1 General properties

By this point, we’ve seen many examples of exponential families as well as some algorithms related to them:

1. maximum likelihood estimation $\Lambda^*$ via Newton-Raphson, constrained to remain in $\mathcal{D}(\Lambda)$;
2. constrained MLE: the Lagrange multiplier method;
3. stochastic approximation: when the $\Lambda$ is too complicated to compute;
4. Gibbs sampler: to simulate from $\mathbb{P}_\eta$;
5. pseudo-likelihood.

We now remind ourselves of a few general properties before considering the most popular class of exponential families: generalized linear models.

1.1 Likelihoods, scores, and Fisher information

The definitions introduced for one-parameter families are readily generalized to the multiparameter situation. Indeed, we have already been using some of them. Below, we assume that we have sampled $Y_i \overset{IID}{\sim} \mathbb{P}_\eta, 1 \leq i \leq n$.

1. The log-likelihood is
$$\ell(\eta; y) = \ell(\eta) = n[\eta^T \bar{y} - \Lambda(\eta)].$$

2. The score function for $\eta$ is
$$\nabla \ell(\eta) = n(\bar{y} - \mu).$$

3. The Fisher information for $\eta$, denoted $i^{(n)}_\eta$, is
$$i^{(n)}_\eta \equiv \mathbb{E}_\eta \left( \nabla \ell_\eta \nabla \ell_\eta^T \right) = nV_\eta = V^{(n)}_\eta = -\nabla^2 \ell_\eta.$$

4. The score function for $\mu$ is
$$\frac{\partial}{\partial \mu} \ell_\eta(y) = \left( \frac{V_\eta}{n} \right)^{-1}(\bar{y} - \mu).$$

5. The Fisher information for $\mu$, denoted $i^{(n)}_\eta(\mu)$, is
$$i^{(n)}_\eta(\mu) = V^{-1}_\eta i^{(n)}_\eta V^{-1}_\eta = nV^{-1}_\eta.$$

By differentiating the score function for $\eta$, we see that $\ell_\eta(y)$ is a concave function of $\eta$, since $\ddot{\ell}_\eta = -nV_\eta \leq 0$. In other words, the density
$$\frac{d\mathbb{P}_\eta}{dm}$$

is log-concave.
1.2 Cramer-Rao lower bound

For real-valued $\zeta = h(\eta)$, the Cramer-Rao lower bound for any unbiased estimator $\bar{\zeta}$ is

$$\text{Var}_\eta(\bar{\zeta}) \geq \frac{\nabla h_{\eta}^T V_{\eta}^{-1} \nabla h_{\eta}}{n} = \frac{\nabla h_{\eta}^T (\mathcal{i}_{\eta}^{(n)})^{-1} \nabla h_{\eta}}{n},$$

For $\zeta = H(\mu)$, the bound takes the form

$$\text{Var}_\eta(\bar{\zeta}) \geq \nabla H_{\mu}^T [\mathcal{i}_{\eta}^{(n)}(\mu)]^{-1} \nabla H_{\mu}.$$  

Notice that $\nabla H$

For the $\mu$ itself, $\hat{\mu} = \bar{y}$ is unbiased. We also see that it achieves the CRLB:

$$\text{Var}_\eta(\hat{\mu}) \overset{\text{CRLB}}{=} \frac{V_{\eta}}{n} = \mathcal{i}_{\eta}^{(n)}(\mu)^{-1}. \quad (1)$$

1.2.1 Distribution of the MLE

The CLT based on repeated sampling asserts that

$$\hat{\mu} \approx N(\mu, V_{\eta}/n).$$

While the mean and variance are correct, the approximation above is in the distribution, i.e. the distributional sense.

Applying the delta rule yields

$$\hat{\eta} \approx N(\eta, V_{\eta}^{-1}/n).$$

In this case, the mean and variance are not exact, nor is the distribution exactly normal.

1.2.2 Skewness and kurtosis

For multiparameter families, the mean is a $p$-vector, the covariance a $p \times p$ matrix. These are, of course, just derivatives of $\Lambda$. Higher order derivatives are tensors.

For skewness, there is a $p \times p \times p$ tensor, or multidimensional array that takes three vectors as argument. That is,

$$\nabla^3 \Lambda_{\eta}(U,V,W) \in \mathbb{R}.$$  

Formally, the power series for $\Lambda$ can be written as

$$\Lambda(\eta + \Delta) = \sum_{j=0}^{\infty} \frac{1}{j!} \nabla^j \Lambda_{\eta}(\otimes^j \Delta).$$

1.2.3 Exercise: skewness of a one-parameter subfamily

Suppose we consider the one-parameter subfamily

$$\{\eta = \eta_0 + s \cdot v, s \in \mathbb{R}, \eta \in \mathcal{D}(\Lambda)\}.$$  

1. What is the sufficient statistic of this one-parameter subfamily (denote the sufficient statistic of the original family by $t(x)$).

2. Express the skewness and kurtosis of this sufficient statistic in terms of the derivatives of $\Lambda$.  

2
1.3 Stein’s least favorable family

Suppose we want to estimate \( \zeta = h(\eta) = h(\nabla^* \mu) = s(\mu) \), a real-valued function of \( \eta \).

Stein’s least favorable family is a one-parameter subfamily of the original family, determined by some hypothesized true value, say, \( \eta_0 \) and the function \( h \) (or, equivalently, \( s \)).

The family is

\[
\{ \eta : \eta = \eta_0 + \theta \cdot V_{\eta_0}^{-1} \nabla h(\eta_0), \theta \in \mathbb{R}, \eta \in \mathcal{D}(\Lambda) \}.
\]

1.3.1 Exercise: Stein’s least favorable family

1. Show that the CRLB for the above one-parameter family is the same as the CRLB in the full \( p \)-parameter family.

1.3.2 Deviance

Everything we saw previously about deviance remains true in the multiparameter case.

\[
D(\eta_1; \eta_2) = D(\mathbb{P}_{\eta_1}; \mathbb{P}_{\eta_2}) = 2 \left[ \Lambda(\eta_2) - \Lambda(\eta_1) - \nabla \Lambda(\eta_1)^T(\eta_2 - \eta_1) \right] = 2 \left[ \Lambda^*(\mu_1) - \Lambda^*(\mu_2) - \nabla \Lambda^*(\mu_2)^T(\mu_1 - \mu_2) \right] = \tilde{D}(\mu_1; \mu_2).
\]

with \( \mu_i = \nabla \Lambda(\eta_i) \).

The deviance and Fisher information are both related to Taylor series expansions of \( \Lambda \).

\[
D(\eta_1; \eta_2) = \nabla^2 \Lambda_{\eta_1}(\eta_2 - \eta_1, \eta_2 - \eta_1) + O(\|\eta_2 - \eta_1\|^3) = (\eta_2 - \eta_1)^T \tilde{\mathcal{I}}_{\eta_1}^{(n)}(\eta_2 - \eta_1) + O(\|\eta_2 - \eta_1\|^3).
\]

where the constant in the remainder can be expressed in terms of a Lipschitz constant of \( \nabla^3 \Lambda \) near \( \eta_1 \).

