# Deep Learning

*A course about theory & practice*

UNIVERSITÀ DI PAVIA

## Regression vs. Classification

Marco Piastra

# Function Approximation (a.k.a. Regression)

# Regression

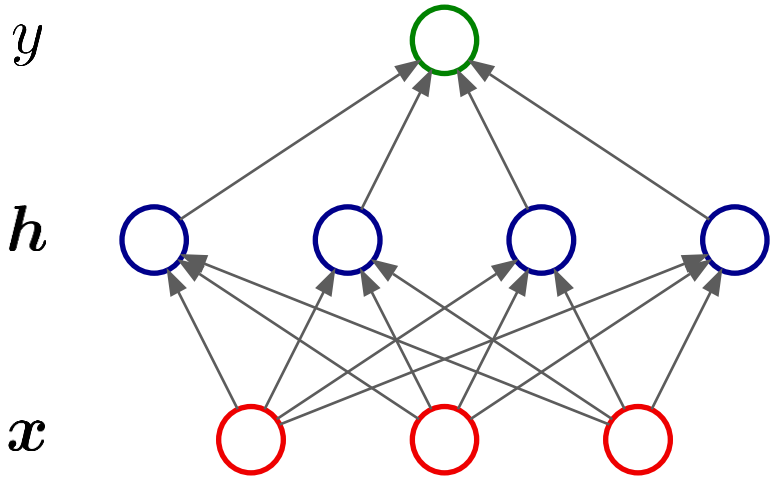
- ***Function approximation***

$$y = f^*(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d$$

Feed-forward neural network

$$\tilde{y} = \boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b$$

*The Feed-forward neural network is used as an approximator of* $f^*(\boldsymbol{x})$



# Classification

# Classification: Softmax

- **Classification**

$$y = f^*(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d, \quad y \in \{\text{class}_i\}_{i=1}^k$$

Feed-forward neural network with a *Softmax* layer

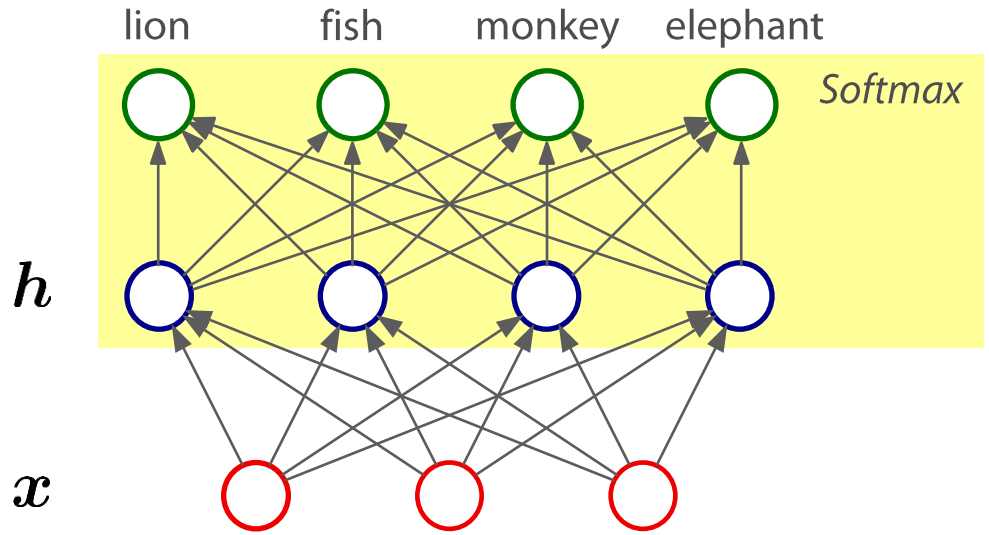
$$P(\tilde{y} = \text{class}_i \,|\, \boldsymbol{x}) := \frac{\exp(\boldsymbol{w}_i \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b_i)}{\sum_{j=1}^k \exp(\boldsymbol{w}_j \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b_j)}$$

