3. Modeling
2019년 가을 학기

2019년 08월 25일
https://www.github.com/KU-BIG/KUBIG_2019_Autumn

김승연, 정용주의 '처음 배우는 머신러닝'을 많이 참고하였습니다.

1. Introduction

모델은 머신러닝의 시작점이다. 데이터 분석을 하거나 머신러닝으로 문제를 해결하려면 무엇을 해야하는가? 데이터에서 패턴을 찾아내서 이를 이용하면 되지 않을까라는 생각이 들 것이다. 그런데 과연 패턴은 존재할것인가? 사실 패턴이 있을 것이라는 생각은 데이터 자체에 대한 믿음, 즉 가정에 해당한다. 이러한 여러 가정을 한데 모은 것을 머신러닝에서는 모델이라고 한다.

쉽게 말해 현재 상태를 어떠한 시각으로 바라보고 어떠한 기대를 하고 있는가 하는 것이 모델이다. 이러한 모델은 머신러닝의 전과정에서 활용된다. 모델의 형태를 추측하고 구체화하고 평가하는 것이 머신러닝의 전부이기 때문이다. 머신러닝의 과정은 다음과 같다.

1. 모델 정하기 (데이터가 어떻게 생겼을지 가정하기)
2. 모델의 학습 목표를 수식화하기
3. 실제 데이터로 모델 학습하기 (최적화)
4. 평가하기
5. 필요에 따라 1에서 4의 과정을 반복

모델이란 가정에 따라 생성될 수 있는 함수들의 집합이다. 과정 1에서는 세상에 있는 무수히 많은 함수 중에서 특정한 형태의 함수를 지정하는 것이다. 이 데이터는 1차식의 관계가 있을 것이라고 일종의 가정을 가하는 것이다. 데이터를 우선적으로 살펴보고 그럴듯한 가정을 설정해야한다.

과정 2는 과정 1에서 가정한 함수를 수식화 한다. 그 다음에 과정 3에서 과정 2에서 수식화한 함수를 '학습'해야한다. 예를 들어 과정 2에서 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.

1.1 간단한 모델

만약 우리가 간단한 모델을 세우면 어떻게 될까? 간단한 모델이란 데이터의 구조가 간단하다는 뜻이다. 이는 굉장히 강력한 가정을 한다고 해석할 수 있다. 간단한 모델은 대표적으로 Linear Model이 있다. y와 x의 관계를 1차식으로 표현한 모델이다. Input(입력값)들에 Weight(가중치)를 곱한 값의 합, 즉 선형 결합으로 표현된다.

간단한 모델은 이해하기가 쉽다는 장점이 있다. 또한 간단한 모델은 일반적으로 데이터를 간단하게 설명하려하기에 가정 자체에 제약이 많고 데이터의 변화에 비해 모델 자체의 변화폭(Variance, 분산)이 작다. 모델 자체 변화폭이 작으므로 Outlier나 Noise가 있어도 영향을 적게 받기에 신경을 써야할 부분이 적다.

간단한 모델은 복잡한 관계를 학습할 수 없다. 입출력 데이터의 관계가 단순한 선형 관계가 아니라면 선형 모형으로는 제대로 표현하지 못한다. 간단한 모델은 강력한 가정을 필요로 하므로 데이터 자체가 함수가 요구하는 가정을 만족하지 않을 수 있다.

1.2 복잡한 모델

복잡한 모델은 앞서 설명한 간단한 모델보다 더욱 유연성을 중요시한다. 강력한 가정이 들어가지 않았기에 많은 데이터 형태에서 적용될 수 있다. 예를 들면, Decision Tree는 선형 모형과 달리 복수의 비교식을 무수히 생성할 수 있기 때문에 더 일반적이고 유연한 가정을 할 수 있다.

복잡한 모델은 간단한 모델에 비해 전체 데이터에 대한 일괄적인 가정이 적다. 위에서 언급한 Decision Tree에는 데이터 전체에 대한 가정이 거의 없다시피한다. 그렇다면 포괄적인 복잡한 모델을 사용하면 되지, 왜 굳이 모델의 학습 이후에 데이터가 모델의 가정을 만족하는지까지 일일이 확인해야하는 간단한 모델이 사용되는가?