1.3.3 Exercise: monotone mapping

Show that, in general

\[
(\eta_2 - \eta_1)^T(\mu_2 - \mu_1) = \frac{1}{2} [D(\eta_1; \eta_2) + D(\eta_2; \eta_1)] \geq 0
\]

with \( \mu_i = \nabla \Lambda(\eta_i) \).

This demonstrates that the relationship between is globally monotone, and is an example of the general concept of a monotone mapping, the canonical example being the (sub)gradient of a convex function.

2 Generalized linear models

A generalized linear model (GLM) is a model the conditional distribution

\[
\mathcal{Y} | \mathcal{X}, \mathcal{X} \in \mathbb{R}^p, \mathcal{Y} \in \mathbb{R}.
\]
The distribution of \( Y \mid X \) is assumed to be in some one parameter exponential family \( \mathbb{P}_\eta \) with sufficient statistic \( Y \).

(I am using \( \mathcal{X}, \mathcal{Y} \) to ensure there is no confusion with the observed data \( Y \in \mathbb{R}^n, X \in \mathbb{R}^{n \times p} \) below)

Therefore,

\[
\mathcal{Y} \mid \mathcal{X} \sim \mathbb{P}_{\eta(X)}.
\]

The linear in GLM refers to the assumption that

\[
\eta(x) = x^T \beta, \beta \in \mathbb{R}^p.
\]

The model refers to the assumption that we observe

\[
\mathcal{Y}_i \mid \mathcal{X}_i = x_i \sim \mathbb{P}_{\eta(x_i)}, \quad 1 \leq i \leq n,
\]

where the independence here is a statement on the distribution of the vector \( Y = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \) given the matrix

\[
X_{n \times p} = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix}
\]

The parameters of the GLM are \( \beta \), and the loglikelihood is

\[
\ell(\beta) = \sum_{i=1}^{n} \eta_i(\beta) \cdot y_i - \Lambda(\eta_i(\beta))
\]

\[
= \sum_{i=1}^{n} (x_i^T \beta) \cdot y_i - \Lambda(x_i^T \beta).
\]

We could write this in matrix form as

\[
\ell(\beta) = (X \beta)^T Y - \Lambda(n)(X \beta) = \beta^T (X^T Y) - \Lambda(n)(X \beta)
\]

where

\[
\Lambda(n)(\eta) = \sum_{i=1}^{n} \Lambda(\eta_i), \eta \in \mathbb{R}^n.
\]

Written in this way, we see that the GLM is actually a \( p \)-parameter exponential family with sufficient statistic \( X^T Y \) and CGF \( \Lambda(\beta) = \Lambda(n)(X \beta) \).

Let’s untangle this statement a little more carefully.

1. If it is an exponential family, it should be a collection of distributions. In this case it is a collection of conditional distributions for \( Y \mid X \). These are distributions on \( \mathbb{R}^n \).

2. If it is an exponential family, it should have a reference measure. In this case, the reference measure is the product

\[
\prod_{i=1}^{n} m(dy_i)
\]

where \( m \) is the reference measure of \( \mathbb{P}_\eta \).
3. If it is an exponential family, it should have a natural parameter and a sufficient statistic. In this case, the natural parameter is $\beta$ and the sufficient statistic is $X^T Y$.

4. The fact that $\tilde{A}(\beta) = \Lambda^{(n)}(X\beta)$ follows directly from the likelihood and the modelling assumption that $\eta_i = x_i^T \beta$.

Strictly speaking, this GLM is one in which we have used the \textit{canonical link}, i.e.

$$\eta_i = x_i^T \beta.$$ 

A model with a non-canonical link has

$$\eta_i = F(x_i^T \beta).$$

We will see examples shortly.

\textbf{2.0.4 Exercise: link functions}

Suppose we use a non-canonical link function. Is the collection of distributions $Y|X$ an exponential family with natural parameter $\beta$?

\textbf{2.0.5 Exercise: GLM computations}

1. Show that

$$\mathbb{E}_\beta(X^T Y) = X^T \nabla \Lambda^{(n)}_{X\beta}.$$ 

2. Show that

$$\text{Var}_\beta(X^T Y) = X^T \nabla^2 \Lambda^{(n)}_{X\beta} X = X^T \text{diag}(\tilde{A}(x_i^T \beta)) X.$$ 

\textbf{2.0.6 Score equation, Fisher information for GLM}

The score equation for $\beta$ is

$$\nabla \ell_\beta = X^T (Y - \nabla \Lambda^{(n)}_{X\beta}) = X^T (Y - \mu(\beta))$$

where

$$\mu(\beta) = \nabla \Lambda^{(n)}_{X\beta} \in \mathbb{R}^n.$$ 

The Fisher information for $\beta$ is

$$-\nabla^2 \ell_\beta = X^T \text{diag}(\tilde{A}(x_i^T \beta)) X.$$ 

This diagonal matrix can be thought of as assigning a weight to each case, so we define

$$W_\beta = \text{diag}(\tilde{A}(x_i^T \beta))$$

and the Fisher information has the form

$$-\nabla^2 \ell_\beta = i^{(n)}_\beta = X^T W_\beta X.$$
2.0.7 Asymptotic distribution of MLE

The general asymptotic picture for exponential families yields

\[ \hat{\beta} \approx N(\beta, (i_\beta(n))^{-1}) = N(\beta, (X^TW_\beta X)^{-1}). \]

Above, both the mean and variance are approximate, as is the distribution itself.

The asymptotics above are, strictly speaking, for \( p \) fixed and \( n \to \infty \). After estimation of the MLE, this distribution is usually approximated as

\[ N(\beta, X^TW_\beta X). \]

2.1 Estimation

Most generalized linear models are such that \( \tilde{\Lambda}(\beta) = \mathbb{R}^p \) so unconstrained Newton-Raphson can be used.

The algorithm looks like

\[
\hat{\beta}^{(k+1)} = \hat{\beta}^{(k)} - \nabla^2 \tilde{\Lambda} \left( \hat{\beta}^{(k)} \right)^{-1} \left[ \nabla \tilde{\Lambda} \left( \hat{\beta}^{(k)} \right) - X^T Y \right] \\
= \hat{\beta}^{(k)} - (X^T W_{\hat{\beta}^{(k)}X})^{-1} \left[ X^T \left( \mu(\hat{\beta}^{(k)}) - Y \right) \right]
\]

2.1.1 MLE map

So far, for all of our exponential families, we have written the MLE map in terms of the Fenchel-Legendre transform of a CGF. The GLM should be no different.

The mean value space is \( \nabla \tilde{\Lambda}(\mathbb{R}^p) \subseteq \mathbb{R}^p \) (I am assuming the \( \mathcal{D}(\Lambda) = \mathbb{R} \) here), and the MLE map should be a map from \( M \), the convex hull of \( \nabla \tilde{\Lambda}(\mathbb{R}^p) \) back to \( \mathbb{R}^p \).

