Logistic Regression

September 16, 2020

Abstract

Topics: binary classification, logistic regression, multi-class classification, softmax, generalized linear models.

1 Logistic regression

The probabilistic interpretation of the linear regression can similarly be derived for binary classification problems.

1.1 Binary classification problems

Instead of predicting some real number $y \in \mathbb{R}$, in binary classification problems, one predicts a binary target $y \in \{0, 1\}$ given some input feature vector $\mathbf{x}$.

The linear regression model $h_{\boldsymbol{\theta}}(\mathbf{x})$ is not suitable for predicting binary values since the range of such an h is the entire real line rather than restricted to $\{0, 1\}$. We need to train a model, called binary classifier, that can output values closer to the desired range $\{0, 1\}$. Let's introduce a function to map any real numbers to the range of $(0, 1)$ (the unit interval).

1.2 Sigmoid function

The sigmoid function is commonly used in machine learning and is defined as:

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{1}$$

Some properties of the sigmoid function:

- $\sigma(z)$ is monotonically increasing in z.
- $\sigma(z) \to 1$ if $z \to \infty$ and $\sigma(z) \to 0$ if $z \to -\infty$.
- $1 - \sigma(z) = \sigma(-z)$.
- $\sigma(z)$ is differentiable:

$$\frac{d\sigma(z)}{dz} = \frac{e^{-z}}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}} \frac{e^{-z}}{1 + e^{-z}} = \sigma(z)(1 - \sigma(z)). \tag{2}$$

- $\log \sigma(z)$ is differentiable:

$$\frac{d \log \sigma(z)}{dz} = -\frac{d \log(1 + e^{-z})}{dz} = \frac{e^{-z}}{1 + e^{-z}}. \tag{3}$$

- log-odd is defined as

$$\log \frac{\sigma(z)}{1 - \sigma(z)} = \log \frac{1}{e^{-z}} = z. \tag{4}$$

1.3 Fitting a logistic regression model

A logistic regression model is the parametrized function $h_{\boldsymbol{\theta}}((x)) = \sigma(\boldsymbol{\theta}^\top \mathbf{x})$. $\sigma(\boldsymbol{\theta}^\top \mathbf{x})$ is interpreted as the probability that $y = 1$ given input $\mathbf{x}$ and parameter $\boldsymbol{\theta}$.

$$\sigma(\boldsymbol{\theta}^\top \mathbf{x}) = \Pr(y = 1|\mathbf{x}; \boldsymbol{\theta}). \tag{5}$$

As a result, $1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x}) = 1 - \Pr(y = 1|\mathbf{x}; \boldsymbol{\theta}) = \Pr(y = 0|\mathbf{x}; \boldsymbol{\theta})$ since y can only take binary values. The log-odd is the log of the ratio of the two probabilities:

$$\log \frac{\sigma(z)}{1 - \sigma(z)} = \log \frac{1}{e^{-z}} = z = \boldsymbol{\theta}^\top \mathbf{x}. \tag{6}$$

This is a nice interpretation of the linear function $\boldsymbol{\theta}^\top \mathbf{x}$: it is the log-odd of classifying $\mathbf{x}$ into classes 1 and 0.

Similar to the probabilistic interpretation of linear regression, we can find the likelihood function for the observed training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ for training a binary classifier, where $y^{(i)} \in \{0, 1\}$.

$$\begin{aligned}
L(\boldsymbol{\theta}; \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m) &= \prod_{i=1}^m \Pr(y^{(i)}|\mathbf{x}^{(i)}; \boldsymbol{\theta}) && (7)\\
&= \prod_{i=1}^m \Pr(y = 1|\mathbf{x}^{(i)}; \boldsymbol{\theta})^{y^{(i)}} (1 - \Pr(y = 1|\mathbf{x}^{(i)}; \boldsymbol{\theta}))^{1-y^{(i)}} && (8)\\
&= \prod_{i=1}^m \sigma(z^{(i)})^{y^{(i)}} (1 - \sigma(z^{(i)}))^{1-y^{(i)}} && (9)\\
&\quad \prod_{i=1}^m \sigma(z^{(i)})^{y^{(i)}} \sigma(-z^{(i)})^{1-y^{(i)}} && (10)
\end{aligned}$$

where $z^{(i)}$ is defined as $\boldsymbol{\theta}^\top \mathbf{x}^{(i)}$. You can verify the last equality holds by discussing the two cases where $y^{(i)} = 1$ and $y^{(i)} = 0$. y is a random variable not tied to any $y^{(i)}$.

