

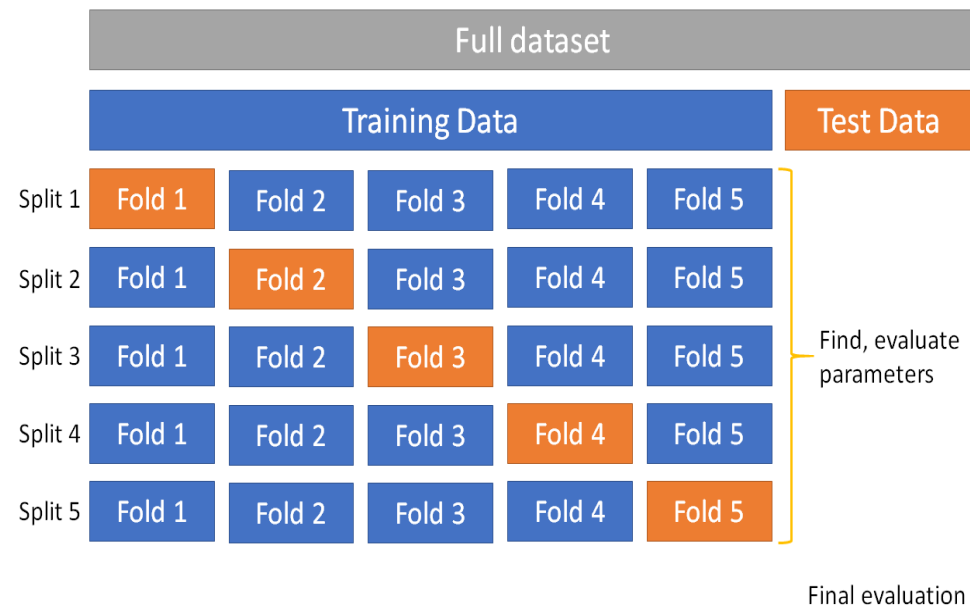
# 머신러닝의 네가지 분류

---

- 지도학습
  - 레이블(정답, target, annotation)이 있는 샘플데이터로 학습.
  - 일반적인 분류, 회귀 문제
  - 이외 시퀀스 생성, object detection image segmentation ...
- 비지도학습
  - 레이블이 없는 샘플데이터로 학습
  - 데이터셋의 이해, 차원축소 등에 활용됨
- 자기지도학습
  - 별도의 레이블이 필요하지 않은 학습
  - 원본데이터를 목표값으로 학습하는 Auto Encoder
  - 일련의 단어가 주어졌을 때 다음 단어를 예측
- 강화학습
  - 주어진 조건에서 보상을 최대화하는 행동을 취하도록 학습
  - 구글 딥마인드에서 Atari 게임 방법 학습에 성공하면서 관심받음.
  - 알파고 구현

# 머신러닝 모델평가

- 훈련, 검증, 테스트 세트
  - 동일 데이터셋을 모델의 설정 튜닝에 사용하면 검증세트에 과대적합된 모델이 만들어지기 쉬움.
  - 튜닝 대상 : 층의 수, 층의 유닛수...
  - 따라서 생성된 모델은 최종적으로 한번도 본 적이 없는 데이터로 평가되어야 함.
- 단순 홀드아웃 검증
  - 데이터셋의 일정량을 테스트셋으로 할당.
  - 데이터량이 적으면 테스트셋의 결과의 신뢰성이 떨어짐.
- k-겹 교차검증
  - 데이터셋을 동일한 크기를 가진 k 개의 집단으로 분할.
  - k-1 개의 집단으로 훈련하고 남은 1 개의 집단으로 검증한 후 평균으로 성능 평가.
  - 모델의 성능이 데이터 분할에 따라 편차가 클때 평가치를 안정적으로 제공.
- 반복 k겹 교차검증(RepeatedKFold)
  - 무작위로 섞은 후 k겹 교차검증을 여러 번 반복 수행.
  - $P(\text{반복}) \times k\text{겹 (교차검증)}$  만큼 수행하므로 계산비용이 많이 듦.
- 주의할 점
  - 데이터셋의 목표값이 편중되는 것을 방지하기 위해 분할 전 무작위로 섞는 것이 바람직함.
  - 시계열 데이터셋은 무작위로 섞으면 예측대상인 미래의 데이터셋이 학습과정에 노출될 수 있으므로 섞으면 안됨.
  - 중복데이터가 훈련세트와 검증세트에 동시에 나타나면 안되므로 제거.
- (실습)
  - 교재는 순수코딩 → scikeras 를 이용하여 KFold 교차검증.



