예측 모델

2020년 봄 학기

2020년 월 일

https://www.github.com/KU-BIG/KUBIG_2019_Autumn

1. 모델이란

모델은 머신러닝의 시작점이다. 데이터 분석을 하거나 머신러닝으로 문제를 해결하려면 무엇을 해야 할까? 데이터에서 패턴을 찾아내고 그것을 이용하겠다는 생각이 들 것이다. 그렇다면 과연 그 패턴은 존재할까? 패턴이 있을 것이라는 생각은 데이터 자체에 대한 믿음, 즉 가정에 해당한다. 이런 다양한 가정을 한데 모은 것을 머신러닝에서는 모델이라고 한다.

쉽게 말해 현재 상태를 어떠한 시각으로 바라보고 어떠한 기대를 하고 있는가를 구체화한 것이 모델이다. 모델은 머신러닝의 전(全)과정에서 활용된다. 모델의 형태를 추측하고 구체화하고 평가는 것이 머신러닝의 전부이기 때문이다. 머신러닝의 과정은 다음과 같다.

1. 모델 정하기(데이터 형태를 가정하고 목적 고려)

2. 모델의 학습 목표를 수식화하기

3. 실제 데이터로 모델 학습하기(최적화)

4. 평가하기

5. 위의 과정을 반복

모델이란 가정에 따라 생성될 수 있는 함수들의 집합이다. 과정 1에서는 세상의 무수히 많은 함수 중에서 특정한 형태의 함수를 지정하는 것이다. 예를 들어 '이 데이터는 1차식의 관계가 있을 것' 이라고 일종의 가정을 하는 것이다. 데이터를 우선적으로 살펴보고 그럴듯한 가정을 설정해야 한다.

과정 2는 과정 1에서 가정한 함수를 수식화 한다. 그 다음, 수식화한 함수를 과정 3에서 학습해야 한다. 예를 들어 과정2에서 y=ax라고 수식화했다면, 과정 3에서는 이 a에 어떤 숫자가 들어갈지 데이터를 통해 추측한다. 이러한 추측을 학습이라고 하며, 여기서 추측의 대상인 a는 모델의 parameter(모수)가 된다. 머신 러닝의 러닝이 바로 이 학습이며, 학습은 모델이 표현하는 함수 집합 중에서 가장 데이터에 적합한 함수를 고르는 과정이다.

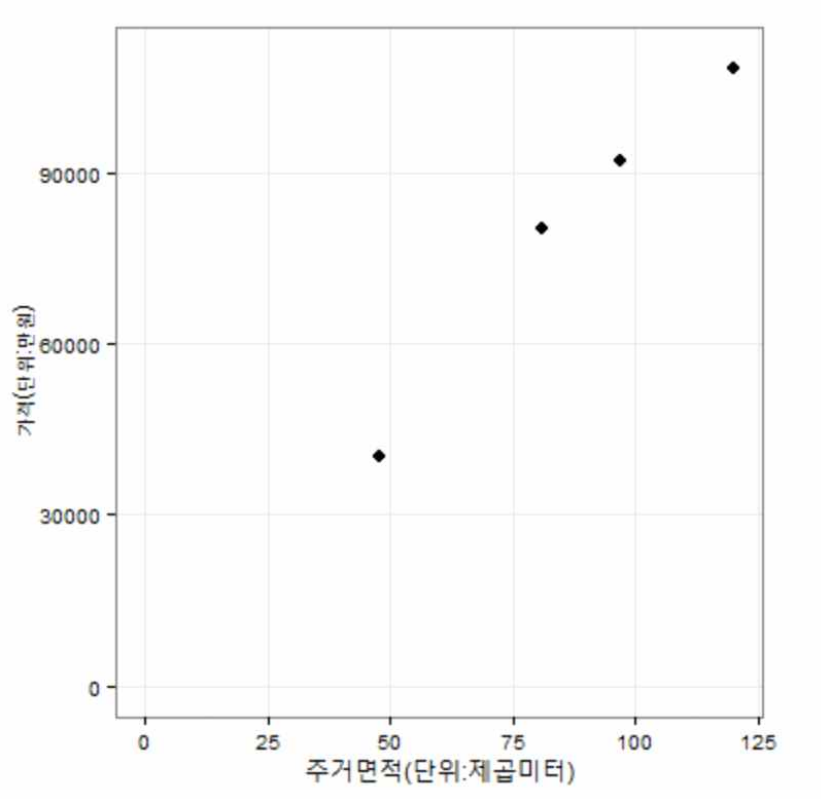
이 단원에서는 그 중에서도 '예측(Prediction)'이라는 특수한 목적을 가진 모델을 중점적으로 다루려고 한다. 그렇다면 예측이란 무엇일까?

2. 예측이란

데이터 분석을 하다 보면 '예측(prediction)'이란 용어가 자주 등장한다. 실제로 데이터 분석의 가장 큰 목적은 '예측(prediction)'이다. 그래서 '예측 분석(predictive analytics)'라는 말을 자주 사용하기도 하고 구글의 클라우드 기반 기계 학습 서비스의 이름도 'Prediction API' 인 것을 확인할 수 있다.

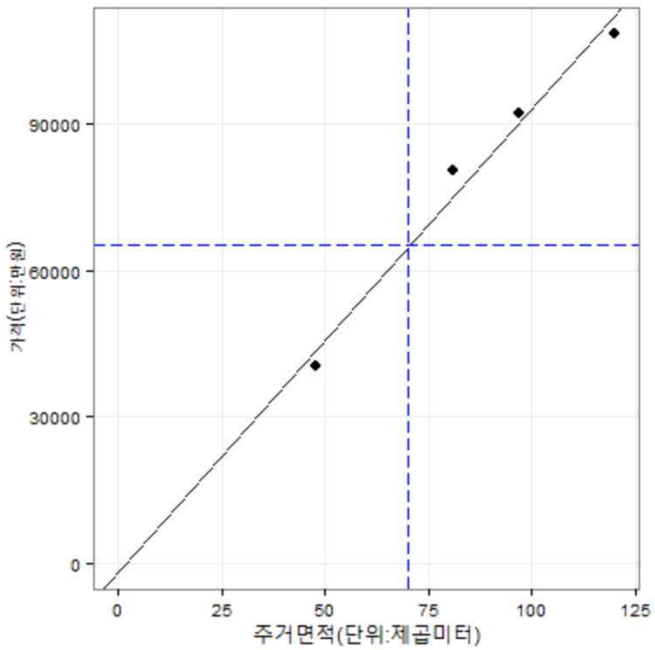
그런데 데이터 분석에서 말하는 예측은 우리가 일반적으로 말하는 예측과 그 의미가 조금 다르다. 보통 예측이라 하면 미래의 사건을 예상하는 것, 즉 '예보(forecast)'를 상상한다. 물론 예측과 비슷한 성격을 가지지만 엄밀히 말해 데이터 분석에서 말하는 예측은 미래를 예

견하는 것과는 전혀 다르다. 그렇다면 데이터 분석에서 말하는 예측은 무엇을 의미할까? 한마디로 말하자면 '내삽(Interpolation)'을 의미한다. '내삽'이라는 용어는 흔히 사용하는 단어가 아니기 때문에 쉽게 와 닿지 않을 수 있으니 예를 들어 설명하자면 다음과 같다.