복잡한 모델은 그 형태가 복잡하기 때문에 Input과 Output(출력값)간의 관계 규명이 잘 되지 않아 결과가 어떻게 도출되었는지 이해하기 어렵다. 이러한 모델은 형태가 복잡하기에 학습해야하는 Parameter가 많아서 학습속도가 느리다. 또한 한정된 데이터에서만의 변화를 그대로 학습하기에 새로운 데이터에 대한 성능이 떨어질 수 있다. 무엇보다도 가정이 강력할수록 데이터가 그 가정에 만족하기만 한다면 강력한 가정만큼 강력한 예측력을 갖출 수 있다.

2. Loss function (Cost function)

모델이 실제로 데이터를 바르게 표현했는지 혹은 얼마나 예측이 정확한지에 대한 수학적 표현이 Loss function(Cost function, 손실 함수, 비용 함수)이다. 그리고 모델이 실제로 데이터와 얼마나 다른지에 대한 값, 즉 Loss function의 결과값인 Loss(cost, error)이다. 앞서 다루었던 선형모형이 만든 직선이 데이터와 얼마나 떨어져 있는지 계산하는 함수가 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 추가적인 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 데이터 셋에서 각각의 데이터가 서로 확률적 독립이고 같은 분포를 가진다는 가정인 i.i.d(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, $\hat{y}$는 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 Evalutation 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 구글링 하길 바란다.

3. Optimization

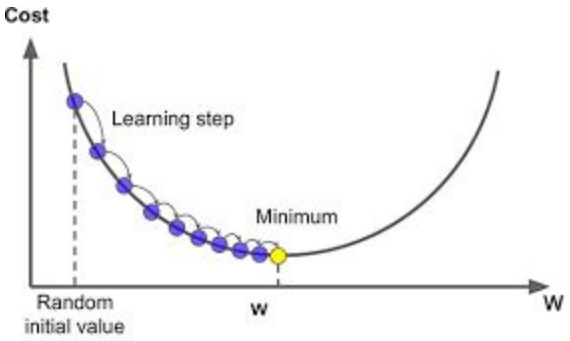
지금까지 데이터를 바라보는 시각인 모델과 모델이 데이터를 얼마나 잘 표현하는지를 측정하는 Loss function에 대해 알아봤다. 그렇다면 Loss function을 통해 모델을 학습하는 방법에 대해 알아보겠다.

Optimization(최적화)란 숫자, 함수, system의 extrema(minima or maxima)를 찾아가는 수학의 한 분야이다. 머신 러닝에서는 주로 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 최적화는 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 겪어야한다. 따라서 보통은 컴퓨터가 이러한 시행착오를 하도록 하는 알고리즘을 사용한다.

3.1 Gradient descent

Gradient descent(경사하강법)은 간단한 최적화 방법 중 하나로 임의의 지점에서 시작해서 경사를 따라 내려갈 수 없을 때까지 반복적으로 내려가며 최적화를 수행한다.

