2. 인공신경망 기초 및 학습

Hosik Choi

Dept. of Applied Statistics, Kyonggi Univ.

June, 2019

Outline

1. 인경신경망의 기초
   - 인경신경망의 기본 개념
   - XOR 분류문제
2. 인공신경망 모형
3. 인공신경망의 학습
   - 역전파 알고리즘의 이해
   - 연습: 단순회귀분석에서의 SGD방법
   - 실습1: 다중클래스 분류
4. 정리
5. Toward 딥러닝
   - 실습2: 이진분류(계속)

인공 신경망 (Artificial Neural Network)

- 기계 학습 (Machine Learning)의 한 분야로, 생물학의 뇌 구조 (신경망)를 모방하여 만든 수학적 모형
- 인간의 뇌가 문제를 해결하는 방식과 유사하게 구현

인공 신경망의 역사

- 1943년, W.S.McCulloch와 W.Pitts의 MP-model
- 1957년, F.Rosenblatt의 single layer perceptron
- 1986년, D.E.Rumelhart의 multilayer perceptrons
- 2006년, G.E.Hinton의 Deep Neural Network에서의 혁신적인 학습 방법 개발

생물학적 뉴런과 모방

- 뇌는 수많은 뉴런이 시냅스를 통해 네트워크를 이루며, 사고 · 판단 · 기억과 학습 등의 고등 기능부터 잠 · 욕구 등과 같은 원초적 기능까지 모든 기능을 관장.
- 뇌에는 수많은 뇌세포가 있으며, 이 중 약 10%는 뉴런이 차지.
- 각 뉴런은 정보 수용, 연산처리 및 출력 전송 등과 같은 여러 기능을 담당.

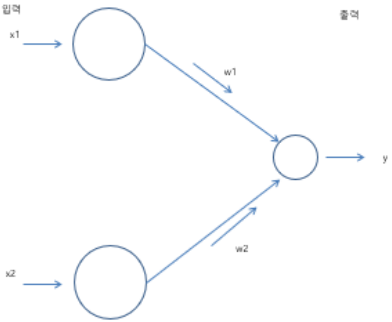
생물학적 신경망	인공신경망
세포체	노드(Node)
수상돌기	입력 (Input)
축삭 (Axon)	출력 (Output)
시냅스	가중치 (Weight)

- 뉴런은 전기신호를 통해 다른 뉴런에게 정보를 전달

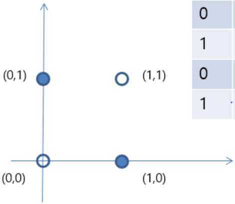
1. 두 개의 뉴런은 어느 정도의 전기신호를 전달하는가?
2. 임계값(critical value, cutoff)은 어느 정도로 설정해야 하는가?
3. 임계값을 초과했을 때 어느 정도의 신호를 보내야 하는가?

- 결합의 강도를 w1과 w2라 할 때, 두 개의 뉴런으로부터 전달되는 전기신호의 총량은?
- $w_1 * x_1 + w_2 * x_2$ 여기서, w_1와 w_2를 네트웍의 웨이트라 부름.

4. 임계값 b : $w_1 * x_1 + w_2 * x_2 \geq b$를 만족하면 해당 뉴런이 activation됨.
5. 수신 신호의 양: $y = I(w_1 * x_1 + w_2 * x_2 \geq b)$



XOR 이범주 분류문제

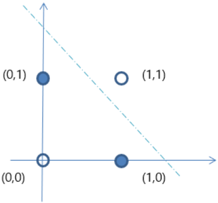


x1	x2	y
0	0	0
1	0	1
0	1	1
1	1	0

-Exclusive OR
-XOR 문제는 매우 단순하지만 선형적으로 구분할 수 없음.

