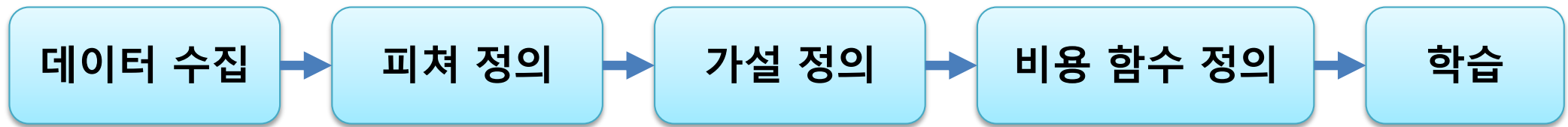


# MNIST 손글씨 데이터 인식



# ANN Code Preparation

## 학습단계 (Training)



## 모델 정의




## 검증 (Evaluation)



## 예측단계 (Prediction)

# 학습 알고리즘 구현

- 전제
  - 신경망에는 가중치와 편향이 있음
  - 가중치와 편향을 훈련 데이터에 적응하는 과정 → **학습**
- 1단계 – 미니배치
  - 훈련 데이터 중 일부를 무작위로 가져와서 사용
  - 미니배치의 손실 함수 값을 줄이는 것이 목표**확률적 경사 하강법**  
(*Stochastic Gradient Descent*)
- 2단계 – 기울기 산출
  - 미니배치의 손실 함수 값을 줄이기 위해 각 가중치 매개변수의 기울기를 구함
  - 기울기는 손실 함수의 값을 가장 작게 하는 방향 제시
- 3단계 – 매개변수 갱신
  - 가중치 매개변수를 기울기 방향으로 아주 조금 갱신
- 4단계 – 반복
  - 1~3 단계 반복

# Simple Neural Network

- 초기화 : 입력, 은닉, 출력 노드의 수 결정
- 학습 : 학습 데이터들을 통해 학습하고 이에 따라 가중치 업데이트
- 테스트 : 입력을 받아 연산한 후 출력 노드에서 답을 전달

```
import numpy
import scipy.special
import matplotlib.pyplot as plt

class NerualNetwork:
    # 신경망 초기화
    def __init__(self, inputnodes, hiddennodes, outputnodes, learnrate):
        pass

    # 신경망 학습
    def train(self, inputs_list, targets_list):
        pass

    # 신경망 테스트(질의)
    def query(self, inputs_list):
        pass
```

*Neural\_1.py*

# Simple Neural Network

- 신경망 초기화

```
# 신경망 초기화
def __init__(self, inputnodes, hiddennodes, outputnodes, learnrate):
    self.inodes = inputnodes
    self.hnodes = hiddennodes
    self.onodes = outputnodes

    self.lr = learnrate

    pass
```

```
input_nodes = 3
hidden_nodes = 3
output_nodes = 3
learning_rate = 0.3

n = NeuralNetwork(input_nodes, hidden_nodes, output_nodes, learning_rate)
```

*Neural\_2.py*

# Simple Neural Network

- 가중치
  - forward : 전달 신호
  - backward : 오차 계산
- (은닉 노드 x 입력 노드)의 크기를 가지는 입력 계층과 은닉 계층 사이의  
가중치 행렬  $\rightarrow W_{\text{input\_hidden}}$
- (출력 노드 x 은닉 노드)의 크기를 가지는 은닉 계층과 출력 계층 사이의  
가중치 행렬  $\rightarrow W_{\text{hidden\_output}}$

# Simple Neural Network

- 정교한 가중치

```
# 가중치 행렬 wih, who
self.wih = numpy.random.normal(0.0, pow(self.hnodes, -0.5), (self.hnodes, self.inodes))
self.who = numpy.random.normal(0.0, pow(self.onodes, -0.5), (self.onodes, self.hnodes))
```

노드로 들어오는 연결 노드의 개수  
루트를 씌우고 역수를 취한 표준편차

# Simple Neural Network

- training
  - 입력 계층과 은닉 계층 사이의 가중치 행렬 계산
  - $\mathbf{X}_{\text{hidden}} = \mathbf{W}_{\text{input\_hidden}} \cdot \mathbf{I}$

```
# 은닉계층으로 들어오는 값 계산  
hidden_inputs = numpy.dot(self.wih, inputs)
```

- $\mathbf{O}_{\text{hidden}} = \text{sigmoid}(\mathbf{X}_{\text{hidden}})$



# Simple Neural Network

- training

```
# 신경망 학습
def train(self, inputs_list, targets_list):
    # 입력 리스트를 2차원으로 행렬로 변환
    inputs = numpy.array(inputs_list, ndmin=2).T
    targets = numpy.array(targets_list, ndmin=2).T

    # 은닉계층으로 들어오는 값 계산
    hidden_inputs = numpy.dot(self.wih, inputs)
    # 은닉계층으로 나가는 값 계산
    hidden_outputs = self.activation_function(hidden_inputs)

    # 최종계층으로 들어오는 값 계산
    final_inputs = numpy.dot(self.who, hidden_outputs)
    # 최종계층으로 나가는 값 계산
    final_outputs = self.activation_function(final_inputs)

    pass
```

