Bernoulli Distribution as a tiny Neural Network

Ankur Gupta[*]

April 28, 2019

1 Introduction

Logistic regression is often considered the smallest neural network for binary classification. We can think of Bernoulli distribution as an even smaller neural network – one that doesn't even depend on the input data. Such a neural network would likely not be useful in practice. However, given it's simplicity, it serves as an illuminating example to help us understand the statistical assumptions underlying a neural network model. The assumptions we require for modeling Bernoulli distribution as a neural network are also required for larger neural networks. As an example, using Bernoulli distribution as a tiny neural network, we can easily demonstrate how the famous cross-entropy loss comes into being. We can even extend this Bernoulli distribution model framework to recreate the familiar logistic regression model by simply replacing a constant parameter by a sigmoid-affine function.

We will start with Bernoulli distribution fundamentals first (Section 2) to get ourselves familiar with the various equivalent forms of Bernoulli distribution. In Section 3, we describe the familiar binary classification problem with relevant notation. We model the binary classification problem as a Bernoulli distribution in Section 4 and extend the Bernoulli distribution model to the familiar logistic regression in Section 5. Finally, provide some notable points in the summary at the end.

2 Bernoulli distribution

Bernoulli distribution, owing to its simplicity, is used more often than it is noticed. A random variable $X \sim \text{Bernoulli}(p)$ has the following probability mass function

[*]ankur@perfectlyrandom.org

(*pmf*):

$$\begin{aligned} P(X=1) &= p \\ P(X=0) &= 1-p \\ P(X \notin \{0,1\}) &= 0 \end{aligned} \tag{1}$$

in which, the only parameter, p, is a probability and therefore must satisfy $0 \leq p \leq 1$. We call Equation 1 the *raw form pmf* of the Bernoulli distribution.

The *raw form pmf* is simple to understand but its different case structure makes it difficult to use in other derivation. We can combine the two of the three cases in Equation 1 into one equation without changing anything about the distribution. This results in the following two forms of *pmf* – the *additive form* shown in Equation 2 and the *multiplicative form* shown in Equation 3.

$$P(X=x) = \begin{cases} px + (1-p)(1-x) & x \in \{0,1\} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$P(X=x) = \begin{cases} p^x(1-p)^{(1-x)} & x \in \{0,1\} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

All three forms — raw, additive, multiplicative[1] — are equivalent to each other and represent the same exact distribution. This implies that no matter which of the three forms we use for our analysis, we should get the exact same analytical result. However, one form may be easier to work with than the others when wrangling algebraic equations. The multiplicative form is the most common one used in both statistical analysis and with neural networks.

3 Binary classification

Let's consider a familiar application of supervised binary classification in computer vision – image classification. We would like to classify a given image into one of two classes – a *cat* image versus a *dog* image, as shown in Figure 1.

In a supervised setting, we usually have training data available, which is represented as:

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(i)}, y^{(i)}), \ldots, (x^{(m)}, y^{(m)})\} \tag{4}$$

in which, $x^{(i)} \in \mathbb{R}^{n_x}$ is the input data and $y^{(i)} \in \{0,1\}$ is the output label. For the cat *vs* dog example, $x^{(i)}$ is a vector of pixel values obtained by flattening the tensor that represents an image and $y^{(i)}$ represents the label – cat ($y=1$) or dog ($y=0$).

[1]Note that we interpret 0^0 as 1 and $\log 0^0 = 0 \log 0 = 0$.

(a) Cat (b) Dog

Figure 1: Binary classification: cat *versus* dog. *Images from Pexels.*

4 Modeling the binary classification problem as Bernoulli distribution

4.1 Modeling binary classification

We aim to *fit* a function to describe the input-output relationship in the training data. We could attempt to find a suitable deterministic function $y = f(x, \theta)$ and minimize (w.r.t the model parameters θ) some appropriate measurement of discrepancy ($\Phi(\theta)$) between the function's predicted labels and true labels[2]. Alternatively, we could model the output label as a random variable[3]

$$Y \sim \text{SomeDistribution}(x, \theta) \tag{5}$$

in which, $\theta \in \mathbb{R}^{n_t}$ is the set of model parameters. The training data in Equation 4 is interpreted as a list of m statistical samples of Y generated along with the corresponding values of x.[4] When we choose to model the output label as random variable, we have a well-established approach to minimize the discrepancy between the predicted and true labels – maximum likelihood estimation.

4.2 Modeling with Bernoulli distribution

Since the true output labels only take values in $\{0, 1\}$, it would be ideal if our choice of random variable in Equation 5 also assumes values in $\{0, 1\}$. Bernoulli distribution is one such choice:

$$Y \sim \text{Bernoulli}(p) \tag{6}$$

[2]For certain choices of $f(x, \theta)$ and $\Phi(\theta)$, the estimated model parameters may equal the estimated model parameters obtained using the statistical modeling approach, analytically.