To find the Fenchel-Legendre transform, we would have to find

\[ \tilde{\Lambda}^*(\xi) = \sup_{\beta} \xi^T \beta - \tilde{\Lambda}(\beta). \]

Or, we would solve

\[ \min_{\beta} \tilde{\Lambda}(\beta) - \xi^T \beta. \]

In the GLM case, it is somewhat easier to use a dual function rather than the \( \tilde{\Lambda}^* \) itself, because we have an implicit constraint, namely

\[ \eta_i = x_i^T \beta. \]

Our problem could then be stated formally as

\[
\min_{\eta, \beta : \eta = X\beta} \Lambda^{(n)}(\eta) - \eta^T Y \equiv \min_{\eta, \beta : \eta = X\beta} \tilde{\Lambda}(\beta) - (X\beta)^T Y
\]

Above, we have used

\[ \tilde{\Lambda}(\beta) = \Lambda^{(n)}(X\beta) = \sum_{i=1}^{n} \Lambda(x_i^T \beta). \]
As we did in the one parameter case, let’s form the Lagrangian
\[
L(\eta, \beta; r) = \eta^T Y - \Lambda^{(n)}(\eta) + r^T (X\beta - \eta)
\]
\[
= \sum_{i=1}^{n} [\eta_i y_i - \Lambda(\eta_i) + r_i (x_i^T \beta - \eta_i)]
\]
\[
= L_1(\eta; r) + L_2(\beta; r)
\]
where we have grouped together terms involving \(\eta\) into \(L_1(\eta; r)\) and those involving \(\beta\) into \(L_2(\beta; r)\).

Specifically,
\[
L_1(\eta; r) = \eta^T (Y - r) - \Lambda^{(n)}(\eta)
\]
\[
L_2(\beta; r) = r^T X\beta
\]

Minimizing \(L_1(\eta; r)\) over \(\eta\) yields
\[
- \sum_{i=1}^{n} \Lambda^*(y_i - r_i)
\]
while minimizing \(L_2(\beta; r)\) over \(\beta\) yields a constraint
\[
\sum_{i=1}^{n} r_i x_i^T = 0.
\]
Or, \(r \in \text{null}(X^T)\).

**Dual problem**  This is the general dual function
\[
G(r) = - \inf_{\eta} L_1(\eta; r) - \inf_{\beta} L_2(\beta; r)
\]
and dual problem
\[
\text{minimize } G(r) \equiv \text{minimize } \sum_{i=1}^{n} \Lambda^*(y_i - r_i).
\]

Having found \(\hat{r}\), the same calculation we saw for the Lagrange multipliers in the one-parameter case tell us that
\[
\hat{\eta}_i = \hat{\Lambda}^*(y_i - \hat{r}_i)
\]
\[
= X\hat{\beta}_i.
\]
In vector form, this reads as
\[
\hat{\eta} = \nabla \left(\Lambda^{(n)}\right)^* (Y - \hat{r}).
\]
This yields
\[
\hat{\beta} = X^\dagger \left[\left(\nabla \Lambda^{(n)}\right)^* (Y - \hat{r})\right].
\]
where \(X^\dagger\) is the pseudoinverse with
\[
X^\dagger = (X^T X)^{-1} X^T
\]
when \(X^T X\) is non-singular.
2.1.2 Exercise: variables in the dual problem

The notation $r$ for the variables in the dual problem suggest residual.

1. Show that

$$\hat{r} = Y - \nabla \Lambda^{(n)}(X\hat{\beta}) = Y - \hat{\mu}.$$ 

2.1.3 Exercise: influence function

This exercise computes the influence function of $\hat{\beta}$, i.e.

$$\frac{\partial \hat{\beta}}{\partial y}.$$ 

1. Show that the influence function satisfies

$$X^T W_\beta X \frac{\partial \hat{\beta}}{\partial y} = X^T.$$ 

2. Suppose $n \geq p$ and $X^T W_\beta X$ is invertible. Conclude that

$$\frac{\partial \hat{\beta}}{\partial y} = \left(X^T W_\beta X\right)^{-1} X^T.$$ 

2.1.4 Projection picture

The GLM is a $p$-parameter subfamily of the $n$-parameter family

$$\prod_{i=1}^{n} \mathbb{P}_{\eta_i}.$$ 

When $p > n$, it can even be the same family in the sense that the map

$$\beta \mapsto X\beta$$

covers all of $\mathbb{R}^n$.

In this case, the MLE is not generally unique. For $p \leq n$, the set

$$\mathcal{M}_X = \left\{ \nabla \Lambda^{(n)}(X\beta), \beta \in \mathbb{R}^p \right\} = \{\mu(\beta) : \beta \in \mathbb{R}^p\}$$

describes a curved subset of $\mathbb{R}^n$.

The score equation for $\beta$ is

$$X^T (Y - \mu(\hat{\beta})) = 0.$$ 

Brad expresses this by saying that the MLE projects $Y$ onto $\mathcal{M}_X$ “orthogonal” to the columns of $X$.

This same score equation almost arises when we consider the least squares projection of $Y$ onto $\mathcal{M}_X$. That is, if we try to solve the following problem

$$\min_{\mu \in \mathcal{M}_X} ||Y - \mu||^2_2.$$ 

Then, we see that critical points in $\mathcal{M}_X$, when parameterizing the problem by $\beta$ satisfy

$$Y - \mu(\hat{\beta}) \perp T_{\mu(\hat{\beta})}\mathcal{M}_X$$

where the last expression is the tangent space to the curved surface $\mathcal{M}_X$ at $\mu = \mu(\hat{\beta})$. 

2.1.5 Exercise: tangent space computation

1. Show that the tangent space at \( \mu(\hat{\beta}) \) is made up of the span of the vectors

\[
\left( x_{1j} \Lambda(x_1^T \beta), \ldots, x_{nj} \Lambda(x_n^T \beta) \right), \quad 1 \leq j \leq p.
\]

2. Write this in matrix form.

3. Use this to fully write out the stationary condition for

\[
\minimize_{\mu \in M_X} \| Y - \mu \|^2_2.
\]

Is this the same as the MLE stationarity equation?

2.2 Example: using Poisson regression for density estimation

In this example, we will use Poisson regression to fit a density to the \( t \)-statistics we saw in our application of Tweedie’s formula. Recall, we had \( N = 6033 \) \( t \)-statistics for comparing a healthy population and to a cancerous one.

Our model will be that the density has the form

\[
f_\beta(x) = \exp \left( \sum_{j=0}^{7} \beta_j x^j - \Lambda(\beta) \right).
\]

This is an 8-parameter exponential family of distributions on \( \mathbb{R} \) with Lebesgue as reference measure.

The basic approach will be to use the Poisson trick we saw in the conditioning section.