*From now on*

$$P(\tilde{y} = \text{class}_i \,|\, \boldsymbol{x})$$

*will be written as*

$$P(\tilde{y} = i \,|\, \boldsymbol{x})$$



# Classification: Softmax

- **Classification**

$$y = f^*(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d, \quad y \in \{\text{class}_i\}_{i=1}^k$$

The *Softmax* layer can be rewritten as:

$$P(\tilde{y} = \text{class}_i \,|\, \boldsymbol{h}) := \frac{\exp(\boldsymbol{w}_i \cdot \boldsymbol{h} + b_i)}{\sum_{j=1}^k \exp(\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j)}$$

where, in this case: $\boldsymbol{h} := g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$

*(yet, more in general, $\boldsymbol{h}$ can be anything)*

# Classification: Softmax

- ***Softmax as a layer***

  The entire *Softmax* layer can be rewritten as:

$$P((\tilde{y} = i)_1^k \mid \boldsymbol{h}) := \frac{\exp(\boldsymbol{W}_S \boldsymbol{h} + \boldsymbol{b}_S)}{\sum \exp(\boldsymbol{W}_S \boldsymbol{h} + \boldsymbol{b}_S)}$$

*Probability distribution (a vector)*

*Sum of all components*

where:

$$\boldsymbol{W}_S := \begin{bmatrix} - \ \boldsymbol{w}_1 \ - \\ \vdots \\ - \ \boldsymbol{w}_k \ - \end{bmatrix} \qquad \boldsymbol{b}_S := \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix}$$

*The vector* $\boldsymbol{W}_S \boldsymbol{h} + \boldsymbol{b}_S$ *is sometimes referred to as the* ***logit (vector)***

# Classification: Softmax

- ***Cross-entropy* (in general)**

  $P$ and $Q$ are probability distributions on a discrete random variable $y \in \{1, \cdots, k\}$

$$H(Q, P) := -\sum_{j=1}^{k} Q(y = j) \log P(\tilde{y} = j)$$

- ***A loss function for Softmax***

  $Q$ describes the 'true' class, i.e., the one in the dataset

$$\hat{y}_j^{(i)} := \begin{cases} 1 & \text{if } y^{(i)} = j \\ 0 & \text{otherwise} \end{cases}$$

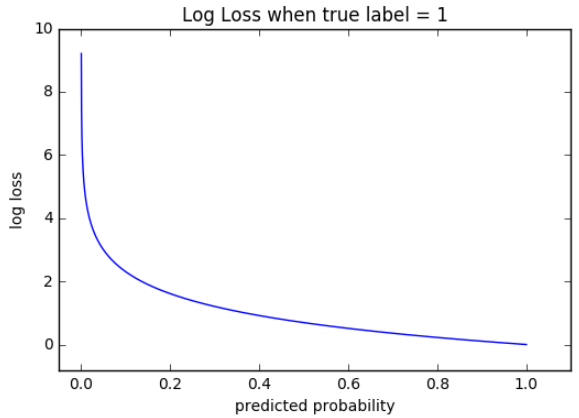
  while $P$ is the output of the Softmax layer

$$P(\tilde{y} = j \,|\, \boldsymbol{h})$$

  Hence, the loss is:

$$L(\boldsymbol{h}^{(i)}, y^{(i)}) := -\sum_{j=1}^{k} \hat{y}_j^{(i)} \log P(\tilde{y} = j | \boldsymbol{h}^{(i)})$$

$$= -\log P(\tilde{y} = y^{(i)} | \boldsymbol{h}^{(i)})$$



# Classification: Softmax

- **Cross-entropy for Softmax**

$$L(\boldsymbol{h}^{(i)}, y^{(i)}) := -\sum_{j=1}^{k} \delta(y^{(i)} = j) \log P(\tilde{y} = j | \boldsymbol{h}^{(i)})$$

Expressing the loss function in vector form:

$$\boldsymbol{y} := \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix}, \; y_j := \delta(y = j) \qquad \boldsymbol{p} := \begin{bmatrix} p_1 \\ \vdots \\ p_k \end{bmatrix}, \; p_j := P(\tilde{y} = j \,|\, \boldsymbol{h})$$