# 머신러닝 기타사항 - 신경망을 위한 전처리



---

- 원본데이터를 신경망에 적용하기 좋은 형태로 변형하는 작업
- 벡터화
  - 모든 입력과 타겟은 부동소수점(또는 정수) 텐서여야 함
  - 사운드, 이미지 텍스트 등 예외없이 텐서로 변환
- 값 정규화
  - 큰값, 작은값이 섞여있으면 역전파시 그래디언트 값이 커져 수렴을 방해함.
  - 모든 특성에 대하여 독립적으로 변환
  - 작은값. 모든 특성이 비슷한 범위의 값을 가져야 함.
  - 예 : 이미지의 경우 0 ~ 255 사이의 값을 0 ~ 1 사이의 값으로 변환
- 누락된 값 처리
  - 학습데이터에 누락된 값들이 존재하면 모델은 누락값을 무시하도록 학습할 수 있음.
  - 학습데이터에 누락된 값들이 없는데 누락값이 테스트데이터에서 나오면 모델은 처리방법을 모름.
  - pandas에 누락값 다루는 많은 처리유형이 제공되고있음.
  - <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html>

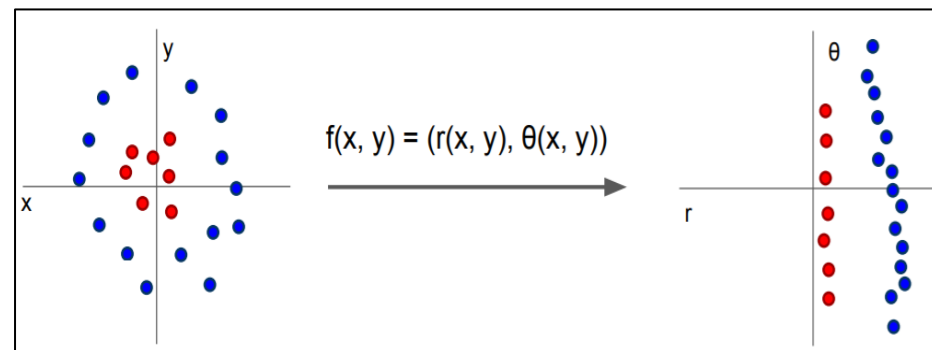
# 머신러닝 기타사항 - 특성공학

- 도메인 지식을 사용하여 모델이 좀 더 쉽게 학습할 수 있도록 하는 변환
- 시계 이미지로 시간을 예측하는 모델의 경우
- 시계 이미지를 바로 모델 입력으로 사용하는 경우
  - 많은 데이터 샘플이 필요하고 학습시 자원소모도 큼.
- 시계바늘 끝의 좌표를 전처리하여 모델입력으로 사용
  - 시계바늘의 끝 지점 확인후 좌표로 변환
  - 시간으로 변환하는 알고리즘 설계 가능
- 극좌표로 전처리하여 모델에 입력
  - 시계바늘의 끝 지점 확인후 각도로 변환
  - 시간으로 변환이 더 쉬워짐.
- 딥러닝에서는,
  - 자동으로 원본데이터에서 유용한 특성을 추출함.
  - 단, 이 경우 충분히 많은 데이터 샘플이 확보되어야 함.
  - 유용한 전처리를 먼저하면, 더 적은 데이터 샘플수, 더 적은 컴퓨팅 자원으로 학습 가능

예 1 : 시계의 바늘 데이터를 좌표 또는 각도로 변환

Raw data: pixel grid		
Better features: clock hands' coordinates	$\{x1: 0.7,$ $y1: 0.7\}$ $\{x2: 0.5,$ $y2: 0.0\}$	$\{x1: 0.0,$ $y2: 1.0\}$ $\{x2: -0.38,$ $2: 0.32\}$
Even better features: angles of clock hands	theta1: 45 theta2: 0	theta1: 90 theta2: 140

예 2 : 두 그룹의 점들을 반지름과 각도 속성으로 변환



[http://cs231n.stanford.edu/slides/2021/lecture\\_4.pdf](http://cs231n.stanford.edu/slides/2021/lecture_4.pdf)

# 머신러닝 기타사항 – 과대적합/과소적합

---

- 일반화(generalization) – 학습을 통하여 본 적이 없는 데이터에 대해 성능을 높이는 작업.
- 과소적합(underfitting)
  - 훈련데이터 손실과 테스트데이터 손실이 같이 개선되는 상태
  - 모델이 훈련데이터에 있는 특성을 모두 학습하지 못한 상태
  - 모델의 성능이 개선될 수 있는 상태
- 과대적합(overfitting)
  - 훈련데이터 손실과 좋아지나 테스트데이터 손실은 나빠지는 상태
  - 훈련데이터를 외우는 상태
  - 학습문제를 맞추는 능력은 우수하나 새로운 문제의 예측성능이 눈에 띄게 떨어지는 상태
- 과대적합 해결방안
  - 훈련데이터 늘임.
  - 모델이 수용하는 정보의 양을 조절 : 네트워크 규모 축소.
  - 저장하는 정보에 제약을 가함(가중치 규제).
  - 이러한 방법을 통하여 모델은 가장 중요한 데이터 패턴에 집중하게 됨.