<출처 - 2015년 19월 기준, 출처: 국토교통부 실거래가 공개시스템>

위의 그림을 보면 주거 면적과 가격 사이에 개략적인 선형 관계가 있다고 추정할 수 있다. 그래서 실제 데이터 상으로는 알 수 없지만 누군가 '주거 면적이 70제곱미터인 아파트는 얼마 정도 할까?'라고 물어본다면 6억 5천만 원 정도 될 것이라 추정할 수 있다. 그 이유는 위 데이터를 토대로 아래와 같은 가상의 선을 그릴 수 있기 때문이다.



아파트 실거래가 데이터에 추정선을 추가한 플롯.
파란색 수직, 수평선은 각각 x=70, y=65000 위치에서 그린 직선이다.

위와 같이 이미 주어진 자료를 토대로 자료 사이에 비어 있는 값을 추정하는 것을 내삽이라고 한다. 기계 학습이나 데이터 마이닝을 이용해서 예측 분석을 하는 것은 바로 이런 내삽을 잘 하기 위해 데이터를 변형하거나 패턴을 찾는 작업이다. 즉, 위의 예에 나온 좌표계에 대각선을 가로지르는 가상의 추정선을 긋는 것을 '예측 모델링'이라고 하며 이렇게 만든 추정선(모델)을 이용해서 주어진 데이터에는 없는 새로운 데이터를 추정하는 것(즉, 70제곱미터 면적의 아파트 가격을 추정하는 것)을 '예측'이라고 말한다. 다시 말해, 예측 모델링은 데이터와 통계를 활용하여 데이터 모델의 결과를 예측하는 절차로, 이 모델들은 스포츠 경기 결과와 TV 시청률부터 기술 진보 및 기업 수익에 이르기까지 모든 것을 예측하는데 사용될 수 있다. 예측 모델링은 '예측 분석', '기계 학습'과 같은 이름으로 불리기도 한다.

따라서 어떤 문제가 예측이 가능한지를 판단하려면 현대 보유한 데이터로 내삽이 가능한지

파악해야한다. 즉, 모델링의 토대가 된 데이터의 차원과 예측하고자 하는 데이터의 차원이 같아야 한다.

그렇다면 과정 3까지 진행하여 만든 모델이 목적에 맞게 잘 작동하는지 혹은 모델이 실제로 데이터를 바르게 표현했으며 얼마나 예측이 정확한지에 대한 수학적 표현이 필요하다. 모델에 대한 이러한 평가는 어떻게 이루어질까?

3. 모델 평가

앞서 말했듯이 실제로 데이터를 바르게 표현했는지 혹은 얼마나 예측이 정확한지에 대한 수학적 표현이 Loss function(Cost function, 손실 함수, 비용함수)이다. 그리고 모델이 실제로 데이터와 얼마나 다른지에 대한 값, 즉 Loss function의 결과값인 Loss(cost, error)이다. 이후 다루겠지만, 예를 들어 선형모형에서 만든 직선이 데이터와 얼마나 떨어져 있는지 계한하는 함수가 바로 Loss function이다. Loss가 작을수록 모델이 더 정확하게 학습된 것이라 볼 수 있다.

Loss function은 거의 같은 모델을 대상으로 하여도 중요하게 생각하는 데이터의 특성에 따라 변형될 수 있다. 예를 들어 데이터의 전체적인 패턴을 중시할지 지엽적인 패턴을 중시할지에 따라 기본적인 Loss function에 축적인 Loss function을 덧붙일 수 있다. 엄밀히 따지면 데이터를 보는 관점이 조금 변화했기에 다른 모델이라고 할 수 있지만, 이러한 Loss function의 조합은 같은 모델의 변형으로 여겨지는 경우가 많으며 자주 사용된다.

Loss function은 데이터 전체에 대해 계산하는 함수지만, 간단하게 표시하기 위한 한 개의 데이터에 대해서만 정의하기도 한다. 이때 각 데이터에 대한 Loss function 계산 결과의 총합이 그 모델과 데이터 전체에 대한 Loss function의 결과이다. 물론 이런 경우에는 몇 가지 가정이 추가된다. 예를 들어 데이터셋에서 각각의 데이터가 서로 확률적 독립이고 같은 분포를 가진다는 가정인 iid(Independent and Identically Distributed)가 대표적이다.

Loss function은 주로 실제값과 예측값과의 차이를 중점적으로 구성된다. 가장 간단한 Loss function인 제곱 손실함수를 살펴보자. f는 모델, y는 데이터로부터 주어진 실제 출력값, $\hat{y}$은 모델에 입력값을 넣어 계산한 예측값, 즉 f(x)이다.

$$loss(f) = (y - \hat{y})^2$$

위 식은 데이터 하나에 대해 정의한 것으로 전체 데이터에 대해서는 각 Loss function의 값을 전부 합산하거나 평균을 낸 값을 사용한다. 여기서 평균을 낸 값은 Mean Squared Error(MSE)로 주로 output 데이터가 연속적일 때 적용하는 회귀(Regression)의 Evaluation metric으로 사용된다.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad \text{Cross entropy} = -\sum_{i=1}^{n}(y_i \log \hat{y}_i)^2$$

Cross entropy는 주로 분류(Classification)의 문제나 딥러닝에서 자주 활용된다. 이외에도 MLE(Maximum Likelihood Estimation, 최대가능도방법), KL-Divergence(쿨백-라이블러 발산)등 많은 Loss function들이 있으며 이들은 각기 다른 상황에서 적용된다. 이에 대한 자세한 설명은 여기서 하지 않기로 한다.

각 모델은 이러한 Loss function을 최소화하는 최적화 문제를 진행해야한다. 위에서 언급했듯이 Loss function은 주로 실제값과 예측값의 차이로 구성되어있기에 이 둘의 차이가 작으면 작을수록 좋은 모델이라고 평가할 수 있기 때문이다.

다시 말해 Loss function의 결과값을 최소화하는 모델의 parameter(θ)를 찾는 것이다. 모델에 임의의 parameter을 설정하여 구한 예측 값으로 Loss를 줄이거나 늘릴 수 있다. 이렇게 임의로 여러 값을 대입하며 손실을 최소화하는 방법도 있지만 수많은 시행착오를 겪어야 한다. 따라서 보통은 컴퓨터가 이러한 시행착오를 하도록 하는 알고리즘을 사용한다. 그렇다면

이러한 과정을 거치는 예측 모델에는 어떤 것들이 있을까?