Using MLE to train a logistic regression model on $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ is then equivalent to

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}). \tag{11}$$

The log-likelihood is

$$\ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = \sum_{i=1}^m \left\{ y^{(i)} \log \sigma(z^{(i)}) + (1 - y^{(i)}) \log \sigma(-z^{(i)}) \right\}. \tag{12}$$

Note: the negative log-likelihood loss $-\ell(\boldsymbol{\theta})$ for logistic regression is also called "cross-entropy loss".

Maximizing the log-likelihood can be done using gradient ascent

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} + \alpha \frac{\partial}{\partial \boldsymbol{\theta}} \ell(\boldsymbol{\theta}), \tag{13}$$

where $\alpha > 0$ is a learning rate and

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \ell(\boldsymbol{\theta}) &= \sum_{i=1}^m \left\{ y^{(i)} \frac{\partial}{\partial \boldsymbol{\theta}} \log \sigma(z^{(i)}) + (1 - y^{(i)}) \frac{\partial}{\partial \boldsymbol{\theta}} \log \sigma(-z^{(i)}) \right\}\\
&= \sum_{i=1}^m \left\{ y^{(i)} (1 - \sigma(z^{(i)})) \frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{\theta}^\top \mathbf{x}^{(i)} - (1 - y^{(i)})(1 - \sigma(-z^{(i)})) \frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{\theta}^\top \mathbf{x}^{(i)} \right\}\\
&= \sum_{i=1}^m \left\{ y^{(i)} - \sigma(z^{(i)}) \right\} \mathbf{x}^{(i)} && (14)\\
&= \sum_{i=1}^m \left\{ y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \right\} \mathbf{x}^{(i)} && (15)
\end{aligned}$$

The above gradient is the same as that of MSE, but the function $h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$ is now the sigmoid function $\sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(i)})$ rather than $h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = \boldsymbol{\theta}^\top \mathbf{x}^{(i)}$.

2 Newton method

The goal of Newton method is to find the roots (zeros) of the functions $f(\boldsymbol{\theta})$, e.g., it helps to find $\boldsymbol{\theta}^*$ where $f(\boldsymbol{\theta}^*) = 0$. To achieve the goal, it produces and updates better (closer to zero) approximation iteratively.

2.1 Newton method in one variable

Consider the first-order Taylor expansion of $f(\theta)$ at point $\theta^{(t-1)}$, we have

$$f(\theta) \approx f(\theta^{(t-1)}) + f'(\theta^{(t-1)})(\theta - \theta^{(t-1)}), \tag{16}$$

and we desire the updated $\theta^{(t)}$ satisfies $f(\theta^{(t)}) = 0$, thus we have

$$0 = f(\theta^{(t-1)}) + f'(\theta^{(t-1)})(\theta^{(t)} - \theta^{(t-1)}). \tag{17}$$

So at each iteration, we update $\theta^{(t)}$ by

$$\theta^{(t)} = \theta^{(t-1)} - \frac{f(\theta^{(t-1)})}{f'(\theta^{(t-1)})} = \theta^{(t-1)} - \frac{f}{f'}|_{\theta=\theta^{(t-1)}}. \tag{18}$$

2.2 Newton method in multi-variables

We generalize Newton method in Eq. (18) to multi-variables:

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \mathbf{H}^{-1}\frac{\partial f}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t-1)}}, \tag{19}$$

where $\frac{\partial f}{\partial \boldsymbol{\theta}} = [\frac{\partial f}{\partial \theta_0}, \frac{\partial f}{\partial \theta_1}, ...\frac{\partial f}{\partial \theta_n}]$ is the gradient of $f(\boldsymbol{\theta})$, and $\mathbf{H}$ is *Hessian* matrix of $f(\boldsymbol{\theta})$, with the element on the j-th row and k-th column being $H_{jk} = \frac{\partial^2 f}{\partial \theta_j \partial \theta_k}$. Specifically, the Hessian matrix of $\ell(\boldsymbol{\theta})$ is

$$\begin{aligned} H_{jk} &= \frac{\partial}{\partial \theta_j}\frac{\partial \ell(\boldsymbol{\theta})}{\partial \theta_k} && (20)\\ &= \frac{\partial}{\partial \theta_j}\sum_{i=1}^{m}(y^{(i)} - \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(i)}))x_k^{(i)} && (21)\\ &= -\sum_{i=1}^{m}\sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(i)})(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(i)}))x_j^{(i)}x_k^{(i)}. && (22) \end{aligned}$$