*'one hot' representation*

$$L(\boldsymbol{h}^{(i)}, \boldsymbol{y}^{(i)}) = -\boldsymbol{y}^{(i)} \cdot \log(\boldsymbol{p}^{(i)})$$

which implies that also the dataset has to be transformed in the 'one hot' representation

$$D := \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N} \qquad \Longrightarrow \qquad D := \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}$$

# Classification: Softmax

- **Gradient of *Softmax*** (layerwise)

$$L(D) = \sum_{i=1}^{N} L(\boldsymbol{h}^{(i)}, \boldsymbol{y}^{(i)}) = -\sum_{i=1}^{N} \boldsymbol{y}^{(i)} \cdot \log(\boldsymbol{p}^{(i)})$$

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} L(D) = \frac{\partial}{\partial \boldsymbol{\vartheta}} \sum_{i=1}^{N} L(\boldsymbol{h}^{(i)}, \boldsymbol{y}^{(i)}) = -\frac{\partial}{\partial \boldsymbol{\vartheta}} \sum_{i=1}^{N} \boldsymbol{y}^{(i)} \cdot \log(\boldsymbol{p}^{(i)})$$

$$= -\sum_{i=1}^{N} \boldsymbol{y}^{(i)} \cdot \frac{\partial}{\partial \boldsymbol{\vartheta}} \log(\boldsymbol{p}^{(i)})$$

*This is a matrix*

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} \log(\boldsymbol{p}) = \begin{bmatrix} \frac{\partial}{\partial \vartheta_1} \log(p_1) & \dots & \frac{\partial}{\partial \vartheta_d} \log(p_1) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \vartheta_1} \log(p_k) & \dots & \frac{\partial}{\partial \vartheta_d} \log(p_k) \end{bmatrix} = \begin{bmatrix} - & \frac{\partial}{\partial \boldsymbol{\vartheta}} \log(p_1) & - \\ & \vdots & \\ - & \frac{\partial}{\partial \boldsymbol{\vartheta}} \log(p_k) & - \end{bmatrix}$$

# Classification: Softmax

- **Gradient of *Softmax*** (layerwise)

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\vartheta}} \log(p_j) &= \frac{\partial}{\partial \boldsymbol{\vartheta}} \log P(\tilde{y} = j \mid \boldsymbol{h}) \\
&= \frac{\partial}{\partial \boldsymbol{\vartheta}} \log \frac{\exp(\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j)}{\sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)} \\
&= \frac{\partial}{\partial \boldsymbol{\vartheta}} \left( \log \exp(\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j) - \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l) \right) \\
&= \frac{\partial}{\partial \boldsymbol{\vartheta}} (\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j) - \frac{\partial}{\partial \boldsymbol{\vartheta}} \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)
\end{aligned}$$

# Classification: Softmax

- **Gradient of *Softmax*** (layerwise)

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} \log(p_j) = \frac{\partial}{\partial \boldsymbol{\vartheta}} (\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j) - \frac{\partial}{\partial \boldsymbol{\vartheta}} \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)$$

*Case 1*: $\boldsymbol{\vartheta} = \boldsymbol{w}_r$ or $\boldsymbol{\vartheta} = b_r$

*Case 2*: $\boldsymbol{h}(\boldsymbol{\vartheta})$ i.e. $\boldsymbol{\vartheta}$ is a generic parameter on which $\boldsymbol{h}$ depends

$$\frac{\partial \boldsymbol{h}^{[i]}}{\partial \boldsymbol{\vartheta}^{[i]}}$$

$$\frac{\partial \boldsymbol{h}^{[i]}}{\partial \boldsymbol{\vartheta}^{[j]}}, \; j < i$$

Let's compute the two contributions separately

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} (\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j)$$

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)$$

# Classification: Softmax

- **Gradient of *Softmax*** (layerwise)

$$\frac{\partial}{\partial \boldsymbol{\vartheta}}(\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j)$$