1. We discretize \( \mathbb{R} \) into \( K = 50 \) equal sized bins of size \( \Delta \) : \( B_i = (L_i, U_i] \), \( 1 \leq i \leq K \) and define

\[
Y_i = \# \{ Z_j : Z_j \in B_i \}.
\]

2. We model \( Y_i \sim \text{Poisson}(\mu_i) \) where

\[
\mu_i = N \cdot \int_{(L_i, U_i]} f_\beta(x) \, dx \approx e^\alpha \exp \left( \sum_{j=1}^{7} \beta_j M_i^j \right) = h(M_i)
\]

where \( N \) is the number of observed \( Z \)'s and \( M_i = (L_i + U_i)/2 \) is the midpoint of the interval.

3. We fit a Poisson GLM to the \( Y_i \) with design matrix

\[
X = \begin{pmatrix}
1 & M_1 & M_1^2 & \ldots & M_1^7 \\
1 & M_2 & M_2^2 & \ldots & M_2^7 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & M_K & M_K^2 & \ldots & M_K^7
\end{pmatrix}
\]

Actually, this design matrix is highly singular. We use R’s \texttt{poly} function which produces a design matrix with the same column span but is much better conditioned.
4. This yields an estimated density

\[ \hat{f}_\beta(x) \propto \exp \left( \hat{\alpha} + \sum_{j=1}^{7} \hat{\beta}_j x^j \right). \]

```r
library(sda)
data(singh2002)
labels = singh2002$y
print(summary(labels))
expression_data = singh2002$x
tvals = c()
for (i in 1:6033) {
  tvals = c(tvals, t.test(expression_data[,i] ~ labels, var.equal=TRUE)$statistic)
}
```

Loading required package: entropy
Loading required package: corpcor
Loading required package: fdrtool
cancer healthy
  52   50
Warning messages:
1: package ‘sda’ was built under R version 2.15.3
2: package ‘entropy’ was built under R version 2.15.3
3: package ‘corpcor’ was built under R version 2.15.3

As in the previous exercise, we will convert to Z-scores, though this has hardly any effect.

```r
zvals = qnorm(pt(tvals, 100))
hist = plt.hist(zvals,bins=50)
f = plt.gcf()
```
Next, let’s split the data into bins, and compute the counts and the midpoints.

\begin{verbatim}
%%R
bins = c(-Inf, seq(-4,4,length=51), Inf)
counts = c()
for (i in 1:(length(bins)-1)) {
    counts = c(counts, sum((zvals > bins[i]) * (zvals <= bins[i+1])))
}
midpoints = (bins[1:length(bins)-1] + bins[2:length(bins)])/2
counts = counts[2:(length(counts)-1)]
midpoints = midpoints[2:(length(midpoints)-1)]
\end{verbatim}

To fit a GLM in R, we use the \texttt{glm} function. The argument \texttt{family} says that we are using a Poisson regression model, which sets the family $\mathbb{P}_\eta$ above to be a Poisson family.

\begin{verbatim}
%%R
density.glm = glm(counts ~ poly(midpoints,7), family=poisson(link='log'))
print(summary(density.glm))
X = model.matrix(density.glm)
C = density.glm$coef
SE = summary(density.glm)$coef[,2]
\end{verbatim}

Call:
glm(formula = counts ~ poly(midpoints, 7), family = poisson(link = "log"))

Deviance Residuals:
               Min      1Q  Median      3Q     Max
-1.79431 -0.69422  0.01274  0.61035  2.08374

Coefficients:

                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                   11
poly(midpoints, 7) 1
poly(midpoints, 7) 2
poly(midpoints, 7) 3
We see that R produces a standard error for each parameter we estimated. (Where does this come from?)

For completeness, we show that using the Newton-Raphson algorithm above yields the same estimates of coefficients.

Here is a Poisson regression model in 30 or so lines (excluding comments...)

```python
class poisson(object):
    def __init__(self, Y):
        self.Y = Y

    def value(self, eta):
        .. math::
            \Lambda^{(n)}(\eta) - \eta^T Y
        
        return (np.exp(eta) - eta*Y).sum()

    def grad(self, eta):
        .. math::
            \dot{\Lambda}(\eta) - Y

        mu = np.exp(eta)
        return mu - self.Y
```
def hess(self, eta):
    
    .. math::
    \ddot{\Lambda}(\eta)

    dmu_deta = np.exp(eta)
    return dmu_deta

class GLM(object):

    def __init__(self, Y, X, family):
        self.Y = Y
        self.X = X
        self.n, self.p = X.shape
        self.family = family(Y)

    def value(self, beta):
        
        .. math::
        \Lambda^{(n)}(X\beta) - (X\beta)^TY

        eta = np.dot(self.X, beta)
        return self.family.value(eta)

    def grad(self, beta):
        
        .. math::
        X^T \left(\nabla \CGF^{(n)}(X\beta) - Y\right)

        eta = np.dot(self.X, beta)
        return np.dot(self.X.T, self.family.grad(eta))

    def hess(self, beta):
        
        .. math::
        X^T \nabla^2 \CGF^{(n)}(X\beta)X

        eta = np.dot(self.X, beta)
        D = self.family.hess(eta)
        return np.dot(self.X.T, D[:,np.newaxis] * self.X)

    def constraint(self, beta):
        
        Check to see whether the constraint is satisfied or not.
Most GLMs will have no constraint on $\beta$. 

```python
return True
```

Let’s instantiate the model and verify we’ve computed our gradient correctly. If correct, the vector $C$ from R’s solution should have gradient approximately 0.

```python
model = GLM(counts, X, poisson)
model.grad(C)
anp.array([[0.0000001 , -0.00000002, 0.00000003, -0.00000002, 0.00000003, -0.00000002, 0.00000002, -0.00000001]])
```

As for the Hessian, the diagonal diagonal of its inverse should yield the standard errors. (Why?)

```python
H = model.hess(C)
print 'Hessian inverse:', np.sqrt(np.diag(np.linalg.inv(H)))
print 'SE:', SE
Hessian inverse: [ 0.03435823 0.35327947 0.35132316 0.32856422 0.30931663 0.30917959 0.20410041 0.20382491]
SE: [ 0.03435753 0.35327081 0.35131134 0.3285515 0.30930698 0.30917085 0.2040974 0.20382201]
```

Now, let us fit the model using Newton-Raphson.

```python
def fit(model, niter=10):
    beta = np.ones(model.p)
    value = model.value(beta)
    for _ in range(niter):
        step = 1.
        count = 0
        while True:
            proposed_beta = beta - step * np.dot(np.linalg.pinv(model.hess(beta)), model.grad(beta))
            if (model.value(proposed_beta) > value and model.constraint(proposed_beta)):
                step *= 0.5
            else:
                break
            beta = proposed_beta
            value = model.value(beta)
        return beta
beta = fit(model)
```

If we’ve converged, the gradient should be zero.
```python
print poisson(counts).value(np.dot(X, beta)) - poisson(counts).value(np.log(counts))
model.grad(beta)
```

20.5394350112

```python
array([0., 0., -0., -0., 0., 0., -0., -0.])
```

The coefficients are the same as R, reassuringly...

```python
print 'R:', C
print 'Newton-Raphson:', beta
```

R: 
```
[ 3.89067037 -0.19805055 -10.89545629 -0.14374003 1.99021697
  0.01894121  0.31586279  0.07490286]
```

Newton-Raphson: 
```
[ 3.89067037 -0.19805055 -10.8954563 -0.14374002 1.99021696
  0.01894122  0.31586278  0.07490286]
```

Let's add our density estimate to our earlier histogram.

```r
%%R -o xval,dval
xval=seq(-4,4,length=201)
dval=exp(predict(density.glm, list(midpoints=xval)))
```

```python
hist = plt.hist(zvals,bins=50, normed=True)
delta = xval[1]-xval[0]
plt.plot(xval, (dval / dval.sum() / delta), linewidth=3)
```

![Histogram with density estimate added](image)
2.3 Some useful extensions

2.3.1 Offset vector

Suppose we change the constraint \( \eta = X\beta \) to \( \eta = a + X\beta \).

