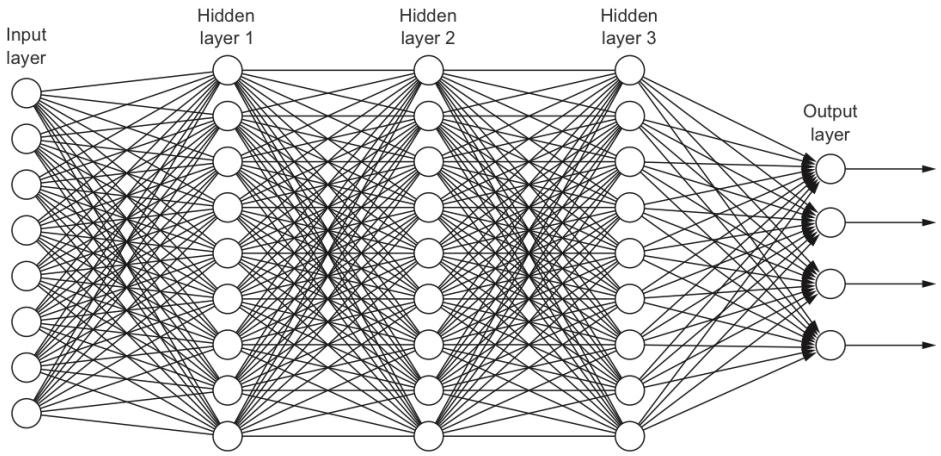
ESC 2024 Winter Session 5th Week

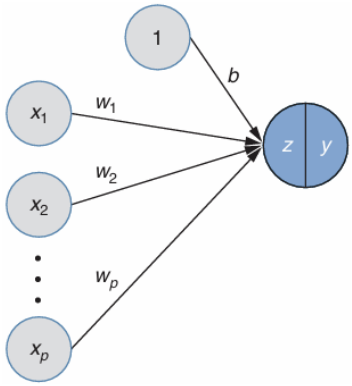
Probabilistic Neural Network 1

김근영, 김준엽, 김태영, 정예준

1. Fully Connected Neural Networks (fcNNs)



위 그림은 각각 9개의 neuron으로 구성된 3개의 hidden layer을 가지고 있는 **fully connected neural network(fcNN)**의 예시이다. Layer 안에 있는 각 neuron들이 다음 layer에 있는 neuron들과 모두 연결되어 있어서 fully connected neural network 혹은 densely connected NN이라고 부른다.



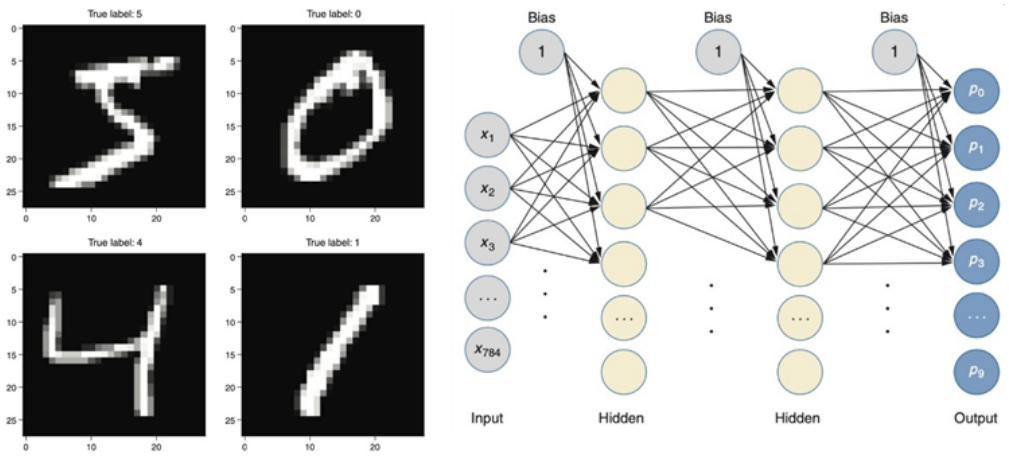
각 뉴런은 입력값 x_i들과 가중치 w_i들을 모두 곱한 값들을 더하고 bias b를 더한 값 z를 생성한다.

