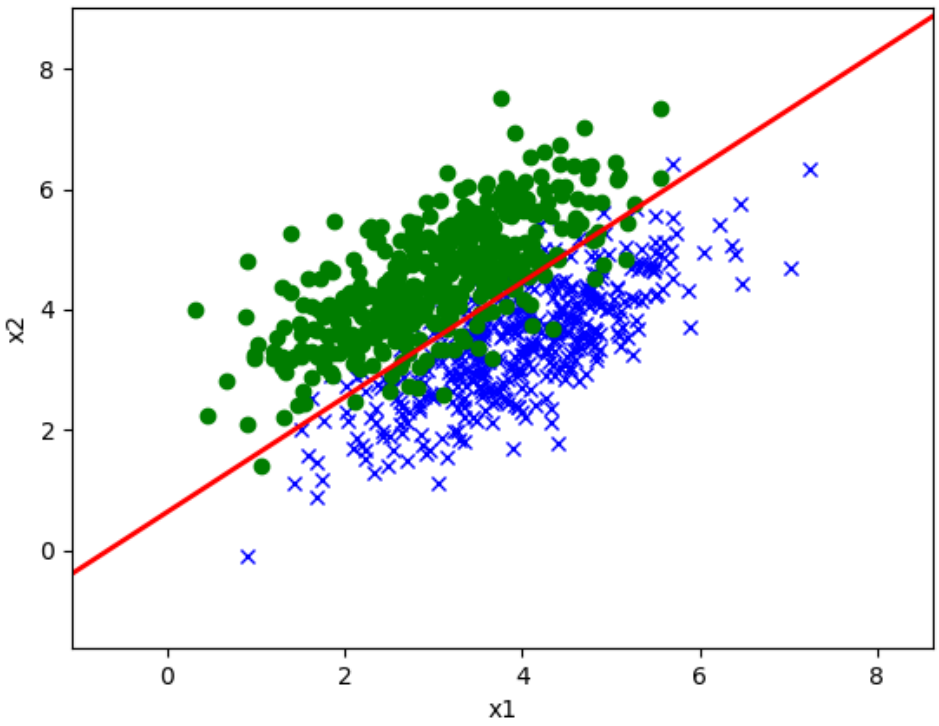
where $\theta = \Sigma^{-1}(\mu_1 - \mu_0)$ and $\theta_0 = \frac{1}{2}(\mu_0 + \mu_1)^T \Sigma^{-1}(\mu_0 - \mu_1) - \log\left(\frac{1-\phi}{\phi}\right)$

```
class GDA(LinearModel):
    def fit(self, x, y):
        phi = np.mean(y)
        # axis=0 means normal array of vectors avg
        mu_0 = np.mean(x[y == 0], axis=0)
        mu_1 = np.mean(x[y == 1], axis=0)

        m, n = x.shape

        sigma = np.zeros((n, n))
        for i in range(m):
            sigma += np.outer(x[i] - (mu_1 if y[i] == 1 else mu_0), x[i] -
(mu_1 if y[i] == 1 else mu_0))
        sigma /= m

        sigma_inv = np.linalg.inv(sigma)

        self.theta = np.zeros(n + 1)
        self.theta[1:] = sigma_inv @ (mu_1 - mu_0)
```

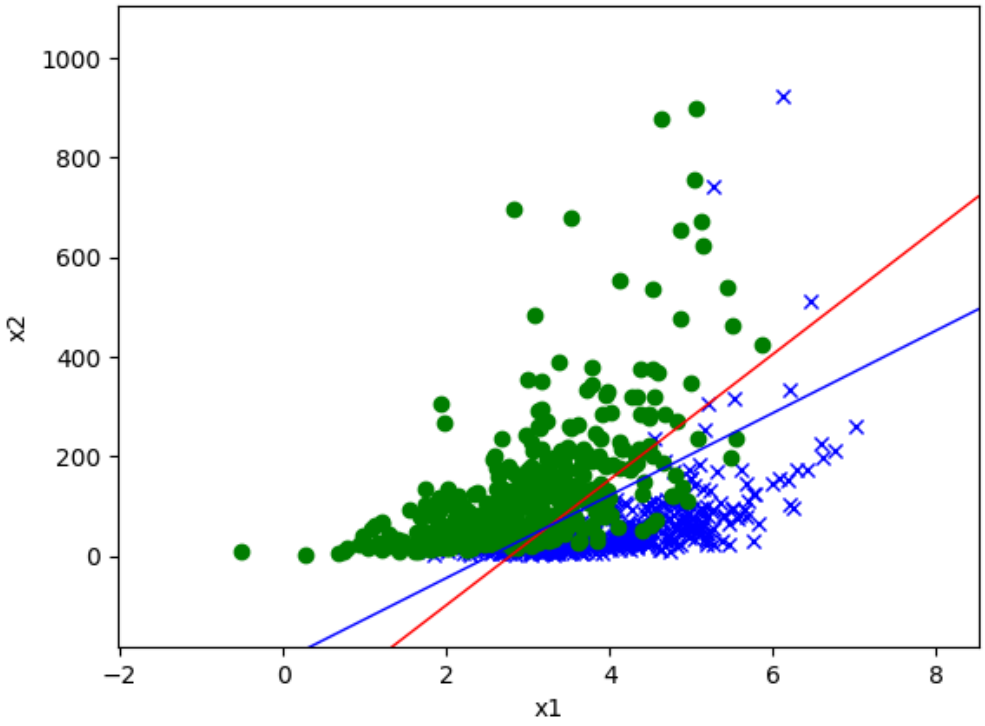
```
        self.theta[0] = 0.5 * (mu_0 + mu_1).T @ sigma_inv @ (mu_0 - mu_1) -
np.log((1 - phi) / phi)
```

```
    def predict(self, x):
        x_mod = np.zeros((x.shape[0], x.shape[1] + 1))
        x_mod[:, 1:] = x
        x_mod[:, 0] = 1

        return 1 / (1 + np.exp(-x_mod @ self.theta))
```

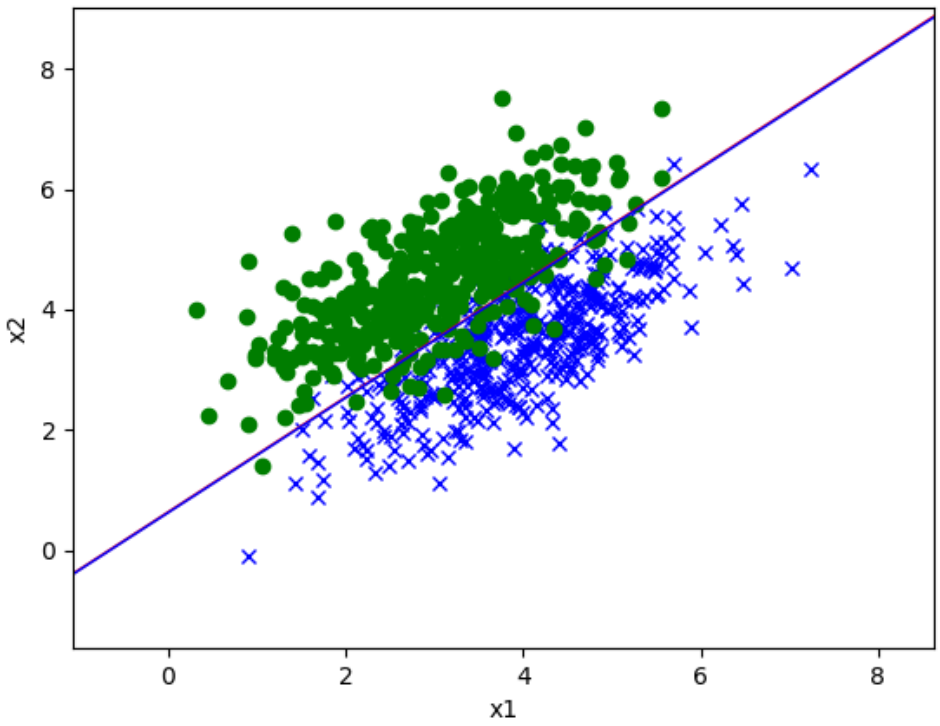





(f) Dataset 1:




(g) Dataset 2:




In Dataset 1, GDA (red) performs worse, possibly due to data not being Gaussian.

(h) To make GDA perform better in Dataset 1, we need a transformation which makes it more "normal". Note that the data points in Dataset 1 are strictly non-negative.

A **power transform** is a data mapping which is used to normalise data and stabilise variance.

The Box-Cox transform is given as

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^{(\lambda)} - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln y_i & \text{if } \lambda = 0 \end{cases}$$

where $y_i > 0$ is the data being transformed, which in our case is $x_j^{(i)}$.