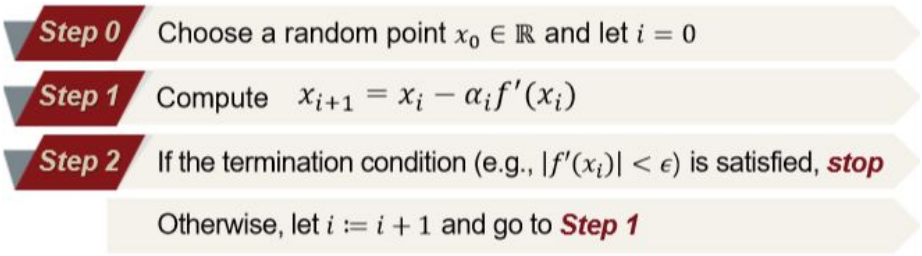


https://saugatbhattarai.com.np/what-is-gradient-descent-in-machine-learning/

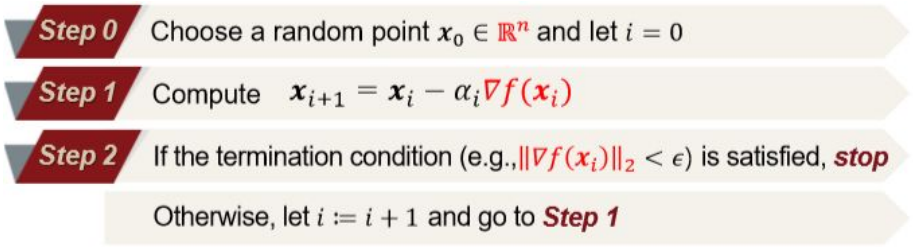
위 그림을 보면 그림에 점선으로 표시된 초기값으로 Parameter를 초기화한 후, 경사를 따라 차츰차츰 내려가면서 최저점에 도달한다. 위의 그림에서 표시된 Learning step은 주로 learning

rate(α_i)라고 하며 학습속도를 조정하는 Hyperparameter이다. 이는 경사를 따라 내려갈 때 얼만큼 내려갈지를 정해주는 척도이다.

- Single-variable case



- Multi-variable case



고려대학교 2019-1학기 데이터과학기초 14강

위의 알고리즘을 살펴보자.

Step 0) 임의의 초기치를 잡는다.

Step 1) 1차 미분 혹은 gradient로 최적화 방향(경사도)를 구한다. 기울기의 반대 방향으로 α_i 만큼 내려간다. (즉, 경사값에 -1을 곱한 방향으로 α_i 만큼 움직인다.)

step 2) Loss function의 출력값이 많이 줄었는지(ε 이상으로 줄어들었는지) 확인하고 그렇다면 step 1로 돌아가 계속 하강한다. 많이 줄어들지 않았다면 경사를 다 내려온 것으로 반복을 그만둔다.

Gradient descent에서는 초기값을 최적 모수와 얼마나 가까이 잡느냐에 따라서 학습 속도가 달라진다. 따라서 좋은 초기값을 잡는 것이 중요하다. 또한 learning rate도 잘 정해야한다. 만약 learning rate가 너무 작다면 알고리즘의 속도가 매우 느리다. 만약 learning rate가 너무 크다면 알고리즘은 최적값을 지나쳐서 진자 운동을 하게 될 것이고 수렴하지 않을 확률이 크다.

Learning rate를 설정하는 방법은 다양하다. 수렴이 안 될 시 좀 더 작은 값을 선택하면 된다. 혹은 매 iteration이 돌때마다 learning rate가 줄어들게 설정하면 된다. (ex. C/t, Ce^{-yt}, ...) 혹은 아래의 식을 선택해도 좋다.

$$\alpha_i = \text{argmin}_{s \geq 0} f(x_i - sf'(x_i))$$

3.2 Newton-Raphson's method

Newton-Raphson 방법은 Taylor 급수를 활용하였다. x에서의 Taylor 2차 근사식을 활용하여 다음 iteration의 값을 업데이트하고 이를 계속적으로 수렴할 때 까지 반복한다.

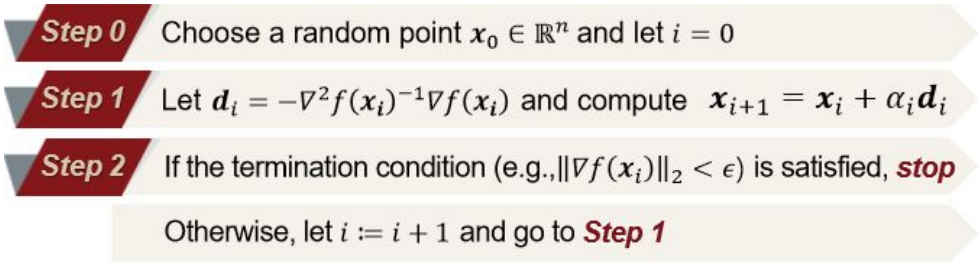
$$f(x) = f(\bar{x}) + f'(\bar{x})(x - \bar{x}) + \frac{f''(\bar{x})}{2}(x - \bar{x})^2$$

$$f(\boldsymbol{x}) = f(\bar{\boldsymbol{x}}) + \nabla f(\bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}}) + \frac{1}{2}(\boldsymbol{x} - \bar{\boldsymbol{x}})^T \nabla^2 f(\bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})$$

