3-2 Regularization

Zhonglei Wang

WISE and SOE, XMU, 2025

Contents

Intuition

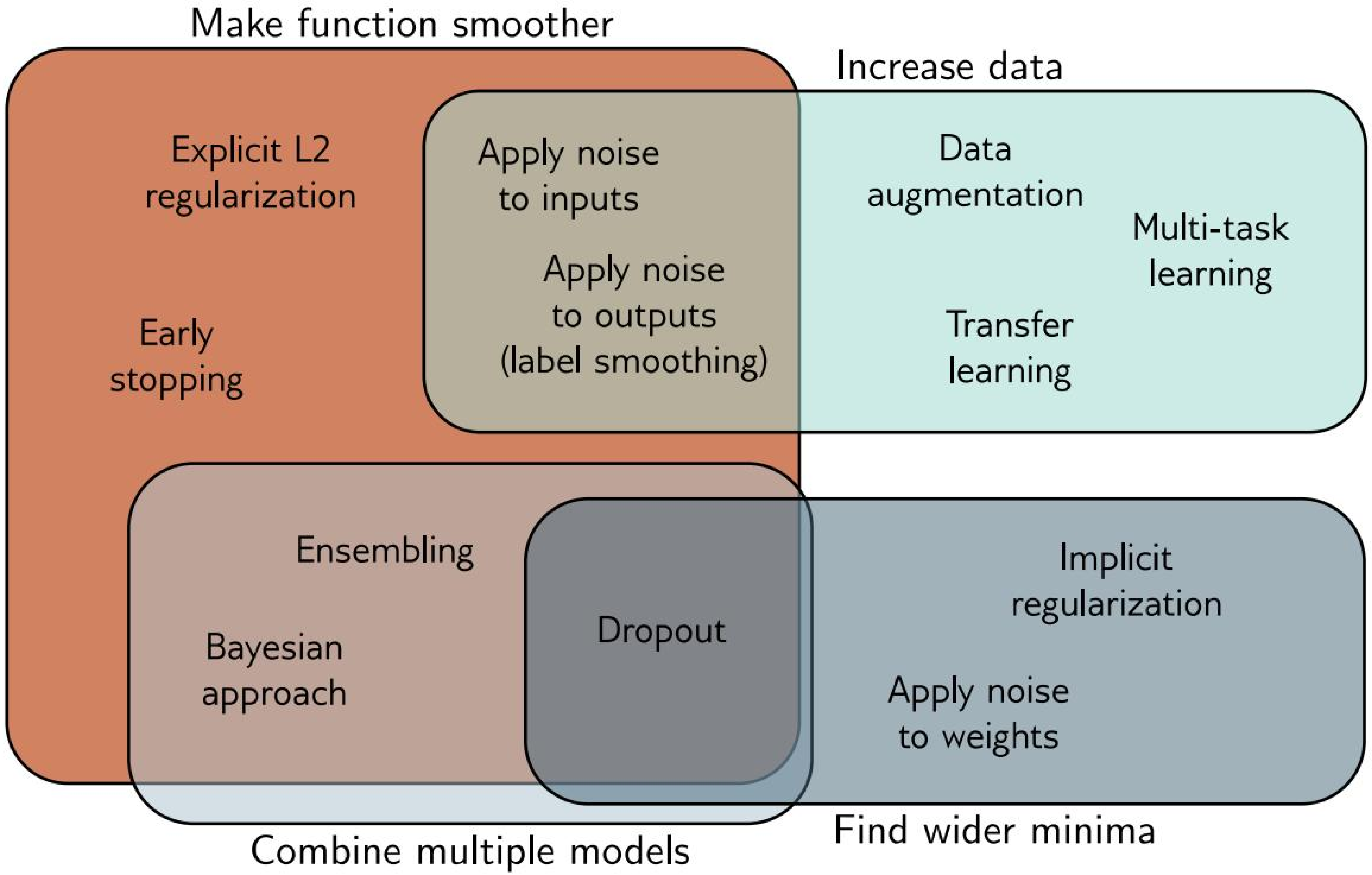
1. Fully connected neural network often involves too many model parameters
    - Complex models tend to overfit the training data
    - Decrease the generality of the trained model
2. Definitely, we can consider a simpler neural network
    - Simpler models may underfit the training data, however
3. Thus, we may would like to obtain a kind of less complex model without changing the architecture