The problem changes to

\[
\text{minimize } \Lambda^{(n)}(\eta) - \eta^T Y.
\]

In R, this offset can be included by adding an offset argument to glm.

2.3.2 Exercise: adding an offset

1. How do the Newton-Raphson iterations changes for fitting the GLM with an offset \( a \)?

2. Thinking of the prostate cancer \( Z \)-statistics as a perturbation of a \( N(0,1) \) distribution, refit the model above to estimate this perturbation from \( N(0,1) \) as a 7th order polynomial. What offset vector does this correspond to?

3. Does it change the fitted density values? Does it change the estimate of the parameters \( \hat{\beta} \)?

2.3.3 Multiple observations

In some modelling scenarios, one observes several observations at the same \( x_i \).

For example, suppose \( X \in \mathbb{R}^2 \) is a pair of binary variables and \( Y \) is also binary. (This would be a logistic regression model – coming shortly).

Then, for a sample of any reasonable size, one will eventually observe each of the 4 combinations of \( X \). The data can be summarized in a \( 2 \times 2 \) table:

<table>
<thead>
<tr>
<th>( X_1 = 0 )</th>
<th>( X_1 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_2 = 0 )</td>
<td>( Y_{11} )</td>
</tr>
<tr>
<td>( X_2 = 1 )</td>
<td>( Y_{21} )</td>
</tr>
</tbody>
</table>

where

\[
Y_{ij} = \sum_{k: X_k = (i,j)} Y_k
\]

This affects the likelihood in a fairly simple way. Let

\[
\text{unique}(X) = \{ X[i] : 1 \leq i \leq n \}
\]

denote the set of observed rows of \( X_{n \times p} \).

Then, we can write

\[
\tilde{\Lambda}(\beta) = \Lambda^{(n)}(X\beta)
\]

\[
= \sum_{x_u \in \text{unique}(X)} \Lambda(x_u^T \beta) : \#\{ i : X[i] = x_u \}.
\]

\[
= \sum_{x_u \in \text{unique}(X)} w(x_u) \Lambda(x_u^T \beta)
\]
where
\[ w(x_u) = \# \{ i : X[i] = x_u \}. \]

The negative log-likelihood then has the form
\[
\sum_{x_u \in \text{unique}(X)} w_u \left[ \Lambda(x_u^T \beta) - \bar{Y}(x_u) \right]
\]
with
\[ \bar{Y}(x_u) = \frac{1}{w_u} \sum_{i: X[i] = x_u} Y_i \]

This has decreased the dimension of the problem from \( n \) samples to \( \# \text{ unique}(X) \). In the binary example above, it is reduced from \( n \) to 4. In R, this information can be used using the weights argument to glm. Below, our logistic regression example will use these weights.

This leads us to a GLM with case weights, which solves the corresponding problem
\[
\minimize_{\eta, \beta} \sum_{j=1}^{m} \left[ w_j \cdot \Lambda(\eta_j) - \eta_j^T Y_j \right].
\]

### 2.3.4 Exercise: adding case weights

1. Describe the Newton-Raphson steps for a glm with case weights.

2. When the weights come from having observed ties in the \( X_i \)'s, does using the weighted version of the GLM result in the same estimates of \( \eta, \beta \)?

3. How does this change our estimate of the variance of \( \hat{\beta} \)?

### 2.3.5 Exercise: prostate cancer

In the prostate cancer example, suppose we used unequally spaced bins in our density estimate.

1. Use what we have seen about weights and offset vectors to modify the algorithm above to fit a histogram to our Z-statistics for the prostate cancer data set.

2. Try your modified algorithm out by refitting the model using bins spaced out evenly on the normal quantile scale.

### 2.4 Lindsey’s method

The following is a formal generalization of what we had done above Suppose we observe \( N \) samples \((X_1, \ldots, X_N)\) from an exponential family of distributions on \( \Omega \).

Our exponential family
\[
\mathbb{P}_\beta(dx) = e^{\beta^T t(x) - \Lambda(\beta)} \cdot m(dx).
\]

The score equation for \( \beta \) is, as usual
\[
\nabla \Lambda(\hat{\beta}) = \frac{1}{N} \sum_{i=1}^{N} t(x_i).
\]
Lindsey’s method, which we describe now, is useful when there is no closed form solution for Λ. Whatever Ω is, we assume that we can partition (finely) as \( \Omega = \bigcup_{k=1}^{K} B_k \) with each element having an associated center \( x_k \).

The data is reduced to the counts \( N_k = \# \{ i : X_i \in B_k \} \).

Standard calculations show that \((N_1, \ldots, N_K) \sim \text{Mult}(N, \pi(\beta))\)

with

\[
\pi_k(\beta) = \int_{B_k} e^{\beta^T t(x) - \Lambda(\beta)} m(dx) \\
\approx m(B_k) \cdot e^{\beta^T t(x_k) - \Lambda(\beta)} \\
= \tilde{\gamma}_k(\beta) \cdot m(B_k) e^{-\Lambda(\beta)} \\
= \tilde{\pi}_k(\beta)
\]

where

\[
\tilde{\pi}_k(\beta) = \frac{\tilde{\gamma}_k(\beta) \cdot m(B_k)}{\sum_{j=1}^{K} \tilde{\gamma}_j(\beta) \cdot m(B_j)}.
\]

This leads to an approximate multinomial model

\((N_1, \ldots, N_K) \approx \text{Mult}(N, \tilde{\pi}(\beta))\).

where the approximation is replacing \( \pi(\beta) \) with \( \tilde{\pi}(\beta) \).

Corresponding to our multinomial model is a Poisson model

\((N_1, \ldots, N_K) \sim \text{Poisson}(e^\theta \pi(\beta))\)

and an approximate Poisson model

\((N_1, \ldots, N_K) \approx \text{Poisson}(e^\alpha \tilde{\gamma}(\beta)W)\)

where \( \theta \) and \( \alpha \) are free rate parameters and

\[
W_k = m(B_k)
\]

which are fixed.

We can think of our observed multinomial as coming from the above Poisson model and we simply condition on the total number of samples.

Finding the MLE in the approximate Poisson model, yields \((\hat{\alpha}, \hat{\beta})\) where \( \hat{\beta} \) is the MLE for the original approximate multinomial problem.