위의 두 식은 f(x)의 x에서의 2차 근사식(quadratic approximation)이다. 위의 식을 미분하여 좌변에 x만을 남기고 정리하자. 위의 식에서 미분할 때 필요한 가정은 $f'(x) = 0$이다. x는 Loss function의 최소값이므로 이의 미분은 0이 되기 때문이다. 마찬가지로 밑의 multi-variable의 식에도 다음과 같은 가정을 필요로 한다. $\nabla^2 f(\bar{x})$가 PD(Positive Definite)이다.

$$x = \bar{x} - \frac{f'(\bar{x})}{f''(\bar{x})} \qquad \mathbf{x} = \bar{\mathbf{x}} - \nabla^2 f(\bar{\mathbf{x}})^{-1} \nabla f(\bar{\mathbf{x}})$$

위의 식을 이용하여 모수를 지속적으로 업데이트 하면 된다. 위의 식은 아래의 알고리즘에서 d_i에 해당한다. 아래의 α_i는 learning rate이다. 아래의 알고리즘은 Gradient descent와 유사하므로 자세히 설명하지 않겠다.



고려대학교 2019-1학기 데이터과학기초 14강

이외에도 확률적 경사하강법(Stochastic Gradient Descent, SGD), 역전파(Backpropagation) 등이 있으며 이 둘은 뒤에 각각 12강, 11강에서 간단히 다룰예정이다. 더 자세한 내용은 구글에 Convex Optimization(볼록 최적화)을 검색해보길 바란다.

4. Model

4.1 Linear based model

Linear based model은 선형 모형의 변형에 해당한다. 선형 모델은 가장 간단하기에 머신러닝의 입문으로 적합하다. 이 모형은 예측력이 비교적 떨어져 이를 보완하는 많은 방법이 있다.

4.1.1 Linear Regression

Linear Regression은 각 attribute마다 weight가 주어진 linear combination형식이다.

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

위의 Linear Regression의 weight를 optimize하기 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.

4.1.2 Generalized Linear Model

일반화 선형모형(Generalized Linear Model)은 연속형 반응변수에 대한 보통의 회귀모형과 분산분석모형(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 반응 변수와 설명변수들을 결합시키면 $-\infty < \mu < \infty$의 범위를 가지게 되나 반응 변수는 0개부터 무한대 까지 '개수' 값을 가진다. 따라서 Poisson Regression의 경우 아래의 log link function을 사용한다. GLM에는 무척이나 다양한 종류가 존재하는데, 이후에 다룰 Logistic Regression도 GLM의 일종이다.

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

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

4.1.3 Logistic Regression

Logistic Regression은 범주형 변수를 예측하는 데에 있어서 가장 기본적인 통계적 방법론이다. Y가 범주형일 때 선형회귀 모형을 그대로 적용하면 y의 범위가 $(-\infty, +\infty)$가 되기에 적절하지 않다.

반면, 로지스틱 함수는 x값이 어느 범위이든 출력 결과 y가 항상 [0, 1] 사이가 되도록 하는 확률밀도함수로 규정되어있다. 단변량일 경우, $y = \frac{1}{1+e^{-x}}$ 라는 확률밀도 함수를 가지고 있다.
y는 [0,1] 사이의 값을 가져, 일반적으로 0.5 이상을 1, 이하를 0으로 변환하여 출력값을 제공한다.