*Neural\_3.py*

# Simple Neural Network

- 오차

```
# 오차는 (실제 값 - 계산 값)  
output_errors = targets - final_outputs  
  
# 은닉계층의 오차는 가중치에 의해 나뉜 출력 계층의 오차들의 재조합  
hidden_errors = numpy.dot(self.who.T, output_errors)
```

$$\text{error}_{\text{hidden}} = W^T_{\text{input\_hidden}} \text{error}_{\text{output}}$$

# MNIST 손글씨 데이터 인식

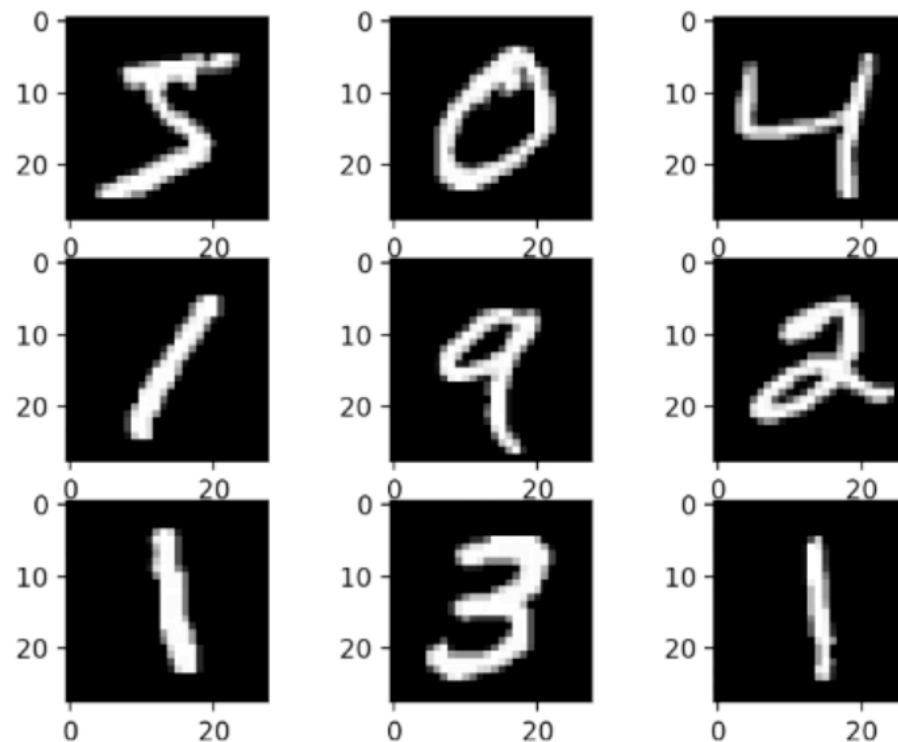
- Yann LeCun 교수 (<http://yann.lecun.com/exdb/mnist>)
- 학습 데이터 – [http://www.pjreddie.com/media/files/mnist\\_train.csv](http://www.pjreddie.com/media/files/mnist_train.csv)
- 테스트 데이터 – [http://www.pjreddie.com/media/files/mnist\\_test.csv](http://www.pjreddie.com/media/files/mnist_test.csv)



- 100개 레코드를 가지는 MNIST 학습 데이터 모음: <https://git.io/vySZ1>
- 10개 레코드를 가지는 MNIST 테스트 데이터 모음: <https://git.io/vySZP>

# MNIST 손글씨 데이터 인식

- 모든 숫자는  $28 * 28 = 784$  pixels (grayscale)
- $x\_train.shape, x\_test.shape = N * 28 * 28$
- $y\_train.shape, y\_test.shape = N$



# MNIST 손글씨 데이터 인식

- 1) 0 ~ 255 사이의 입력 색상 값들의 범위를 0.01 ~ 1.0 사이의 숫자로 조정
  - 데이터를 특정 범위로 변환하는 처리 → **정규화 (Normalization)**
  - 신경망의 입력 데이터에 특정 변화를 가하는 것을 → **전처리 (Pre-processing)**
- 2) 모든 값을 255로 나누고 0.99를 곱하기 → 0.01 ~ 1.00

# MNIST 손글씨 데이터 인식

출력계층	레이블	5	0	9
0	0	0.00	<b>0.95</b>	0.02
1	1	0.00	0.00	0.00
2	2	0.01	0.01	0.01
3	3	0.00	0.01	0.01
4	4	0.01	0.02	<b>0.40</b>
5	5	<b>0.99</b>	0.00	0.01
6	6	0.00	0.00	0.01
7	7	0.00	0.00	0.00
8	8	0.02	0.00	0.01
9	9	0.01	0.02	<b>0.86</b>