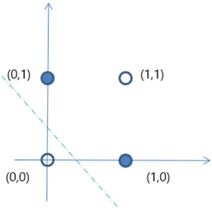
AND gate

x1	x2	y
0	0	0
1	0	0
0	1	0
1	**1**	**1**



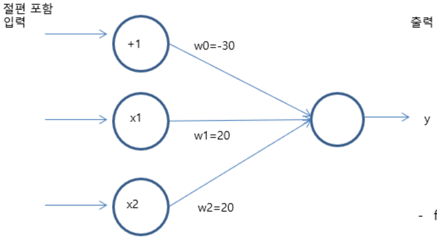
OR gate

x1	x2	y
0	0	0
1	**0**	**1**
0	**1**	**1**
1	**1**	**1**



XOR 문제 II

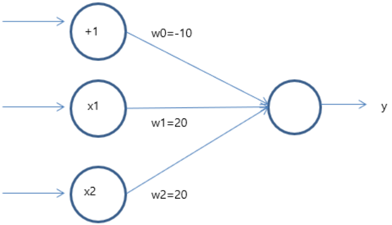
AND gate



x1	x2	f(x)	p(x)
0	0	-30	<1/2
1	0	-10	<1/2
0	1	-10	<1/2
1	1	10	>1/2

- f(x)= w0 * (+1) + w1*x1 + w2*x2

OR gate

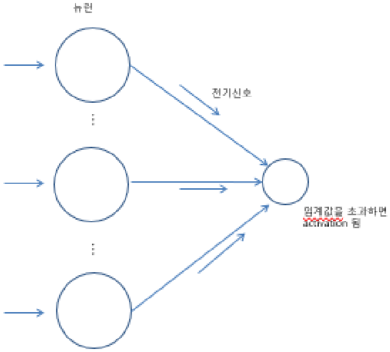


x1	x2	f(x)	p(x)
0	0	-10	<1/2
1	0	10	>1/2
0	1	10	>1/2
1	1	30	>1/2

Not(negation) gate

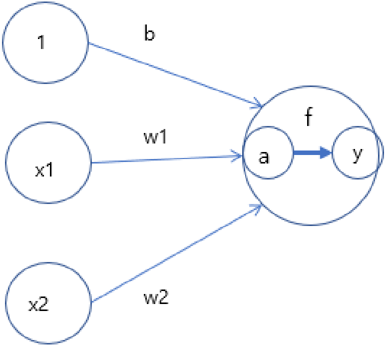
인공신경망 모형 I

- 입력층, 출력층, 은닉층



뉴런간의 정보 전달

인공 신경망의 함수구조(activation function)



- 앞의 예제에서 $a = b + w_1x_1 + w_2x_2$라 두면 $y = I(a \geq 0)$
- 여기서 $I(\cdot)$를 활성화함수(activation function)이라 함.

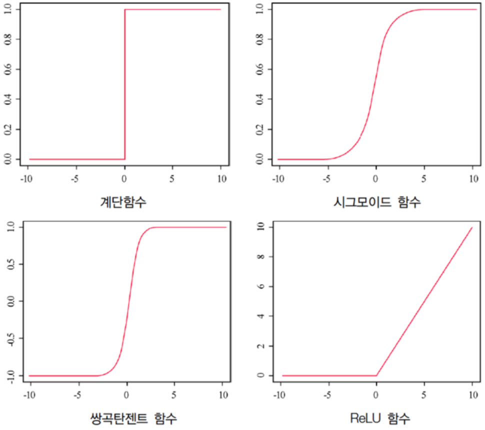


Figure: General activation functions

```
indicator <- function(a){
    if(a >= 0)
        1
    else
        0
}

relu <- function(a){
    if(a >= 0)
        a
    else
        0
}
```

인공신경망의 학습 I

- 손실함수(loss function)
- 미분과 기울기
- 연쇄법칙
- 계산그래프: 순전파와 역전파
- 가중치 갱신
    - 학습률 설정
    - 오버피팅(overfitting)

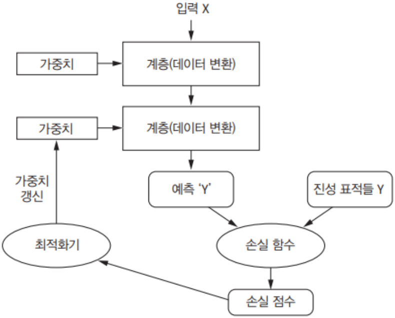


Figure: 구조, dlr book, 보기1.9

손실함수(loss function)

- target: t, predicted value y
- 제곱손실
  - 연속형 반응변수에 대한 손실의 제곱: $(t - y)^2$
- 교차엔트로피(cross-entropy)
  - 범주형 반응변수에 대한 손실: $-t \log y$

= label의 수가 2일 때 특별히, negative binomial log-likelihood(또는 logistic loss)라 함.