● 예측 모델의 유형

구체적으로, 다음은 예측 모델의 몇 가지 상이한 유형들을 보여준다.

최소 제곱 (Ordinary Least Squares)

일반화 선형 모델 (Generalized Linear Models, GLM)

로지스틱 회귀 (Logistic Regression)

랜덤 포레스트 (Random Forests)

결정 트리 (Decision Trees)

신경망 (Neural Networks)

이 모델과 이외의 여러 예측 모델은 모두 특정한 용도가 있으며, 특정 질문에 응답하거나 특별한 유형의 데이터 집합을 이용한다. 모델 유형 간에 방법론적, 수학적 차이가 있음에도 각 모델의 포괄적인 목표는 유사하다. 바로 과거의 결과에 대한 데이터를 이용하여 미래의 또는 미지의 결과를 예측하는 것이다. 그 중 몇 가지 모델을 소개하겠다.

3. 회귀 분석

회귀분석(Regression Analysis)은 어떤 결과값이 존재할 때, 그 결과값을 결정할 것이라고 추정되는 입력값과 결과값의 연관관계를 찾아 결과값을 예측하는 기법을 말한다. 회귀분석에서는 데이터를 아래와 같은 모델로 수식화할 수 있다고 본다.

$$y = h(x_1, x_2, \cdots, x_k\,;\,\beta_1, \beta_2, \cdots, \beta_k) + e$$

회귀모델의 본래 정의는 '어떤 자료에서 그 값에 영향을 주는 조건들을 고려하여 구한 평균'이다. 따라서, 위 수식에서 h()는 평균을 구하는 함수를 의미한다. 각 조건, $x_1, x_2, \ldots, x_k$에 대하여, 그들 각각의 영향력 $\beta_1, \beta_2, \ldots, \beta_k$을 구하여 평균을 계산한다. e는 오차항(error)을 의미하는데, 측정 상의 오차나 현실적인 한계로 인해 발생한느 불확실성('잡음')을 의미한다.

회귀분석에서는 데이터에 대해 여러 가지 가정을 필요로 한다. 특정 조건이 만족되는지의 여부에 따라 사용하는 회귀모델이 달라지게 된다. 이 자료에서는 회귀모델의 두 종류인 기본 선형 회귀모형(Linear Regression)과 GLM(Generalized Linear Model)에 대해 다루고자 한다.

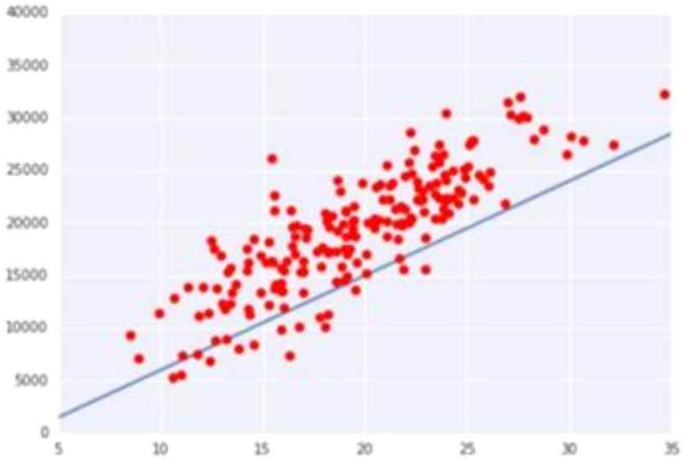
3.1. 선형 회귀 모형

Linear Rgression은 각 attribute마다 weight가 주어진 linear combination 형식이다.

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \varepsilon_i, \varepsilon_i \sim N(0, \sigma), \quad \hat{y} = E[Y|X = \mathbf{x}] = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

위의 식에서 유추해볼 수 있듯이 종속 변수 y와 한 개 이상의 독립변수 x와의 선형 관계를 통해 결과값을 예측하는 방법을 말한다. 예를 들어, 종속변수가 '택시 요금', 독립변수가 '이동 거리'라고 하자. 거리와 요금이 서로 비례하기 때문에 거리(x)와 요금(y)간의 상관관계는 다음과 같이 일차함수의 형태로 나타낼 수 있다.

$$y_i = \alpha + \beta x_i + \varepsilon_i.$$



Linear Regression의 weight를 optimize 하기 위해서는 최소제곱법(LSM) 즉, 잔차의 제곱합인 $\sum(y_i - \hat{y_i})^2$를 최소화 하는 β_i들을 찾아야한다. 이때 잔차의 제곱합인 $\sum(y_i - \hat{y_i})^2$는 Sum of Squared Error(SSE)라고 부른다.

선형 회귀모형은 그 형태가 단순한 만큼 데이터에 대해 많은 가정을 필요로 한다. 대표적으로 '종속변수와 독립변수는 선형관계이다', '오차항은 평균이 0이고 분산이 일정한 정규분포를 따른다', '독립변수와 오차항은 서로 독립이다' 등이 있다. 이 조건들 중 두 번째 오차항의 정규성 조건이 만족하지 않는 경우 사용할 수 있는 회귀모델이 GLM이다.

3.1. GLM(Generalized Linear Model)

일반화 선형모형(Generalized Linear Model)은 연속형 반응변수에 대한 보통의 회귀모형과 분산북석모형(ANOVA) 뿐만 아니라 범주형 반응변수에 대한 모형들을 포함하는 광범위한 모형의 집합이다. 각 반응변수의 특성에 맞게 연결함수를 이용한 Linear Regression이다.

예를 들어 반응 변수가 Poisson 분포를 따르는 경우, 바로 반응변수와 설명변수들을 결합시

키면 $-\infty < \mu < \infty$ 의 범위를 가지게 되나 반응변수는 0개부터 무한대까지 '개수' 값을 가진다. 따라서 Poisson Regression의 경우 아래의 log link function을 사용한다. GLM에는 무척이나 다양한 종류가 존재하는데 여기서 모두 다루지는 않기로 한다.

$g(\mu) = log(\mu)$: log link function, $0 \leq \mu < \infty$

$log(\mu) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$ => Poisson Regression

4. Classification based Prediction

Classification은 Supervised Learning 의 분야로 Target Variable 혹은 Label이 주어졌을 때 활용하는 기법이다. Target Variable(Label)이 연속적인 데이터로 구성된 Regression과 달리 Target Variable이 연속적이지 않은 범주형(Categorical Data)으로 이루어져있을 때 활용된다. 이런 Classification을 기반으로 새로운 범주형 데이터의 Label을 예측할 수 있다.

대표적인 Classification 모델에는 K-nn, Decision Tree, Support Vector Machine(SVM)이 있다.