*Case 1*: $\boldsymbol{\vartheta} = \boldsymbol{w}_r$ or $\boldsymbol{\vartheta} = b_r$

$$\frac{\partial}{\partial \boldsymbol{w}_r}(\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j) = \begin{cases} \mathbf{0} & \text{if } r \neq j \\ \boldsymbol{h} & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial b_r}(\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j) = \begin{cases} 0 & \text{if } r \neq j \\ 1 & \text{otherwise} \end{cases}$$

*Case 2*: $\boldsymbol{h}(\boldsymbol{\vartheta})$ i.e. $\boldsymbol{\vartheta}$ is a generic parameter on which $\boldsymbol{h}$ depends

$$\frac{\partial}{\partial \boldsymbol{\vartheta}}(\boldsymbol{w}_j \cdot \boldsymbol{h} + b_j) = \boldsymbol{w}_j \cdot \frac{\partial}{\partial \boldsymbol{\vartheta}}\boldsymbol{h}$$

# Classification: Softmax

- **Gradient of *Softmax*** (layerwise)

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)$$

*Case 1*: $\boldsymbol{\vartheta} = \boldsymbol{w}_r$ or $\boldsymbol{\vartheta} = b_r$

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{w}_r} \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l) &= \\
&= \frac{1}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m \cdot \boldsymbol{h} + b_m)} \frac{\partial}{\partial \boldsymbol{w}_r} \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l) \\
&= \frac{1}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m \cdot \boldsymbol{h} + b_m)} \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l) \frac{\partial}{\partial \boldsymbol{w}_r} (\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l) \\
&= \frac{\exp(\boldsymbol{w}_r \cdot \boldsymbol{h} + b_r)}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m \cdot \boldsymbol{h} + b_m)} \boldsymbol{h} \quad = p_r \boldsymbol{h}
\end{aligned}$$

# Classification: Softmax

- **Gradient of *Softmax*** (layerwise)

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)$$

*Case 1*: $\boldsymbol{\vartheta} = \boldsymbol{w}_r$ or $\boldsymbol{\vartheta} = b_r$

$$\frac{\partial}{\partial b_r} \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l) =$$

$$= \frac{1}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m \cdot \boldsymbol{h} + b_m)} \frac{\partial}{\partial b_r} \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)$$

$$= \frac{1}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m \cdot \boldsymbol{h} + b_m)} \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l) \frac{\partial}{\partial b_r} (\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)$$

$$= \frac{\exp(\boldsymbol{w}_r \cdot \boldsymbol{h} + b_r)}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m \cdot \boldsymbol{h} + b_m)} \quad = p_r$$

# Classification: Softmax

- **Gradient of *Softmax*** (layerwise)

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)$$

*Case 2*: $\boldsymbol{h}(\boldsymbol{\vartheta})$ i.e. $\boldsymbol{\vartheta}$ is a generic parameter on which $\boldsymbol{h}$ depends

$$\frac{\partial}{\partial \boldsymbol{\vartheta}} \log \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l) =$$

$$= \frac{1}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m \cdot \boldsymbol{h} + b_m)} \frac{\partial}{\partial \boldsymbol{\vartheta}} \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)$$

$$= \frac{1}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m \cdot \boldsymbol{h} + b_m)} \sum_{l=1}^{k} \exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l) \frac{\partial}{\partial \boldsymbol{\vartheta}} (\boldsymbol{w}_l \cdot \boldsymbol{h} + b_l)$$

$$= \sum_{l=1}^{k} \frac{\exp(\boldsymbol{w}_l \cdot \boldsymbol{h} + b_r)}{\sum_{m=1}^{k} \exp(\boldsymbol{w}_m \cdot \boldsymbol{h} + b_m)} \boldsymbol{w}_l^T \frac{\partial}{\partial \boldsymbol{\vartheta}} \boldsymbol{h} \quad = \left( \sum_{l=1}^{k} p_l \, \boldsymbol{w}_l^T \right) \frac{\partial}{\partial \boldsymbol{\vartheta}} \boldsymbol{h}$$