$$z = x_1 w_1 + ... + x_p w_p + b$$

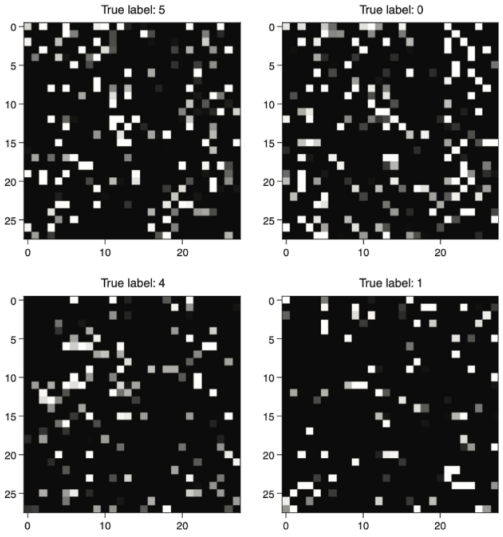
그 후 activation function(e.g. sigmoid function, softmax function, ReLU function)에 z값을 넣어 최종 출력값(혹은 다음 뉴런의 입력값)을 생성한다.

$$y = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$$

이제 fcNN을 image classification 문제에 적용해보자.



왼쪽 그림은 MNIST digit 데이터셋으로, 70000개의 0부터 9까지의 손글씨 digit image 데이터가 들어있다. 각 image는 28×28 pixels로 구성되었고, 각 pixel은 명암에 따라 0부터 255까지의 값으로 할당되었다. 데이터셋을 60000개의 training 데이터와 10000개의 test 데이터로 나누었고, simple neural network는 2D image를 사용할 수 없어서 28·28=784 크기의 1D input vector로 flatten하여 오른쪽 fcNN에 training 데이터를 이용해 fitting 시켰다. 그 후 test 데이터를 이용해 accuracy를 구하였다.

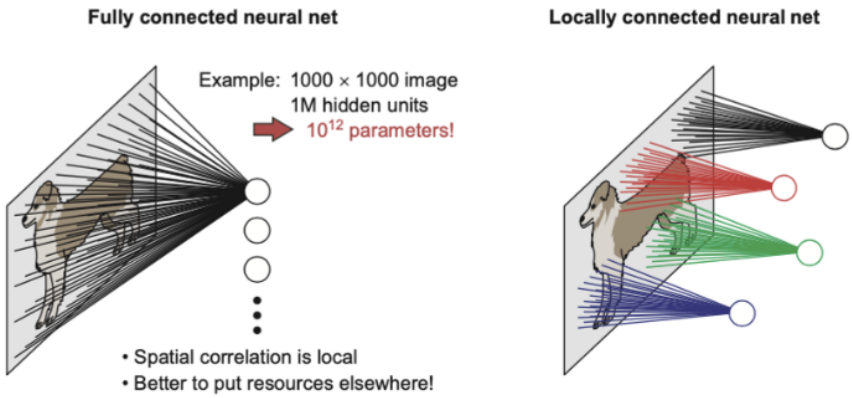


이제 하나의 실험을 해볼 것이다. 위 그림은 MNIST 데이터셋을 network에 사용하기 전 784개의 pixel들을 무작위로 섞은 것이다. 이 데이터를 이용하여 accuracy를 구해보면 어떻게 될까? 놀랍게도 기존 데이터를 이용해 구한 accuracy와 같게 나온다.
그러므로 fcNN은 input 데이터의 order에 영향을 받지 않는다는 것을 알 수 있다. 이러한 특성 때문에 fcNN을 **permutation invariant NN**이라고도 부른다. 그러나 실제 image 데이터는 permutation

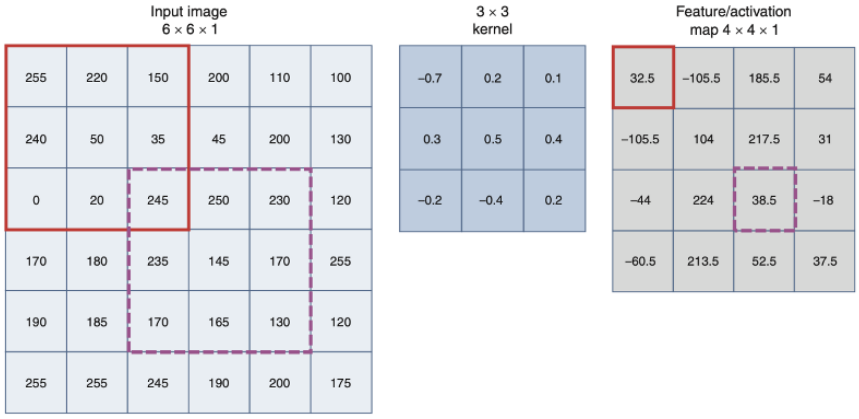
invariant하지 않다. Pixel들을 무작위로 섞으면 사람이 image를 인식하기 어려워진다. 따라서 스프레드시트 데이터같은 열의 순서가 상관없는 데이터엔 fcNN을 사용하기에 적합하지만, image 데이터같은 순서나 정렬이 중요한 데이터엔 fcNN을 사용하기에 적합하지 않다.

