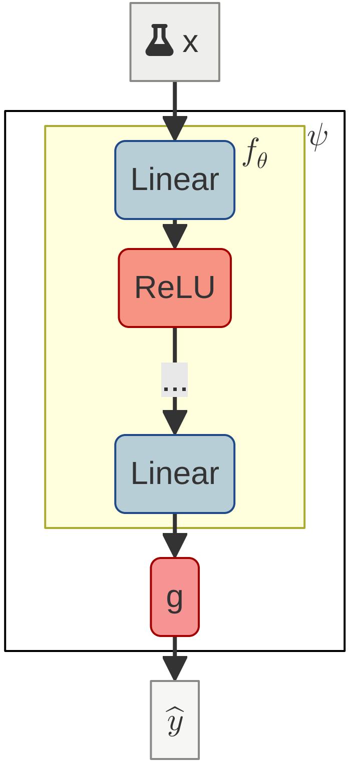
Loss Functions

Recap: Output Transformations

Input: $\mathbf{x}$

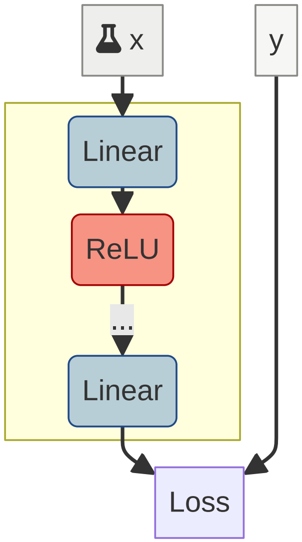
Output: $\mathbf{o}$

Output transformation: g

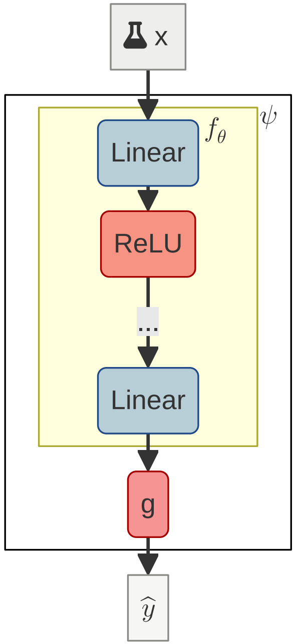


Training Deep Networks

Training



Inference



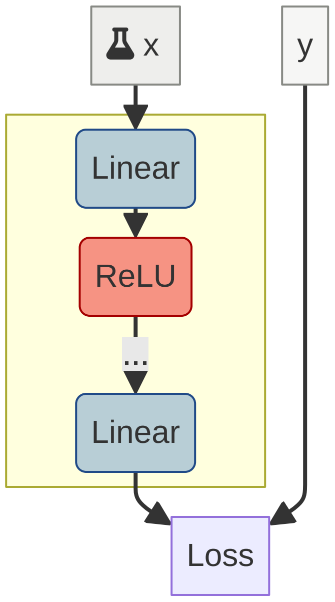
Recap: Loss

Loss function:

$$l(\theta|\mathbf{x}_i, \mathbf{y}_i)$$

Expected loss:

$$L(\theta|\mathcal{D}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[l(\theta|\mathbf{x},\mathbf{y})]$$

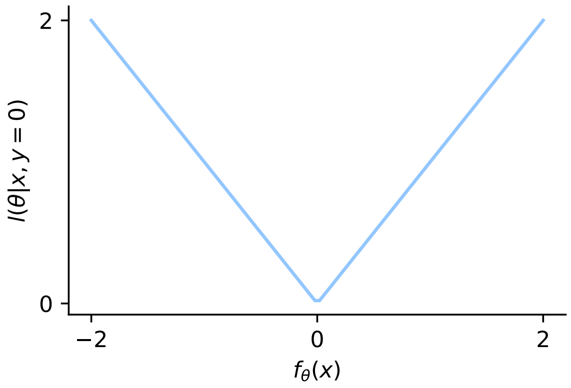


Regression

Regression $\psi : \mathbb{R}^n \to \mathbb{R}$

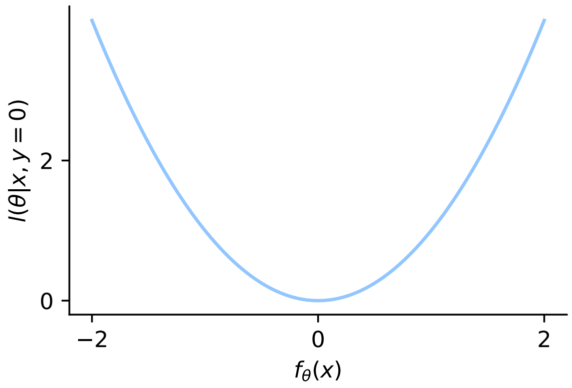
L1 Loss

$$l(\theta|\mathbf{x},\mathbf{y}) = \|\mathbf{y} - \mathbf{o}\|_1 = \|\mathbf{y} - f_\theta(\mathbf{x})\|_1$$