For Lindsey’s method, the negative Poisson log-likelihood is

\[
\ell_{\text{Poisson}}(\beta, \alpha) = \sum_{k=1}^{K} \left( N_k \left[ \beta^T t(x_k) + \log(W_k) + \alpha \right] - e^{\beta^T t(x_k)W_k\alpha} \right).
\]
This can be optimized over $\alpha$ with $\beta$ fixed directly

$$e^{\hat{\alpha}(\beta)} = \frac{N}{\sum_{j=1}^{K} e^{\beta^T t(x_j)} W_j}.$$ 

Plugging this into $\ell_{\text{Poisson}}$ yields the so-called profiled likelihood for $\beta$

$$\tilde{\ell}(\beta) = \inf_\alpha \ell_{\text{Poisson}}(\beta, \alpha)$$

$$= \left[ \sum_{k=1}^{K} N_k \left( \beta^T t(x_k) + \log(W_k) + \log(N) - \log \left( \sum_{j=1}^{K} e^{\beta^T t(x_j)} W_j \right) \right) \right] - N$$

$$= \sum_{k=1}^{K} N_k \log (\tilde{\pi}_k(\beta)) + N \log N - N$$

This is exactly the likelihood for the approximate multinomial model (up to a constant) and maximizing it will yield $\hat{\beta}$ which is the MLE for the multinomial model. From $\hat{\beta}$ we can form $\hat{\alpha}(\hat{\beta})$ yielding an MLE for the Poisson model.

### 2.4.1 Exercise: Poisson trick and Lindsey’s method

1. Above, we solved for $\alpha(\beta)$ in the approximate Poisson model with observations $(N_1, \ldots, N_K)$. Can you also solve for $\theta(\beta)$ explicitly in the exact Poisson model?

2. Compare the resulting function of $\beta$ with the exact multinomial likelihood.

3. Give a formula for the estimated variance of $\hat{\beta}$ in the approximate model Poisson model. Does this agree with the multinomial estimate of variance?

4. Given more and more data, and finer and finer partitions, informally argue that the limiting estimator is the MLE of the original problem, i.e. with no discretization.

### 2.4.2 Exercise: galaxy data

Repeat the analysis of the galaxy data found in Section 4 of Using specially designed exponential families for density estimation. You can find the count data in galaxy.csv.

1. Fit the same Poisson model used in this paper, and compare the estimate to a kernel density estimate.

2. Include contour and heat map plots of both the Poisson estimated density and the kernel density estimate, evaluated on a 100x100 grid.

### 2.5 Logistic regression

The other common form of a GLM (perhaps more common than the Poisson model) is the logistic regression model

$$Y | \mathcal{X} \sim \text{Bernoulli}(\pi(\mathcal{X})).$$
Optionally, one might consider the model
\[ Y|X \sim \text{Binomial}(n(X), \pi(X)) \]
but this is equivalent to using weights for each case.

The GLM assumes an independent sample from this model (again, independence refers conditional independence of \( Y_i \)'s given \( X \)):
\[ Y_i|X_i, n_i \sim \text{Binomial}(n_i, \pi(X_i)). \]

Recall that the Binomial(\( n, \pi \)) is a one-parameter exponential family with \( t(Y) = Y \) and CGF
\[ \Lambda(\eta) = n \log (1 + e^\eta) \]

In terms of fitting these models, what is nice about the GLM world, is that only the family has to be changed. Above, we defined the Poisson family. For logistic regression we might define a Bernoulli and a Binomial family:

```python
class binomial(object):
    def __init__(self, YN):
    def value(self, eta):
    def grad(self, eta):
        E = np.exp(eta)
        return self.N * E / (1. + E) - self.Y
    def hess(self, eta):
        E = np.exp(eta)
        mu = E / (1. + E)
        return self.N * mu * (1 - mu)
    @classmethod
    def bernoulli(cls, Y):
        n = np.asarray(Y).shape[0]
        YN = np.ones((n,2))
        YN[:,0] = Y
        return cls(YN)
```

Below are some examples of the success rate for field goal kicker Don Cockroft, comparing his success rate with the league's success rate.

```python
cockroft = np.array([[1,4], [8,27], [15,32], [22,25], [10,12]], np.float)
league = np.array([[29,124], [176,449], [238,372], [222,306], [226,243]], np.float)
design = np.array([[1,55], [1,45], [1,35], [1,25], [1,12]], np.float)
design[:,1] = design[:,1] - 30
Cmodel = GLM(cockroft, design, binomial)
```
\[
C_\beta = \text{fit}(C_{\text{model}})
\]

\[
C_\beta
\]

\[
\begin{array}{c}
0.65048232 \\
-0.09867412
\end{array}
\]

As for standard errors, they are derived from the diagonal of the Hessian

\[
\text{np.sqrt(np.diag(np.linalg.inv(C_{\text{model}}.\text{hess}(C_\beta))))}
\]

\[
\begin{array}{c}
0.25656038 \\
0.02428946
\end{array}
\]

Just to be sure, let's check with R

```r
%%R -i cockroft,design,league
c = cockroft[,1] / cockroft[,2]
yards_over_30 = design[,2]
L = league[,1] / league[,2]
print(summary(glm(C ~ yards_over_30, weights=cockroft[,2], family=binomial())))
```

Call:
\[
glm(formula = C \sim \text{yards}_{\text{over}}_{\text{30}}, \text{family} = \text{binomial}, \text{weights} = \text{cockroft[,2]})
\]

Deviance Residuals:

```
1    2    3    4    5
0.58261 -0.08412 -0.79786 1.53133 -0.96155
```

Coefficients:

```
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.65048 0.25656   2.535  0.0112 *
yards_over_30 -0.09867 0.02429  -4.062 4.86e-05 ***
```

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 26.4758 on 4 degrees of freedom
Residual deviance: 4.2526 on 3 degrees of freedom
AIC: 22.796

Number of Fisher Scoring iterations: 4

Let's compare this to the league rate

```r
%%R print(summary(glm(L ~ yards_over_30, weights=league[,2], family=binomial())))
```

Call:
\[
glm(formula = L \sim \text{yards}_{\text{over}}_{\text{30}}, \text{family} = \text{binomial}, \text{weights} = \text{league[,2]})
\]

Deviance Residuals:
Coefficients:

|                  | Estimate | Std. Error | z value | Pr(>|z|) |
|------------------|----------|------------|---------|----------|
| (Intercept)      | 0.826993 | 0.068884   | 12.01   | <2e-16   *** |
| yards_over_30    | -0.081682| 0.005373   | -15.20  | <2e-16   *** |

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 309.9220 on 4 degrees of freedom
Residual deviance: 7.8145 on 3 degrees of freedom
AIC: 40.12

Number of Fisher Scoring iterations: 4

As above, our fit agrees with R

```r
Lmodel = GLM(league, design, binomial)
Lbeta = fit(Lmodel)
Lbeta
```

```r
array([ 0.82699314, -0.08168239])
```

```r
np.sqrt(np.diag(np.linalg.inv(Lmodel.hess(Lbeta))))
```

```r
array([ 0.06888424, 0.00537335])
```

### 2.5.1 Exercise: is Cockcroft better or worse than the league average?

It looks like Cockcroft has a lower success rate than the league average from 45 yards away.