2. Convolutional Neural Networks (CNNs)

앞서 살펴본 fcNN을 image classification에 적용할 때 발생한 문제점을 보완할 수 있는 **convolutional neural networks(CNNs)**에 대해 알아보자.



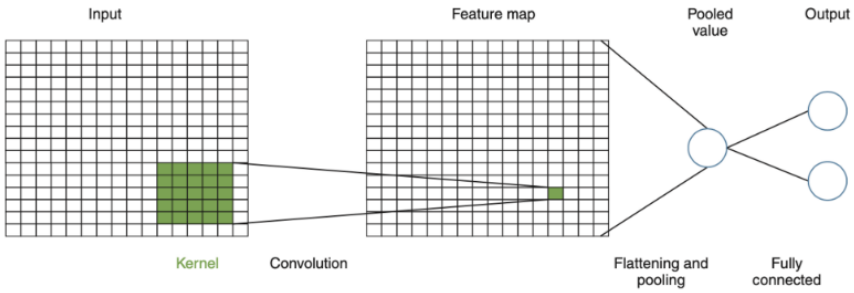
CNN의 가장 핵심 아이디어는 연속한 두 layer의 모든 neuron을 연결하는 대신, 이웃한 pixel들의 neuron들만을 다음 뉴런 하나에 각각 연결시키는 것이다. 이런 간단한 트릭을 통해 image의 local structure을 학습할 수 있게 되고, 모델에 필요한 parameter들의 수 역시 줄일 수 있다.



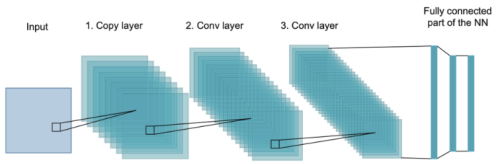
위 그림은 6×6 image 데이터와 3×3 kernel인데, 다음 layer에 속한 각각의 neuron은 이전 layer의 한 **kernel** 안에 속한 neuron들과 연결된다.

$$z = x_1 w_1 + ... + x_9 w_9 + b$$

이렇게 계산된 z는 output matrix의 한 원소가 된다. Kernel을 다음 위치로 shift 시켜서 다음 z를 계산해나간다. 이때 각 위치에서 같은 kernel을 사용하므로 사용되는 가중치 w_i들은 모두 같다. 이렇게 계산된 최종 output matrix를 **activation map** 혹은 **feature map**이라고 하고, 이 과정을 **convolution**이라고 부른다.



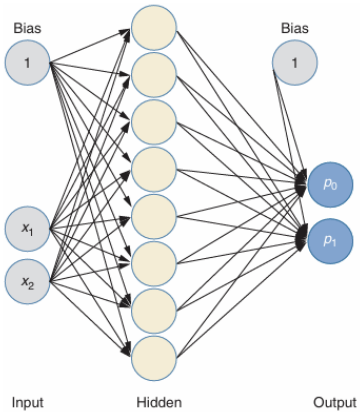
앞서 구한 feature map의 대푯값(e.g. 최댓값)을 추출하는 과정을 **pooling**이라고 한다. 위 그림은 feature map 전체 크기에서 pooling하여 한 값을 추출하였지만, kernel이 작동하는 방식과 비슷하게 크기를 $n \times n$으로 설정할 수 있고, 이동하는 보폭(stride)도 설정하여 최종적으로 matrix형태로 추출할 수 있다.
이제 CNN의 전체 과정을 살펴보자.



Input 데이터에 kernel을 여러 개 사용하게 된다면 feature map 역시 각각의 kernel마다 생성할 수 있다. 이렇게 생성된 feature map들을 각각 pooling하여 다음 layer의 input으로 넘긴다. 이렇게 convolution과 pooling을 여러번 반복하여 image의 특징만을 남긴 후, 최종 pooling된 데이터를 flatten하여 1차원의 벡터로 만들어서 fully connected NN의 input으로 넘겨 최종 output을 생성한다.

3. Deriving a Loss Function for a Classification Problem

우리는 linear regression problem에선 mean squared error(MSE), 즉 데이터와 커브 사이의 square distances를 이용하여 loss function을 만들었다. 이때, square distance 대신 absolute difference를 이용하여 loss function을 만들 수 있지 않나 하는 의문이 들 수 있다. 이에 대해 **maximum-likelihood approach(MaxLike)**를 통해 근거를 찾을 수 있다. Maxlike approach를 통해 classification problem에서 categorical cross entropy를 loss function으로 사용하는 근거 역시 찾을 수 있다.
먼저 **binary classification** problem을 생각해보자.