L2 Loss

$$l(\theta|\mathbf{x},\mathbf{y}) = \|\mathbf{y} - \mathbf{o}\|_2^2 = \|\mathbf{y} - f_\theta(\mathbf{x})\|_2^2$$



Binary Classification

Binary classification $\psi : \mathbb{R}^n \to [0, 1]$

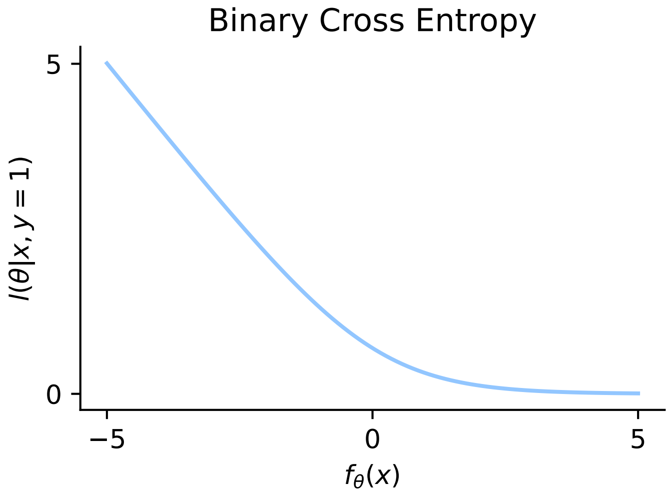
- labels $y \in \{0, 1\}$

Likelihood estimation

- $p(0) = 1 - \sigma(f_\theta(x))$
- $p(1) = \sigma(f_\theta(x))$

Binary cross entropy (negative log-likelihood)

$$
\begin{aligned}
l(\theta|\mathbf{x}, \mathbf{y}) &= -\log p(y) \\
&= -[y \log p(1) + (1 - y) \log p(0)]
\end{aligned}
$$



Binary Classification Loss in Practice

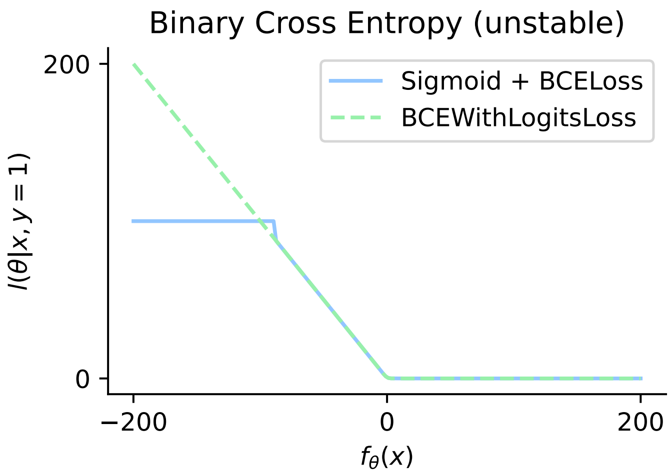
Numerical stability

- $\sigma(o) = 0$ for $o \to -100$
- $\log(\sigma(o)) = \log(0) =$ `NaN` !!

Combine $\log$ and σ

$$l(\theta|\mathbf{x}, y) = -[y \log \sigma(o) + (1 - y) \log(1 - \sigma(o))]$$

- Use `BCEWithLogitsLoss` !!
- Numerically more stable than `Sigmoid` + `BCELoss`



Multi-Class Classification

Binary classification $\psi : \mathbb{R}^n \to [1 \dots C]$

- labels $y \in \{1, \dots, C\}$

Likelihood estimation

$$\mathbf{p} = \mathrm{softmax}(\mathbf{o}) = \begin{bmatrix} p(1) \\ p(2) \\ \vdots \\ p(C) \end{bmatrix}$$

Cross entropy (negative log-likelihood)

$$l(\theta|\mathbf{x}, \mathbf{y}) = -\log p(y)$$

Multi-Class Classification Loss in Practice

Numerical stability

- $\mathrm{softmax}(o)_i \to 0$ for $o_j - o_i > 100$
- $\log(\mathrm{softmax}(o)_i) = \log(0)$ is `NaN`

Combine $\log$ and $\mathrm{softmax}$

$$l(\theta|\mathbf{x}, \mathbf{y}) = -\log \mathrm{softmax}(\mathbf{o})_y$$

- Use `CrossEntropyLoss` !!
- numerically more stable

Loss Functions - TL;DR

Regression: L1 loss `torch.nn.L1Loss`, L2 loss `torch.nn.MSELoss`

Binary classification: binary cross-entropy loss `torch.nn.BCEWithLogitsLoss`

Multi-class classification: cross-entropy loss `torch.nn.CrossEntropyLoss`

Always use PyTorch loss for better numerical stability!