4.1. K-nn

KNN 알고리즘은 대표적인 게으른 학습(lazy learning)이다. 여기서 게으른 학습의 반대는 열정적인 학습(eager learning)이다. 두 학습을 비교하면 다음과 같다.

● **게으른 학습(lazy learning) :** 학습 데이터(training data)를 단순하게 저장하고 테스트 데이터(test data)가 올 EO까지 기다렸다가 데이터가 input 되면 그 때 모델을 실행하는 경우, 학습 시간보다 예측 시간이 더 걸린다.

● **열정적인 학습(earger learning) :** 학습 데이터가 주어지면 테스트 데이터가 input 되기 전부터 바로 모델을 생성하는 경우.

따라서, 새로운 데이터가 주어지면 그와 비슷한 혹은 그에 가까운 것 k 개를 찾아 Majority Voting / Weighted Voting 등의 방법을 활용해서 Class를 결정하는 방식이다. 가장 주의해야 하는 부분은 scale에 민감하기 때문에 Normalize/Scaling 등의 처리가 필요하다.

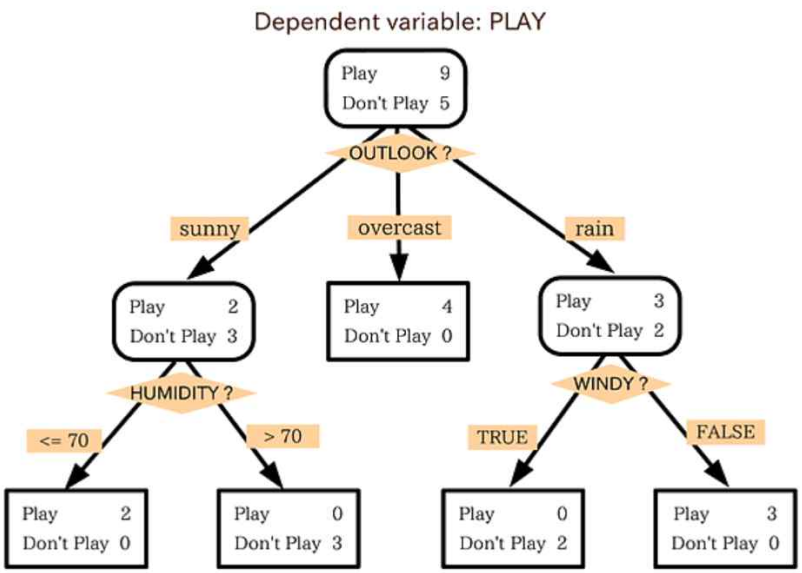
KNN 알고리즘의 장단점을 비교하면 다음과 같다.

● **장점 :** 데이터 분산에 대한 추정을 만들 필요가 없다. 훈련 단계가 빠르다. 특별한 모델을 생성하지 않는다. 단순하며 효율적이다.

● **단점 :** 많은 메모리가 필요하다. 명목형 데이터와 결측치에 대해서는 추가적인 처리가 필요하다.

4.2. Decision Tree

의사결정나무는 데이터를 분석하여 이들 사이에 존재하는 패턴을 예측 가능한 규칙들의 조합으로 나타내며, 그 모양이 '나무'와 같다고 해서 의사결정나무라 불린다. 질문을 던져서 대상을 좁혀나가는 '스무고개 놀이'와 비슷한 개념이라고 할 수 있다.



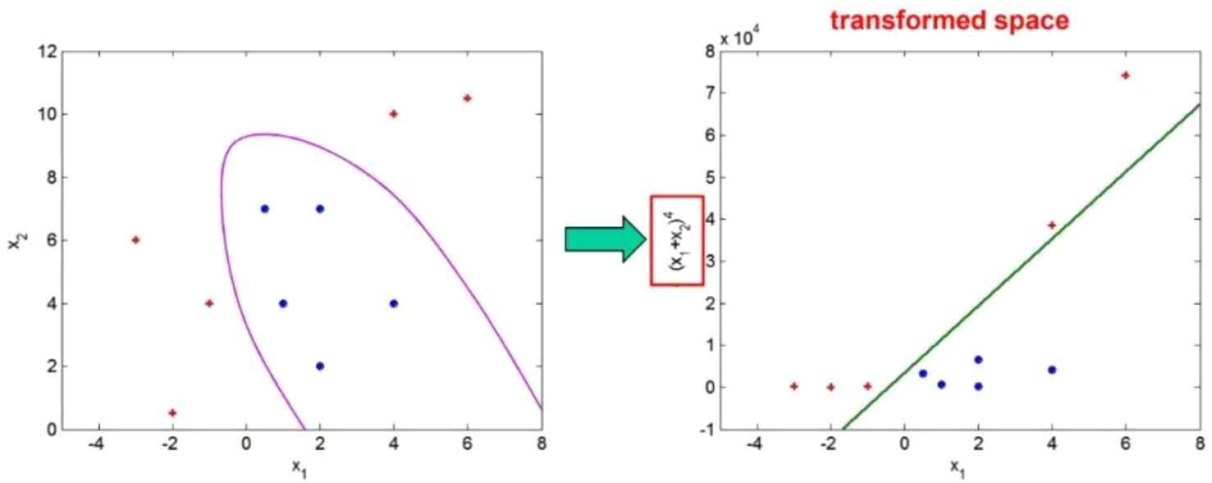
<출처 – https://ratsgo.github.io/machine%20learning/2017/03/26/tree/>

전체적으로 보면 나무를 뒤집어놓은 것과 같은 모양이다. 초기지점은 root node이고 분기가 거듭될수록 그에 해당하는 데이터의 개수는 줄어든다. 각 terminal node에 속하는 데이터의 개수를 합하면 root node의 데이터수와 일치한다. 바꿔 말하면 terminal node 간 교집합이 없다는 뜻이다. 의사결정나무는 분류(classification)와 회귀(regression) 모두 가능하다. 범주나 연속형 수치 모두 예측할 수 있다는 것이다.

Decision Tree는 Training set을 이용해 매 단계마다 불순도(불확실성)를 최대한 감소하게 하는 설명변수의 값들을 찾아놓고 새로운 관측치가 주어졌을 때 이 기준에 따라 분류한다. 새로운 관측치과 최종적으로 위치한 노드에서 가장 많은 비율을 차지하고 있는 클래스로 그 관측치의 클래스를 예측하는 방법이다. 불순도를 측정하는 지표로는 대표적으로 엔트로피(Entropy), 지니 불순도(Gini Impurity)가 있으며 그 외에도 카이제곱통계량, 분산의 감소량 등이 있다. 이런 지표에 따라 관측치를 분류해나가는 Decision Tree는 모델의 해석이 쉽고 계산량 대비 좋은 성능을 내지만 과적합 문제가 발생할 수 있다는 단점이 있다. 과적합을 막기 위해 가지치기(Pruning)의 과정이 필요하며 앙상블 기법이 대안으로 사용될 수 있다.