* 참고: 베르누이 시행과 최대우도추정방법(maximum likelihood estimation)

여러 손실(loss)함수들

- It measures "loss" or "cost" for the use of f

$$L(t, f(\mathbf{x}))$$

- regression: $t \in \mathbb{R}$
  - square: $(t - f(\mathbf{x}))^2$
  - quantile: $\rho_\tau(t - f(\mathbf{x})), \ 0 < \tau < 1$
- classification: $t \in \{-1, 1\}$
  - zero-one: $I[t \neq \phi(\mathbf{x})] = I[tf(\mathbf{x}) < 0]$
  - logistic: $\log\{1 + \exp(-tf(\mathbf{x}))\}$
  - exponential: $\exp(-tf(\mathbf{x}))$
  - hinge: $\max\{0, 1 - tf(\mathbf{x})\}$

- 여러 손실함수를 도시하여 보자.

미분과 기울기

수치미분(numerical differentiation)

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

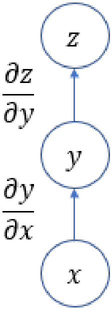
편미분: 예를 들어, 인자가 두 개(x, y)인 경우

$$\frac{\partial f(x,y)}{\partial x} = \lim_{h \to 0} \frac{f(x+h,y) - f(x,y)}{h}$$
$$\frac{\partial f(x,y)}{\partial y} = \lim_{h \to 0} \frac{f(x,y+h) - f(x,y)}{h}$$

```
numerical_diff <- function(f,x){
  h <- 1e-4
  u <- f(x+h)-f(x-h)
  l <- 2*h
  return(u/l)
}

numerical_pdiff <- function(f,x,y){
  h <- 1e-4
  u_x <- f(x+h,y) - f(x-h,y)
  u_y <- f(x,y+h) - f(x,y-h)
  l <- 2*h
  return(c(u_x,u_y)/l)
}
f2 <- function(x,y){
  return(3*x^2+y^2)
}
numerical_pdiff(f2,x=1,y=4)
```

연쇄법칙



$$\Delta z = \frac{\partial z}{\partial y}\Delta y$$

$$\Delta y = \frac{\partial y}{\partial x}\Delta x$$

$$\Delta z = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\Delta y$$

계산그래프: 순전파와 역전파

- 계산그래프(computational graph): 계산과정을 그래프로 표현
- 노드(node)와 에지(edge)로 구성
- 그래프상에서 국소적 계산을 연결(전파)하면 전체 계산이 완성

- 예제1. 한개 당 100원인 사과 2개를 샀을 때, 지불금액은? 단, 10%의 소비세 부과됨.

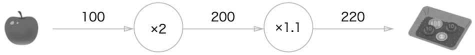


Figure: 계산그래프 (https://github.com/WegraLee/deep-learning-from-scratch)

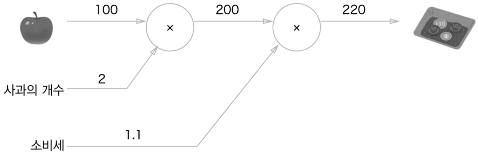


Figure: 계산그래프2 (https://github.com/WegraLee/deep-learning-from-scratch)

- 예제2. 100원/개인 사과 2개, 150원/개 귤을 3개를 샀을 때, 지불금액은? 단, 10%의 소비세 부과됨.

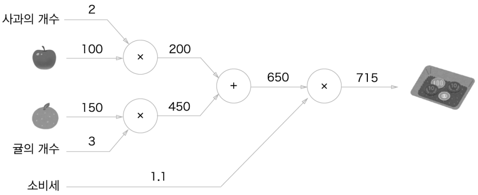


Figure: 계산그래프3 (https://github.com/WegraLee/deep-learning-from-scratch)

- 예시 그래프에서 왼쪽에서 오른쪽을 진행하는 단계를 순전파(forward propagation)이라 부름.
- 반대 방향을 역전파(backward propagation)이라 함.
- 사과값(x) 변동에 따른 지불금액(L)의 변화량은?

$$\frac{\partial L}{\partial x}$$

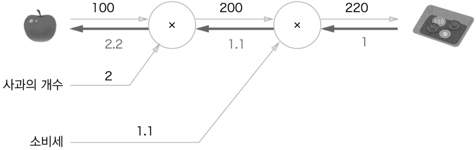


Figure: 역전파에 따른 미분값 전달 (https://github.com/WegraLee/deep-learning-from-scratch)