The last equation uses $\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$. (dot't forget the negative sign). It is easy to obtain H in matrix form:

$$\begin{aligned} \mathbf{H} &= -\sum_{i=1}^{m}\mathbf{x}^{(i)}(\mathbf{x}^{(i)})^\top \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(i)})(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(i)})) && (23)\\ &= -\mathbf{X}\boldsymbol{\Sigma}\mathbf{X}^\top, && (24) \end{aligned}$$

where $X = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}]$ is a matrix with m columns and $n+1$ rows and

$$\begin{aligned} \boldsymbol{\Sigma} &= \text{diag}([\sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(1)})(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(1)})), \ldots, \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(m)})(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(m)}))]) && (25)\\ &= \begin{bmatrix} \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(1)})(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(1)}) & & \\ & \ddots & \\ & & \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(m)})(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(m)}) \end{bmatrix}, && (26) \end{aligned}$$

Example 2.1 *Let the training data be*

$$X = \begin{bmatrix} 1 & 1 \\ 2 & -2 \\ 4 & 1 \end{bmatrix}, y = [1, 2] \tag{27}$$

each column of X and y is for one training example. With $\boldsymbol{\theta} = [1, 1, 1]^\top$, $\sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(1)}) = \sigma(7) = 0.999$ and $\sigma(\boldsymbol{\theta}^\top \mathbf{x}^{(2)}) = \sigma(0) = 0.5$.

The diagonal matrix

$$\Sigma = \begin{bmatrix} 0.000999 & 0 \\ 0 & 0.25 \end{bmatrix} \tag{28}$$

The Hessian is then

$$H = -X\Sigma X = \begin{bmatrix} 1 & 1 \\ 2 & -2 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 0.000999 & 0 \\ 0 & 0.25 \end{bmatrix} \begin{bmatrix} 1 & 2 & 4 \\ 1 & -2 & 1 \end{bmatrix} \tag{29}$$

In numpy, you can use np.dot to implement the matrix multiplications.

The Newton method for updating $\boldsymbol{\theta}$ for logistic regression is then:

$$\boldsymbol{\theta}^{(t+1)} \rightarrow \boldsymbol{\theta}^{(t)} - \mathbf{H}^{-1} \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{30}$$

Note that $\mathbf{H}$ is an $(n+1) \times (n+1)$ square matrix and the inversion of $\mathbf{H}$ can be expensive when n is large. The gradient $\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ is a column vector of $n+1$ elements so that the product $\mathbf{H}^{-1} \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ is also a column vector of length $n+1$, matching the shape of the parameter vector $\boldsymbol{\theta}$.

3 Multi-class classification

We have been working on binary classification problems. In many machine learning applications, the problems usually have more than two classes. For example, in news classification, you can classify a piece of news into *politics, entertainment, religion, health*, and *tech*.

Formally, let $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ be the training set with $y^{(i)} \in \{1, \ldots, k\}$ where $k > 2$ is the number of classes. The goal is to learn a model $h_{\boldsymbol{\theta}}$ to map a test example $\mathbf{x}$ to the ground truth class $y \in \{1, \ldots, k\}$ as accurately as possible. The parameter $\boldsymbol{\theta}$ is now a matrix with $n+1$ rows and k columns, with the j-th column denoted by $\boldsymbol{\theta}_j$ which is a vector of length $n+1$.

3.1 Multi-class logistic regression using Softmax function

To predict a probability distribution over the k classes, we use the so-called softmax function here. Let $z_j = \boldsymbol{\theta}_j^\top \mathbf{x} \in \mathbb{R}$. Then the softmax function is defined as

$$\text{softmax}([z_1, \ldots, z_k]) = \left[\frac{\exp(z_1)}{\sum_j \exp(z_j)}, \ldots, \frac{\exp(z_k)}{\sum_j \exp(z_j)} \right] \triangleq [\phi_1, \ldots, \phi_k]. \tag{31}$$

That is, the softmax maps the vector $[z_1, \ldots, z_k]$ to another vector $[\phi_1, \ldots, \phi_k]$. It can be easily verified that $[\phi_1, \ldots, \phi_k]$ is indeed a probability distribution over k classes:

- $\phi_j > 0$;
- $\sum_{j=1}^k \phi_j = 1$.

With the model $h_{\boldsymbol{\theta}}(\mathbf{x}) = \text{softmax}([z_1, \ldots, z_k])$, the class with highest predicted probability can be selected as the predicted label of $\mathbf{x}$. The label y is distributed according to the so-called "multinomial" distribution $y \sim \prod_{j=1}^k \phi_j^{\mathbb{1}[y=j]}$.