# Likelihood
# (or the Mother of All Loss Functions)

# MSE: Choosing the Noise Model

- ***Function approximation***

$$y = f^*(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d$$

Feed-forward neural network

$$\tilde{y} = \tilde{f}(\boldsymbol{x}; \boldsymbol{\vartheta}) = \boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b$$

Assume that the dataset is *noisy*

$$D := \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$$

*Gaussian noise, constant variance*

$$\text{where} \quad y^{(i)} \sim \mathcal{N}(\mu^{(i)}, \sigma^2), \quad \mu^{(i)} = f^*(\boldsymbol{x}^{(i)}), \quad \forall i$$

Then, the approximator function can be interpreted as:

$$\mu = \tilde{f}(\boldsymbol{x}; \boldsymbol{\vartheta})$$

# MSE: Choosing the Noise Model

- ***Likelihood of the dataset***

  Assuming all data items $(\boldsymbol{x}^{(i)}, y^{(i)})$ are **Independent and Identically Distributed (I.I.D.)**

*Joint Probability of the dataset, given $\boldsymbol{\vartheta}$*

$$\mathcal{L}(D;\boldsymbol{\vartheta}) = \prod_{i=1}^{N} P(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\vartheta})$$

*Gaussian noise, constant variance*

$$= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)}-\mu^{(i)})^2}{2\sigma^2}\right)$$

$$\log \mathcal{L}(D;\boldsymbol{\vartheta}) = \sum_{i=1}^{N} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)}-\mu^{(i)})^2}{2\sigma^2}\right)\right]$$

$$= \sum_{i=1}^{N} \left(\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\right] + \log\left[\exp\left(-\frac{(y^{(i)}-\mu^{(i)})^2}{2\sigma^2}\right)\right]\right)$$

$$= \sum_{i=1}^{N} \left(-\frac{1}{2}\log(2\pi\sigma^2) - \frac{(y^{(i)}-\mu^{(i)})^2}{2\sigma^2}\right)$$

# MSE: Choosing the Noise Model

- ***Likelihood of the dataset***

  Assuming all data items $(\boldsymbol{x}^{(i)}, y^{(i)})$ are **Independent and Identically Distributed (I.I.D.)**

$$\log \mathcal{L}(D; \boldsymbol{\vartheta}) = \sum_{i=1}^{N} \left( -\frac{1}{2} \log(2\pi\sigma^2) - \frac{(y^{(i)} - \mu^{(i)})^2}{2\sigma^2} \right)$$

  Actually, $\sigma^2$ is constant and it is not involved in the maximization:

$$\underset{\boldsymbol{\vartheta}}{\operatorname{argmax}} \ \log \mathcal{L}(D; \boldsymbol{\vartheta}) = -\sum_{i=1}^{N} (y^{(i)} - \mu^{(i)})^2$$

  By convention, *minimization* (i.e., gradient descent) is preferred:

$$\underset{\boldsymbol{\vartheta}}{\operatorname{argmin}} \ \mathrm{MSE}(D; \boldsymbol{\vartheta}) = \sum_{i=1}^{N} (y^{(i)} - \mu^{(i)})^2$$

  Moral: *Mean Squared Error* (MSE) is the right loss function when the ***noise model*** is Gaussian with constant $\sigma^2$

# Softmax: End-toEnd Likelihood

- ***Classification***

$$y = f^*(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d, \quad y \in \{\text{class}_i\}_{i=1}^k$$

The *Softmax* layer (as a function of the *logit vector*):

*class* $c$ *logit vector*

$$P(\tilde{y} = y_c \,|\, \boldsymbol{z}; \boldsymbol{\vartheta}) := \frac{\exp(z_c)}{\sum_{j=1}^k \exp(z_j)}$$

# Softmax: End-toEnd Likelihood

- ***Likelihood of the dataset***

Assuming all data items $(\boldsymbol{x}^{(i)}, y^{(i)})$ are **Independent and Identically Distributed (I.I.D.)**