Training 데이터 (x_i, y_i)가 있다고 가정하자(*위 그림에서의 x_1과 데이터셋의 x_1은 다르다. 데이터셋 $x_i = (x_{i1}, x_{i2})$라고 생각하자.). NN에서 weight값들과 input x_i가 주어져 있을때, likelihood는 모델이 correct class y_i에 할당될 확률, 즉 $y_i = 0$이면 $p_0(x_i)$, $y_i = 1$이면 $p_1(x_i)(= 1 - p_0(x_i))$이다. 따라서 전체 training set의 likelihood는 다음과 같이 x_i, weight들에 대한 식으로 표현된다.

$$\prod_{j \in \{i \in [n] | y_i = 0\}} p_0(x_j) \prod_{j \in \{i \in [n] | y_i = 1\}} p_1(x_j)$$

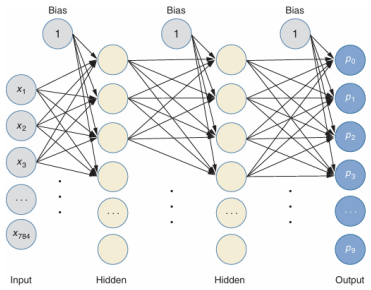
MaxLike approach에 따라, x_i들을 given이라고 생각하고, 위의 식이 최댓값을 갖게 하는 NN의 weight 값들을 찾아주면 된다. 그런데 training 데이터 수가 많아지면 0과 1사이에 있는 확률 값들을 더 많이 곱해주게 되어 underflow 문제가 발생할 수 있다. 이러한 문제를 해결하기 위해 likelihood에 자연로그를 취한다. 자연로그를 취해도 식이 최댓값을 갖게 하는 NN의 weight값들은 보존된다.

$$\sum_{j \in \{i \in [n] | y_i = 0\}} log(p_0(x_j)) + \sum_{j \in \{i \in [n] | y_i = 1\}} log(p_1(x_j))$$

위의 식은 training 데이터 수 n값에 의존하게 되므로 $1/n$으로 나눠준다. 또한, DL framework에서는 일반적으로 loss function을 minimize하도록 $-$를 취해준다. 따라서, 식의 형태를 통해 MaxLike approach 결과는 우리가 아는 crossentropy loss function 결과와 동치임을 알 수 있다.

$$crossentropy = -\frac{1}{n}(\sum_{j \in \{i \in [n] | y_i = 0\}} log(p_0(x_j)) + \sum_{j \in \{i \in [n] | y_i = 1\}} log(p_1(x_j)))$$

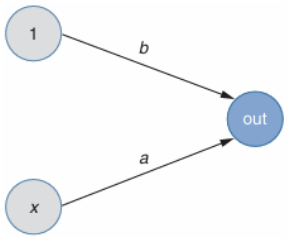
위와 동일한 과정을 통해 **Multiclassification** problem에서 역시 MaxLike approach 결과가 crossentropy loss function의 결과와 동치임을 알 수 있다.



$$crossentropy = -\frac{1}{n}(\sum_{j \in \{i \in [n] | y_i = 0\}} log(p_0(x_j)) + \sum_{j \in \{i \in [n] | y_i = 1\}} log(p_1(x_j)) + ... + \sum_{j \in \{i \in [n] | y_i = K-1\}} log(p_{K-1}(x_j)))$$

4. Deriving a Loss Function for a Regression Problem

MaxLike approach를 통해 **linear relationship**을 갖는 regression problem을 살펴보자.



Training 데이터 (x_i, y_i)가 있다고 가정하자. 각 input x_i에 대해 y_i는 normal distribution을 따른다.

$$f(y_i; \mu_{x_i}, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \mu_{x_i})^2}{2\sigma^2}}$$

위의 simple network에서 $\mu_{x_i} = ax_i + b$의 linear dependency를 가지므로 전체 training set의 likelihood는 다음과 같다.

$$\prod_{i=1}^{n} f(y_i; \mu_{x_i}, \sigma) = \prod_{i=1}^{n} f(y_i; ax_i + b, \sigma)$$

이 식을 최대화하는 a, b를 구하면 되는데, 앞서 얘기한 underflow 문제에 따라 likelihood에 자연로그를 취해준다. DL framework에서는 일반적으로 loss function을 minimize하도록 $-$를 취해준다. 다음 식을 최소화하는 a, b는 위의 식을 최대화하는 a, b랑 같게 된다.