1. How would you test the null hypothesis of “no difference at 45 yards”?
2. Describe two tests, one using all the data, and one using only the 45 yard data.
3. Carry out these tests for the above data.

### 2.5.2 Exercise: undoing case weights

For the field goal data, produce a dataset with 100 cases that has the same fitted value as Don Cockcroft’s, but each case has weight 1. Refit the binomial model. Do the results match with our example above?

### 2.5.3 Other link functions

So far, we have considered only the canonical link in our GLMs. With this choice, we saw $\eta_i = x_i^T \beta$. 

22
There are other choices, though. The usual definition of link function for a GLM is $g: \{M\} \to \mathbb{R}$ with $g(\mu_i) = x_i^T \beta$.

The canonical link corresponds to $g = \hat{\Lambda}^* = (\hat{\Lambda})^{-1}$.

The general model could be written as

$$\min_{\eta, \beta} \Lambda^{(n)}(\eta) - \eta^T Y$$

where

$$F = (g \circ \hat{\Lambda})^{-1}.$$

For the canonical link, we see that $F$ is the identity function as $g = \hat{\Lambda}^{-1}$.

### 2.5.4 Fisher scoring

Generally, this is a nonlinear equality constraint so the problem is non-convex, but is typically solved with (a modified) Newton-Raphson algorithm known as Fisher scoring.

Plugging in the nonlinear constraint above yields the problem

$$\min_{\beta} \Lambda^{(n)}(F(X\beta)) - F(X\beta)^T Y.$$

By the chain rule, the gradient of what we're trying to minimize is

$$X^T \left[ \nabla F_{X\beta} \left( \nabla \Lambda^{(n)}(F(X\beta)) - Y \right) \right].$$

where

$$\nabla F_{X\beta} = \text{diag}(\hat{F}(x^T_i \beta), 1 \leq i \leq n).$$

The Hessian has two terms. The first has the form

$$H_1(\beta) = X^T \left[ \nabla^2 F_{X\beta} \nabla^2 \Lambda^{(n)}(F(X\beta)) \nabla F_{X\beta} \right] X$$

and is non-negative definite.

While the second has the form

$$H_2(\beta) = X^T \left[ \nabla^2 F_{X\beta} \left( \nabla \Lambda^{(n)}(F(X\beta)) - Y \right) \right] X$$

where

$$\nabla^2 F_{X\beta} = \text{diag}(\hat{F}(x^T_i \beta), 1 \leq i \leq n).$$

Fisher scoring uses the observation that

$$\mathbb{E}_{F(X\beta)} (H_2(\beta)) = 0.$$

The updates take the form

$$\hat{\beta}^{(k+1)} = \hat{\beta}^{(k)} - H_1(\hat{\beta}^{(k)})^{-1} \nabla \left[ \Lambda^{(n)}(F(X\hat{\beta}^{(k)})) - F(X\hat{\beta}^{(k)})^T Y \right].$$

One advantage $H_1(\beta)$ has is that it is non-negative definite, while $H_2(\beta)$ is not, though, it seems likely that $H_1(\beta) + H_2(\beta)$ is also non-negative definite as $H_1(\beta)$ is growing with $n.$
2.5.5 Exercise: Fisher scoring

1. For \( p \) fixed, with \( n \) large, give an estimate of the size of the two terms \( H_1(\beta) \) and \( H_2(\beta) \) for \( \beta \) near \( \beta_0 \). That is, what is their rough mean and variance?

2. Does this justify the Fisher scoring algorithm, which simply ignores \( H_2 \)?

2.5.6 Probit analysis

The probit model is a common example of a GLM that does not use the canonical link. The probit model says that

\[
\pi(\beta) = \Phi(x_i^T \beta) = \int_{-\infty}^{x_i^T \beta} \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz.
\]

Therefore, \( g = \Phi^{-1} \), the normal quantile function.

2.5.7 Exercise: probit model

1. For the probit model, determine the map \( F \) that relates \( x_i^T \beta \) to \( \eta_i \). Plot it. Is it convex?

2. Describe the Fisher scoring steps for the probit model in full detail.

2.5.8 Exercise: coefficient interpretation

One of the important things to consider when changing the link is the interpretation of the parameters \( \beta \).

1. In the logistic model, show that \( e^{\beta_j} \) represents the change in log-odds of \( \pi \) when variable \( x_j \) is increased by one unit.

2. What interpretation can you give to \( \beta_j \) in the probit model?

3. What interpretation can you give to \( \beta_j \) in the Poisson regression model with the canonical link?

2.6 Analysis of deviance

The total deviance of a GLM is a function of \( \mu \), or, equivalently \( \eta \) and is defined to be

\[
D_{+}(\mathcal{Y}, \mu) = \sum_{i=1}^{n} \hat{D}(\mathcal{Y}_i, \mu_i) = \hat{D}(\mathcal{Y}; \mu)
\]

\[
= 2 \left[ \sum_{i=1}^{n} \Lambda^*(\mathcal{Y}_i) - \Lambda^*(\mu_i) - \hat{\Lambda}^*(\mu_i)(\mathcal{Y}_i - \mu_i) \right]
\]

\[
= 2 \left[ \sum_{i=1}^{n} \Lambda(\eta_i) - \Lambda(\mathcal{Y}_i) - \hat{\Lambda}(\eta_i)(\eta_i - \eta(\mathcal{Y}_i)) \right]
\]

\[
= 2 \left[ \sum_{i=1}^{n} \Lambda(\eta_i) - \Lambda(\mathcal{Y}_i) - Y_i(\eta_i - \eta(\mathcal{Y}_i)) \right]
\]
The choice is such that, for an instance of the GLM \((Y, X)\) if we fit a saturated model, i.e. \(\text{col}(X) = \mathbb{R}^n\), then \(\hat{\mu} = Y\) and \(D_+(Y, \hat{\mu}) = 0\).

As we saw before, we can use the deviance to arrive at Hoeffding’s formula

\[
\frac{d\mathbb{P}_{\hat{\eta}(Y)}}{d\mathbb{P}_\eta} = e^{D(\hat{\eta}(Y); \eta)/2} = e^{D_+(Y; \mu)/2}
\]

where

\[
\hat{\eta}(Y) = \nabla \left( \Lambda^{(n)} \right)^* (Y).
\]

The total deviance replaces the error sum of squares from ordinary least squares regression. Specifically,

\[
D_+(Y, \hat{\mu}_X) = 2 \cdot \inf_{\eta, \beta: X\beta = \eta} \left[ \Lambda^{(n)}(\eta) - \eta^T Y - \left( \Lambda^{(n)}(\hat{\eta}(Y) - \hat{\eta}(Y)^T Y) \right) \right]
\]

where

\[
\hat{\mu}_X = \nabla \Lambda^{(n)}(X\hat{\beta})
\]

is the vector of fitted values for the GLM.

