### MLE in Linear regression

Let's consider linear regression. Imagine that each single prediction y^ produces a "conditional" distribution pmodel(y^∣x), given a sufficiently large train set. The goal of the learning algorithm is again to match the distribution pdata(y∣x).

Now we need an assumption. We hypothesize the neural network or any estimator f as y^=f(x,θ). The estimator approximates the mean of the normal distribution N(μ,σ) the we choose to parametrize pdata. Specifically, in the simplest case of linear regression we have μ=θTx. We also assume a fixed standard deviation σ of the normal distribution. These assumptions immediately causes MLE to become Mean Squared Error (MSE) optimization. Let's see how.

\begin{aligned} & \hat{y}=f(\textbf{x} , \boldsymbol{\theta}) \\ y & \sim \mathcal{N}\left(y , \mu=\hat{y}, \sigma^{2}\right) \\ p(y \mid \textbf{x} , \boldsymbol{\theta}) &=\frac{1}{\sigma \sqrt{2 \pi}} \exp \left(\frac{-(y-\hat{y})^{2}}{2 \sigma^{2}}\right) \end{aligned}

In terms of log-likelihood we can form a loss function:

\begin{aligned} L &=\sum\_{i=1}^{m} \log p(y \mid \textbf{x} , \boldsymbol{\theta}) \\ &=\sum\_{i=1}^{m} \log \frac{1}{\sigma \sqrt{2 \pi}} \exp \left(\frac{-\left(\hat{y}^{(i)}-y^{(i)}\right)^{2}}{2 \sigma^{2}}\right) \\ &=\sum\_{i=1}^{m}-\log (\sigma \sqrt{2 \pi})-\log \exp \left(\frac{(\hat{y}^{(i)}-y^{{(i)}} )^{2}.}{2 \sigma^{2}}\right) \\ &=\sum\_{i=1}^{m}-\log (\sigma)-\frac{1}{2} \log (2 \pi)-\frac{(\hat{y}^{(i)}-y^{{(i)}})^{2}}{2 \sigma^{2}} \\ &=-m \log (\sigma)-\frac{m}{2} \log (2 \pi)-\sum\_{i=1}^{m} \frac{\left(\hat{y}^{(i)}-y^{{(i)}}\right)^{2}}{2 \sigma^{2}} \\ \end{aligned}

By taking the partial derivative with respect to the parameters, we get the desired MSE.

\begin{aligned} \nabla\_{\theta} L &=-\nabla\_{\theta} \sum\_{i=1}^{m} \frac{\left\|\hat{y}^{(i)}-y^{(i)}\right\|^{2}}{2 \sigma^{2}} \\ &=-m \log (\sigma)-\frac{m}{2} \log (2 \pi)-\sum\_{i=1}^{m} \frac{\left\|\hat{y}^{(i)}-y^{(i)}\right\|^{2}}{2 \sigma^{2}} \\ &=-m \log (\sigma)-\frac{m}{2} \log (2 \pi)- \frac{m}{2 \sigma^{2}} MSE \end{aligned}

### MLE in supervised classification

In linear regression, we pmodel(y∣x,θ) as a normal distribution. More precisely, we parametrized the mean to be μ=θTx.

It is possible to convert linear regression to a classiﬁcation problem. All we need to do is encode the ground truth as a one-hot vector:

\begin{cases}1 & \text { if } y=y\_{i} \\ 0 & \text { otherwise }\end{cases} ,

where ii refer to a single data instance.

\begin{aligned} H\_{i}\left(p\_{data}, p\_{model}\right) &=-\sum\_{y \in Y} p\_{data}\left(y \mid \textbf{x}\_{i}\right) \log p\_{model}\left(y \mid \textbf{x}\_{i}\right) \\ &=-\log p\_{model}\left(y\_{i} \mid \textbf{x}\_{i}\right) \end{aligned}

For simplicity let's consider the binary case of two labels, 0 and 1.

\begin{aligned} L &=\sum\_{i=1}^{n} H\_{i}\left(p\_{data}, p\_{model}\right) \\ &=\sum\_{i=1}^{n}-\log p\_{model}\left(y\_{i} \mid \textbf{x}\_{i}\right) \\ &=-\sum\_{i=1}^{n} \log p\_{model}\left(y\_{i} \mid \textbf{x}\_{i}\right) \end{aligned}

\begin{aligned} =\underset{\boldsymbol{\theta}}{\arg \min } L &= \underset{\boldsymbol{\theta}}{\arg \min } -\sum\_{i=1}^{n} \log p\_{model}\left(y\_{i} \mid \textbf{x}\_{i}\right) \end{aligned}

This is in line with our definition of conditional MLE:

θML=θargmaxi=1∑mlogpmodel(y(i)∣x(i),θ)

Broadly speaking, MLE can be applied to most (supervised) learning problems, by specifying a parametric family of (conditional) probability distributions.

Another way to achieve this in a binary classification problem would be to take the scalar output y of the linear layer and pass it from a sigmoid function. The output will be in the range [0,1] and we define this as the probability of p(y=1∣x,θ).

p(y=1∣x,θ)=σ(θTx)=sigmoid(θTx)∈[0,1]

Consequently, p(y=0∣x,θ)=1−p(y=1∣x,θ). In this case binary-cross entropy is practically used. No closed form solution exist here, one can approximate it with gradient descend. For reference, this approach is surprisingly known as ""logistic regression".