2. **Incomplete, Positive-Only Labels**

Binary Classification Problem.

Dataset $\left\{\left(x^{(i)}, t^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$ where $t^{(i)} = \{0, 1\}$ is the label, and $y^{(i)} = \{0, 1\}$ indicate whether it is labelled.

$y^{(i)} = 0$ - example is unlabeled, maybe positive or negative. $y^{(i)} = 0$ - example is labeled and is positive.

$$p\big(t^{(i)} = 1 | y^{(i)} = 1\big) = 1$$

Construct binary classifier $h$

$$h\big(x^{(i)}\big) = p\big(t^{(i)} = 1 | x^{(i)}\big)$$

(a) Given that

$$p\big(y^{(i)} = 1 | t^{(i)} = 1, x^{(i)}\big) = p\big(y^{(i)} = 1 | t^{(i)} = 1\big)$$

or equivalently

$$p\big(x^{(i)}, y^{(i)} = 1 | t^{(i)} = 1\big) = p\big(x^{(i)} | t^{(i)} = 1\big) p\big(y^{(i)} = 1 | t^{(i)} = 1\big)$$

(i.e. given a positive example, the probability of it being labelled is independent of the example itself, labelled examples are a random sample of positive examples.)

To show:

$$p\big(t^{(i)} = 1 | x^{(i)}\big) = \frac{p\big(y^{(i)} = 1 | x^{(i)}\big)}{\alpha}$$

Properties:

$$p(X|A) = \frac{p(X, A)}{p(A)}$$

$$p(X) = p(X|A_1)p(A_1) + ... + p(X|A_n)p(A_n)$$

for $A_i$ mutually exclusive and collectively exhaustive.

$$\alpha = \frac{p\big(y^{(i)} = 1, x^{(i)}\big)}{p\big(t^{(i)} = 1, x^{(i)}\big)}$$

$$= \frac{p\big(y^{(i)} = 1, x^{(i)} | t^{(i)} = 1\big) p\big(t^{(i)} = 1\big) + p\big(y^{(i)} = 1, x^{(i)} | t^{(i)} = 0\big) p\big(t^{(i)} = 0\big)}{p\big(t^{(i)} = 1, x^{(i)}\big)}$$

$$= \frac{p\big(y^{(i)} = 1 | t^{(i)} = 1\big) p\big(x^{(i)} | t^{(i)} = 1\big) p\big(t^{(i)} = 1\big) + \cancel{p\big(y^{(i)} = 1 | t^{(i)} = 0\big)} p\big(x^{(i)}, t^{(i)} = 1\big) p\big(t^{(i)} = 0\big)}{p\big(t^{(i)} = 1, x^{(i)}\big)}$$

$$= p\big(y^{(i)} = 1 | t^{(i)} = 1\big)$$

(fraction of positive examples that are labelled)

This proves that $p\big(y^{(i)} = 1 | x^{(i)}\big)$ and $p\big(t^{(i)} = 1 | x^{(i)}\big)$ are proportional. Note that for all training examples we know the value of $y^{(i)}$ but not $t^{(i)}$. So with our usual methods we can model $p\big(y^{(i)} = 1 | x^{(i)}\big)$ and using it find $p\big(t^{(i)} = 1 | x^{(i)}\big)$.

(b) Validation set $V$

$V_+$ - set of labelled (hence positive) examples in $V$

Assume $h\big(x^{(i)}\big) \approx p\big(y^{(i)} = 1 | x^{(i)}\big) \forall x^{(i)}$

Assume $p\left(t^{(i)}=1|x^{(i)}\right) \approx 1$ (all examples are positive)

Show that $h\left(x^{(i)}\right) \approx \alpha$ for $x^{(i)} \in V_+$

For $x^{(i)} \in V_+$,

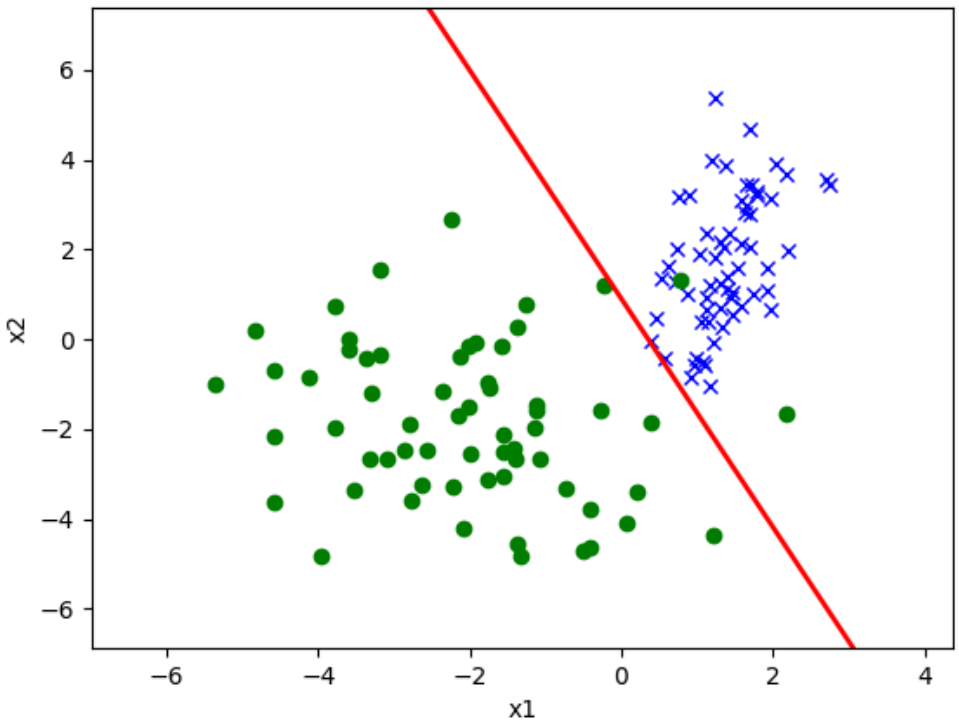
$$\begin{aligned} h\left(x^{(i)}\right) &= p\left(y^{(i)}=1|x^{(i)}\right) = \alpha \times p\left(t^{(i)}=1|x^{(i)}\right) \\ &= \alpha \times 1 = \alpha \end{aligned}$$

This shows if we model $p\left(y^{(i)}=1|x^{(i)}\right)$, then prediction on any labelled value will output a probability approximately $\alpha$. This can be used to estimate the proportionality factor $\alpha$ which is necessary to estimate $p\left(t^{(i)}=1|x^{(i)}\right)$. (This is an unknown quantity in a $\left(x^{(i)}, y^{(i)}\right)$ dataset.)

(c) **Coding Problem**

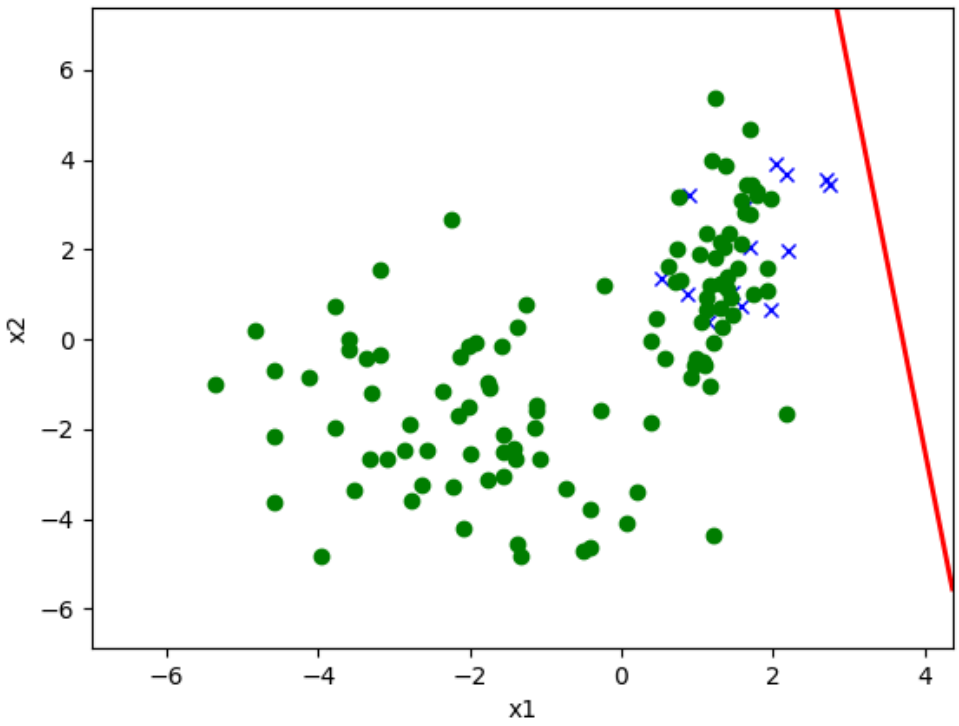
To train model on t-labels.

```
model = LogisticRegression()
x_train, y_train = util.load_dataset(train_path, label_col='t',
                                     add_intercept=True)
x_test, y_test = util.load_dataset(test_path, label_col='t',
                                   add_intercept=True)
model.fit(x_train, y_train)

t_pred = model.predict(x_test)

util.plot(x_test, y_test, model.theta,
          pred_path_c.replace('.txt', '.png'))

np.savetxt(pred_path_c, t_pred > 0.5, fmt='%d')
```