$$-\sum_{i=1}^{n} log(f(y_i; ax_i + b, \sigma))$$

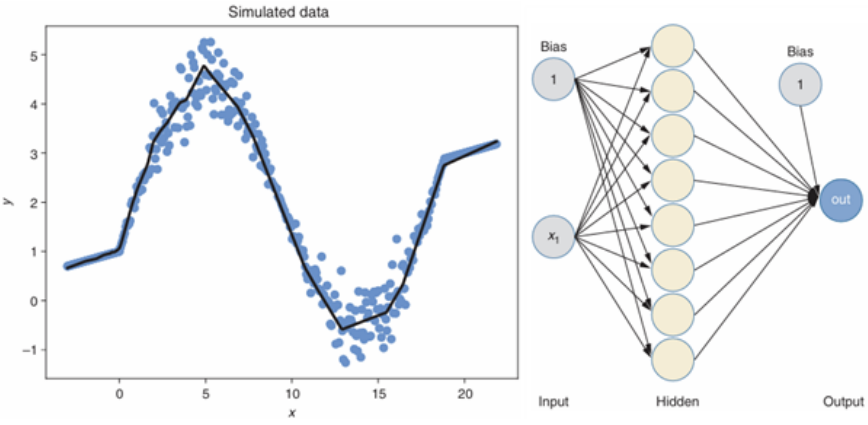
$w = (a, b)$라 할 때 이제 다음 MaxLike approach를 통해 구한 결과가 MSE의 결과랑 동치임을 알 수 있다.

$$argmin_w\{\sum_{i=1}^{n} -log(f(y_i; ax_i + b, \sigma))\} \iff argmin_w\{\sum_{i=1}^{n} -log(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \mu_{x_i})^2}{2\sigma^2}})\}$$

$$\iff argmin_w\{\sum_{i=1}^{n} -log(\frac{1}{\sqrt{2\pi\sigma^2}}) + \frac{(y_i - \mu_{x_i})^2}{2\sigma^2}\} \iff argmin_w\{\sum_{i=1}^{n} \frac{(y_i - \mu_{x_i})^2}{2\sigma^2}\}$$

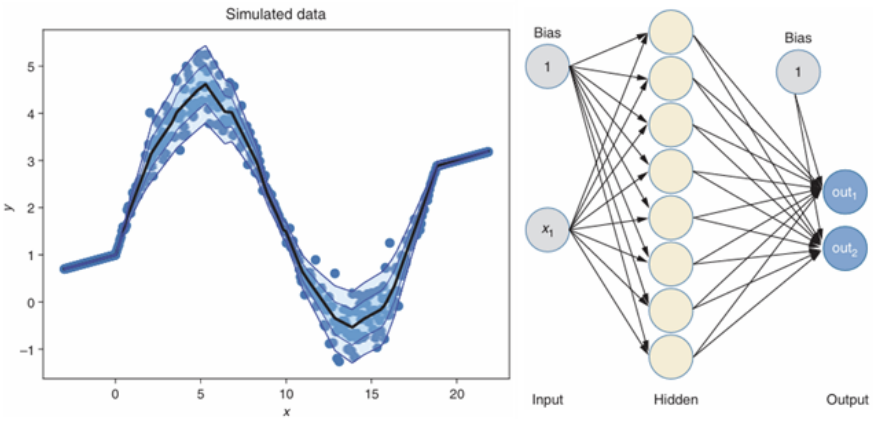
$$\iff argmin_w\{\frac{1}{n}\sum_{i=1}^{n}(\mu_{x_i} - y_i)^2 \iff argmin_w\{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2\} = MSE$$

위의 식에서 linear relationship의 가정이 없어도, 즉 **non-linear relationship**이더라도 두번째부터 마지막 동치가 성립하므로 MaxLike approach의 결과가 MSE의 결과랑 동치임을 알 수 있다.



$$Y_{x_i} \sim N(\mu_{x_i}, \sigma^2)$$

마지막으로 **nonconstant variance**일 때의 NN까지 살펴보자.



$$Y_{x_i} \sim N(\mu_{x_i}, \sigma^2_{x_i})$$

σ_x도 x에 의존하게 되므로 output을 두개 가지도록 NN을 설계해야 한다. 위의 과정과 비슷한 과정을 거친 MaxLike approach를 통해 최종 loss function을 다음과 같이 나타낼 수 있다.

$$loss = \sum_{i=1}^{n} -log(\frac{1}{\sqrt{2\pi\sigma^2_{x_i}}}) + \frac{(y_i - \mu_{x_i})^2}{2\sigma^2_{x_i}}$$