4.1.4 Penalized linear model

전통적인 통계학에서는 unbiasedness를 기본으로 두고 모델링을 사용하였다. 하지만 과연 unbiasedness가 절대적인 진리마냥 꼭 지켜야하는가에 대해 의문이 있었다. 이에 bias는 조금 생기더라도 획기적으로 예측치의 variance를 줄여 예측 정확성을 올리는 모형들을 개발하였다.

1) Ridge

Ridge Regression은 기존의 최소제곱법의 objective function에 L2-norm penalty를 부여했다. 따라서 아래의 식과 같은 objective function을 가지게 되었다.

$$\text{Minimize} \quad \sum_{j=1}^{n}(y - \beta_0 - \sum_{i=1}^{n}\beta_i x_{ij}) + \frac{\lambda}{2}\sum_{j=1}^{n}\beta_j^2$$

Shrinkage penalty(L2 penalty)는 Linear Regression의 coefficient들이 0에 가까울수록 작아지며, 추정치들을 0으로 축소하는 경향을 갖는다. λ에 곱해진 1/2는 최소화하는데 큰 영향을 끼치지 않으며 미분이 편하도록 하는 상수이다. $\lambda = 0$ 일 때, penalty의 효과가 없기에 일반 Linear Regression과 같은 효과를 지닌다. λ가 ∞에 가까울수록, penalty의 효과가 강해지기에, Ridge Regression의 beta coefficient들을 0으로 당기려는 힘이 더욱 강하게 작용한다.

2) Lasso

LASSO Regression은 기존의 최소제곱법 objective function에 L1-norm penalty를 부여했다.

$$\text{Minimize} \quad \sum_{j=1}^{n}(y - \beta_0 - \sum_{i=1}^{n}\beta_i x_{ij}) + \lambda\sum_{j=1}^{n}|\beta_j|$$

LASSO Regression는 Ridge와 다르게 0에 가까워 의미가 없는 변수들을 아예 0으로 잡아당기기에 별도의 variable selection없이 model 하나로 variable selection의 효과를 가진다. $\lambda = 0$ 일 때, penalty의 효과가 없기에 일반 Linear Regression과 같은 효과를 지닌다. λ가 ∞에 가까울수록, penalty의 효과가 강해지기에, 더 많은 coefficient들을 0으로 만든다.

3) Elastic net

Ridge와 Lasso의 단점을 극복하기 위해서 L2과 L1 penalty를 섞은 모형이다. 이를 통해 Ridge와 Lasso의 특징이 섞여진 모델이 형성되었다.

$$\text{Minimize } \sum_{j=1}^{n}(y-\beta_0-\sum_{i=1}^{n}\beta_i x_{ij})+\lambda[\frac{1}{2}(1-\alpha)\sum_{j=1}^{n}\beta_j^2+\alpha\sum_{j=1}^{n}|\beta_j|],\ \alpha\in[0,1]$$

각 penalty term 앞에 곱해진 α값을 통해서 Ridge의 특성을 더 많이 가질 지 Lasso의 특성을 더 많이 가질지를 결정할 수 있다. α은 보통 [0,1] 사이에서 결정하게 된다. 만약 위의 식에서 α가 0이라면 일반적인 Lasso Regression이 된다. 반대로 α가 1 이라면 Ridge Regression이 된다. λ는 penalty term의 강도를 나타내면 커질수록 0으로 잡아당기려는 경향도 커진다.

4.2 Classification

Classification은 Supervised Learning의 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.