- 사과 가격의 (편)미분은 2.2
- 사과 개수의 (편)미분은 110
- 소비세의 (편)미분은 200

- x와 y의 덧셈($z = x + y$)에 대한 역전파

$$\frac{\partial z}{\partial x} = 1, \frac{\partial z}{\partial y} = 1$$

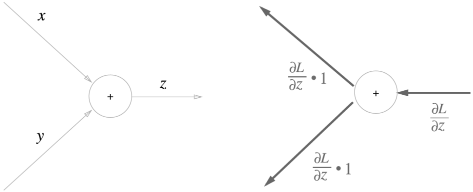


Figure: 덧셈노드의 역전파 (https://github.com/WegraLee/deep-learning-from-scratch)

- x와 y의 곱셈($z = xy$)에 대한 역전파

$$\frac{\partial z}{\partial x} = y, \frac{\partial z}{\partial y} = x$$

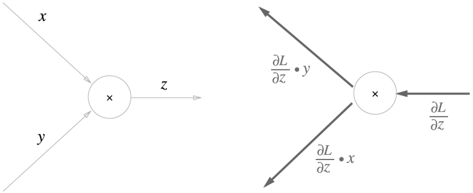


Figure: 곱셈노드의 역전파 (https://github.com/WegraLee/deep-learning-from-scratch)

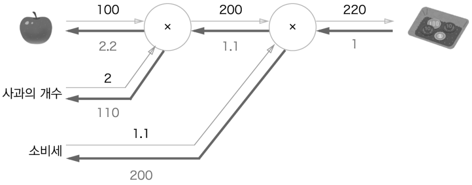


Figure: 예제1의 역전파 (https://github.com/WegraLee/deep-learning-from-scratch)

연습

(1) 예제2에 대한 역전파 그래프를 완성하여보자.

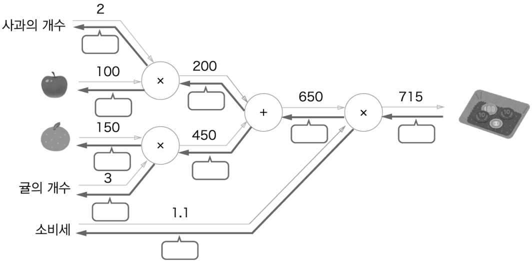


Figure: 예제2의 역전파 (https://github.com/WegraLee/deep-learning-from-scratch)

(2) 나눗셈노드, exp노드, log노드 등에 대한 역전파 그래프를 생각하여보자.

가중치 갱신

- 순전파 후 역전파를 통해 가중치 수정함
- Gradient descent algorithm (기울기 강하 알고리즘)

- 특정 목적 함수 $L(\theta)$를 최소화하는 θ를 찾기 힘든 경우에 사용하는 대표적인 반복 알고리즘.
- $-\frac{\partial L(\theta)}{\partial \theta}|_{\theta=\theta_0}$는 $\theta = \theta_0$일 때 $L(\theta)$를 가장 빠르게 감소시키는 방향이라는 점에 착안.

1. 초기값 설정: $\theta = \theta_0$
2. 현재해 $\theta^{(t)}$와 학습률(η)로 해를 갱신

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \frac{\partial L(\theta)}{\partial \theta}|_{\theta=\theta^{(t)}}$$

3. 수렴할 때까지 2과정 반복

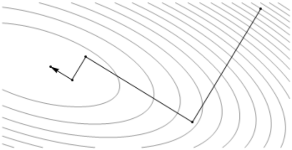


Figure: trace of gradient descent algorithm(https://takisword.wordpress.com)

확률적 경사하강법

- SGD(stochastic gradient descent)
- batch, mini-batch 등 자료의 일부분을 활용하여 경사하강법을 반복 적용하는 최적화방법
- 업데이터할 때, 모든 자료, batch를 사용할 때보다 적은 계산량을 필요로 함.
- 수렴성이 이론적으로 보장됨
- 모멘텀(momentum) / AdaGrad / RMSProp / Adam

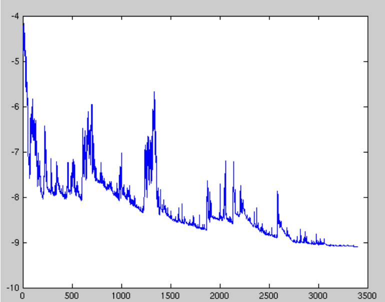
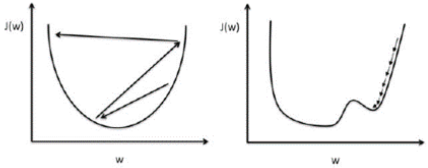