[3]A random variable is also a function (a measurable function) but we choose to highlight the difference between any deterministic function and the restrictive measurable function that is a random variable.

[4]For this problem, we choose to consider the list of different images (*i. e.*, x's) as deterministically fixed. Therefore, x is not random and we do not condition on x.

in which, $\theta = \left[p\right]$. Note how the model above doesn't depend on the input x at all.

Now that we have a statistical model to describe the output, we can write down the likelihood as follows:

$$\begin{aligned}
&\mathcal{L}(\theta) \\
&= P\left((Y^{(1)} = y^{(1)}) \cap (Y^{(2)} = y^{(2)}) \cap \ldots (Y^{(i)} = y^{(i)}) \cap \ldots (Y^{(m)} = y^{(m)}); \theta\right) \quad (7) \\
&= P\left((Y^{(1)} = y^{(1)}) \cap (Y^{(2)} = y^{(2)}) \cap \ldots (Y^{(i)} = y^{(i)}) \cap \ldots (Y^{(m)} = y^{(m)}) \mid \Theta = \theta\right) \quad (8)
\end{aligned}$$

Equation 8 is the Bayesian form of likelihood, in which we choose to model the parameters as a random variable Θ. Equation 7 is also the likelihood but it doesn't consider the model parameters as random variables. We will only use the Equation 7 form of likelihood in this document because we have no need to model the parameter(s) as random variable(s) at this time.

Assuming independence, we can re-write Equation 7 as:

$$\mathcal{L}(\theta) = \prod_{i=1}^{i=m} P\left(Y^{(i)} = y^{(i)}; \theta\right) \quad (9)$$

Substituting the multiplicative form (Equation 3) and applying the knowledge that the output labels $y^{(i)} \in \{0, 1\}$, we obtain:

$$\mathcal{L}(\theta) = \prod_{i=1}^{i=m} p^{y^{(i)}} (1-p)^{(1-y^{(i)})} \quad (10)$$

Taking logarithm, the log-likelihood is:

$$\log \mathcal{L}(\theta) = \sum_{i=1}^{i=m} \left[y^{(i)} \log p + (1-y^{(i)}) \log(1-p)\right] \quad (11)$$

The expression above is the famous cross-entropy loss[5]. Maximizing the log-likelihood:

$$\begin{aligned}
0 &= \frac{\partial \log \mathcal{L}(\theta)}{\partial p} \\
0 &= \sum_{i=1}^{i=m} \left[\frac{y^{(i)}}{p} - \frac{(1-y^{(i)})}{(1-p)}\right] \\
\implies \hat{p} &= \frac{\sum_{i=1}^{i=m} y^{(i)}}{m} \quad (12)
\end{aligned}$$

[5]This is a demonstration of the equivalence between maximizing the likelihood and minimizing the KL divergence.

in which $\hat{p}$ is the maximum likelihood estimate for p. In order to confirm that this value of $\hat{p}$ actually maximizes the log-likelihood, we can show that the second order derivative is negative as follows:

$$\frac{\partial^2 \log \mathcal{L}(\theta)}{\partial p^2} = -\sum_{i=1}^{i=m} \left[\frac{y^{(i)}}{p^2} + \frac{(1-y^{(i)})}{(1-p)^2} \right] < 0 \tag{13}$$

See Appendix A for the same result obtained using the additive form.

5 Extension to logistic regression

In the previous subsection, we didn't even consider the input x in our model. If we want to include the input x, we could replace the previously constant p with a function of x. A simple way to include x is to model p as an *affine*[6] function of x instead of a constant

$$p = w^T x + b \tag{14}$$

in which, $w \in \mathbb{R}^{n_x}$ and $b \in \mathbb{R}$ are model parameters. However, there is a problem with Equation 14 – there is no guarantee that the expression $w^T x + b$ would be within 0 and 1, as required for p.[7] We can solve this problem easily by passing $w^T x + b$ through a *sigmoid* function to obtain the following *sigmoid-affine* function:

$$p = \sigma(w^T x + b) \tag{15}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{16}$$

The resulting model for Y becomes:

$$Y \sim \text{Bernoulli}(\sigma(w^T x + b)) \tag{17}$$

which is exactly the logistic regression model. The difference is that instead of directly assuming the logistic form, we have chosen to interpret logistic regression model as an extension of the Bernoulli distribution.

We can now perform log-likelihood maximization as usual. The likelihood is

[6] A linear function $p = w^T x$ would be even simpler than affine function $p = w^T x + b$.