4.2.1 K-nn

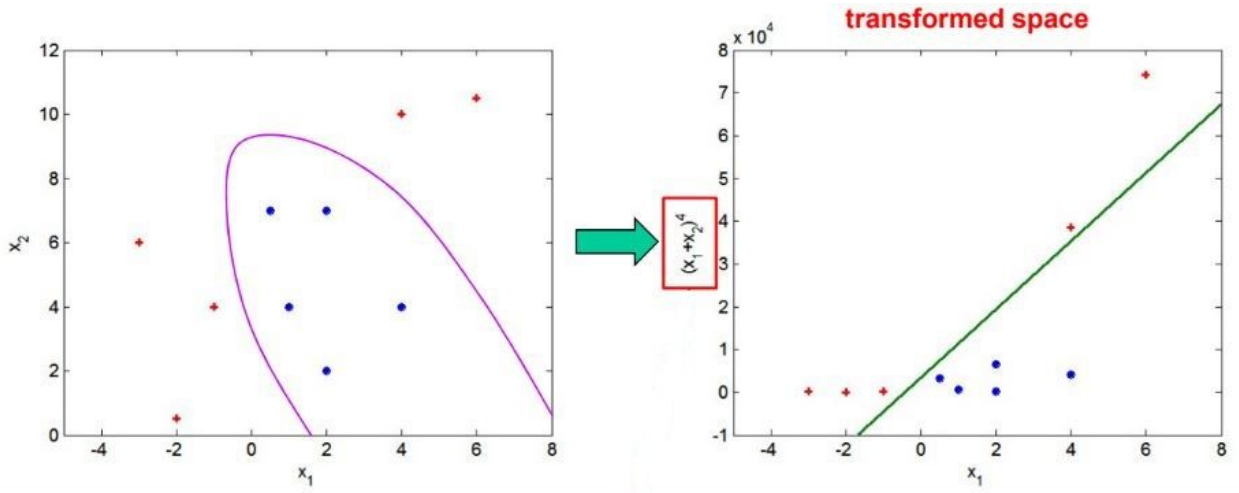
KNN은 대표적인 Lazy Learning 방법이다. 즉, model을 적합시키는 과정 대신 데이터를 저장하고 새로운 instance(data)가 주어지면 그에 비슷한, 그에 가까운 것 k개를 찾아 Majority Voting / Weighted Voting 등의 방법을 활용해서 Class를 결정하는 방식이다. 가장 주의해야 하는 부분은 scale에 민감하기 때문에 Normalize/Scaling 등의 처리가 필요하다.

4.2.2 Decision Tree

Decision Tree는 Training set을 이용해 매 단계마다 불순도를 최대한 감소하게 하는 설명변수의 값들을 찾아놓고 새로운 관측치가 주어졌을 때 이 기준에 따라 분류한다. 새로운 관측치가 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 낸다는 장점이 있지만, 과적합 되기 쉽다는 단점이 있다. 과적합을 막기 위해 가지치기(Pruning)의 과정이 필요하며, 앙상블 기법이 대안으로 사용될 수 있다.

4.2.3 Support Vector Machine (with kernel trick)

Kernel trick을 사용한 SVM은 다음 그림을 통해 쉽게 이해할 수 있다.



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

이와 같이 Linear Boundary를 통해서 데이터를 완벽하게 분류할 수 없는 경우에 사용하는 방법 중 하나가 Kernel SVM이다. Mapping function으로 고차원의 공간으로 데이터들을 Transformation 시키고 그 차원에서 데이터들을 완전히 분류하는 Linear Boundary를 찾아내는 것이다. SVM에 대해서는 추가적인 설명 세션이 없기 때문에 더 알고 싶다면 구글링을 하길 바란다.

4.3 Ensemble

Ensemble은 단일 모델의 성능을 높이기 위한 방법이다. 단일 모델 보다는 여러 모델의 결과를 결합함으로써 단일 모델의 성능은 떨어지더라도 결합 모델의 성능의 획기적으로 향상시킨다.

4.3.1 Bagging

Bagging은 이미 주어진 data에 Bootstrap Sampling을 이용하여 많은 model들을 생성하는 방법이다. 각각의 Bootstrap Sample에 model을 적합하여 결과를 도출한 후 결과들을 다수결 혹은 평균을 내는 방법으로 최종 결론을 계산한다. 이를 통해 단일 모델보다 성능이 향상되지만 앙상블의 특징인 여러 모델의 결합으로 인해 설명력은 떨어져 원인 결과 분석에는 적합하지 않다.