Figure: function value trace of stochastic gradient algorithm

(https://en.wikipedia.org/wiki/Stochastic_gradient_descent)

학습률(η)의 선택

- η_t: t시점에서의 학습률(learning rate)
- 학습률이 너무 크거나 너무 작으면 좋은 추정값을 얻을 수 없음.
- Learning rate가 너무 크거나 너무 작을 경우 생기는 문제점들



- 처음에는 비교적 큰 값으로 시작하지만, t가 증가함에 따라 η_t를 줄여나간다.
- 이론적으로 $\eta_t \propto 1/t$이면 역전파 알고리즘을 사용하였을 때, 손실함수가 국소최소값으로 수렴한다는 사실이 알려져 있음.

연습: 단순회귀분석에서의 SGD방법

n개의 (input, output) 짝으로 이루어진 훈련자료 $(x_i, y_i)_{i=1}^{n}$가 주어져 있다고 하자.

- 절편이 0인 단순 회귀분석에서의 gradient descent 방법
  1. 절편이 0인 단순회귀모형
  2. 적합한 손실함수
  3. 모수(매개변수)에 대한 미분
  4. 가중치 갱신
- 확률적 경사하강방법으로 수정하여보자.

실습1: 다중클래스 분류 I

- iris 자료는 붓꽃 3종의 4개의 꽃받침, 꽃잎의 길이 넓이 4개로 조사된 자료
  - 다중클래스 분류(multi-class classification)
  - multi-label vs. multi-class classification
  - softmax 분류방법

자료 살펴보기

```
> library(ggplot2); library(GGally)
>  pairs(iris[1:4],
      main = "Iris, 3 species",
      pch = 21, bg = c(1, 2, 3)[unclass(iris$Species)])

> ggpairs(iris, columns=1:4, aes(color=Species)) +
ggtitle("Iris, 3 species")
```

자료준비

```
> library(keras)
> iris[,5] <- as.numeric(as.factor(unlist(iris[,5]))) - 1
> iris <- as.matrix(iris)
> dimnames(iris) <- NULL
> iris_x <- normalize(iris[,1:4])

# 훈련표본과 검증표본
> ind <- sample(2, nrow(iris_x), replace=TRUE, prob=c(0.67, 0.33))
> tr_iris <- iris_x[ind==1, ]
> te_iris <- iris_x[ind==2, ]

# 모형 예측변수
> tr_targets <- iris[ind==1, 5]
> te_targets <- iris[ind==2, 5]

> to_one_hot <- function(labels, dimension = 3) {
  results <- matrix(0, length(labels), dimension)
  ulevels <- unique(labels)
  for(j in 1:length(ulevels))
    results[labels==ulevels[j], j] <- 1
```

```
  results
}
> one_hot_labels <- to_one_hot(iris$Species)

# One-Hot 인코딩: 훈련예측변수/ 검증예측변수
> tr_labels <- to_categorical(tr_targets)
> te_labels <- to_categorical(te_targets)
```

모형 개발

```
> set.seed(369)
# 초기화
> model <- keras_model_sequential()

# 망 구축
# 4 inputs -> [8 hidden nodes] -> 3 outputs
> g1 <- model %>%
    layer_dense(units = 8, activation = 'relu', input_shape = c(4)) %>%
    layer_dense(units = 3, activation = 'softmax')

> summary(g1)
```

망 컴파일

```
> g1 %>% compile(
    loss = 'categorical_crossentropy',
    optimizer = 'adam', metrics = 'accuracy')

> history1 <- g1 %>% fit(
    tr_iris, tr_labels,
    epochs = 200, batch_size = 5, validation_split = 0.1)

> listviewer::jsonedit(history1, model="view")
```

모형 수렴

```
> plot(history1$metrics$loss, main="Model Loss",
  xlab = "epoch", ylab="loss", type="l")
> lines(history1$metrics$val_loss, col=2, lty=2)
> legend("topright", c("train","test"), col=c(1, 2), lty=c(1,2))
```

모형 정확성

```
> plot(history1$metrics$acc, main="Model Accuracy",
  xlab="epoch", ylab="accuracy", type="l", ylim=c(0,1))
> lines(history1$metrics$val_acc, col=2, lty=2)
> legend("bottomright", c("train","test"), col=c(1,2), lty=c(1,2))
```

```
> pred_mat <- g1 %>% predict(te_iris)
> pred <- apply(pred_mat,1,which.max)
# confusion Matrix
> table(te_targets, pred)
```

- confusion matrix를 작성하여 보자.