(d) To train model on y-labels and to predict $p(y^{(i)} = 1|x^{(i)})$

```
x_train, y_train = util.load_dataset(train_path, label_col='y',
                                     add_intercept=True)
model.fit(x_train, y_train)

x_test, y_test = util.load_dataset(test_path, label_col='y',
                                   add_intercept=True)
y_pred = model.predict(x_test)


util.plot(x_test, y_test, model.theta,
          pred_path_d.replace('.txt', '.png'))

np.savetxt(pred_path_d, y_pred > 0.5, fmt='%d')
```

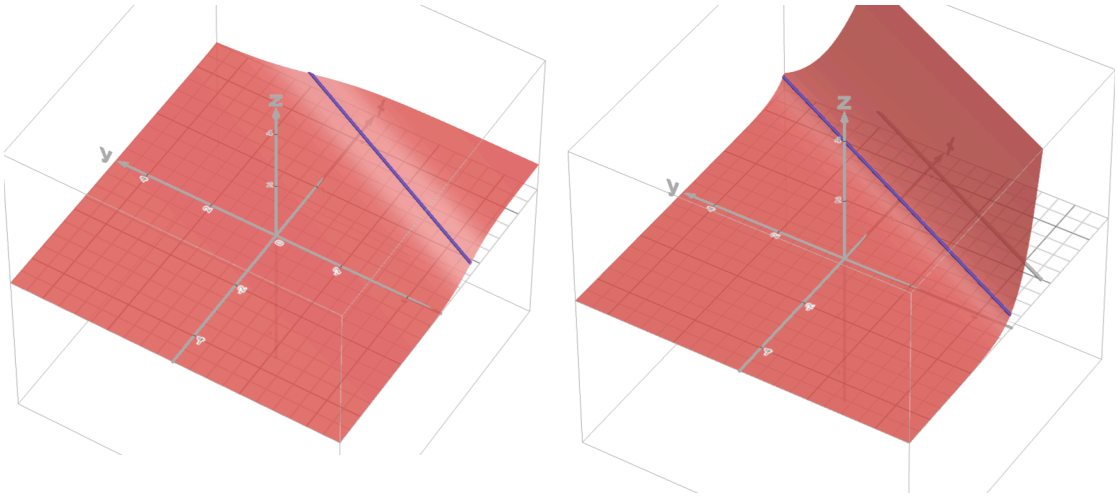



Here, blue data points are labelled ones. Green data points are unlabelled and thus contain both positive and negative examples.

The red line is the decision boundary where $p(y = 1|x) = 0.5$. (The model has 'learned' (albeit very badly) to distinguish between labelled and unlabelled examples).

(e) Now since $p(t = 1|x) = \frac{1}{\alpha} p(y = 1|x)$, we can scale up the probability values and predict based on $p(t = 1|x) > 0.5$.

To visualise, the following two plots shows an example of possible sigmoid curve for $p(y = 1|x)$ (left) with its decision boundary marked, and its scaled version $p(t = 1|x)$ (right). As it is clear from the figure, by doing so the decision boundary shifts towards the unlabelled (negative) examples. (This is what should happen, because earlier it predicted all examples as unlabelled (negative).)



```
x_valid, y_valid = util.load_dataset(valid_path, label_col='y',
add_intercept=True)
y_valid_pred = model.predict(x_valid)
alpha = np.mean(y_valid_pred)

t_pred = y_pred / alpha

np.savetxt(pred_path_e, t_pred > 0.5, fmt='%d')
```

However we need to figure out how to draw the boundary of this corrected model.

Normally, the decision boundary is drawn by solving $p(y = 1|x) = 0.5$, which is equivalent to $\theta^T x = 0$.

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$$

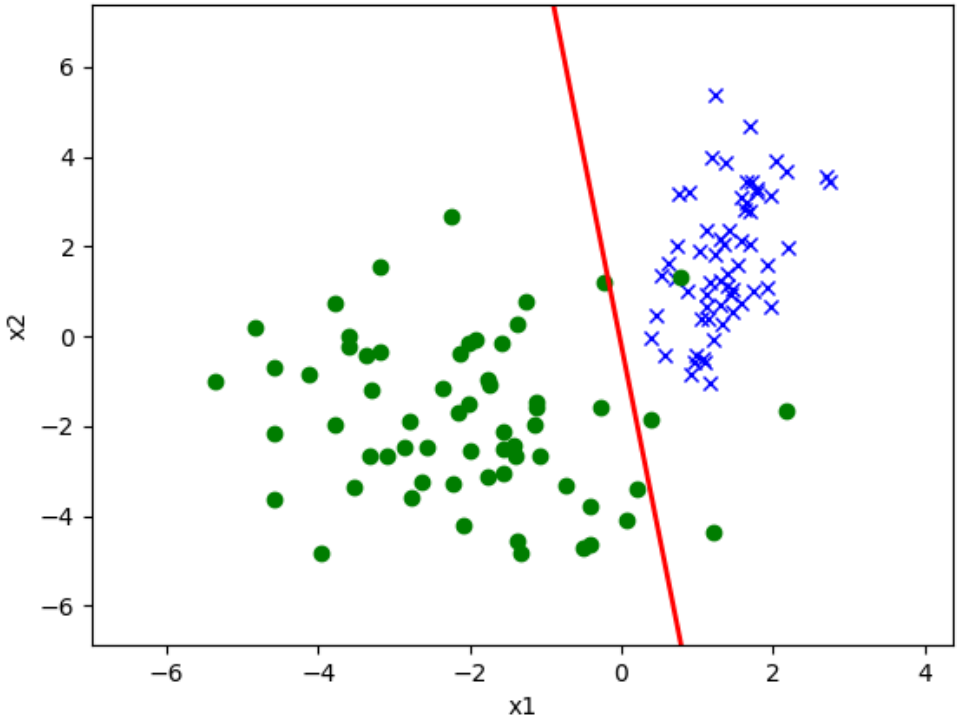
However, here we need $p(t = 1|x) = 0.5$, which means

$$\begin{aligned} p\left(t^{(i)} = 1|x^{(i)}\right) = \frac{1}{\alpha} p\left(y^{(i)} = 1|x^{(i)}\right) &= 0.5 \\ \Rightarrow \frac{1}{1 + \exp\left(-\theta^T x^{(i)}\right)} &= \frac{\alpha}{2} \\ \Rightarrow \exp\left(-\theta^T x^{(i)}\right) &= \frac{2}{\alpha} - 1 \\ \Rightarrow \theta^T x^{(i)} &= -\ln\left(\frac{2}{\alpha} - 1\right) \\ \Rightarrow \theta_0 + \theta_1 x_1 + \theta_2 x_2 &= -\ln\left(\frac{2}{\alpha} - 1\right) \\ \Rightarrow \theta_0 \left(1 + \frac{1}{\theta_0} \ln\left(\frac{2}{\alpha} - 1\right)\right) + \theta_1 x_1 + \theta_2 x_2 &= 0 \end{aligned}$$

The `plot()` function takes in a `correction` argument for this purpose, which is a 'scaling' factor for the constant term.

In this case, the `correction` is $1 + \frac{1}{\theta_0} \ln\left(\frac{2}{\alpha} - 1\right)$.

```
correction = 1 + np.log(2 / alpha - 1) / model.theta[0]
util.plot(x_test, y_test, model.theta,
          pred_path_e.replace('.txt', '.png'), correction=correction)
```




In the above image we are back to representing the positive examples in blue and negative examples in green.

Notice that since our correction only affected the constant term of our equation of the line, the decision boundary here is parallel to the one for $p(y = 1|x)$.

(Remark) In case sorting is required instead of classification, we can directly sort based on $p(y = 1|x)$ (without needing to estimate $\alpha$). In the above example, this is equivalent to sorting based on probability prediction of (d), so examples near the top-right of the x1-x2 space will be sorted towards the beginning.

