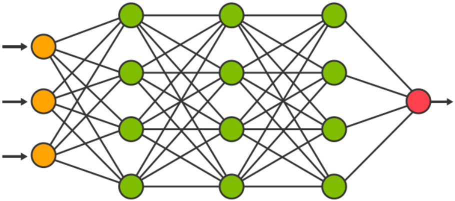
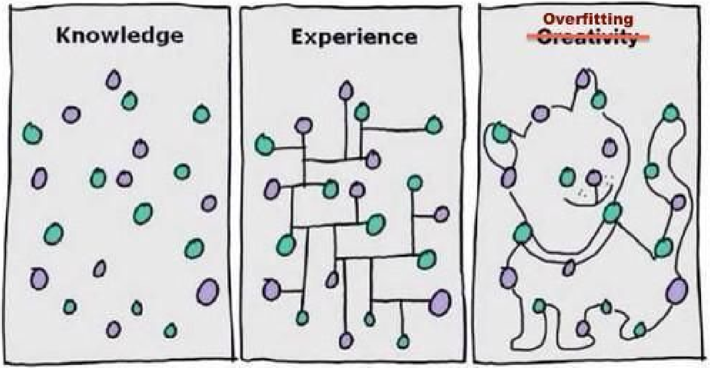
Introduction to Neural Networks

ML Instruction Team, Fall 2022

CE Department
Sharif University of Technology



Problem: Over-fitting in a Neural Network



Solution 1: L1/L2 regularization

- just like a classic regularizer
- sum the regularizer term for every layer weight

$$L = \frac{1}{N}\sum_{i=1}^{N} L(\phi(x_i), y_i) + \lambda \sum_{i,j,k} R(W_{j,k}^{(i)})$$

Solution 1: L1/L2 regularization

- review

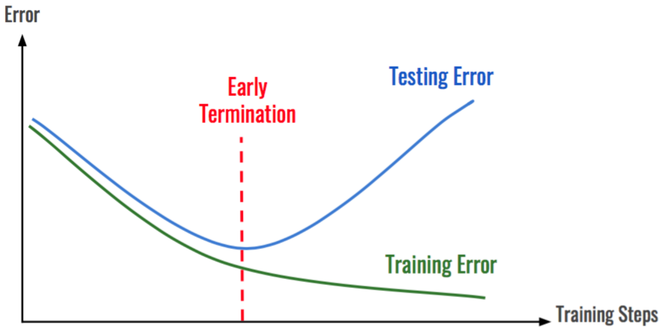
$$L1 : R(w) = |w|$$
$$L2 : R(w) = w^2$$

- you can also combine the two different regularizers (Elastic net)

$$R(w) = \beta w^2 + |w|$$

Solution 2: Early Stopping

- Stop learning when the validation error is Minimum



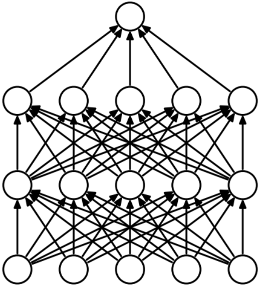
Solution 3: Dropout

Problem: Vanishing/Exploding Gradients

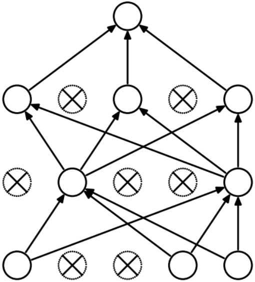
Solution: Batch Norm layer

Regularization: Dropout

- Randomly set some of neurons to zero in forward pass.



(a) Without using dropout



(b) After using dropout.

Figure: Behavior of dropout at training time. Source

Regularization: Dropout

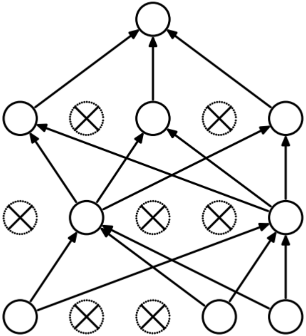


Figure: Source

- Dropout:
  - Prevents co-adaptation of features (forces network to have redundant representations).
  - Can be considered a large ensemble of models sharing parameters.

Dropout: Test Time

- Dropout makes output of network random!

$$y = f_W(x, z)$$

z: random mask
x: input of the layer
y: output of the layer

- We want to ”average out” the randomness at test time:

$$y = f(x) = \mathbb{E}_z[f(x, z)] = \int p(z) f(x, z) dz$$

- Can we calculate the integral exactly?

Dropout: Test Time

- Dropout makes output of network random!

$$y = f_W(x, z)$$

z: random mask
x: input of the layer
y: output of the layer

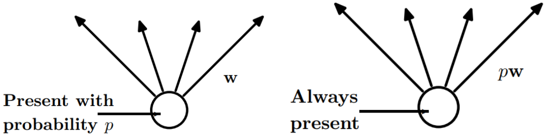
- We want to "average out" the randomness at test time:

$$y = f(x) = \mathbb{E}_z[f(x, z)] = \int p(z) f(x, z) dz$$

- Can we calculate the integral exactly?
- We need to approximate the integral.

Dropout: Test Time

- At test time neurons are always present and its output is multiplied by dropout probability:



(a) Without using dropout (b) After using dropout.

Figure: Behavior of dropout at test time. Source

Regularization: Adding Noise

- We've seen a common approach for regularization thus far:
  - **Training**: Add some kind of randomness (z) :

$$y = f_W(x, z)$$

  - **Testing**: Average out the randomness:

$$y = f(x) = \mathbb{E}_z[f(x, z)] = \int p(z) f(x, z) dz$$

Regularization: Adding Noise

- We've seen a common approach for regularization thus far:
  - **Training**: Add some kind of randomness (z) :

$$y = f_W(x, z)$$

  - **Testing**: Average out the randomness:

$$y = f(x) = \mathbb{E}_z[f(x, z)] = \int p(z)f(x, z)dz$$

- Adding noise is another way to prevent a neural network from overfitting on the training data. In every iteration, a random noise is added to the outputs of the layer, preventing consequent layers from co-adapting too much to the outputs of this layer.

Batch Normalization

Input: $x : N \times D$ Learnable Parameters: $\gamma, \beta : D$
Output: $y : N \times D$ Intermediates: $\mu, \sigma : D$, $\hat{x} : N \times D$

$$\mu_j = \text{(Running) average of values seen during training}$$
$$\sigma_j^2 = \text{(Running) average of values seen during training}$$
$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$
$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Batch Normalization

- Batch normalization is done along with **C** axis in convolutional networks:

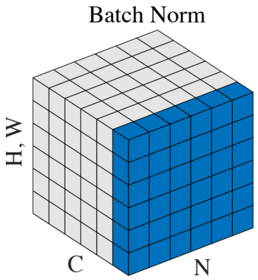


Figure: Batch normalization in CNNs Source.

- ▷ BN for FCNs: $x, y : N \times D \rightarrow \mu, \sigma, \gamma, \beta : 1 \times D$
- ▷ BN for CNNs: $x, y : N \times C \times H \times W \rightarrow \mu, \sigma, \gamma, \beta : 1 \times C \times 1 \times 1$
- ▷ In both cases: $y = \gamma(x - \mu)/\sigma + \beta$

Thank You!

Any Question?

References