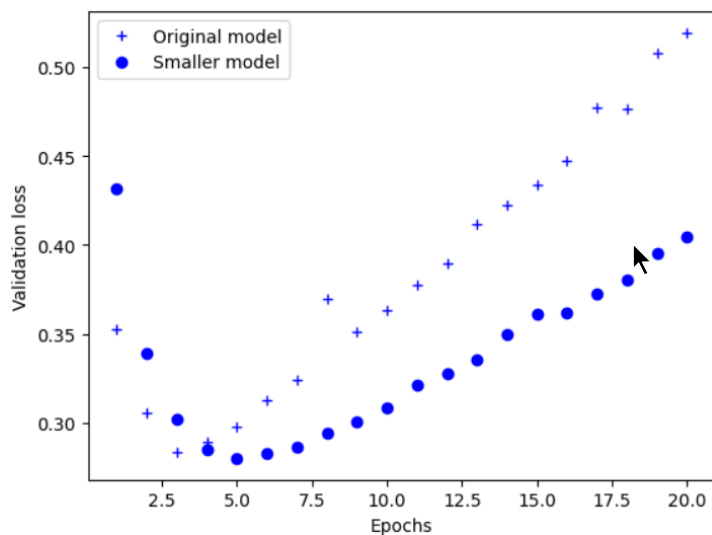
## 과대적합/과소적합 - 네트워크 크기 축소

- 모델의 크기(= 모델의 용량)를 축소하여 과대적합을 방지할 수 있음.
- 모델의 크기: 층의 수, 층단위 유닛 수
- 모델이 클수록 기억용량이 커지기 때문에 학습데이터를 외우기 쉬움.
- 모델을 작으면 학습 시 꼭 필요한 정보만 학습하여 일반화능력이 높아짐.

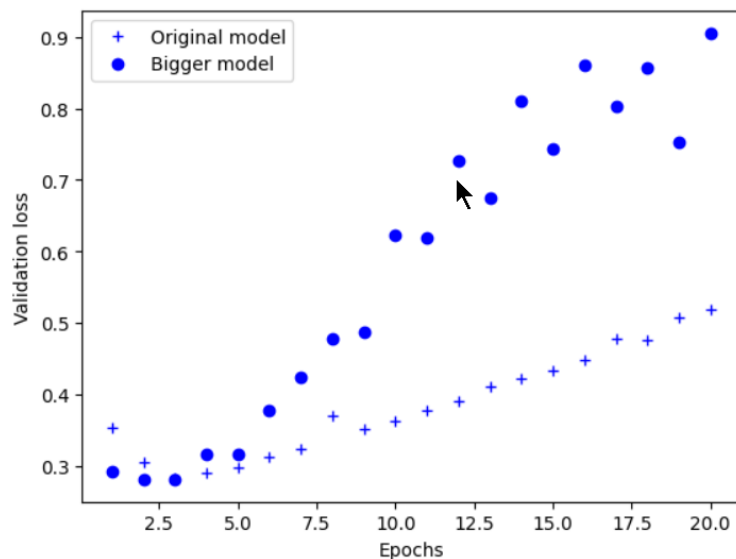
- (실습)