4.3.2 Ada Boosting

Ada Boosting은 전체 데이터로부터 sample을 무작위 복원추출으로 구성하여 다양한 분류기에 적합시킨다. 이전의 model에서 잘못 분류된 data를 중점적으로 뽑아 다음 모델을 fitting하는데 활용한다. 하지만 오분류된 data만 집중적으로 학습하기 때문에, Outlier와 Noise가 많은 데이터에서는 분석에 장애만 되는 Outlier와 Noise만 집중적으로 학습하는 경향이 크다.

4.3.3 Gradient Boost

Gradient Boosting은 boosting에 Gradient descent를 적용하여 Ada boosting의 Outlier와 Noise에 취약한 점을 극복한 방법이다. Loss function을 최소화하는 negative gradient를 활용했다. 다만 Gradient descent 특성으로 인하여 Ada boosting보다 연산량이 많아져서 이를 극복하는 여러 기법들이 나오게 되었다.

4.3.4 XGBoosting

XGBoosting는 Gradient boosting의 많은 연산량을 CPU 등의 시스템 자원을 효율적으로 분산 / 병렬 처리를 시켜 획기적으로 속도와 성능을 향상시킨 모델이다. 현존하는 머신러닝 기법 중에서도 그 성능이 매우 뛰어난 편이기에 많은 대회에서 이 모델을 활용하여 결과를 도출한다.

4.3.5 Random Forest

Random Forest는 여러 개의 Decision Tree를 구성하여 Majority Voting을 통해서 의사결정을 하는 방식이다. Random Forest는 각각의 Decision Tree를 만드는 데에 있어 attribute 요소를 무작위로 선정하여 무작위로 linear combination하여 새로운 feature들을 생성한 후 Tree를 구성한다. Decision tree계열에서는 가장 성능이 좋다고 알려져 있다.

4.4 Clustering

Clustering은 비지도학습 기법으로 Target Variable(Label)이 주어지지 않았을 때 활용한다. 주로 비슷한 데이터들을 묶어서 하나의 큰 Cluster를 형성하는 기법이다. Cluster는 이를 중심으로 데이터의 패턴을 파악하는데 주로 사용된다.

4.4.1 K-means

K-means Clustering은 Partitioning Clustering 방법론으로, 하나의 데이터가 하나의 Cluster에만 포함되는 exclusive한 Cluster를 형성한다. 각 Cluster의 Centroid와 그 Cluster 내 점들 간의

거리를 최소화하고, Cluster들의 Centroids 간의 거리를 최대화 하는 방식으로 수렴한다. 다만 Cluster 개수인 K가 사전에 결정되어야 하고, Initial Centroid에 따라 성능의 차이가 있다.

4.4.2 Hierarchical Clustering

Hierarchical Clustering은 Partitioning Clustering (ex K-means)과 다르게 nested된 Cluster들을 형성한다. 모든 Cluster가 nested 구조를 띄고 있으며, Dendrogram을 보고 적절하다고 생각되는 Cluster 개수를 정할 수 있다는 장점이 있다. 다만, 연산량이 많고 성능이 떨어진다는 단점이 있다.

4.4.3 DBSCAN

DBSCAN은 위의 모델과 달리 밀도를 기반으로 하는 Clustering기법이다. 밀도가 높은 곳을 중심으로 Cluster을 구성하고 밀도가 낮은 곳은 Noise로 무시함으로써 Noise와 Outlier에 취약한 기존의 거리 기반의 Clustering의 한계를 극복하였다.

4.5 Neural Network

구글의 딥마인드가 알파고로 이세돌을 꺾은 뒤로 사람들의 관심을 끌기 시작한 모델이다. 이 인공신경망은 Perceptron을 단위로 한다. Perceptron을 여러 개를 묶은 모델이 Multi Layer Perceptron(MLP)이다. MLP 내부의 Hidden Layer을 2개 이상으로 깊게 구성한 신경망이 Deep Neural Network(DNN) 즉 Deep Learning이다. 딥러닝에는 다양한 기법이 있으며, 최근에는 VAE와 GAN이 핫한 기법으로 자리하고 있다.