Model Part

2018320135 윤영로

2019.12.08

1 Model

1.1 Preknowledge

1. First Order necessary Condition(FONC) :
   $f(x) =$ local minimum $\implies \frac{\mathrm{d}}{\mathrm{d}x}(f(x)) = 0$
2. Convex Set : $\forall v, w \in Set, (t * v + s * w) \subseteq Set,$ **(such that.** $t + s = 1$**)**
3. Epi-Graph : $S = \{y | y >= f(x)\}$
3. Convex function : Epi-Graph is convex set.
4. Convex Properties :
   - $x^T \cdot Hessian \cdot x \succeq 0$,**(such that,** $x \geq 0$**)**
   - $f(x)$ has global minimum $\iff \frac{\mathrm{d}}{\mathrm{d}x}(f(x)) = 0$
   - A Concatenation of convex functions make convex functions.
   - Sub derivative :
     If $f(x)$ is convex function, then the only not derivative point of $f(x)$ is global minimum. And the derivative of this point(x^*) can be anything where $value \in (\lim_{x \to x^*-} f(x), \lim_{x \to x^*+} f(x))$
5. Convex Optimization :
   - **expression** : $argmin_x f(x)$,**(where,** $Ax \leq b$**)**
   - **f(x)** : convex function
   - **constraint** : convex set
6. Gradient Descent(Momentum Method) :
   - **정의** : 이전 갱신을 현재 갱신에 반영하여 Loss 값이 빠르게 감소하도록 w를 찾는 방법
   - **Implementation** :
     1. Find unit gradient ($g = G/\|G\|$)

2. Set direction for decreasing a gradient ($p = -g$)
3. update next weight
   $w^{i+1} = w^i + step * direction + gravity * (w^i - w^{i-1})$
4. If the size of gradient converge on 0, then this algorithm terminates.
5. otherwise, repeat this until loop¡max iter.

7. Gaussian Distribution(Normal Distribution) :

   - **Definition** : $N(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$
   - **mean, median, mode** : μ
   - **variance** : σ^2
   - **Characteristic** :

     A. μ를 기준으로 대칭이다.
     B. μ에서 Likelyhood가 최대가 된다.
     C. $mean, median, mode = \mu$이다.
     D. $Domain \in (-\infty, +\infty)$일 때, 여러 Discrete Distribution은 Gaussian에 근접한다.
     E. $-N(x|\mu, \sigma^2)$ is convex because an Epi-Graph of this is convex set.

8. Laplace Distribution

   - **Definition** : $Laplace(\mu, b) = P(y|\mu, b) = \frac{1}{2b} exp(-\frac{|x-\mu|}{b})$
   - **mean, median, mode** : μ
   - **variance** : $2b^2$
   - **Characteristic**

     A. μ를 기준으로 대칭이다.
     B. μ에서 Likelyhood가 최대가 된다.
     C. $mean, median, mode = \mu$이다.
     D. μ에서 미분 불가능 하다.
     E. $-Laplace(\mu, b)$ is convex because an Epi-Graph of this is convex set.

1.2 Linear Regression

1.2.1 Definition

a **초평면** : N차원 공간을 분할하는 (N-1)차원의 평면

b **Linear Regression** :

   - features와 targets으로 이루어진 공간에서, targets을 예측하는 초평면을 찾는 모델

1.2.2 Expression

1. **expression** : $\boldsymbol{y = Xw + b}$
2. **symbol**
   - $\boldsymbol{X_{m,n}} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix}$: feature matrix
   - $\boldsymbol{y} = ({}_1, {}_2, \ldots, {}_n)$: target vector (예측해야 하는 값)
   - $\boldsymbol{w} = ({}_1, {}_2, \ldots, {}_n)$: weight vector (feature의 중요도)
   - $\boldsymbol{b} = (\_1, \_2, \ldots, \_n)$: bias vector (초평면의 위치)
3. **loss function**
   - **Residual Sum of Squares** :
     * **Definition** : $RSS(w) := \sum_{i=1}^{m}(y_i - (X_i w + b_i))$
     * **Matrix Form** : $RSS(w) := (y - (Xw + b))^2$
   - **expression** : $L(w) = RSS(w)$

1.2.3 Properties

1. Space의 basis를 변주하여 $P_n(R)$의 vector 또한 관측할 수 있다.
2. Convex Optimization Program.
3. parameteric model이므로, weight를 계산해야 한다.
4. non-parameteric model과 비교했을 때, fitting에는 시간이 더 소요되지만, 빠른 예측이 가능하다.(후에 모델 비교로 이동)