*Joint Probability of the dataset, given* $\boldsymbol{\vartheta}$

$$\mathcal{L}(D;\boldsymbol{\vartheta}) = \prod_{i=1}^{N} P(\tilde{y}^{(i)} = y^{(i)} \mid \boldsymbol{z}^{(i)};\boldsymbol{\vartheta})$$

$$= \prod_{i=1}^{N} \frac{\exp(z_c^{(i)})}{\sum_{j=1}^{k} \exp(z_j^{(i)})}$$

Define an *indicator* (i.e., one-hot encoding)

$$\hat{y}_c^{(i)} := \mathbb{I}(y^{(i)} = c) = \begin{cases} 1 & \text{if } y^{(i)} = c \\ 0 & \text{otherwise} \end{cases}$$

The likelihood becomes:

$$\mathcal{L}(D;\boldsymbol{\vartheta}) = \prod_{i=1}^{N} \prod_{c=1}^{K} \left[ \frac{\exp(z_c^{(n)})}{\sum_{j=1}^{K} \exp(z_j^{(n)})} \right]^{\hat{y}_c^{(n)}}$$

# Softmax: End-toEnd Likelihood

- ***Likelihood of the dataset***

  Assuming all data items $(\boldsymbol{x}^{(i)}, y^{(i)})$ are **Independent and Identically Distributed (I.I.D.)**

$$\log \mathcal{L}(D; \boldsymbol{\vartheta}) = \sum_{i=1}^{N} \sum_{c=1}^{K} \hat{y}_c^{(i)} \log \left[ \frac{\exp(z_c^{(i)})}{\sum_{j=1}^{K} \exp(z_j^{(i)})} \right]$$

$$= \sum_{i=1}^{N} \sum_{c=1}^{K} \hat{y}_c^{(i)} \log P(\tilde{y}^{(i)} = y_c \mid \boldsymbol{z}^{(i)}; \boldsymbol{\vartheta})$$

$$= \sum_{i=1}^{N} \log P(\tilde{y}^{(i)} = y^{(i)} \mid \boldsymbol{z}^{(i)}; \boldsymbol{\vartheta})$$

*the log of a probability is always negative or zero*

By convention, *minimization* (i.e., gradient descent) is preferred:

$$\underset{\boldsymbol{\vartheta}}{\operatorname{argmin}} \, \mathrm{CE}(D; \boldsymbol{\vartheta}) = -\sum_{i=1}^{N} \log P(\tilde{y}^{(i)} = y^{(i)} \mid \boldsymbol{z}^{(i)}; \boldsymbol{\vartheta})$$

Moral: *Cross Entropy* (CE) is the right loss function
when the class attribution is ***uncertain (noisy)***