영화 리뷰 분류하기 - 모델크기 대비 검증손실, 훈련손실 비교

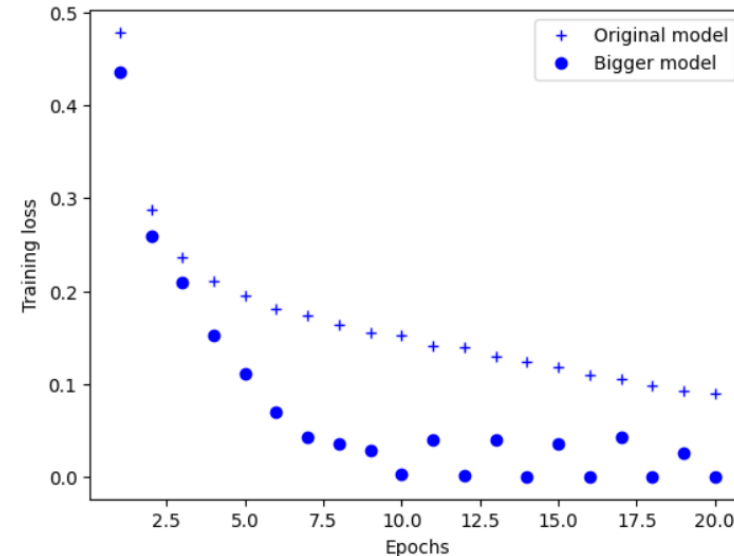
기본모델(16 unit) - 작은모델(6 unit) 비교



기본모델(16 unit) - 큰모델(128 unit) 비교



기본모델 - 큰모델 훈련손실 비교



## 과대적합/과소적합 – 가중치 규제

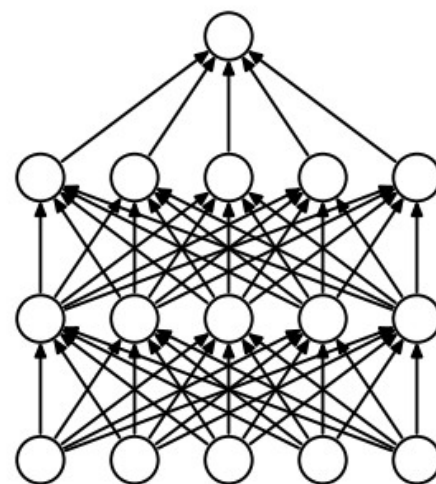
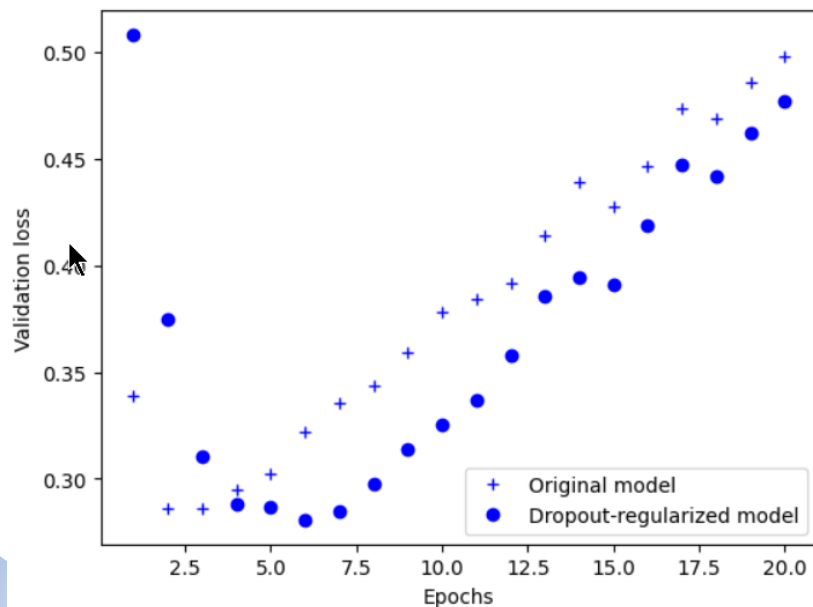
---

- 오컴의 면도날(Occam's razor) 이론
- 어떤 현상에 대한 두가지 설명이 있다면 더 적은(단순한) 가정을 가진 설명이 좋은 설명이다.
- 간단한 모델을 만들기 위해 네트워크의 복잡도에 제한을 두어 가중치 값을 억제
- 손실함수에 가중치 관련 비용을 추가
  - L1 규제 : 가중치의 절대값에 비례하는 비용 추가
  - L2 규제 : 가중치의 제곱에 비례하는 비용 추가
- L1, L2 규제는 훈련시에만 적용하므로 훈련손실이 테스트손실보다 높을 수 있음.
- (실습) – L2 규제 적용  
규제 미사용 모델에 비해 과대적합 경향이 현저히 낮아짐.