3. **Poisson Regression**

(a)
$$p(y; \lambda) = \frac{e^{-\lambda}\lambda^y}{y!}$$

The exponential family is characterised by the following PDF:

$$p(y; \eta) = b(y) \exp(\eta^T T(y) - a(\eta))$$

where $y$ is the data, $\eta$ is the natural parameter, $T(y)$ is the sufficient statistic (mostly $T(y) = y$), $b(y)$ is the base measure, and $a(\eta)$ is the log-partition function, playing the role of normalising factor.

The Poisson distribution is

$$\begin{aligned} p(y;\lambda) &= \frac{e^{-\lambda}\lambda^y}{y!} \\ &= \frac{1}{y!}\exp(-\lambda + y\ln\lambda) \end{aligned}$$

It can be written in terms of an exponential distribution, with

$$\begin{aligned} \eta &= \ln\lambda \\ T(y) &= y \\ b(y) &= \frac{1}{y!} \\ a(\eta) &= \lambda = e^{\eta} \end{aligned}$$

(b) If $y|x;\theta \sim \text{ExponentialDistribution}(\eta)$, to model $y|x;\theta$ we can use a GLM, with $\eta = \theta^T x$. We train $h_\theta(x) = E[y|x;\theta] = E[y|\theta^T x]$

To learn, $\max_\theta \log p\big(y^{(i)}, \theta^T x^{(i)}\big)$.

Canonical response function $g$ is the one that gives the distributions mean as a function of the natural parameter $\eta$.

For Poisson's GLM, $E[y] = \mu = \lambda = g(\eta) = e^\eta$.

Then hypothesis (predictor to be learned)

$$h_\theta(x) = \mu = e^{\theta^T x}$$

(c) Stochastic Ascent Rule: (For any GLM)

$$\theta_j := \theta_j + \alpha\frac{\partial}{\partial\theta_j}l(\theta)$$

$$\begin{aligned} \log p(y|x;\theta) = \log\frac{e^{-\lambda}\lambda^y}{y!} &= -\lambda + y\ln\lambda - \sum_{i=1}^{y}\ln i \\ &= -e^{\theta^T x} + y\theta^T x - \ln y! \end{aligned}$$

Then, log likelihood:

$$l(\theta) = \sum_{i=1}^{m}\log p\big(y^{(i)}|x^{(i)};\theta\big) = \sum_{i=1}^{m} -e^{\theta^T x^{(i)}} + y^{(i)}\theta^T x^{(i)} - \ln y^{(i)}!$$

By MLE:

$$\frac{\partial}{\partial\theta_j}l(\theta) = \sum_{i=1}^{m} -x_j^{(i)}e^{\theta^T x^{(i)}} + x_j^{(i)}y^{(i)} = x_j^{(i)}\big(y^{(i)} - e^{\theta^T x^{(i)}}\big)$$

Thus update rule:

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - e^{\theta^T x^{(i)}}\right)x_j^{(i)}$$

(d) **Coding Problem**

Stochastic Gradient Descent:

loop:

for $i = 1$ to $m$

$$\theta := \theta + \alpha\left(y^{(i)} - \exp\left(\theta^T x^{(i)}\right)\right)x^{(i)}$$

Batch Gradient Descent

loop:

$$\theta := \theta + \alpha \sum_{i=1}^{m}\left(y^{(i)} - \exp\left(\theta^T x^{(i)}\right)\right)x^{(i)}$$

$$:= \theta + \alpha \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \begin{bmatrix} y^{(1)} - \exp\left(\theta^T x^{(1)}\right) \\ y^{(2)} - \exp\left(\theta^T x^{(2)}\right) \\ \vdots \\ y^{(m)} - \exp\left(\theta^T x^{(m)}\right) \end{bmatrix}$$

$$:= \theta + \alpha X^T(y - \exp(X\theta))$$

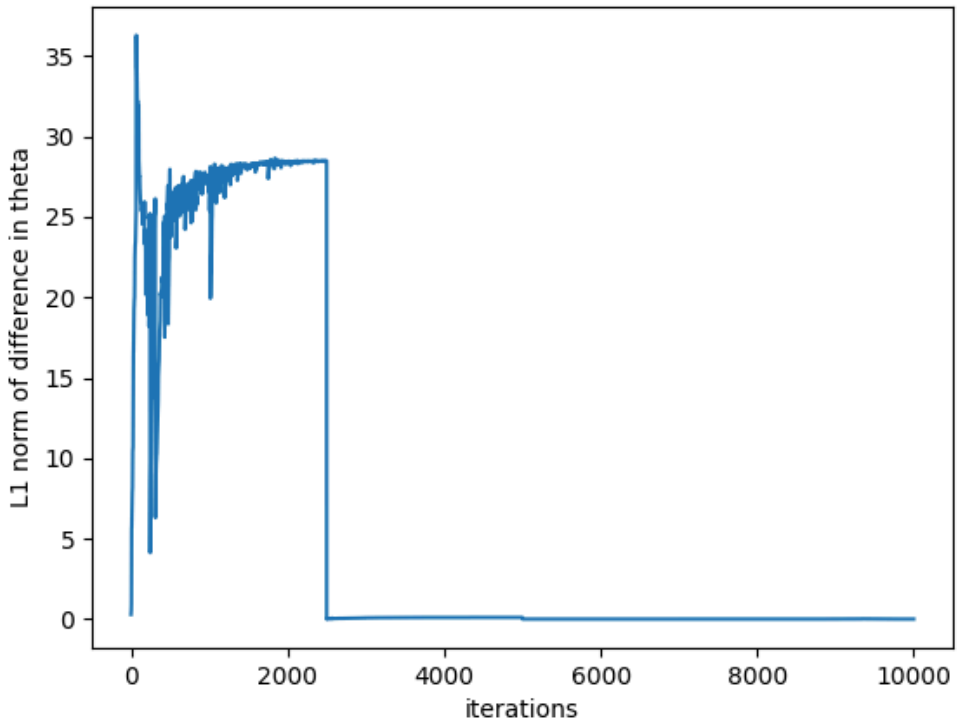
(The sum is a linear combination of $x^{(i)}$ vectors!)

```
class PoissonRegression(LinearModel):
  def fit(self, x, y, stochastic=False):
      m, n = x.shape
      self.theta = np.zeros(n)
      c = 0
      while True:
          c += 1
          prev = np.copy(self.theta)

          # Stochastic gradient ascent
          if stochastic:
              for i in range(m):
                  self.theta += self.step_size *
                      (y[i] - np.exp(x[i] @ self.theta)) * x[i]
          # Batch gradient ascent
          else:
              self.theta += self.step_size *
                  (x.T @ (y - np.exp(x @ self.theta)))

          diff = np.linalg.norm(self.theta - prev, ord=1)
          if diff < self.eps or c >= self.max_iter:
              break

  def predict(self, x):
      return np.exp(x @ self.theta)
```

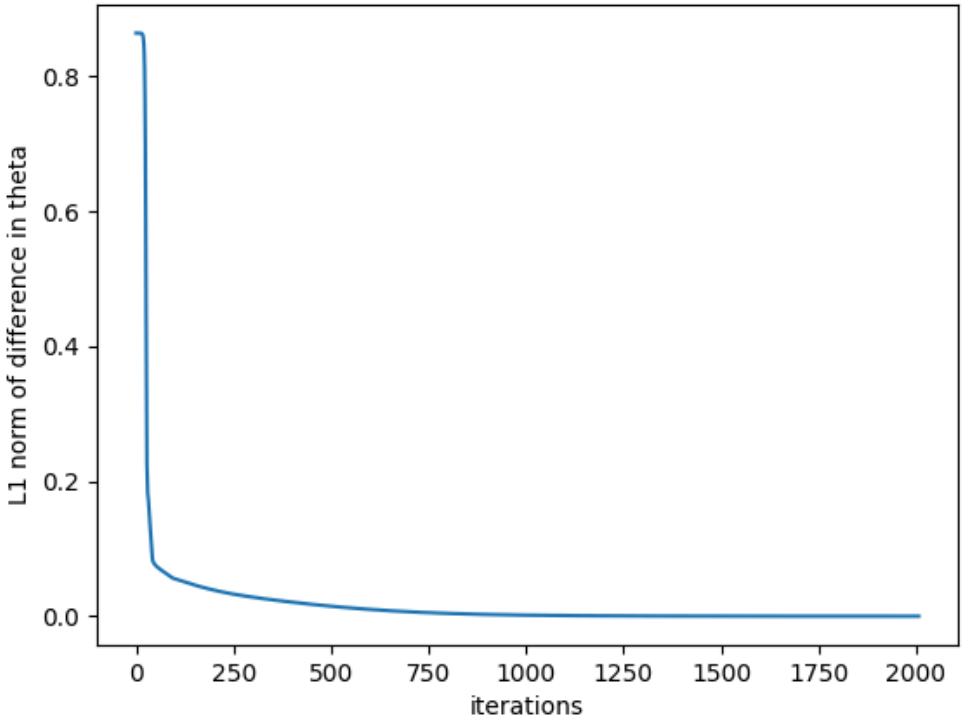
Ran on parameters:

```
Stochastic Gradient Ascent:
  - lr: 5e-8
  - converges in: 4 iterations

Batch Gradient Ascent:
  - lr: 2e-11
  - converges in: 2008 iterations
```