That is, the total deviance is twice the difference between the minimized (negative) log-likelihood and its unrestricted minimum value.

### 2.6.1 Nested GLMs

The total deviance also is used to compare two models, one nested within the other.

Suppose we have a GLM for which we can partition the design \(X\) as

\[
X = \begin{pmatrix} X^{(1)}_{n \times p_1} & X^{(2)}_{n \times p_2} \end{pmatrix}
\]

The full model achieves \(D_+(Y, \hat{\mu}_X)\).

We might fit the reduced model with \(\beta^{(2)} \equiv 0\). That is,

\[
\minimize_{\eta, \beta: X^{(1)}\beta = \eta, \beta^{(2)} = 0} \Lambda^{(n)}(\eta) - Y^T \eta.
\]

This model achieves \(D_+(Y, \hat{\mu}_{X^{(1)}})\).

From our calculations above, we see that the difference

\[
D_+(Y, \hat{\mu}_{X^{(1)}}) - D_+(Y, \hat{\mu}_X)
\]

is twice the difference in the maximized log-likelihoods. By Wilks’ theorem, we see that under \(H_0: \beta^{(2)} = 0\)

\[
D_+(Y, \hat{\mu}_{X^{(1)}}) - D_+(Y, \hat{\mu}_X) \xrightarrow{n \to \infty} \chi^2_{p_2}.
\]

The term on the left above is actually

\[
D_+(\hat{\mu}_X, \hat{\mu}_{X^{(1)}}) = D_+(Y, \hat{\mu}_{X^{(1)}}) - D_+(Y, \hat{\mu}_X).
\]

This is the additivity theorem.
2.6.2 Exercise: additivity theorem

Prove the additivity theorem.

2.6.3 Analysis of deviance table

Consider now a partition of the design matrix into $K$ blocks of columns

$$X = \left( X^{(1)}_{n \times p_1} \ldots X^{(K)}_{n \times p_K} \right)$$

with a corresponding partition of

$$\beta = \left( \begin{array}{c} \beta^{(1)} \in \mathbb{R}^{p_1} \\ \vdots \\ \beta^{(K)} \in \mathbb{R}^{p_K} \end{array} \right) \in \mathbb{R}^{p_1 + \cdots + p_K}$$

We consider the model

$$\eta = \alpha + X \beta.$$ 

For $1 \leq j \leq K - 1$ set $\hat{\beta}^j$ to be the MLE under $H_j : \beta^{(l)} = 0, j + 1 \leq l \leq K$. Here, the model $\hat{\beta}^0$ corresponds to $\eta = \alpha$, an intercept but no other covariates in the model. The deviance of this model is referred to as the null deviance.

This leads to an analysis of deviance table

<table>
<thead>
<tr>
<th>MLE</th>
<th>deviance difference</th>
<th>degrees of freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{\beta}^1$</td>
<td>$D_+ (\hat{\mu}^1, \hat{\mu}^0)$</td>
<td>$p_1$</td>
</tr>
<tr>
<td>$\hat{\beta}^2$</td>
<td>$D_+ (\hat{\mu}^2, \hat{\mu}^1)$</td>
<td>$p_2$</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>$\hat{\beta}^K$</td>
<td>$D_+ (\hat{\mu}^K, \hat{\mu}^{K-1})$</td>
<td>$p_K$</td>
</tr>
</tbody>
</table>

The residual deviance of the model is $D_+ (Y, \hat{\mu}^0)$. The additivity formula now implies that

$$D_+ (\hat{\mu}^K, \hat{\mu}^0) = \sum_{j=0}^{K-1} D_+ (\hat{\mu}^{j+1}, \hat{\mu}^j) = D_+ (Y, \hat{\mu}^0) - D_+ (Y, \hat{\mu}^K).$$

This is the difference between the null deviance and residual deviance in R's output.

---

%%R

```r
print(summary(density.glm))
```

Call:
glm(formula = counts ~ poly(midpoints, 7), family = poisson(link = "log"))

Deviance Residuals:

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1.79431</td>
<td>-0.69422</td>
<td>0.01274</td>
<td>0.61035</td>
<td>2.08374</td>
</tr>
</tbody>
</table>

Coefficients:

| Estimate | Std. Error | z value | Pr(>|z|) |
|----------|------------|---------|----------|

26
(Intercept) 3.89067 0.03436 113.241 < 2e-16 ***
poly(midpoints, 7)1 -0.19805 0.35327 -0.561 0.575
poly(midpoints, 7)2 -10.89546 0.35131 -31.014 < 2e-16 ***
poly(midpoints, 7)3 -0.14374 0.32855 -0.437 0.662
poly(midpoints, 7)4 1.99022 0.30931 6.434 1.24e-10 ***
poly(midpoints, 7)5 0.01894 0.30917 0.061 0.951
poly(midpoints, 7)6 0.31586 0.20410 1.548 0.122
poly(midpoints, 7)7 0.07490 0.20382 0.367 0.713
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 6707.369 on 49 degrees of freedom
Residual deviance: 41.079 on 42 degrees of freedom
AIC: 342.61

Number of Fisher Scoring iterations: 4

R recognizes that all the variables in its model were derived from midpoints. Here is a slightly tedious way to see the full table.

```r
X1 = midpoints
X2 = midpoints^2
X3 = midpoints^3
X4 = midpoints^4
X5 = midpoints^5
X6 = midpoints^6
X7 = midpoints^7
print(anova(glm(counts ~ X1 + X2 + X3 + X4 + X5 + X6 + X7, family=poisson())))
```

Analysis of Deviance Table

Model: poisson, link: log

Response: counts

Terms added sequentially (first to last)

<table>
<thead>
<tr>
<th>Df Deviance Resid. Df Resid. Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL    49     6707.4</td>
</tr>
<tr>
<td>poly(midpoints, 7) 7  6666.3     42  41.1</td>
</tr>
</tbody>
</table>

R recognizes that all the variables in its model were derived from midpoints. Here is a slightly tedious way to see the full table.
Response: counts

Terms added sequentially (first to last)

<table>
<thead>
<tr>
<th>Df</th>
<th>Deviance</th>
<th>Resid. Df</th>
<th>Resid. Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>49</td>
<td>6707.4</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>1</td>
<td>0.0</td>
<td>48</td>
</tr>
<tr>
<td>X2</td>
<td>1</td>
<td>6614.3</td>
<td>47</td>
</tr>
<tr>
<td>X3</td>
<td>1</td>
<td>0.3</td>
<td>46</td>
</tr>
<tr>
<td>X4</td>
<td>1</td>
<td>49.0</td>
<td>45</td>
</tr>
<tr>
<td>X5</td>
<td>1</td>
<td>0.1</td>
<td>44</td>
</tr>
<tr>
<td>X6</td>
<td>1</td>
<td>2.3</td>
<td>43</td>
</tr>
<tr>
<td>X7</td>
<td>1</td>
<td>0.1</td>
<td>42</td>
</tr>
</tbody>
</table>