연습

```
# 2번 모형
> g2 <- model %>%
    layer_dense(units = 8, activation = 'relu', input_shape = c(4)) %>%
    layer_dropout(rate = 0.5) %>%
    layer_dense(units = 3, activation = 'softmax')

> summary(g2)

> g2 %>% compile(
    loss = 'categorical_crossentropy',
    optimizer = 'adam', metrics = 'accuracy')

> history2 <- g2 %>% fit(
    tr_iris, tr_labels,
    epochs = 500, batch_size = 5, validation_split = 0.1)

> listviewer::jsonedit(history2, model="view")
```

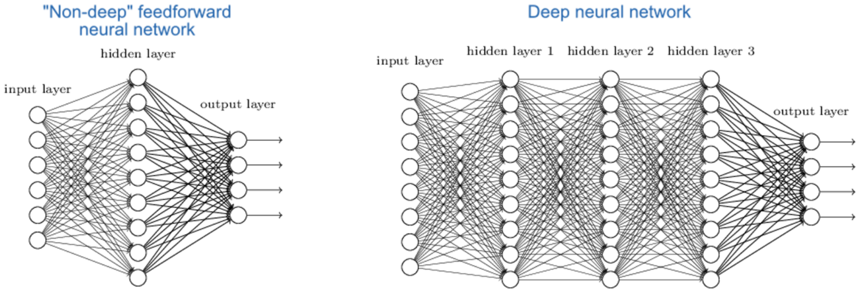
정리

Table: 최상단 층과 손실함수

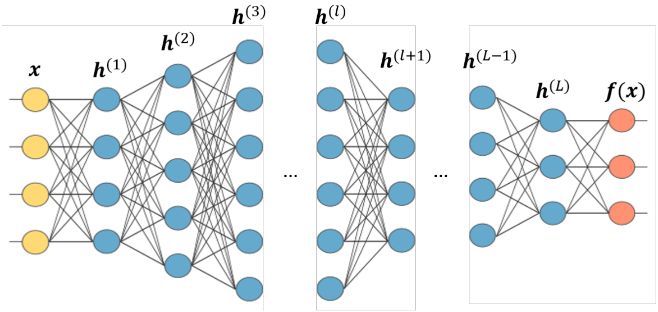
문제유형	최상단 층	손실함수
이항분류	시그모이드(sigmoid)	binary_crossentropy
다중 클래스분류		
단일 레이블	소프트맥스(softmax)	categorical_crossentropy
다중 레이블	시그모이드(sigmoid)	binary_crossentropy
회귀	없음	mse
회귀(0-1)	시그모이드	mse or binary_crossentropy

딥러닝이란? I

- 2개 이상의 중간층을 가지고 있는 신경망 모형



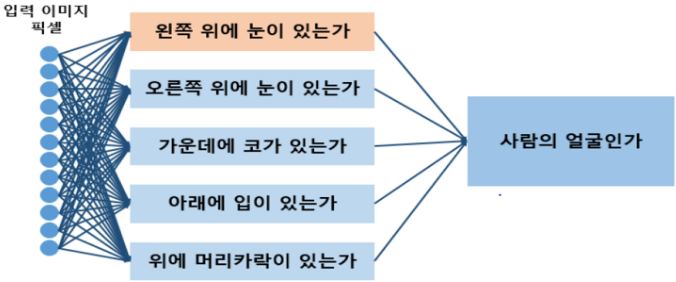
- Fully connected NN의 함수구조
- L개의 중간층 (Hidden layer)을 갖는 Neural Network 모형
  - $h^{(l)}, l = 0, 1, \cdots, L+1$을 l번째 중간층의 노드값
  - $h^{(0)} = \mathbf{x}$, $h^{(L+1)} = f(\mathbf{x})$



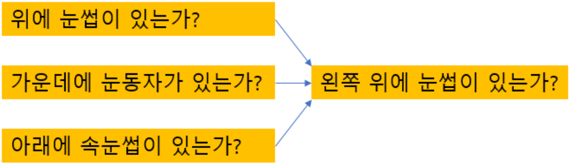
- 다음의 세가지 그림 중 사람의 얼굴을 찾아내는 문제를 생각하자.