Intuition

1. In machine learning, regularization
   - usually adds additional terms to the loss function
   - improves generality of the deep learning model
2. There exist implicit regularization, including dropout, early stopping, ensembling

Intuition

1. The following image is Figure 9.14 of Prince (2024)



Penalties on parameters

1. l_2 penalty

$$\mathcal{J}_2 = \frac{\lambda}{2n} \sum_{l=1}^{L} \| \boldsymbol{W}^{[l]} \|_F^2$$

- λ : hyperparameter
- $\|\boldsymbol{A}\|_F = (\sum_i \sum_j a_{ij}^2)^{1/2}$: Frobenius norm of a real-valued matrix $\boldsymbol{A} = (a_{ij})$
- Derivative

$$\frac{\partial \mathcal{J}_2}{\partial \boldsymbol{W}^{[l]}} = \frac{\lambda}{n} \boldsymbol{W}^{[l]}$$

- It is similar to ridge regression and is used to control the complexity of the model

Penalties on parameters

1. l_1 penalty

$$\mathcal{J}_1 = \frac{\lambda}{n} \sum_{l=1}^{L} \|\boldsymbol{W}^{[l]}\|_1$$

- $\|\boldsymbol{A}\|_1 = \sum_i \sum_j |a_{ij}|$
- Derivative

$$\frac{\partial \mathcal{J}_1}{\partial \boldsymbol{W}^{[l]}} = \frac{\lambda}{n} \text{sign}(\boldsymbol{W}^{[l]})$$

- It is used to
  - ▷ control the complexity of the model
  - ▷ induce sparsity, which is similar to LASSO

Remarks

1. We only implement l_2 or l_1 penalty on the weight terms $\{\boldsymbol{W}^{[l]} : l = 1, \ldots, L\}$
   - By implementing, we mean that the l_2 or l_1 penalty is directly added on the original cost function
   - We leave the bias terms $\{\boldsymbol{b}^{[l]} : l = 1, \ldots, L\}$ unpenalized
2. We "misuse" $\|\boldsymbol{A}\|_1$ to denote the summation of absolute values of elements in $\boldsymbol{A}$
   - More generally, $\|\boldsymbol{A}\|_1 = \max\limits_j \sum\limits_i |a_{ij}|$ denotes the maximum column summation of absolute elements

