NN parameter estimation for regression

- Model: $\mathbf{Y} = \hat{\mathbf{Y}} + \boldsymbol{\epsilon} = f(\mathbf{X};\boldsymbol{\theta}) + \boldsymbol{\epsilon}$ *where* $f(\mathbf{X};\boldsymbol{\theta})$ expresses our neural network

$$\boldsymbol{\epsilon} \sim N(\mathbf{Y} - f(\mathbf{X};\boldsymbol{\theta}), \boldsymbol{\Sigma})$$

$$= \sqrt{\frac{1}{(2\pi)^N \det \boldsymbol{\Sigma}}} e^{-\frac{1}{2}(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta}))^T \boldsymbol{\Sigma}^{-1}(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta}))}$$

$$:= P(\mathbf{Y} \,|\, \mathbf{X};\boldsymbol{\theta}) \leftarrow \mathscr{L}(\mathbf{D};\boldsymbol{\theta})$$

- Optimization:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; - \log P(\mathbf{Y} \,|\, \mathbf{X};\boldsymbol{\theta}) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; - \log\left(\sqrt{\frac{1}{(2\pi)^N \det \boldsymbol{\Sigma}}} \exp\left[-\frac{1}{2}\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)\right]\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; - \frac{1}{2}\log((2\pi)^N \det \boldsymbol{\Sigma}) - \frac{1}{2}\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; - \left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)^T\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; - \sum_{i=1}^{M} \left(\mathbf{y}^{(i)} - f(\mathbf{x}^{(i)};\boldsymbol{\theta})\right)^T\left(\mathbf{y}^{(i)} - f(\mathbf{x}^{(i)};\boldsymbol{\theta})\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; - \sum_{i=1}^{M}\sum_{j=1}^{N} \left(\mathbf{y}_j^{(i)} - f(\mathbf{x}^{(i)};\boldsymbol{\theta})_j\right)^2 \quad \leftarrow \text{least squares}
\end{aligned}$$

$\boldsymbol{\Sigma}$ is diagonal, strictly positive, independent of $\mathbf{x}$; it doesn't affect $\hat{\boldsymbol{\theta}}$

NN parameter estimation for classification

- Model: $P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta}) \quad where \quad \mathbf{x}, \mathbf{y} \sim P_D \,, \quad \mathbf{y} \in \{0,1\}^N$

one hot encoding

- Optimization:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{M} -\log P(\mathbf{y}_i \mid \mathbf{x}_i; \boldsymbol{\theta}) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; -\mathbb{E}_{\mathbf{x},\mathbf{y} \sim P_D}\left[\log P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})\right] \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; \mathbb{E}_{\mathbf{x},\mathbf{y} \sim P_D}\left[\log \frac{1}{P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})}\right]
\end{aligned}$$

Cross entropy between data distribution and the model output distribution, $H\big(P_D(\mathbf{y} \mid \mathbf{x}), P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})\big)$. Note, this holds for arbitrary PMFs (softmax, Bernoulli, etc...). Because the labels, $\mathbf{y}$, are one hot, they have zero entropy, and thus in this setting minimizing the cross entropy is equivalent to minimizing the KL divergence, $D_{KL}\big(P_D(\mathbf{y} \mid \mathbf{x}), P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})\big)$.