# Regularization

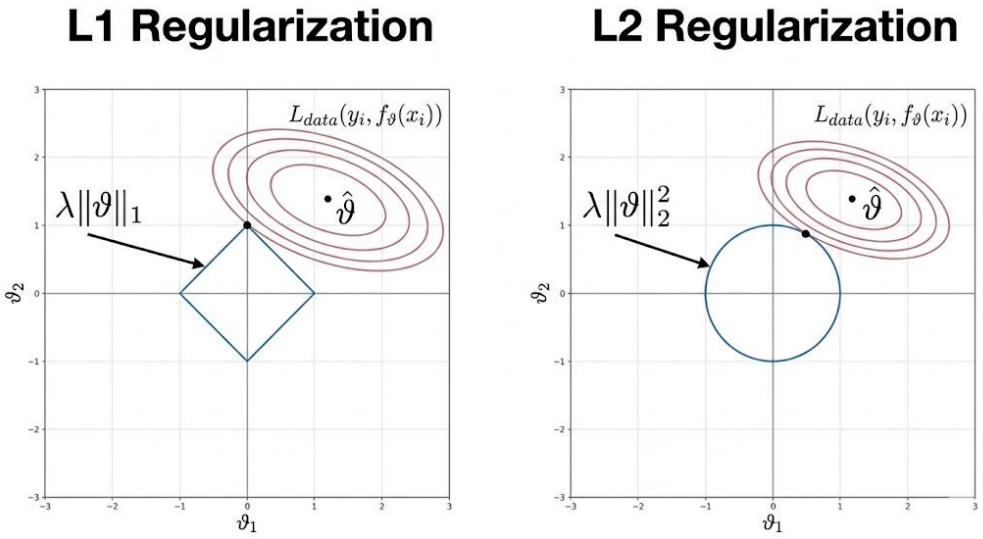
# Beyond Likelihood: Regularization

- ***A penalty on complexity***

*coefficient*

*Penalty on complexity*

$$L_{total} = L_{data} + \lambda R(\boldsymbol{\vartheta})$$




- ***Norm-based regularization***

$L_2$ Regularization *(Weight Decay)*

$$R(\boldsymbol{\vartheta}) = \frac{1}{2}\|\boldsymbol{\vartheta}\|_2^2 = \frac{1}{2}\boldsymbol{\vartheta}^T\boldsymbol{\vartheta} = \frac{1}{2}\left(\vartheta_1^2 + \vartheta_2^2 + \cdots + \vartheta_n^2\right)$$

*Geometrically, it pushes the solution toward a sphere centered at the origin*
*Smoothes the function by preventing any single feature from dominating*

$L_1$ Regularization *(Lasso)*

$$R(\boldsymbol{\vartheta}) = \|\boldsymbol{\vartheta}\|_1 = \sum_{i=1}^{n} |\vartheta_i| = |\vartheta_1| + |\vartheta_2| + \cdots + |\vartheta_n|$$

*Promotes sparsity (forces many weights to exactly zero). Useful for feature selection.*

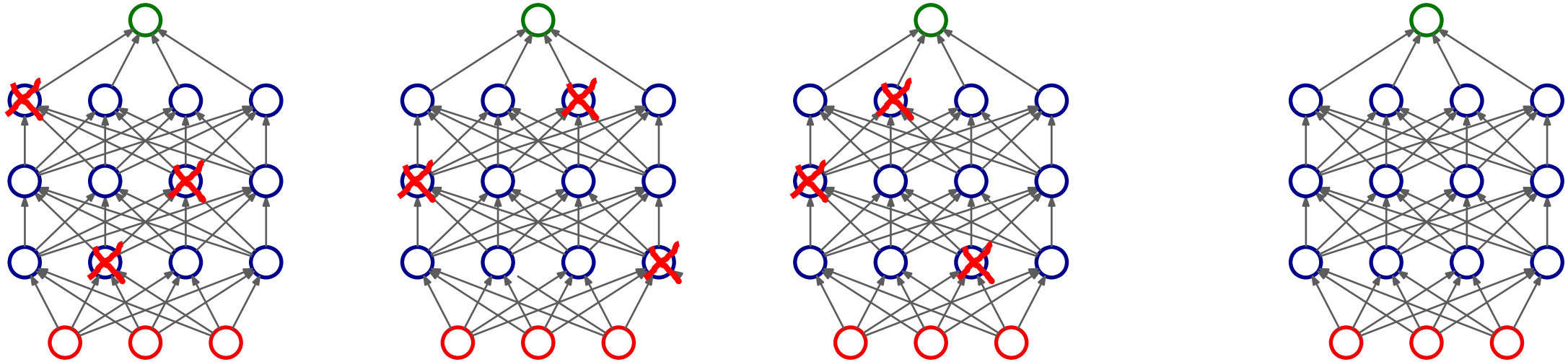
[https://medium.com/@iambeingferoz/l1-vs-l2-regularization-intuition-geometry-and-bayesian-view-da18a2b74de1]

# Stochastic Regularization: Dropout

- **Knocking-out at random**

  For each mini-batch, a small percentage of 'units' is de-activated

*At inference time, dropout is not active*



***Training: mini-batch 1*** ***Training: mini-batch 2*** ***Training: mini-batch 3*** ***Inference Time***

*Instead of penalizing the values of weights, it promotes the reliability of the network*
*It prevents* ***co-adaptation:*** *each unit must learn robust, independent features*