[7] We could always treat the problem as a constrained optimization problem in which w, b can only take values such that $0 \leq w^T x + b \leq 1$ but we do not pursue this line of analysis.

given by the following expression:

$$\begin{aligned}
&\mathcal{L}(\theta)\\
&= P\left((Y^{(1)} = y^{(1)}) \cap (Y^{(2)} = y^{(2)}) \cap \ldots (Y^{(i)} = y^{(i)}) \cap \ldots (Y^{(m)} = y^{(m)}); \theta\right)\\
&= \prod_{i=1}^{i=m} P\left(Y^{(i)} = y^{(i)}; \theta\right) \quad \text{(independence)}\\
&= \prod_{i=1}^{i=m} \left[\{\sigma(w^T x + b)\}^{y^{(i)}} \{1 - \sigma(w^T x + b)\}^{(1-y^{(i)})}\right] \quad (y^{(i)} \in \{0,1\}, \text{for all } i)
\end{aligned} \tag{18}$$

in which $\theta = \begin{bmatrix} w & b \end{bmatrix}$. The log-likelihood may be written as:

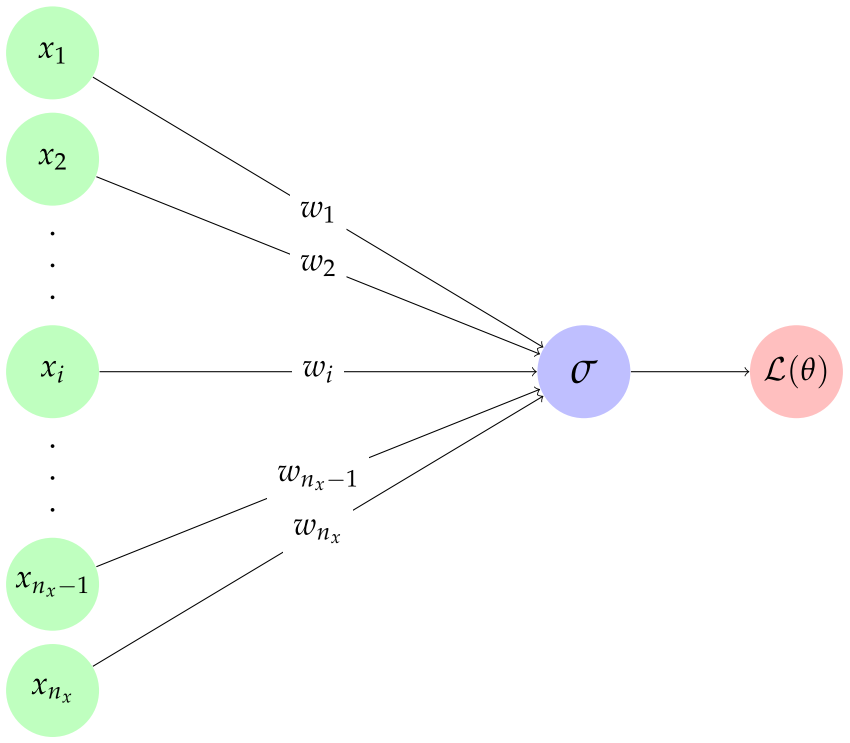
$$\log \mathcal{L}(\theta) = \sum_{i=1}^{i=m} \left[y^{(i)} \log \sigma(w^T x + b) + (1 - y^{(i)}) \log(1 - \sigma(w^T x + b))\right] \tag{19}$$

Log-likelihood may be maximized via any of the numerical optimization algorithms such as gradient descent.