Stochastic Gradient Ascent:




Batch Gradient Ascent:




4. **Convexity of Generalised Linear Models**

To show: NLL loss of GLM is a convex function w.r.t. model parameters

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta)))$$

Simplified:

$$p(y;\eta) = b(y)\exp(\eta y - a(\eta))$$

(a)
$$\mathbb{E}(Y \mid X;\theta) = \mathbb{E}(Y|\eta) = \int y\, p(y;\eta)\,\mathrm{d}y$$

$$\mu = \int y\, b(y)\exp(\eta y - a(\eta))\,\mathrm{d}y$$

$$\Rightarrow \mu e^{a(\eta)} = \int y\, b(y)\exp(\eta y)\,\mathrm{d}y$$

Normalisation Condition:

$$\int p(y;\eta)\,\mathrm{d}y = \int b(y)\frac{e^{\eta y}}{e^{a(\eta)}}\,\mathrm{d}y = 1$$

$$\Rightarrow e^{a(\eta)} = \int b(y)e^{\eta y}\,\mathrm{d}y$$

$$\Rightarrow e^{a(\eta)}a'(\eta) = \int y\, b(y)e^{\eta y}\,\mathrm{d}y$$

On comparison,

$$e^{a\eta}\mu = e^{a\eta}a'(\eta)$$
$$\Rightarrow \mu = a'(\eta)$$

(b) $$\mathrm{Var}(Y|X;\theta) = \mathrm{Var}(Y;\eta) = \int y^2 p(y;\eta)\,\mathrm{d}y - \mu^2$$
$$= \int y^2 b(y)\exp(\eta y - a(\eta))\,\mathrm{d}y - \mu^2$$

From Normalisation Condition, we have

$$a'(\eta) = \int y b(y)\exp(\eta y - a(\eta))\,\mathrm{d}y$$
$$\Rightarrow a''(\eta) = \int y b(y)\exp(\eta y - a(\eta))(y - a'(\eta))\,\mathrm{d}y$$
$$= \int y^2\, b(y)\exp(\eta y - a(\eta))\,\mathrm{d}y - a'(\eta)\int y\, b(y)\exp(\eta y - a(\eta))\,\mathrm{d}y$$
$$= \int y^2\, b(y)\exp(\eta y - a(\eta)\,\mathrm{d}y) - \mu^2$$

On Comparison,

$$\mathrm{Var}(Y|X;\theta) = a''(\eta)$$

(c) $$\mathcal{L}(\theta) = p(y \mid x;\theta) = b(y)\exp(\eta y - a(\eta))$$
$$\Rightarrow \ell(\theta) = -\log b(y) + a(\theta^T x) - \theta^T x y$$
$$\nabla_\theta \ell(\theta) = x\left(a'(\theta^T x) - y\right)$$
$$H = \nabla_\theta\left(x\, a'(\theta^T x)\right) = a''(\theta^T x) x x^T$$

For $H$ to be PSD,

$$z^T H z = z^T a''(\theta^T x) x x^T z = a''(\theta^T x) z^T x x^T z = a''(\theta^T x)(x^T z)^T (x^T z)$$
$$= a''(\theta^T x)(x^T z)^2 \geq 0 \forall z$$

(since $a''(\eta) = \mathrm{Var}(Y \mid X;\theta) \geq 0$)

Thus NLL loss of a GLM is convex.

Given that gradient of NLL is $x(a'(\theta^T x) - y)$, we have gradient descent update rule:

$$\theta := \theta - \alpha\left(a'\left(\theta^T x^{(i)}\right) - y^{(i)}\right)x^{(i)}$$

and prediction function $h_\theta(x) = a'(\theta^T x)$.

Thus, for any GLM, we find the update function to be similar:

$$\theta := \theta + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x^{(i)}$$

5. **Locally weighted linear regression**

(a) $$J(\theta) = \frac{1}{2}\sum_{i=1}^{m} w^{(i)}\big(\theta^T x^{(i)} - y^{(i)}\big)^2$$

i. To show that $J(\theta) = (X\theta - y)^T W (X\theta - y)$ for diagonal $W$.

$$J(\theta) = \frac{1}{2}\big[w^{(1)}\theta^T x^{(1)} - y^{(1)} \;\; w^{(2)}\theta^T x^{(2)} - y^{(2)} \;\; \cdots \;\; w^{(m)}\theta^T x^{(m)} - y^{(m)}\big] \cdot \begin{bmatrix} \theta^T x^{(1)} - y^{(1)} \\ \theta^T x^{(2)} - y^{(2)} \\ \vdots \\ \theta^T x^{(m)} - y^{(m)} \end{bmatrix}$$

$$= \frac{1}{2}\big[\theta^T x^{(1)} - y^{(1)} \;\; \theta^T x^{(2)} - y^{(2)} \;\; \cdots \;\; \theta^T x^{(m)} - y^{(m)}\big] \begin{bmatrix} w^{(1)} & & & \\ & w^{(2)} & & \\ & & \ddots & \\ & & & w^{(m)} \end{bmatrix} \begin{bmatrix} \theta^T x^{(1)} - y^{(1)} \\ \theta^T x^{(2)} - y^{(2)} \\ \vdots \\ \theta^T x^{(m)} - y^{(m)} \end{bmatrix}$$

$$= \frac{1}{2}(X\theta - y)^T W (X\theta - y)$$

where $W = \frac{1}{2}\operatorname{diag}\big(w^{(1)}, w^{(2)}, ..., w^{(m)}\big)$

ii.
$$\begin{aligned} J(\theta) &= \frac{1}{2}(X\theta - y)^T W (X\theta - y) \\ &= \frac{1}{2}\big((X\theta)^T W X\theta - y^T W X\theta - (X\theta)^T W y + y^T W y\big) \\ &= \frac{1}{2}\big(\theta^T X^T W X\theta - y^T W X\theta - \theta^T X^T W y + y^T W y\big) \\ \nabla_\theta J(\theta) &= X^T W X\theta - \frac{1}{2}X^T W^T y - \frac{1}{2}X^T W y \\ &= X^T W X\theta - X^T W y = 0 \\ &\Rightarrow X^T W X\theta = X^T W y \\ &\Rightarrow \theta = \big(X^T W X\big)^{-1} X^T W y \end{aligned}$$

($\nabla_x x^T A x = 2Ax$ for symmetric $A$)

iii. Assume model:

$$p\big(y^{(i)}|x^{(i)};\theta\big) = \frac{1}{\sqrt{2\pi}\sigma^{(i)}}\exp\left(-\frac{\big(y^{(i)} - \theta^T x^{(i)}\big)^2}{2\big(\sigma^{(i)}\big)^2}\right)$$

NLL:

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{m} \frac{\big(y^{(i)} - \theta^T x^{(i)}\big)^2}{2\big(\sigma^{(i)}\big)^2} \\ &= \frac{1}{2}\sum_{i=1}^{m} \frac{1}{\big(\sigma^{(i)}\big)^2}\big(y^{(i)} - \theta^T x^{(i)}\big)^2 \end{aligned}$$

Thus it is equivalent to weighted linear regression with $w^{(i)} = \frac{1}{(\sigma^{(i)})^2}$.