4.2. Support Vector Machine

SVM(Support Vector Machine) 또한 분류와 회귀에 모두 사용된다. SVM 알고리즘은 확실한 간격으로 구분되는 다른 클래스가 포함된 트레이닝 데이터 포인트를 이용해 모델을 만든다. 분리된 값은 최대가 되도록 최적화시키다. 샘플들 간의 간격을 보통 support vector라고 부르며 두 개의 클래스를 분리한 간격의 중심점은 최적분리 초평면(Separating Hyperplan)이라고 한다. 다음 그림을 보자.



Data mining - chapter 4. Classification : Alternative Techniques by Jun-Geol Baek

위의 그림은 Kernel trick을 사용한 SVMd이며, Kernel SVM은 이와 같이 Linear Boundary를 통해 데이터를 완벽하게 분류할 수 없는 경우에 사용하는 방법 중 하나이다. Mapping function으로 고차원의 공간에 데이터들을 Transformation 시키고 그 차원에서 데이터들을 완전히 분류하는 Linear Boundary를 찾아내는 것이다. 이후 새롭게 input 되는 데이터가 속하는 클래스에 따라 예측을 진행할 수 있다.

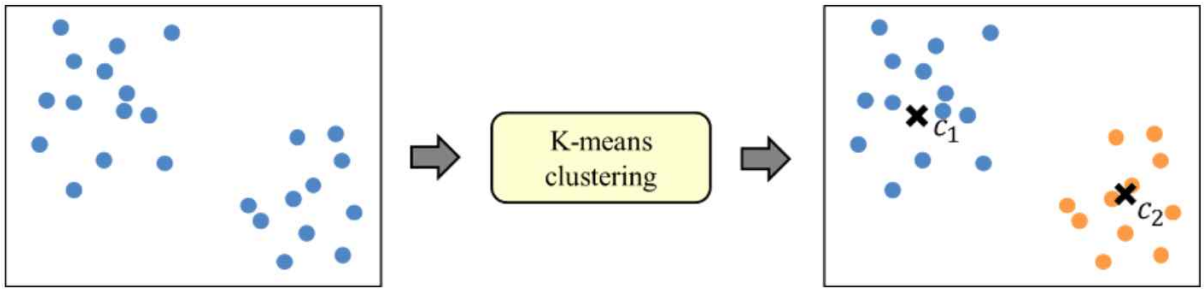
5. Clustering Based Prediction

Clustering은 비지도학습 기법으로 Target Variabel(Label)이 주어지지 않았을 때 활용한다. 주로 비슷한 데이터들을 묶어서 하나의 큰 Cluster를 형성하는 기법이다. Clsuter는 이를 중심으로 데이터의 패턴을 파악하는데 주로 사용된다. 대표적인 Clustering 기법에는 K-means, Hierarchical Clustering, DBSCAN이 있다.

5.1. K-means

K-means clustering은 데이터를 입력받아 이를 소수의 그룹으로 묶는 알고리즘이다. 이 알고리즘은 아래의 그림처럼 label이 없는 데이터를 입력받아 각 데이터에 label을 할당함으로써

군집화를 수행한다. K-means clustering은 개념과 구현이 매우 간단한 기본적인 clustering 알고리즘이면서도 실행 속도가 빠르고, 특정한 형태의 데이터에 대해서는 매우 좋은 성능을 보여주기 때문에 많이 이용되고 있다.



K-means clustering은 벡터의 형태로 표현된 N개의 데이터 X=$\{x_1, x_2, \ldots, x_n$에 대하여 데이터가 속한 cluster의 중심과 데이터 간의 거리의 차이가 최소가 되도록 데이터들을 K개의 cluster S=$\{s_1, s_2, \ldots, s_k\}$에 할당한다. 많은 연구에서 K를 자동으로 설정하기 위한 시도가 이루어졌지만, 기본적으로 K는 데이터를 분석하고자 하는 사람이 직접 설정해주어야 한다. 어떠한 방법을 통해 K를 설정하였다고 가정할 때, k-means clustering의 동작은 아래의 최적화 문제로 표현될 수 있다.

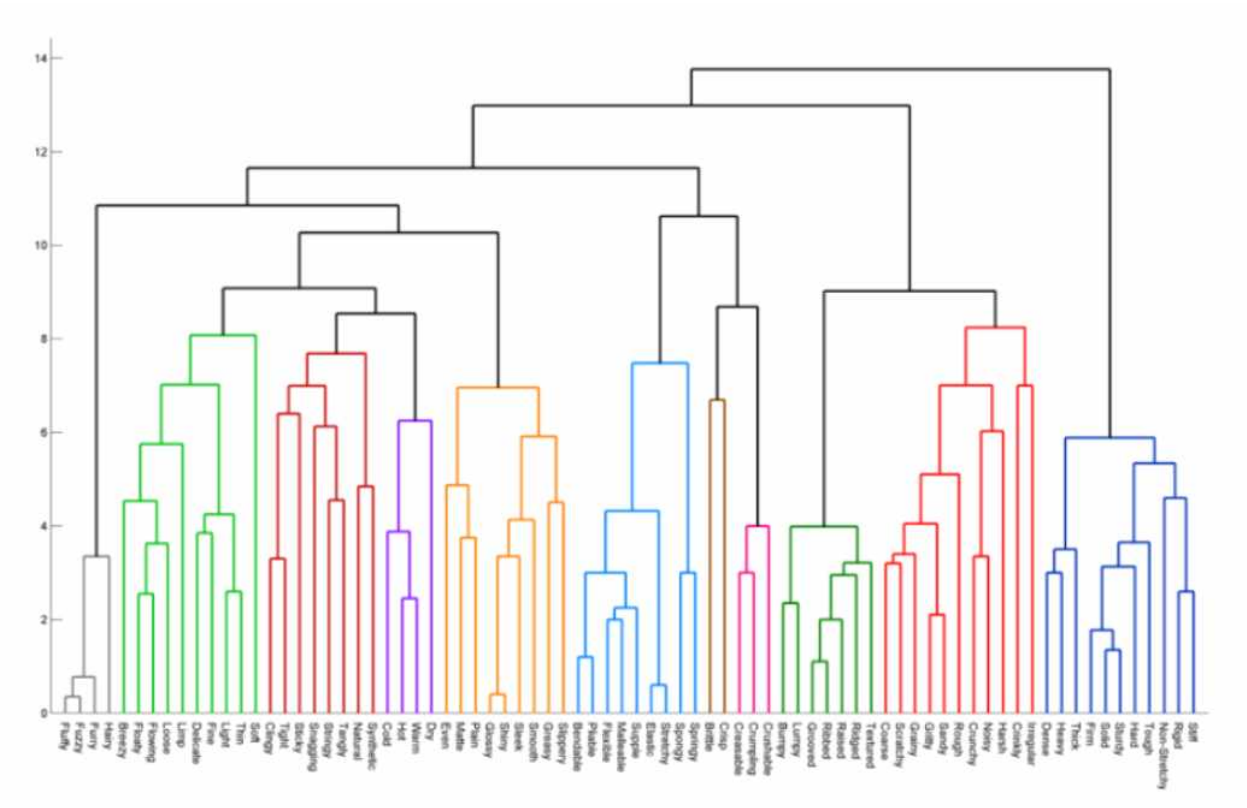
$$\underset{\boldsymbol{r},\boldsymbol{c}}{\operatorname{argmin}} \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - c_k\|^2$$