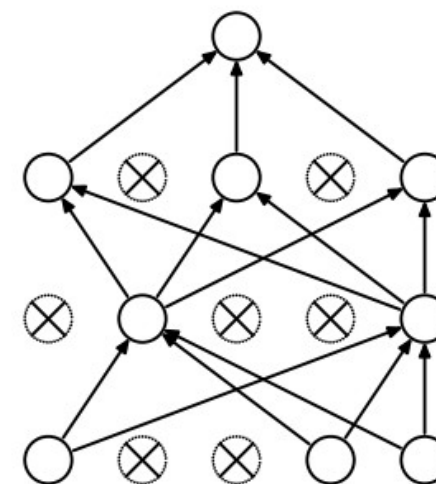
# 과대적합/과소적합 - Dropout

- 저장하는 정보에 제약을 가함(drop-out)
- 훈련하는 동안 무작위로 층의 일부 출력 특성을 제외시킴(가중치를 0으로 하는 것이 아님).
- 모든 데이터 샘플에 대하여 적용
- 훈련할 때만 적용
- 테스트시에는 모든 유닛 활성화
- (실습)

`model.add(layers.Dropout(0.5))` 적용



Standard Neural Net



After applying dropout



# 머신러닝 작업흐름

---

## 1. 문제 정의

- 입력데이터 및 레이블
- 회귀, 이진분류, 다중분류, 강화학습?

## 2. 성능평가지표 선택

- 정확도, 정밀도/재현율, 평균정밀도, ROC/AUC 등

## 3. 평가방법 선택

- 홀드아웃-검증세트 분리 : 데이터가 충분히 많을 때
- k-겹 교차검증, 반복 k-겹 교차검증 : 샘플데이터 수가 적을 때

## 4. 데이터 전처리

- 숫자로 변환, 정규화, 특성공학 적용

## 5. 기본 모델 설정

- 마지막 층의 활성화함수 : sigmoid(이진분류), softmax(다중분류)
- 손실함수 : categorical/binary\_crossentropy, mse (미분가능 함수)...
- 옵티마이저 : rmsprop, adam 및 기본 학습률

# 머신러닝 작업흐름

---

## 6. 과대적합 모델 구축

- 먼저 과대적합모델을 구축하고 이 후 최적화 진행
- 검증데이터에 대한 성능이 최고가 된 후 하향할 때
- 층 또는 유닛 추가
- 학습 epoch 증가

## 7. 모델 규제와 하이파파라미터 튜닝

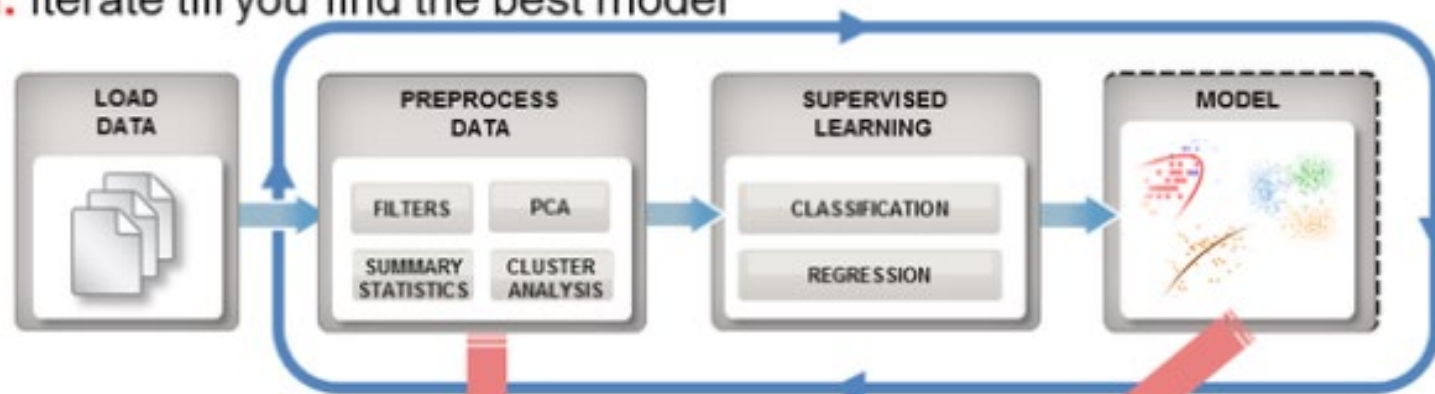
- 반복적으로 모델을 수정.훈련하고 검증데이터에서 평가
- 드롭아웃 추가
- 층 제거 또는 유닛수 축소
- L1/L2 규제 추가
- 특성공학 적용(새로운 특성 추가, 제거)

## 8. 최종모델 완성

- 만족할만한 모델이 만들어지면 테스트셋에서 평가
- (성능차이가 크면 이전 작업 반복)
- 마지막으로 테스트셋을 포함한 모든 데이터셋으로 훈련하여 모델 완성

# 머신러닝 개발 과정

**Train:** Iterate till you find the best model



**Predict:** Integrate trained models into applications

