8 Information Theory

Maximum Likelihood Estimation

Recall that the simple bias network is useful as an easy to compute baseline for regression problems. The simple bias network (see figure below) has no features as inputs. For the simple bias network, we have

$$\boxed{b} \to \hat{\mathbf{y}}$$

Figure 2: Regression Simple Bias Network

that $\hat{y_i} = b$ for $i = 1, 2, \ldots, n$. We showed that the value of the bias, b, that minimizes the mean square error

$$\text{MSE}(b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - b)^2$$

is given by the mean of the target values, i.e.

$$\underset{b}{\text{argmin}}\ \text{MSE(b)} = \bar{y_i}.$$

The simple bias network is a baseline for regression networks. Next, we rigourously derive a similar baseline for classification networks.

Recall that the target for classification networks is one-hot-encoded. So, classification networks must have multiple outputs. The simple bias network for classification networks has no features as inputs and a bias *vector* as outputs. Instead of trying to select **b** by minimizing mean

$$\boxed{\mathbf{b}} \to \hat{\mathbf{P}}$$

Figure 3: Classification Simple Bias Network

square error, we will use a different more general principle called **maximum likelihood estimation.** Maximum likelihood estimation selects the parameters of a network to maximize the probability of observing the training data for the network.

107 Example (Likelihood Function for MNIST)
For the MNIST dataset, the target values are the 10 digits 0 through 9. Let us apply maximum likelihood estimation to a simple bias network. For such a network, we don't need the images since the network has no inputs. Assume the training data target values are the $n = 100$ digits

$$\mathbf{y} = \begin{pmatrix} 2 & 6 & \dots & 9 \end{pmatrix}^\top .$$

The probability of observing the target values is

$$\begin{aligned} \mathbb{P}(\mathbf{y}) &= \mathbb{P}(y_1 = 2, y_2 = 6, \dots, y_{100} = 9) \\ &= \mathbb{P}(y_1 = 2)\mathbb{P}(y_2 = 6) \cdots \mathbb{P}(y_{100} = 9) \\ &= \mathbb{P}(0)^{q_0}\mathbb{P}(1)^{q_1} \cdots \mathbb{P}(9)^{q_9} \end{aligned}$$

where q_0 equals the number of 0's in the training dataset, q_1 equals the number of 1's and so on. Note that

$$q_0 + q_1 + \cdots + q_9 = 100.$$

We have assumed that the training targets are selected at random with replacement so that the joint probability is the product of the probabilities of each digit. Now since the output of the simple bias network for classification is

$$\hat{\mathbf{P}} = \begin{pmatrix} \hat{p}_0 & \hat{p}_1 & \cdots & \hat{p}_9 \\ \hat{p}_0 & \hat{p}_1 & \cdots & \hat{p}_9 \\ \vdots & \vdots & \vdots & \vdots \\ \hat{p}_0 & \hat{p}_1 & \cdots & \hat{p}_9 \end{pmatrix}_{100 \times 10}$$

for the simple bias network, we have that the likelihood of observing the training targets, $\mathbf{y}$ is given by:

$$\mathbb{P}(\mathbf{y}) = \hat{p}_0^{q_0} \hat{p}_1^{q_1} \cdots \hat{p}_9^{q_9}$$

where $\hat{p}_0, \hat{p}_1, \dots, \hat{p}_9$ are equal to the corresponding bias values, $b_1, b_2, \dots, b_{10}$.

Evaulating the likelihood function $\mathbb{P}(\mathbf{y})$ will result in underflows for large datasets because it equals a product of many numbers that are less than one. Instead we will maximize the **log likelihood function**

$$\begin{aligned} \ln(\mathbb{P}(\mathbf{y})) &= \ln(\hat{p}_0^{q_0} \hat{p}_1^{q_1} \cdots \hat{p}_9^{q_9}) \\ &= q_0 \ln(\hat{p}_0) + q_1 \ln(\hat{p}_1) + \cdots + q_9 \ln(\hat{p}_9) \end{aligned}$$

or, since we prefer to minimize, we can minimize the **negative log likelihood function**

$$\mathbb{L}(\mathbf{y}) = -\left[q_0 \ln(\hat{p}_0) + q_1 \ln(\hat{p}_1) + \cdots + q_9 \ln(\hat{p}_9) \right].$$

Note that $\mathbb{L}(\mathbf{y})$ equals the **cross-entropy** function for a simple bias network. In order to determine the values of $\hat{p}_0$ through $\hat{p}_9$ that minimize the negative log likelihood function, we must set the partial derivatives of $\mathbb{L}(\mathbf{y})$ with respect to $\hat{p}_0$ through $\hat{p}_9$ equal to zero. However, there is a constaint that must be satisfied:

$$\hat{p}_0 + \hat{p}_1 + \ldots + \hat{p}_9 = 1.$$

Optimization with constraints can be solved using Lagrange multipliers, so we first review Lagrange multipliers.

108 Example (Lagrange Multipliers)
Determine the point on the parabola $y = x^2$ that is closest to the point $(10, 2)$.

We want to minimize the distance between the point $P(x, y)$ on the parabola and the point $(10, 2)$. Minimizing the square of the distance gives the same minimum value and is simplier since it avoids the square-root.

$$\begin{aligned} \text{minimize } f(x, y) &= (x-2)^2 + (y-4)^2 \\ \text{subject to } g(x, y) &= y - x^2 = 0 \end{aligned}$$

We construct the Lagrangian function

$$\begin{aligned} L(x, y, \lambda) &= f(x, y) + \lambda g(x, y) \\ &= (x-2)^2 + (y-4)^2 + \lambda(y - x^2). \end{aligned}$$

Set the partial derivatives of $L(x, y, \lambda)$ equal to zero and solve for x, y and λ gives

$$x = 2,\ y = 4,\ \lambda = -4.$$

So the point Q(2,4) is the closest point on the parabola $y = x^2$ to the point $(10, 2)$.