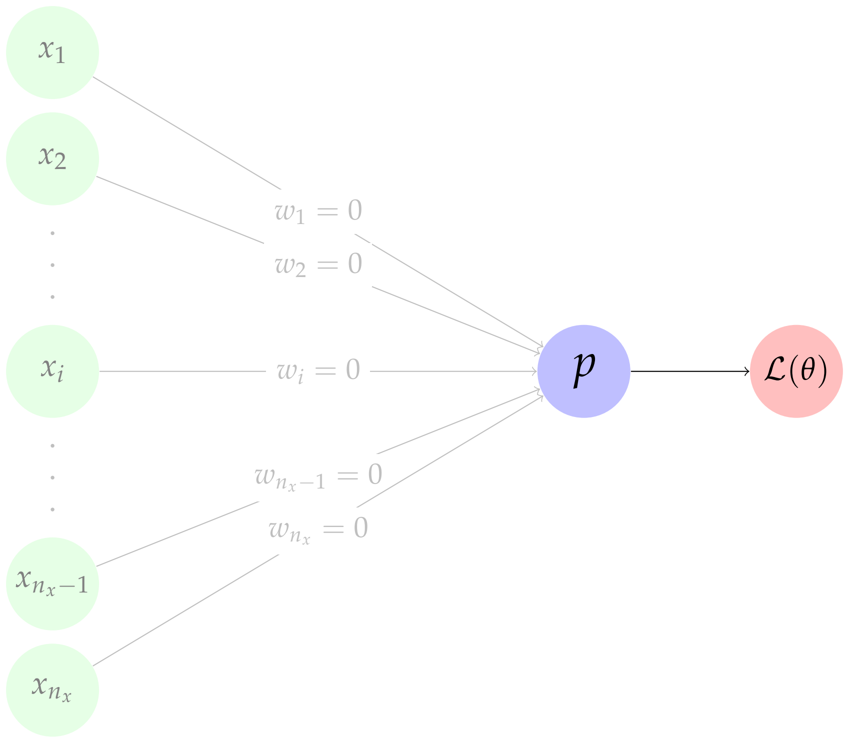
6 Summary

Equations 11 and 19 have the same form. In fact, we can obtain the log-likelihood for logistic regression by simply replacing the $p = \sigma(w^T x + b)$ in the log-likelihood for the Bernoulli model. Logistic regression (in Section 5) is a one-layer[8] neural network as shown in Figure 2a. We can think of the Bernoulli model (Section 4) as an even simpler neural network that isn't connected to the input layer at all (see Figure 2b). Equivalently, Bernoulli model is the same as a logistic regression model that has all the weights w set to zero and $p = \sigma(b)$. Bernoulli model has a constant number of unknown model parameters while logistic regression has a parameter linear complexity in the input image size. Conversely, we can also think of logistic regression as one particular extension of the Bernoulli model framework.

[8]Typically, we don't count input layer at all.



(a) Logistic Regression



(b) Bernoulli Model

A Appendix A: Bernoulli distribution model using the additive form

Continuing from Equation 7 using the additive form (Equation 2):

$$\begin{aligned}
&\mathcal{L}(\theta) \\
&= P\left((Y^{(1)} = y^{(1)}) \cap (Y^{(2)} = y^{(2)}) \cap \ldots (Y^{(i)} = y^{(i)}) \cap \ldots (Y^{(m)} = y^{(m)}); \theta\right) \\
&= \prod_{i=1}^{i=m} P\left(Y^{(i)} = y^{(i)}; \theta\right) \qquad \text{(independence)} \\
&= \prod_{i=1}^{i=m} \left[p y^{(i)} + (1-p)(1-y^{(i)})\right] \qquad (y^{(i)} \in \{0,1\}, \text{for all } i)
\end{aligned} \tag{20}$$

While this expression looks daunting, note that $y^{(i)}$ can only be 0 or 1. Let k denote the number of training samples for which $y^{(i)} = 1$. This means that there are $m-k$ training samples for which $y^{(i)} = 0$. Let's rewrite $py^{(i)} + (1-p)(1-y^{(i)})$ as:

$$\begin{aligned}
py^{(i)} + (1-p)(1-y^{(i)}) &= (2y^{(i)-1})\left[p + \frac{(1-y^{(i)})}{(2y^{(i)}-1)}\right] \\
&= (2y^{(i)-1})\left[p + a^{(i)}\right]
\end{aligned} \tag{21}$$

in which,

$$(2y^{(i)-1}) = \begin{cases} 1 & y^{(i)} = 1 \\ -1 & y^{(i)} = 0 \end{cases} \tag{22}$$

$$a^{(i)} = \frac{(1-y^{(i)})}{(2y^{(i)}-1)} = \begin{cases} 0 & y^{(i)} = 1 \\ -1 & y^{(i)} = 0 \end{cases} \tag{23}$$

Substituting Equation 21 into Equation 20,

$$\mathcal{L}(\theta) = \underbrace{\prod_{i=1}^{i=m}(2y^{(i)-1})}_{(-1)^{(m-k)}} \underbrace{\prod_{i=1}^{i=m}\left[p + a^{(i)}\right]}_{p^k(p-1)^{(m-k)}} = p^k(1-p)^{(m-k)} \tag{24}$$

Finally, realizing that $k = \sum_{i=1}^{i=m} y^{(i)}$ and $m-k = \sum_{i=1}^{i=m}(1-y^{(i)})$, we see that:

$$\mathcal{L}(\theta) = p^{\sum_{i=1}^{i=m} y^{(i)}}(1-p)^{\sum_{i=1}^{i=m}(1-y^{(i)})} = \prod_{i=1}^{i=m} p^{y^{(i)}}(1-p)^{(1-y^{(i)})} \tag{25}$$

which is the same as the likelihood derived using the multiplicative form (Equation 10). The resulting maximum likelihood estimation analysis is identical to the analysis for the multiplicative form.