[0, 0, 0, 0, 0, **1**, 0, 0, 0, 0]

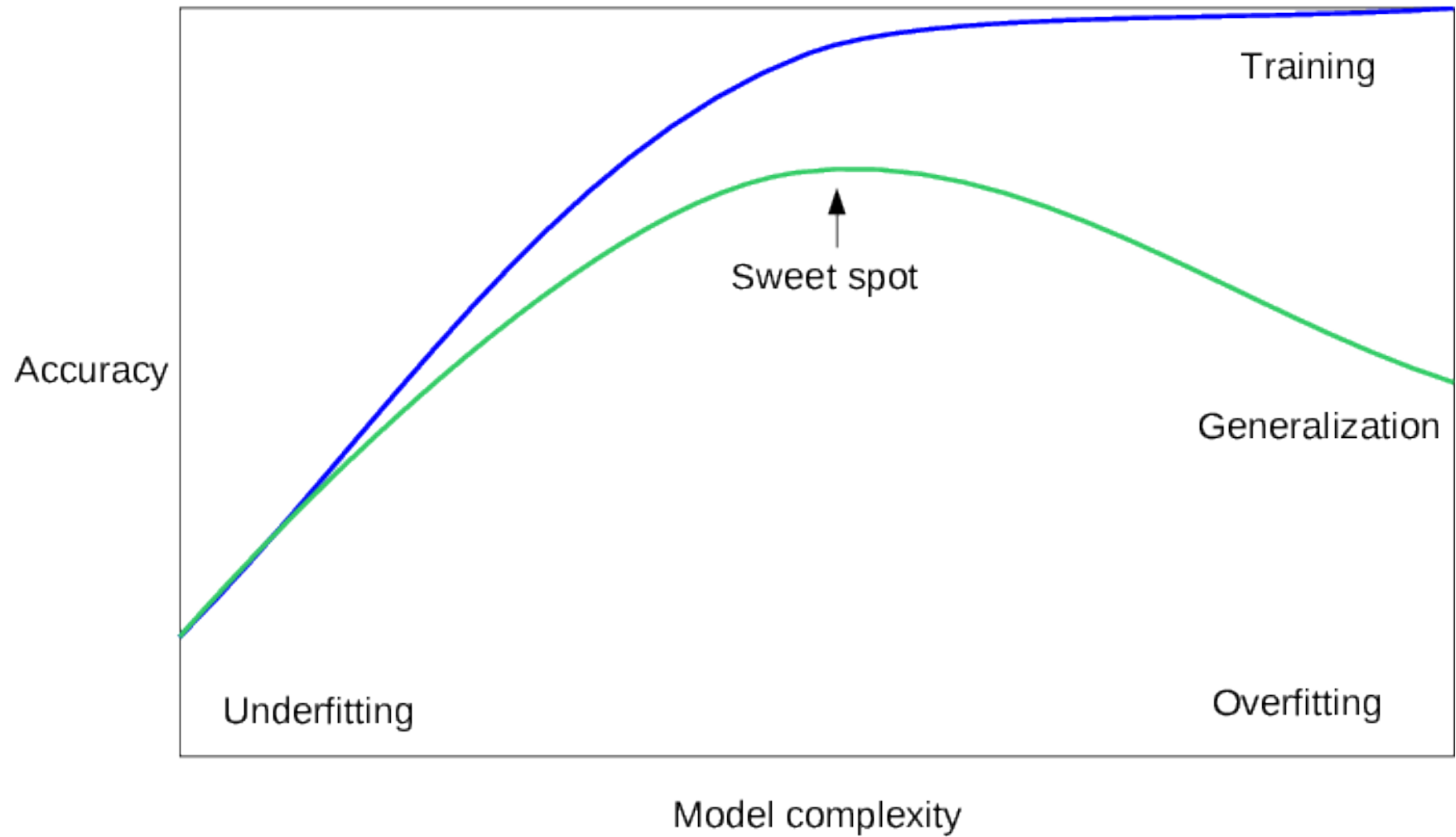
[0.01, 0.01, 0.01, 0.01, 0.01, **0.99**, 0.01, 0.01, 0.01, 0.01]

# MNIST 손글씨 데이터 인식

- 전체 데이터를 이용한 학습 및 테스트
  - mnist\_train\_100.csv → mnist\_train.csv
  - mnist\_test\_10.csv → mnist\_test.csv
- ex) 평균 0.94xxx
- 학습률 변경 → 0.1
- ex) 평균 0.9523
- <http://yann.lecun.com/exdb/mnist>

# MNIST 손글씨 데이터 인식

- 학습률과 성능





# MNIST 손글씨 데이터 인식

- 수행 주기 변경 (epoch)