(b) **Coding Problem**

Use

$$w^{(i)} = \exp\left(-\frac{\|x^{(i)} - x\|_2^2}{2\tau^2}\right)$$

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

```
class LocallyWeightedLinearRegression(LinearModel):
    def fit(self, x, y):
        self.x = x
        self.y = y

    def predict_single(self, x):
        m, n = self.x.shape
        W = np.zeros((m, m))
        for i in range(m):
            W[i, i] = np.exp(-np.sum((x - self.x[i])**2)/(2*self.tau**2))
        theta = np.linalg.inv(self.x.T @ W @ self.x)
                    @ self.x.T @ W @ self.y
        y_pred = x @ theta
        return y_pred

    def predict(self, x):
        m, n = x.shape
        y = np.zeros(m)

        for i in range(m):
            y[i] = self.predict_single(x[i])

        return y
```

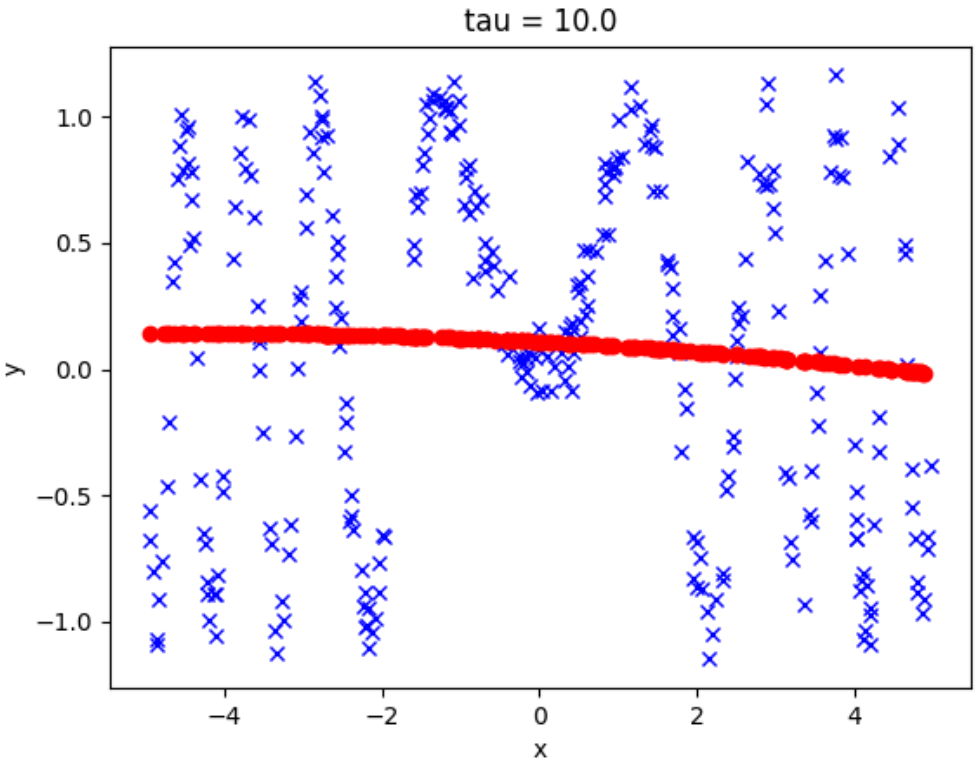
(see plot in (c))
MSE = 0.3305

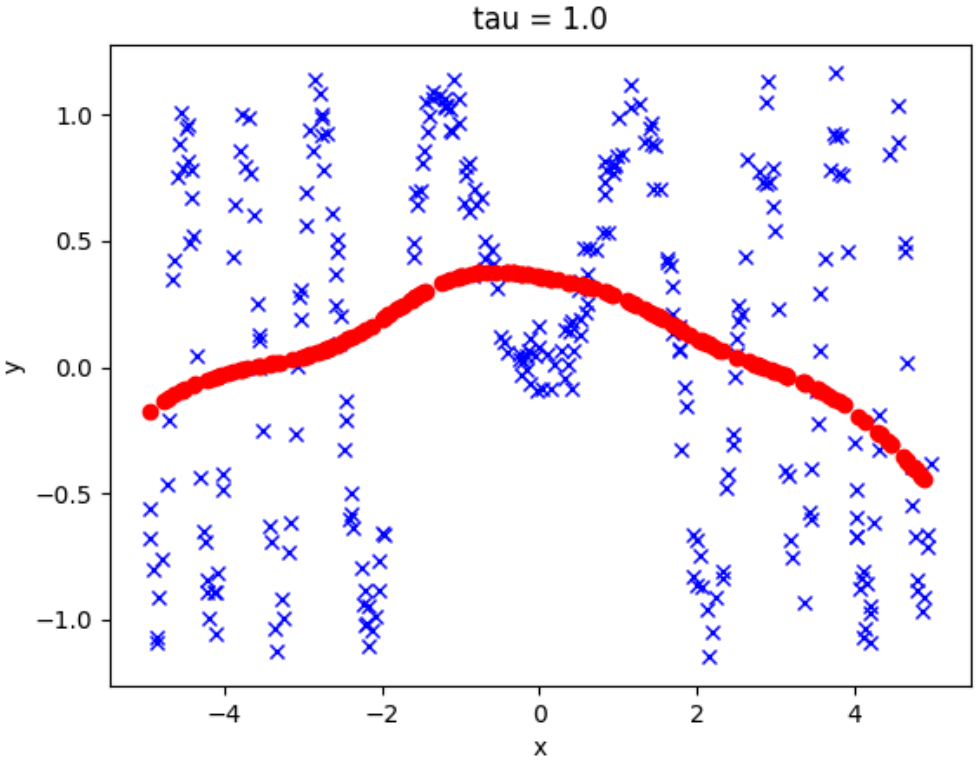
Model seems to be underfitting.

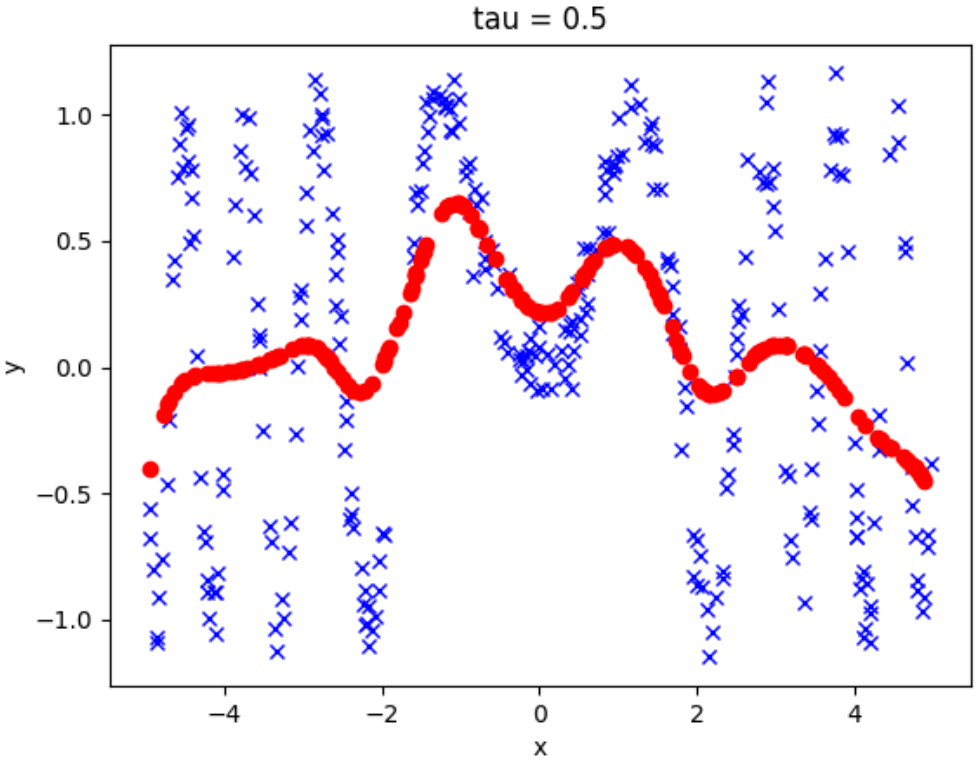
(c)
```
mse_values = []
for tau in tau_values:
    clf = LocallyWeightedLinearRegression(tau)
    clf.fit(x_train, y_train)
    y_pred = clf.predict(x_valid)
    mse = np.mean((y_valid - y_pred)**2)
    mse_values.append(mse)
    plt.figure()
    plt.plot(x_train[:, 1], y_train, 'bx')
    plt.plot(x_valid[:, 1], y_pred, 'ro')
    plt.xlabel('x')
```