r_{nk}는 n번째 데이터가 k번째 cluster에 속하면 1, 아니면 0인 값을 갖는 binary variable이며, c_k는 k번째 cluster의 중심을 뜻한다. k-means clustering을 실행한다는 것은 주어진 데이터 X에 대하여 r_{nk}와 c_k 값을 설정하는 것과 같다. 즉 각 Cluster의 Centroid와 그 Cluster 내 점들 간의 거리를 최소화하고, Cluster들의 Centroids 간의 거리를 최대화하는 방식으로 수렴한다. 다만 Cluster 개수인 K가 사전에 결정되어야 하고, Initial Centroid에 따라 성능의 차이

가 있다.

5.2. Hierarchical Clustering

hierarchical clustering은 계층적 트리 모형을 이용해 개별 개체들을 순차적, 계층적으로 유사한 개체 내지 그룹과 통합하여 군집화를 수행하는 알고리즘이다. K-평균 군집화(K-means Clustering)와 달리 군집 수를 사전에 정하지 않아도 학습을 수행할 수 있다. 개체들이 결합되는 순서를 나타내는 트리형태의 구조인 덴드로그램(Dendrogram) 덕분인데, 아래 그림과 같은 덴드로그램을 생성한 후 적절한 수준에서 트리를 자르면 전체 데이터를 몇 개 군집으로 나눌 수 있게 된다.



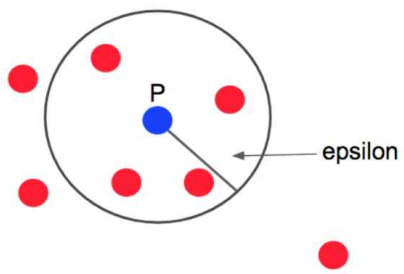
<출처 – https://ratsgo.github.io/machine%20learning/2017/04/18/HC/>

모든 Cluster가 nested 구조를 띄고 있으며, Dendrogram을 보고 적절하다고 생각되는 Cluster 개수를 정할 수 있다는 장점이 있다. 다만, 연산량이 많고 성능이 떨어진다는 단점이 있다.

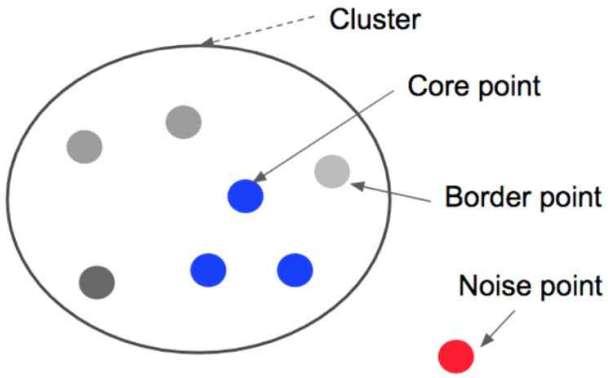
5.3. DBSCAN(Density-based spatial clustering of applications with noise)

앞에서 설명한 K Means나 Hierarchical 클러스터링의 경우 군집간의 거리를 이용하여 클러스터링을 하는 방법인데, 밀도 기반의 클러스터링은 점이 세밀하게 몰려 있어서 밀도가 높은 부분을 클러스터링 하는 방식이다. 쉽게 설명하면, 어느 점을 기준으로 반경 x내에 점이 n개 이상 있으면 하나의 군집으로 인식하는 방식이다.

먼저 점 p가 있다고 할 때, 점 p에서 부터 거리 e (epsilon)내에 점이 m(minPts) 개 있으면 하나의 군집으로 인식한다고 하자. 이 조건 즉 거리 e 내에 점 m개를 가지고 있는 점 p를 core point (중심점) 이라고 한다. DBSCAN 알고리즘을 사용하려면 기준점부터의 거리 epsilon값과, 이 반경 내에 있는 점의 수 minPts를 인자로 전달해야 한다. 아래 그림에서 minPts = 4 라고 하면, 파란점 P를 중심으로 반경 epsilon 내에 점이 4개 이상 있으면 하나의 군집으로 판단할 수 있는데, 아래 그림은 점이 5개가 있기 때문에 하나의 군집으로 판단이 되고, P는 core point가 된다.



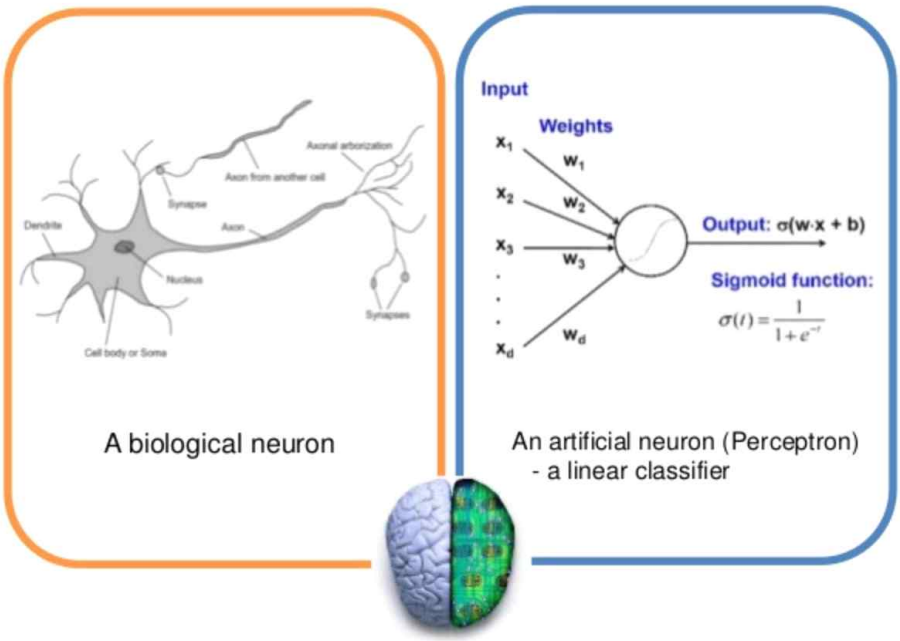
한 점을 중심으로 epsilon 반경 내에 minPts 이상의 점이 있으면 그 점을 중심으로 군집이 형성되고 해당 점을 core point라고 한다. 한 core point가 다른 core point로부터 형성된 군집의 일부가 되면 그 군집을 서로 연결되어 있다고 할 수 있다. 즉 하나의 군집으로 여기게 된다. 군집에는 속하지만 스스로 core point가 되지 못하는 점은 border pooint라고 하며 주로 군집의 외곽을 이루는 점이 된다. 그리고 어느 군집에도 속하지 못하는 점은 noise point가 된다.