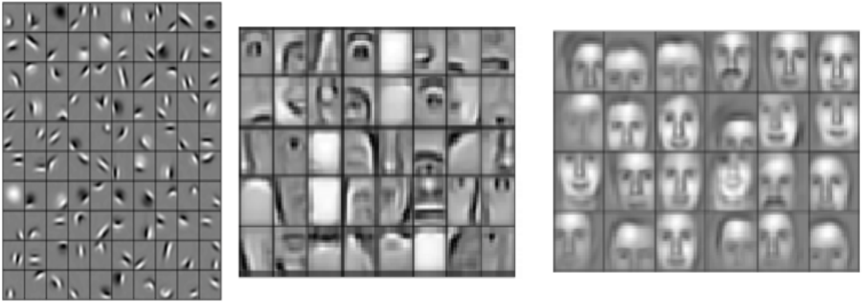
  

- 주어진 문제를 여러 개의 세부 문제로 나누어서 생각할 수 있다.



- 각각의 세부문제를 여러 개의 세세부 문제로 나누어서 생각하는 것이 효과적이다.





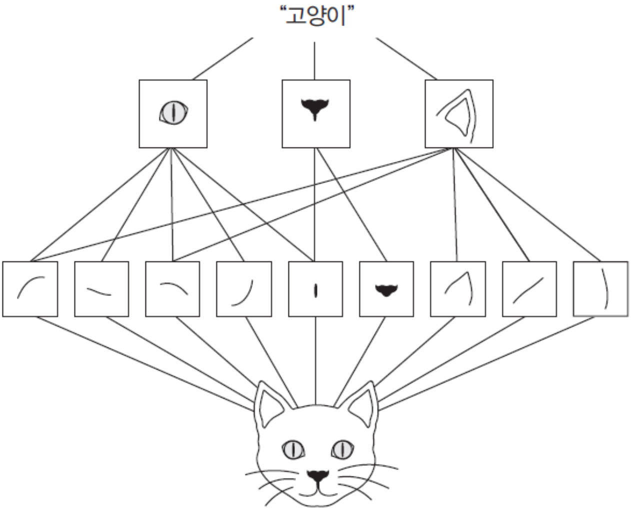


Figure: (https://github.com/WegraLee/deep-learning-from-scratch)

1. 심층 신뢰망(deep belief network): 심층 신뢰망은 입력층과 은닉층으로만 구성된 제한 볼츠만 기계(RBM; restricted boltzmann machine)를 학습한 후, RBM을 쌓아 올려 만든 비지도학습 방식의 생성 그래프 모형

2. 합성곱 신경망: 합성곱 신경망 구조를 사용한 AlexNet이 2012년도 ImageNet 영상자료 분류대회에서 좋은 성과를 거둔 이후, GoogLeNet, VGGNet, ResNet 등과 같이 합성곱 신경망을 기반으로 한 딥러닝이 영상자료 분류의 중심이 됨.

- 이후 합성곱 신경망은 객체탐지, 영상분할 등 컴퓨터비전의 각 분야로 전파, 기존 알고리즘의 성능을 급격하게 향상시킴.

3. 시계열자료: 시간의 흐름에 따라 변하는 자료. (예: 시간에 따라 변하는 주가, 사람의 움직임, 비디오 등)

Deep NN의 여러 모형들 II

- 순환 신경망은 일반 신경망에 시계열 개념이 추가된 신경망, 은닉층에 이전 정보를 기억하게 할 수 있는 장점이 있으나, 시계열자료의 역방향에 은닉층을 추가하여 자료의 인식률을 증가시킬 수 있음.

- 순환 신경망은 일반적 순환 신경망과 양방향 순환 신경망으로 나눌 수 있음.