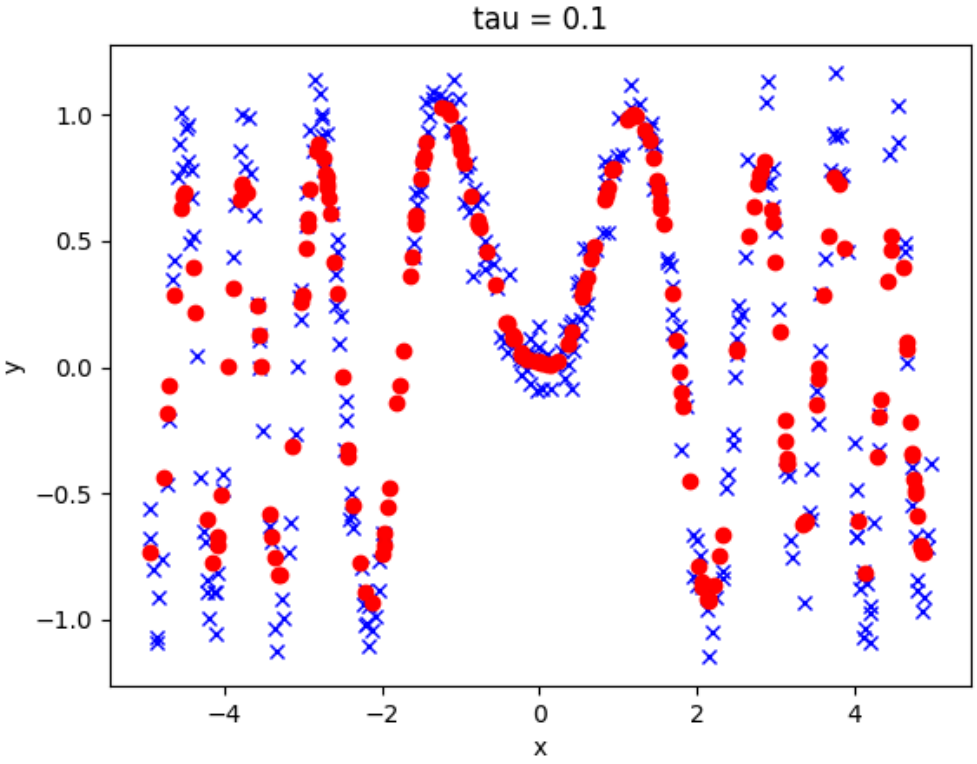
```
    plt.ylabel('y')
    plt.title('tau = ' + str(tau))
    plt.savefig('output/p05c_tau_{}.png'.format(tau))
best_tau = tau_values[np.argmin(mse_values)]
```

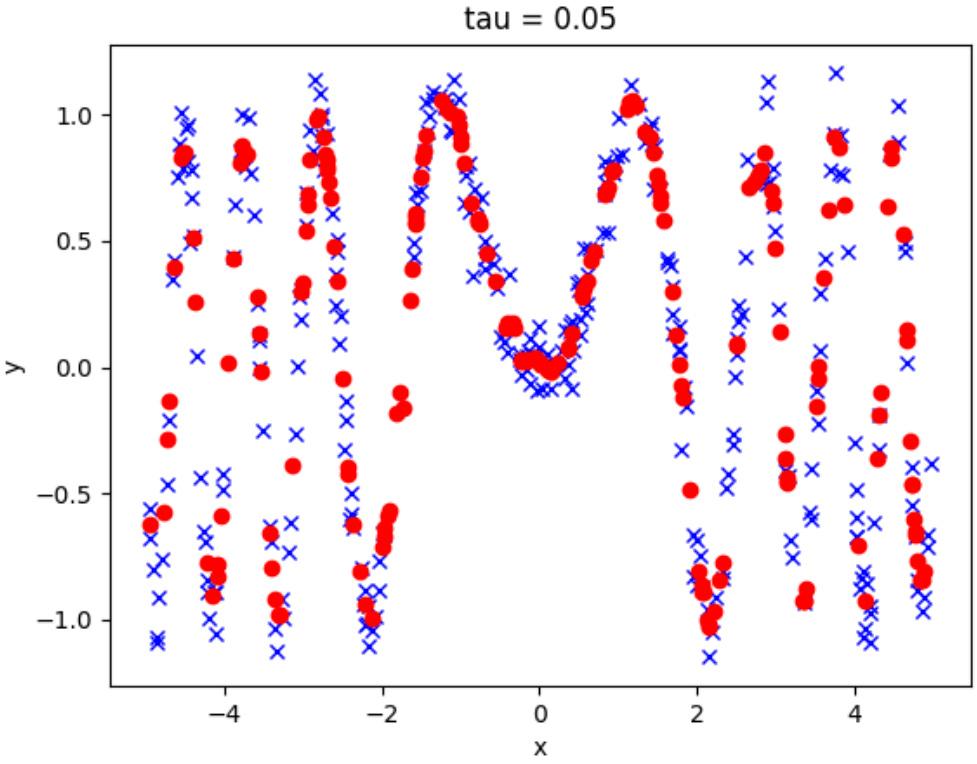
Best $\tau$ value: 0.05, MSE = 0.017 on test set

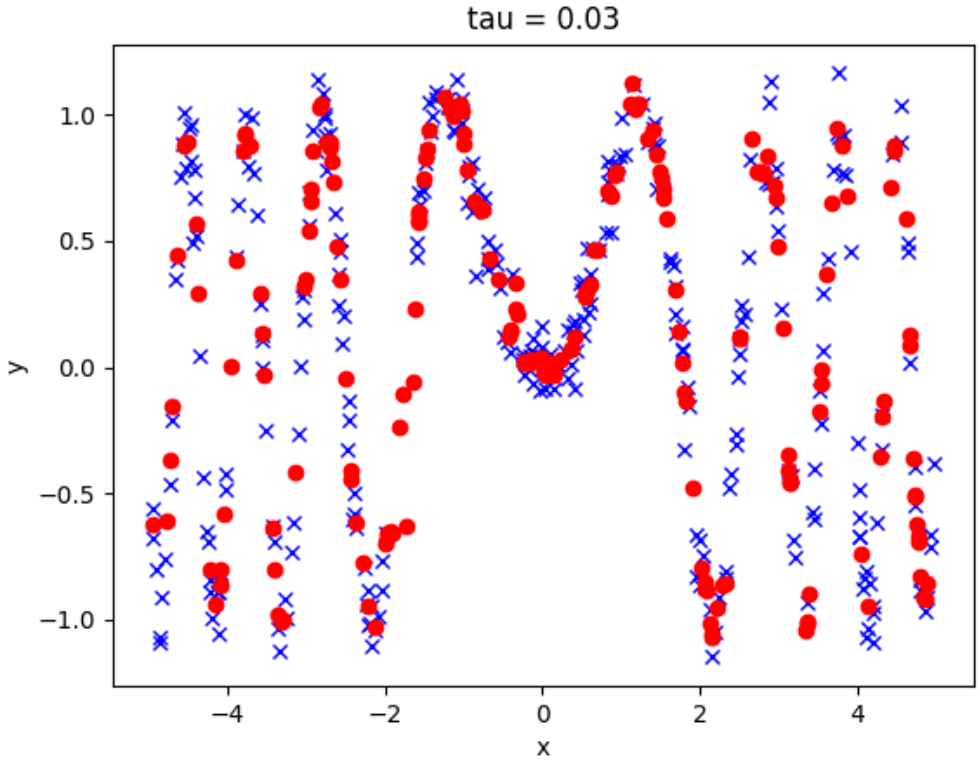



tau = 1.0
1.0
0.5
0.0
−0.5
−1.0
y
−4
−2
0
2
4
x


tau = 0.5
1.0
0.5
0.0
−0.5
−1.0
y
−4
−2
0
2
4
x


tau = 0.1
1.0
0.5
0.0
−0.5
−1.0
y
−4
−2
0
2
4
x


tau = 0.05
1.0
0.5
0.0
−0.5
−1.0
y
−4
−2
0
2
4
x




Note that $\tau$ is the standard deviation of the gaussian for the local weighting. So lower $\tau$ means that weights will be strongly local.

# PSet 1 (Problems from other versions)

1. **Newton's Method for computing least squares** (Public Course, 1)

(a)
$$
\begin{aligned}
J(\theta) &= \frac{1}{2}\sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right)^2 \\
\frac{\partial}{\partial\theta_j}J(\theta) &= \sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right)x_j^{(i)} \\
\nabla_\theta J(\theta) &= \sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right)x^{(i)} \\
&= X^T(X\theta - y) \\
\frac{\partial^2 J(\theta)}{\partial\theta_j\partial\theta_k} &= \sum_{i=1}^{m} x_j^{(i)} x_k^{(i)} \\
H &= X^T X
\end{aligned}
$$

(b) First iteration of Newton's Method:

$$
\begin{aligned}
\theta &:= \theta + H^{-1}\nabla_\theta l(\theta) \\
&:= \theta - \left(X^T X\right)^{-1} X^T(X\theta - y)
\end{aligned}
$$

Assuming initialising with $\theta = 0$

$$
\theta^\star = \left(X^T X\right)^{-1} X^T y
$$

which is the solution to the normal equation.

2. **Naive Bayes** (Public Course, 4)

Input features $x_j \in \{0,1\}$. $x = [x_1 \;\; x_2 \;\; \cdots \;\; x_n]^T$ is the input vector. For each training example, the output $y \in \{0,1\}$.