DBSCAN 알고리즘의 장점은 K-means와 같이 군집의 개수를 정하지 않아도 되며, 군집의 밀도에 따라서 서로 다른 군집을 연결하기 때문에 기하학적인 모양을 가질 수도 있다는 것이다. 따라서 noise와 outlier에 취약한 기존의 거리 기반 clustering의 한계를 극복하였다.

6. Neural Network

Neural Network(신경망) 은 기계학습을 위한 알고리즘으로 인공신경망(ANN, Artificial neural network) 가 더 정확한 표현이지만 보통 줄여서 말한다. 인공신경망은 Perceptron을 단위로 가진다. 인공 신경망 모형의 하나인 퍼셉트론은 1957년에 Rosenblatt라는 사람에 의해서 처음 고안된 아주 오래된 알고리즘이다. 일반적으로 인공 신경망 시스템은 동물의 신경계를 따라 만들었기 때문에 개념적으로나 그 형태로나 비슷한 부분이 많다.



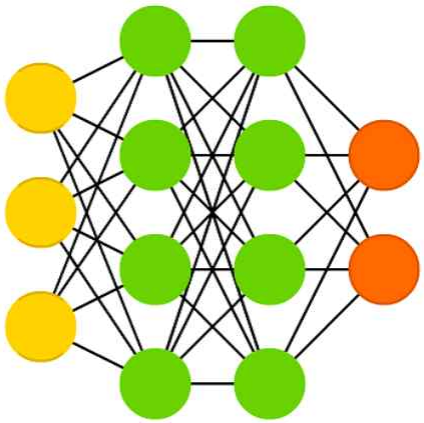
출처: https://www.slideshare.net/jbhuang/lecture-29-convolutional-neural-networks-computer-vision-spring2015

퍼셉트론은 다수의 신호(Input)을 입력받아서 하나의 신호(Output)을 출력한다 이는 뉴런이 전기신호를 내보내 정보를 전달하는 것과 비슷해 보인다. 가중치라고 부르는 이 weight는 각각의 입력신호에 부여되어 입력신호와의 계산을 하고 신호의 총합이 정해진 임계값(θ; theta,세타)을 넘었을 때 1을 출력한다. 넘지 못하면 0 또는 -1을 출력한다.

각 입력신호에는 고유한 weight가 부여되며 weight가 클수록 해당 신호가 중요하다고 볼 수 있다. 여기서 기계학습이 하는 일은 이 weight의 값을 정하는 작업이라고 할 수 있다. 학습 알고리즘에 따라 방식이 다를 뿐 이 weight를 만들어내는 것이 학습이라는 차원에서는 모두 같다고 할 수 있다.

이러한 Perceptron을 여러 개 묶은 모델이 Multi Layer Perceptron(MLP)이다. 그리고 MLP 내부의 Hidden Layer을 2개 이상으로 깊게 구성한 신경망이 Deep Neural Network(DNN) 즉 Deep Learning이다. 딥러닝에는 다양한 기법이 있다.

6.1. DFN(Deep Feedforward Network)



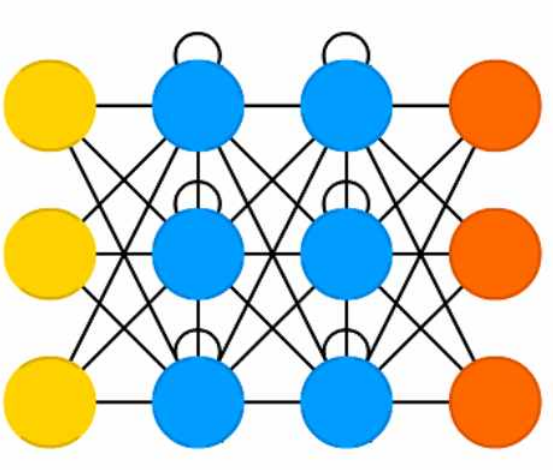
DFN은 딥 러닝에서 가장 기본적으로 이용되는 인공신경망이다. 그림에서도 볼 수 있듯이 DFN은 입력층, 은닉층, 출력층으로 이루어져 있으며, 보통은 2개 이상의 은닉층을 이용한다.

DFN에서 입력 데이터는 입력층, 은닉층, 출력층의 순서로 전파된다. 구조에서 알 수 있듯이 DFN은 현재 입력된 데이터가 단순히 입력층, 은닉층, 출력층을 거치면서 예측값으로 변환된 뒤에 현재 데이터에 대한 정보는 완전히 사라진다. 즉, 입력되었던 데이터들의 정보가 저장되지 않기 때문에 입력 순서에 따라 데이터 간의 종속성이 존재하는 시계열 데이터를 처리는 데는 한계점이 존재한다. 이러한 문제점을 해결하기 위해 제안된 것이 아래에서 소개할 Recurrent Neural Network (RNN)이다.

6.2. RNN(Recurrent Neural Network)

RNN은 시계열 데이터 (예를 들어, 문자열 및 센서 데이터)와 같이 시간적으로 연속성이 있는 데이터를 처리하기 위해 고안된 인공신경망이다. 시계열 데이터나 문자열은 일반적으로 앞에 입력된 데이터 (이전 시간의 데이터)에 의해 뒤에 입력된 데이터에 대한 예측이 영향을 받는다. 따라서, 단순히 현재 입력된 데이터를 입력층, 은닉층, 출력층의 순서로 전파하기

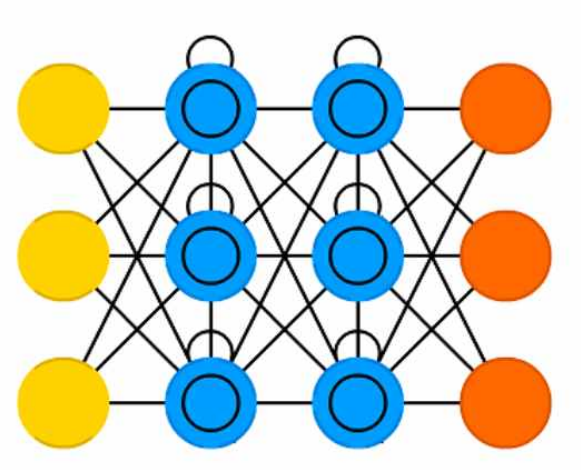
만 하는 DFN으로는 시계열 데이터에 대한 정확한 예측이 어렵다.