3.2 Training multi-class logistic regression

As a notation, the indicator function $\mathbb{1}[\text{P}]$ returns 1 if the predicate P is True and 0 if P is False. To train the model, we form the likelihood function

$$L(\boldsymbol{\theta}; \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m) = \prod_{i=1}^m \Pr(y^{(i)} | \mathbf{x}^{(i)}; \boldsymbol{\theta}) \tag{32}$$

$$= \prod_{i=1}^m \prod_{j=1}^k (\phi_j^{(i)})^{\mathbb{1}[y^{(i)}=j]} \tag{33}$$

$$= \prod_{i=1}^m \phi_{y^{(i)}}^{(i)}, \tag{34}$$

where $\phi_j^{(i)}$ is the j-th entry of the output of the softmax function on $[\boldsymbol{\theta}_1^\top \mathbf{x}^{(i)}, \ldots, \boldsymbol{\theta}_k^\top \mathbf{x}^{(i)}]$. The log-likelihood is

$$\ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left\{ \sum_{j=1}^{k} \mathbb{1}[y^{(i)} = j] \log \phi_j^{(i)} \right\}. \tag{35}$$

The gradient of the log-likelihood with respect to the t-th column of $\boldsymbol{\theta}$ is

$$\frac{\partial}{\partial \boldsymbol{\theta}_t} \sum_{i=1}^{m} \log \phi_{y^{(i)}}^{(i)} = \sum_{i=1}^{m} \frac{\partial}{\partial \boldsymbol{\theta}_t} \log \phi_{y^{(i)}}^{(i)} = \sum_{i=1}^{m} [\mathbb{1}[y^{(i)} = t] - \phi_t] \mathbf{x}^{(i)} \tag{36}$$

The partial derivative $\frac{\partial}{\partial \boldsymbol{\theta}_t} \log \phi_{y^{(i)}}^{(i)}$ can be found by applying chain rule to the $y^{(i)}$-th entry of the softmax on $[\boldsymbol{\theta}_1^\top \mathbf{x}^{(i)}, \ldots, \boldsymbol{\theta}_k^\top \mathbf{x}^{(i)}]$ and discussing the cases when $y^{(i)} = t$ and when $y^{(i)} \neq t$.

To interpret this gradient, consider if $\mathbb{1}[y^{(i)} = t] = 1$ then $y^{(i)}$ takes the t-th class and $\mathbb{1}[y^{(i)} = t] - \phi_t^{(i)}$ measures how much the predicted probability of taking class t ($\phi_t^{(i)}$) is underestimated. The corresponding feature vector $\mathbf{x}^{(i)}$ is multiplied by the amount of underestimation and added to the parameter $\boldsymbol{\theta}_t$ to compensate $\phi_t^{((i)}$. If $\mathbb{1}[y^{(i)} = t] = 0$ then $y^{(i)}$ does not take the t-th class so that $\mathbb{1}[y^{(i)} = t] - \phi_t^{(i)}$ measures the overestimation of the probability of selecting class t for $\mathbf{x}^{(i)}$. $\boldsymbol{\theta}_t$ is updated to remove $\mathbf{x}^{(i)}$ multiplied by the amount of overestimation, so that the $\phi_t^{(i)}$ will be decreased.

4 Exponential family and Generalized linear models

[Refer to PRML Chapter 2.4 for more details.]

A family of distributions is a collection of probability distributions with the same parametrized functional form. By varying the parameter of the function, different distributions can be produced. An exponential family of distributions for a scalar y take the following form:

$$\Pr(y|\eta) = h(y) g(\boldsymbol{\eta}) \exp(\boldsymbol{\eta}^\top \mathbf{u}(y)). \tag{37}$$

In the above equation,

- $\mathbf{u}$ maps the scalar y to a vector $\mathbf{u}(y)$. This is called the "sufficient statistics" for a reason that will be clear next.
- $\boldsymbol{\eta}$ is the "natural parameter". By varying $\boldsymbol{\eta}$ one gets different distributions in the family.
- $g(\boldsymbol{\eta})$ makes sure that the product $h(y)g(\boldsymbol{\eta}) \exp(\boldsymbol{\eta}^\top \mathbf{u}(y))$ integrates to 1 so that $h(y)g(\boldsymbol{\eta}) \exp(\boldsymbol{\eta}^\top \mathbf{u}(y))$ is indeed a probability distribution:

$$g(\boldsymbol{\eta}) \int h(y) \exp(\boldsymbol{\eta}^\top \mathbf{u}(y)) dy = 1, \tag{38}$$

$$\log \int h(y) \exp(\boldsymbol{\eta}^\top \mathbf{u}(y)) dy = -\log g(\boldsymbol{\eta}). \tag{39}$$