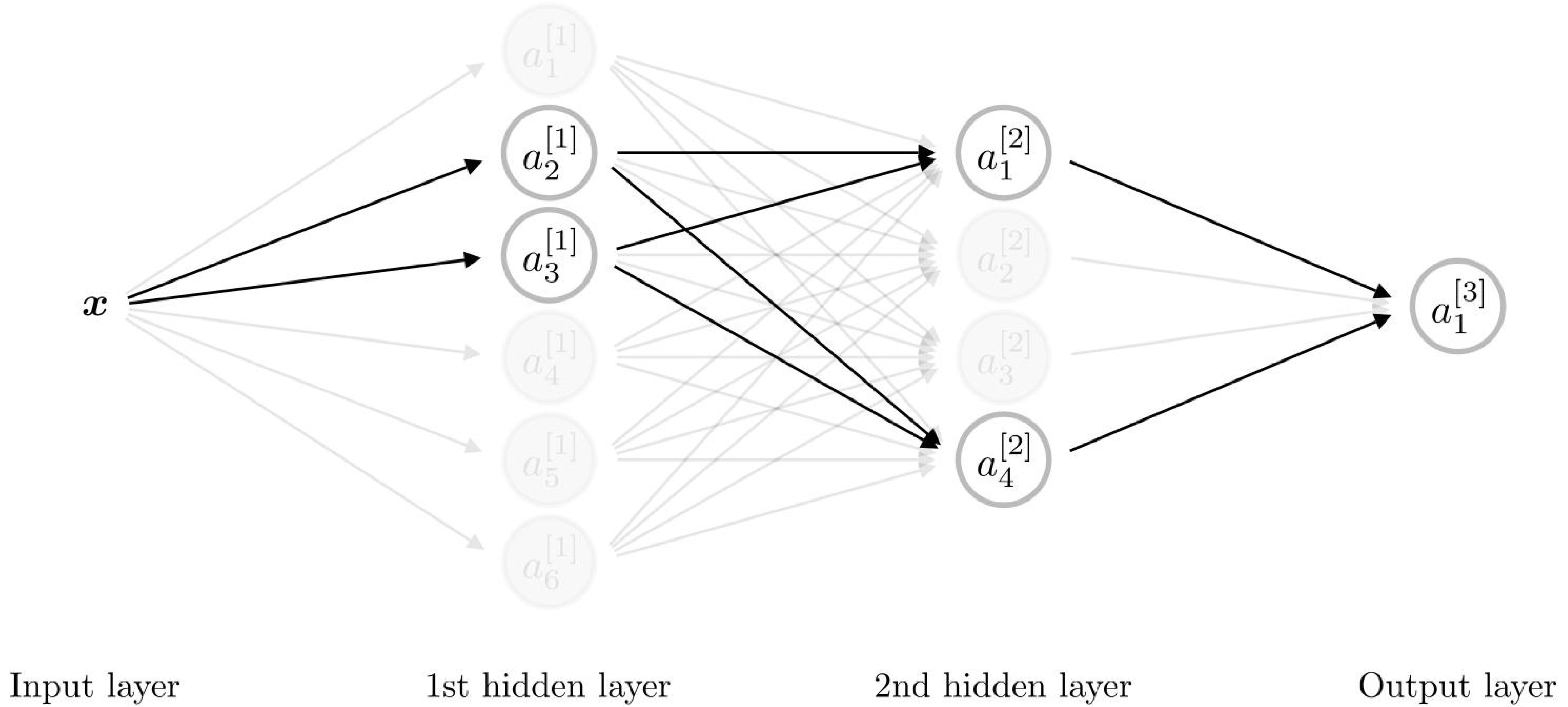
Dropout

1. Commonly used regularization during the training step
    - Randomly remove neurons in each hidden layer
    - If a neuron is removed, its activated value is set to be 0
    - Removal is done for each training example individually
2. Improves the robustness

Dropout

1. For actiavted values of each training exampling in the lth layer ($l = 1, \ldots, L-1$)
   - Set the dropout rate to be $p^{[l]}$
   - Randomly assign them to be 0 using probability $p^{[l]}$
   - Need to rescale the remainings by $(1 - p^{[l]})^{-1}$ to compensate information loss
2. Dropout is implemented using a mask matrix for each layer

Dropout



$\boldsymbol{x}$
$a_1^{[1]}$
$a_2^{[1]}$
$a_3^{[1]}$
$a_4^{[1]}$
$a_5^{[1]}$
$a_6^{[1]}$
$a_1^{[2]}$
$a_2^{[2]}$
$a_3^{[2]}$
$a_4^{[2]}$
$a_1^{[3]}$
Input layer
1st hidden layer
2nd hidden layer
Output layer

Dropout -- training procedure

1. Let $\boldsymbol{M}^{[l]} \in \{0,1\}^{n \times d^{[l]}}$ be a mask matrix
   - Each element of $\boldsymbol{M}^{[l]}$ is a Bernoulli random variable with success probability $1 - p^{[l]}$
2. Forward propagation

$$\boldsymbol{A}_d^{[l]} = \frac{\boldsymbol{A}^{[l]} \circ \boldsymbol{M}^{[l]}}{1 - p^{[l]}}.$$

3. Backpropagation: as usual
   - No additional model parameters are introduced for dropout

Dropout -- training procedure

1. Dropout is multiplying Bernoulli random noise to the neural network
2. Generally, we can add or multiply other noise to the neural network
   - Add noise to the training data
   - Add noise to the weights
   - Perturb the labels

Early stopping

1. The performance on the training dataset improves as training procedure precedes
2. It may overfit the training dataset
   - The performance on the validation/test dataset may not always improve
3. Early stopping monitors the performance on the validataion set
   - Stop the training if the performance does not improve on the validation set

Ensembling

1. To improve generalization of the model, we may
   - Build several models and take their average as the final model
     - ▷ For example, using different initializations
   - Use resampling methods to generate different training sets
     - ▷ This method is commonly used to get uncertainty of the deep learning model