RNN은 그림과 같이 은닉층의 각 뉴런에 순환 (recurrent) 연결을 추가하여 이전 시간에 입력된 데이터에 대한 은닉층의 출력을 현재 시간의 데이터를 예측할 때 다시 은닉층 뉴런에 입력한다. 이러한 방식으로 RNN은 이전 시간에 입력된 데이터를 같이 고려하여 현재 시간에 입력된 데이터에 대한 예측을 수행한다. 그러나 그림 3과 같이 단순한 형태의 RNN은 역전파 (backpropagation) 알고리즘을 기반으로 오랜 시간에 걸쳐 경향성이 나타나는 데이터를 학습할 때 gradient가 비정상적으로 감소하거나 증가하는 vanishing/exploding gradient problem이 발생한다는 문제점이 있다. 이를 해결하기 위해 제안된 것이 다음에 설명할 Long Short-Term Memory (LSTM)이다.

6.3. LSTM(Long Short-Term Memory)

LSTM은 RNN에서 발생하는 vanishing/exploding gradient problem을 해결하기 위해 제안되었으며, 현재까지 제안된 RNN 기반의 응용들은 대부분 이 LSTM을 이용하여 구현되었다.



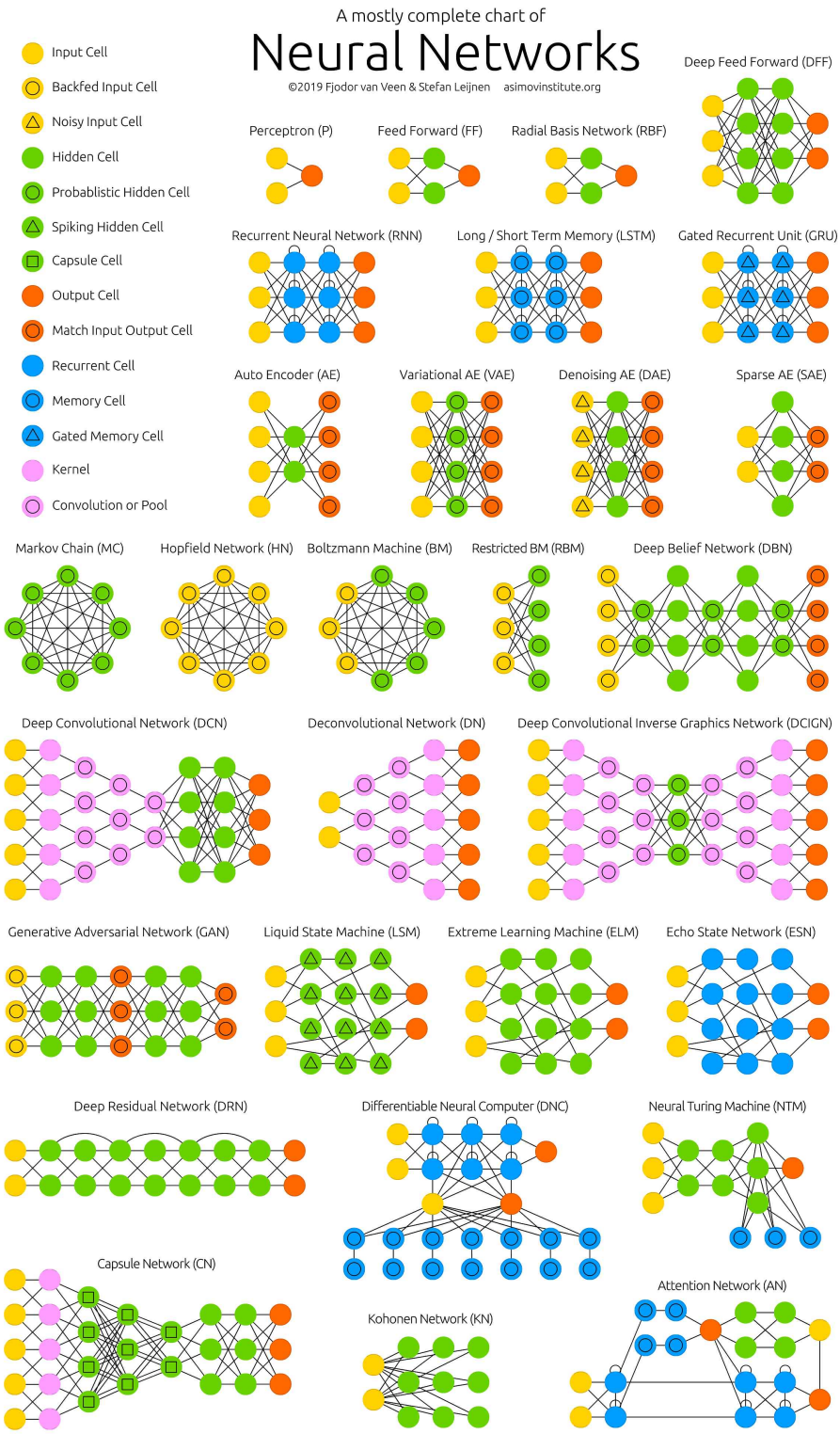
LSTM은 gradient 관련 문제를 해결하기 위해 forget gate, input gate, output gate라는 새로운 요소를 은닉층의 각 뉴런에 추가했다. 그림 4를 보면 기본적인 RNN의 구조에 memory cell이 은닉층 뉴런에 추가된 것을 볼 수 있으며, memory cell은 추가된 3개의 gate를 의미한다. LSTM에서 각 gate의 역할은 아래와 같다.

● **Forget gate :** 과거의 정보를 어느정도 기억할지 결정한다. 과거의 정보와 현재 데이터를 입력 받아 sigmoid를 취한 뒤에 그 값을 과거의 정보에 곱한다. 따라서, sigmoid의 출력이 0일 경우에는 과거의 정보를 완전히 잊고, 1일 경우에는 과거의 정보를 온전히 보존한다.

● **Input gate :** 현재의 정보를 기억하기 위해 만들어졌다. 과거의 정보와 현재 데이터를 입력 받아 sigmoid와 tanh 함수를 기반으로 현재 정보에 대한 보존량을 결정한다.

● **output gate :** 과거의 정보와 현재 데이터를 이용하여 뉴런의 출력을 결정한다.

이외에도 다양한 딥러닝 기법이 있으니 구글링을 통해 참고해보도록 하자. 단원의 마지막에 다양한 딥러닝 기법의 구조를 시각적으로 확인할 수 있는 자료를 첨부하였다.


A mostly complete chart of
Neural Networks
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org
Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)