4.1 Bernoulli's are in an exponential family

Let's prove that the Bernoulli distributions are in an exponential family.

$$\Pr(y = 1; \phi) = \phi^y (1 - \phi)^{1-y} = \exp(y \log \phi + (1 - y) \log(1 - \phi)) = \exp\left(y \log \frac{\phi}{1 - \phi} + \log(1 - \phi) \right).$$

By comparing this equation and Eq. (37), we found that

- $h(y) = 1$;
- $u(y) = y$;
- $\eta = \log \frac{\phi}{1-\phi} \Rightarrow \phi = \frac{1}{1+e^{-\eta}}$. In binary logistic regression, we have $\eta = \boldsymbol{\theta}^\top \mathbf{x}$ so the sigmoid function can be motivated by the exponential family;
- $g(\eta) = 1 - \phi = \frac{e^{-\eta}}{1+e^{-\eta}}$.

4.2 Gaussians are in an exponential family

The 1-dimensional Gaussian is in an exponential family.

$$\Pr(y;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{1}{2\sigma^2}(y-\mu)^2\right\} \tag{40}$$

$$= \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{\mu^2}{2\sigma^2}+\frac{y\mu}{\sigma^2}-\frac{y^2}{2\sigma^2}\right\}. \tag{41}$$

By comparing this equation and Eq. (37), we found that

- $h(y) = 1/\sqrt{2\pi}$;
- $\mathbf{u}(y) = [y, y^2]^\top$;
- $\boldsymbol{\eta} = [\mu/\sigma^2, -1/(2\sigma^2)]^\top$;
- $g(\eta) = (1/\sigma)\exp(-\mu^2/(2\sigma^2))$.

4.3 Multinomials are in an exponential family

Multinomials are used to model the selection of one out of k classes in multi-class classification problems. Let the length k binary vector $\mathbf{y} \in \{0,1\}^k$ indicate the target label of a data instance: the label is j if the j-th entry y_j is 1 and all other entries are zeros. The summation of all entries in $\mathbf{y}$ is exactly 1. The multinomial with parameters $[\phi_1, \ldots, \phi_k]$ is then

$$\Pr(\mathbf{y}|\phi_1,\ldots,\phi_k) = \prod_{j=1}^{k}\phi_j^{y_j} = \exp\left\{\sum_{j=1}^{k} y_j \log\phi_j\right\}. \tag{42}$$

By comparing this equation and Eq. (37), we found that

- $h(y) = 1$;
- $\mathbf{u}(\mathbf{y}) = \mathbf{y}$;
- $\boldsymbol{\eta} = [\log\phi_1, \ldots, \log\phi_k]^\top$;
- $g(\eta) = 1$.

Let $\eta_j = \log\phi_j$, then $\phi_j = \exp(\eta_j)$. Since $\sum_{j=1}^{k}\phi_j = 1$, then $\phi_j = \exp(\eta_j) = \frac{\exp(\eta_j)}{\sum_{j'=1}^{k}\exp(\eta_{j'})}$. You should recognize that this is the softmax function over the vector $\boldsymbol{\eta}$.

4.4 MLE of exponential families

By taking partial derivative of both sides of

$$g(\boldsymbol{\eta})\int h(y)\exp(\boldsymbol{\eta}^\top\mathbf{u}(y))dy = 1, \tag{43}$$

we have

$$0 = \frac{\partial}{\partial\boldsymbol{\eta}}g(\boldsymbol{\eta})\int h(y)\exp(\boldsymbol{\eta}^\top\mathbf{u}(y))dy + g(\boldsymbol{\eta})\int h(y)\exp(\boldsymbol{\eta}^\top\mathbf{u}(y))\mathbf{u}(y)dy \tag{44}$$

$$\frac{\partial}{\partial\boldsymbol{\eta}}\log g(\boldsymbol{\eta}) = \frac{\frac{\partial}{\partial\boldsymbol{\eta}}g(\boldsymbol{\eta})}{g(\boldsymbol{\eta})} = -\int h(y)\exp(\boldsymbol{\eta}^\top\mathbf{u}(y))\mathbf{u}(y)dy = -\mathbb{E}[\mathbf{u}(y)]. \tag{45}$$

In the above, using the chain rule, we have $\frac{d\log f(x)}{dx} = \frac{df(x)/dx}{f(x)}$. The expected value of $\mathbf{u}(y)$ is under the distribution $\Pr(y|\boldsymbol{\eta})$.