Deep NN의 부활

- 심층 신경망은 입력층과 출력층 사이에 여러 개의 은닉층으로 이루어진 신경망
- 심층 신경망은 1개의 은닉층을 가진 천층 신경망과 마찬가지로, 복잡한 비선형 관계를 모형화할 수 있음
- 딥러닝(심층 신경망 학습에 활용되는 기계학습 방법론)이 관심을 받게 된 이유
  1. 기존 신경망의 단점 극복
  2. 하드웨어 성능의 발전
  3. 데이터의 기하급수적 증가
- 토론토대학교의 힌튼 교수가 신경망의 단점을 예비학습(pre-training)으로 해결할 수 있음을 발표
- 딥러닝은 압도적인 성능으로 각종 기계학습 관련 대회의 우승을 휩쓸었으며,
- 현재는 다른 기계학습방법을 통해 영상처리, 음성인식 등을 연구한 연구자들이 딥러닝방법을 기본으로 채택함

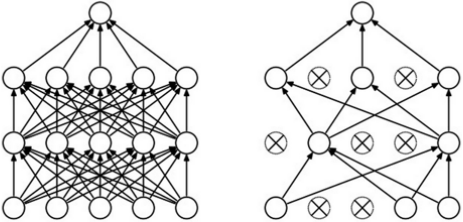
1. pre-training with unsupervised learning
   - 모형의 깊이가 깊기 때문에 역전파 알고리즘만을 이용하면 밑의 층을 이루고 있는 모수들이 잘 추정되지 않음.
   - 일반적으로 각 층마다 RBM 또는 SAE을 이용하여 모수들을 추정한 후 이를 초기값으로 이용하여 역전파 알고리즘 이용. (G.E.Hinton et al., 2006)
2. GPU
   - Deep Learning에서 필요한 계산들은 대부분 병렬처리 할 수 있음
   - 따라서 GPU을 이용하여 계산하면 CPU로 계산할 때보다 훨씬 빠른 계산이 가능.
   - 하드웨어의 발달로 인해 훨씬 많은 중간층과 모수들을 가지는 복잡한 모형을 설계하여 활용할 수 있게 됨

3. 새로운 활성함수(activation function)의 개발

- ReLU (Rectified Linear Units, $f(x) = \max(0, f(x)))$
- 기존의 활성함수 (예: Sigmoid or tanh)는 그 특성상 모수를 추정하는데 어려움이 있었음
- Piece-wise 선형 함수를 활성함수로 사용함으로써 중간층이 깊어질 때 역전파 미분량의 절대량이 작아지는 vanishing gradient 문제를 해결하고, 더 빠르고 좋은 결과 도출이 가능해짐
- Pre-training도 필요 없어짐

4. Regularization(정칙화, 조정화) 방법 개발

- 기존의 fully connected neural network을 학습할 때에 같은 층의 노드간에 높은 상관관계를 가지는 경향이 생김.
- 입력값의 작은 변화에 큰 변화가 생겨 좋지 않은 결과가 생길 수 있음.
- 노드간의 상관관계를 줄이고, 독립적인 역할을 하도록 하기 위해 매번 역전파를 할 때마다 의도적으로 노드의 절반을 끄고 (drop out) 학습함



(a) Standard Neural Net (b) After applying dropout.

5. 고급 최적화 방법 개발

- SGD 단점 개선

6. Batch normalization(Ioffe & Szegedy, 2015)

- 딥러닝 모델이 잘 학습되어지지 않거나 학습이 되어도 좋지 않은 성능을 보이는 이유는 은닉 노드들의 분포 및 scale이 상이하기 때문이라고 판단.
- 각 노드들마다의 분포가 비슷하도록 매번 mini-batch를 이용하여 학습할 때마다 노드들을 정규화(normalization)해줌.

IMDB 자료에 대하여

- 가중치 벌점화(penalization) 또는 정칙화(regularization)와
- 드랍아웃(drop-out)을 적용하여 보자.

```
model <- keras_model_sequential() %>%
  layer_dense(units=16, kernel_regularizer = regularizer_l2(0.001),
              activation="relu", input_shape=c(10000)) %>%
  layer_dense(units=16, kernel_regularizer = regularizer_l2(0.001),
              activation="relu", input_shape=c(10000)) %>%
  layer_dense(units=1, activation="sigmoid")

model <- keras_model_sequential() %>%
  layer_dense(units=16, activation="relu", input_shape=c(10000)) %>%
  layer_dropout(rate=0.5) %>%
  layer_dense(units=16, activation="relu", input_shape=c(10000)) %>%
  layer_dropout(rate=0.5) %>%
  layer_dense(units=1, activation="sigmoid")
```

용어 정리

- loss, cross-entory, learning rate
- computational graph
- forward propagation, back-propagation
- chain rule
- softmax, MLE
- SGD
- batch / mini-batch / epoch
- regularization, drop-out

<u>참고</u>

- 정칙화(regularization)와 l_1, l_2 벌점화의 관계를 확인하여보자.