Model Parameters:

$$
\begin{aligned}
\phi_{j|y=0} &= p\left(x_j = 1 | y = 0\right) \\
\phi_{j|y=1} &= p\left(x_j = 1 | y = 1\right) \\
\phi_y &= p(y = 1)
\end{aligned}
$$

Model joint distribution of $(x, y)$ as

$$
\begin{aligned}
p(y) &= \left(\phi_y\right)^y\left(1 - \phi_y\right)^{1-y} \\
p(x|y=0) &= \prod_{j=1}^{n} p\left(x_j | y = 0\right) &&= \prod_{j=1}^{n}\left(\phi_{j|y=0}\right)^{x_j}\left(1 - \phi_{j|y=0}\right)^{1-x_j} \\
p(x|y=1) &= \prod_{j=1}^{n} p\left(x_j | y = 1\right) &&= \prod_{j=1}^{n}\left(\phi_{j|y=1}\right)^{x_j}\left(1 - \phi_{j|y=1}\right)^{1-x_j}
\end{aligned}
$$

(a) $$l(\varphi) = \sum_{i=1}^{m} \log p\big(x^{(i)}, y^{(i)}; \varphi\big)$$

where $\varphi = \Big\{\phi_y, \phi_{j|y=0}, \phi_{j|y=1}, j = 1, ..., n\Big\}$

$$\begin{aligned} l(\varphi) &= \sum_{i=1}^{m} \log p\big(x^{(i)}, y^{(i)}; \varphi\big) \\ &= \sum_{i=1}^{m} \log p\Big(x^{(i)} | y^{(i)}; \phi_{j|y^{(i)}}\Big) p\big(y^{(i)}; \phi_y\big) \\ &= \sum_{i=1}^{m} \left( \sum_{j=1}^{n} \Big(x_j \log \phi_{j|y^{(i)}} + \big(1 - x_j\big) \log\Big(1 - \phi_{j|y^{(i)}}\Big)\Big) + y^{(i)} \log\big(\phi_y\big) + \big(1 - y^{(i)}\big) \log\big(1 - \phi_y\big) \right) \end{aligned}$$

(b) MLE:

$$\frac{\partial}{\partial \phi_y} l(\varphi) = \sum_{i=1}^{m} \frac{y^{(i)}}{\phi_y} - \frac{1 - y^{(i)}}{1 - \phi_y} = 0$$

$$\Rightarrow \frac{\phi_y}{1 - \phi_y} = \frac{\sum_{i=1}^{m} y^{(i)}}{\sum_{i=1}^{m} 1 - y^{(i)}}$$

$$\Rightarrow \phi_y = \frac{\sum_{i=1}^{m} 1\big\{y^{(i)} = 1\big\}}{m}$$

$$\frac{\partial}{\partial \phi_{j|y^{(i)}}} l(\varphi) = \sum_{i=1}^{m} \frac{x_j}{\phi_{j|y^{(i)}}} - \frac{1 - x_j}{1 - \phi_{j|y^{(i)}}} = 0$$

$$\Rightarrow \sum_{i=1}^{m} \frac{x_j}{\phi_{j|y^{(i)}}} - \frac{1 - x_j}{1 - \phi_{j|y^{(i)}}} = 0$$

$$\Rightarrow \sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\} \left( \frac{x_j}{\phi_{j|y=0}} - \frac{1 - x_j}{1 - \phi_{j|y=0}} \right) + 1\big\{y^{(i)} = 1\big\} \left( \frac{x_j}{\phi_{j|y=1}} - \frac{1 - x_j}{1 - \phi_{j|y=1}} \right) = 0$$

$$\Rightarrow \frac{\phi_{j|y=0}}{1 - \phi_{j|y=0}} = \frac{\sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\} x_j}{\sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\}\big(1 - x_j\big)}$$

$$\Rightarrow \phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\Big\{y^{(i)} = 0, x_j^{(i)} = 1\Big\}}{\sum_{i=1}^{m} 1\big\{y^{(i)} = 0\big\}}$$

$$\Rightarrow \phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\Big\{y^{(i)} = 1, x_j^{(i)} = 1\Big\}}{\sum_{i=1}^{m} 1\big\{y^{(i)} = 1\big\}}$$

(c) To show: $p(y = 1|x) \geq 0.5$ is equivalent to $\theta^T x \geq -\theta_0$ for some $\theta$ and $\theta_0$

$$p(y=1|x) = \frac{p(y=1,x)}{p(x)} = \frac{p(x|y=1)p(y=1)}{p(x)}$$

$$= \frac{\prod_{j=1}^{n} \left(\phi_{j|y=1}\right)^{x_j} \left(1-\phi_{j|y=1}\right)^{1-x_j} \phi_y}{\prod_{j=1}^{n} \left(\phi_{j|y=1}\right)^{x_j} \left(1-\phi_{j|y=1}\right)^{1-x_j} \phi_y + \prod_{j=1}^{n} \left(\phi_{j|y=0}\right)^{x_j} \left(1-\phi_{j|y=0}\right)^{1-x_j} (1-\phi_y)} \geq \frac{1}{2}$$

$$\Rightarrow \prod_{j=1}^{n} \left(\phi_{j|y=1}\right)^{x_j} \left(1-\phi_{j|y=1}\right)^{1-x_j} \phi_y \geq \prod_{j=1}^{n} \left(\phi_{j|y=0}\right)^{x_j} \left(1-\phi_{j|y=0}\right)^{1-x_j} (1-\phi_y)$$

Take log on both sides. (log is a monotonically increasing function)

$$\log \phi_y + \sum_{j=1}^{n} x_j \log \phi_{j|y=1} + (1-x_j) \log\left(1-\phi_{j|y=1}\right)$$

$$= \log(1-\phi_y) + \sum_{j=1}^{n} x_j \log \phi_{j|y=0} + (1-x_j) \log\left(1-\phi_{j|y=0}\right)$$

$$\Rightarrow \sum_{j=1}^{n} x_j \left( \log \frac{\phi_{j|y=1}}{1-\phi_{j|y=1}} \frac{1-\phi_{j|y=0}}{\phi_{j|y=0}} \right) + \log\left( \frac{1-\phi_{j|y=1}}{1-\phi_{j|y=0}} \right) \geq -\log\left( \frac{\phi_y}{1-\phi_y} \right)$$

This is same as $\theta^T x + \theta_0 \geq 0$ with

$$\theta_j = \log \frac{\phi_{j|y=1}}{1-\phi_{j|y=1}} \frac{1-\phi_{j|y=0}}{\phi_{j|y=0}}$$

$$\theta_0 = \sum_{j=1}^{m} \log\left( \frac{1-\phi_{j|y=1}}{1-\phi_{j|y=0}} \right) + \log\left( \frac{\phi_y}{1-\phi_y} \right)$$

# Gradient Descent

Given a scalar valued function $f(x, y) = x^2 - 3y + y^2$, $\nabla f$ is a vector field in $\mathbb{R}^2$.

$$\nabla f = \frac{\partial}{\partial x} f(x, y)\hat{\imath} + \frac{\partial}{\partial y} f(x, y)\hat{\jmath}$$

$$\Rightarrow \nabla f(x, y)\Big|_{(0,0)} = -3\hat{\jmath}$$

In machine language contexts, we have two variables: $n$: number of features, and $m$: number of training examples.

The hypothesis (prediction) function $h_\theta(x)$ takes as input $x \in \mathbb{R}^n$ and returns the guessed output.

$\theta$ are a set of parameters (weights) which control the model's outputs. These are the "turnable dials" which the machine "learns". Number of tunable parameters is $\sim$ number of features, as each parameter is a weight for the corresponding feature.

We have $m$ such training examples: $x^{(1)}, x^{(2)}, ..., x^{(m)}$. Let the corresponding predictions be $y^{(i)}$. Note that depending on the situation $y^{(i)}$ may be a continuous real value, or a specific value from a discrete set.

In supervised learning, the Cost Function ($J(\theta) : \mathbb{R}^n \to \mathbb{R}$) quantifies the difference between the predicted output of the model and the true output. The goal of the training process is to minimise the value of the cost function by tuning the inputs.

Usually, the cost function has terms indicating the error for each training example.

To update our $\theta$s, we notice that if we think of the $\theta$s as input to $J$, then in the $\theta$-space, we want to reach a minima. The direction of steepest descent (in $\theta$-input space) is given by $-\nabla J(\theta)$

Updation function:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

---

# Exponential Family

PDF can be written in the form

$$P(y; \eta) = b(y) \exp(\eta^T T(y) - a(\eta))$$

where $y$ is the data, $\eta$ is the natural parameter, $T(y)$ is the sufficient statistic (mostly $T(y) = y$), $b(y)$ is the base measure, and $a(\eta)$ is the log-partition function.

Canonical Response $g(\eta)$. For linear regression, it is $g(\eta) = \eta$, for logistic regression it is the sigmoid funciton.