Dropout

상태	DNN
담당자	

Overfitting
Dropout

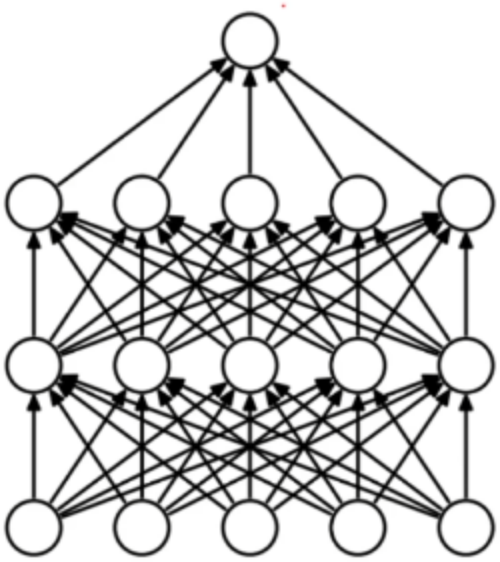
Overfitting

Training set에서는 성능이 좋은 모델이 testing set에서는 성능이 좋지 않을 수 있다. Training data와 testing data가 완전히 같지 않기 때문.

Overfitting을 방지할 수 있는 방법으로 Dropout 을 소개한다.

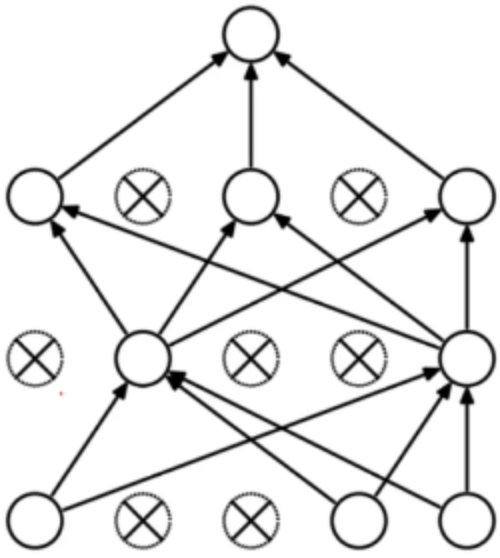
Dropout

Hidden layer의 뉴런을 무작위로 골라 삭제하면서 학습하는 방법



(a) Standard Neural Net

Standard한 fully-connected 신경망은 모든 노드에 신호가 전달된다. 하지만 드롭아웃을 적용하게 된다면?



(b) After applying dropout.

dropout이 설정된 비율에 따라 몇몇 노드들은 신경망에 연결되지 않아 학습에 사용되지 않는다. 이렇게 만든 신경망에 대해 backpropagation을 적용해 신경망의 가중치를 업데이트한다. 단, 테스트를 진행할 때에는 모든 노드들을 고려해서 진행한다. (`model.train()` 모드와 `model.eval()` 모드를 구분해야 한다. 후자에서는 `dropout=False`로 설정됨.)

```
class dnn_dropout(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear1 = nn.Linear(784, 256, bias=True).to(device)
        self.linear2 = nn.Linear(256, 256, bias=True).to(device)
        self.linear3 = nn.Linear(256, 10, bias=True).to(device)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=drop_prob)

        nn.init.xavier_uniform_(self.linear1.weight)
        nn.init.xavier_uniform_(self.linear2.weight)
        nn.init.xavier_uniform_(self.linear3.weight)

        self.model = torch.nn.Sequential(self.linear1, self.relu, self.dropout,
                                         self.linear2, self.relu, self.dropout,
                                         self.linear3)

    def forward(self, x):
        return self.model(x)
```