1.2.4 Optimization

- **goal** : Find $argmin_w L(w)$
- **Perspective of Probability** :
  - **expression** : $argmax_w P(y|X, w)$
  - **Likelyhood Probability** : $P(y|X, w) \sim N(y|\mu, \sigma^2)$
- **Properties** :
  - Convex Optimization.
  - This program has a global minimum at $\frac{\mathrm{d}}{\mathrm{d}x}(f(x)) = 0$
- **search method**

1. $M = (X^tX)$가 가역행렬이라는 가정 하에, Optimal Solution을 직접 구할 수 있다.
   - $w^* = M^{-1}X^ty = X^{-1}y$
2. Gradient Descent
   * **gradient** : $-2X^t(y - Xw)$

- **Implementation** :

$$\begin{aligned} \text{Method} &: \text{Find Optimal Solution directly} \\ \text{Algorithm} &: w^* = Optimal \end{aligned}$$

1.2.5 Prediction

$$\text{Algorithm} : \text{Return} < weight, data >$$

1.3 Ridge Regression

1.3.1 Definition

a **L2 regularization** :

- Definition : L2 norm($\|x\|$)을 이용한 제약
- Expression : $argmin_w f(x) + \lambda\|w\|^2$
- Perspective of Probability :
  1. **base1** : Weight에 대한 Gaussian prior를 도입($P(w|0, \sigma^2) \sim N(w|0, \sigma^2)$)
  2. **base2** : Weight에 대한 Posterior ($P(w|X, y) \sim P(y|X, w)P(w|0, \sigma^2)$)를 이용한다.
  3. **expression** : $argmax_w P(w|X, y)$

b **Ridge Regression** :

- L2 Regularization을 통해, 학습 데이터에 과적합 되지 않도록 만든 Linear Regression

1.3.2 Expression

1. **expression** : $\boldsymbol{y = Xw + b}$
2. **loss function** : $L(w) = RSS(w) + \lambda\|w\|^2$

1.3.3 Properties

1. Convex Optimization Program.
2. Degree of L2 regularization(λ) $\propto$ Simplest of Model
   a. The hypothesis of weight can prevent model from over fitting about training data sets.

   b. Too strong the hypothesis of weight can cause under fitting.

3. Weight Decay : Weight가 Gaussian distribution을 가정이 the entries of weight가 널뛰는 것을 막는다.

1.3.4 Optimization

- **goal** : Find $argmin_w RSS(w) + \lambda \|w\|^2$
- **Perspective of Probability** :
  - **expression** : $argmax_w P(w|X, y)$
  - **Posterior Probability** : $P(w|X, y) \sim P(y|X, w)P(w)$
  - **Likelyhood Probability** : $P(y|X, w) \sim N(y|\mu, {\sigma_1}^2)$
  - **Prior Probability** : $P(w) \sim N(w|0, {\sigma_2}^2)$
- **Properties** :
  - Convex Optimization.
  - This program has a global minimum at $\frac{\mathrm{d}}{\mathrm{d}x}(f(x)) = 0$
- **search method**
  1. $M = X^t X + \lambda I_n$이 가역행렬이라는 가정 하에, Optimal Solution을 직접 구할 수 있다.
     - $w^* = M^{-1} X^t y$
  2. Gradient Descent
     * **gradient** : $-2X^t(y - Xw) + 2w$
- **Implementation**

Method : Optimal Solution과 Gradient Descent를 모두 구현하여, Optimal을 직접 찾을 수 없을 때, Approximate solution을 찾도록 설계했다.

Algorithm :

1. If Optimal can be found directly, then $w_* = Optimal$
2. Otherwise, Use ridge gradient to do gradient descent.

1.3.5 Prediction

Algorithm : Return $< weight, data >$

1.4 LASSO Regression

1.4.1 Definition

a **L1 regularization** :

- Definition : L1 norm($|x|$)을 이용한 제약
- Expression : $argmin_w f(x) + \lambda \sum_{i=1}^{n} |w|$
- Perspective of Probability :
  1. Weight에 대한 Laplace prior를 도입 ($P(w|0,b) \sim Laplace(w|0,b)$)
  2. **base1** : Weight에 대한 Posterior ($P(w|X,y) \sim P(y|X,w)P(w|0,b)$ 를 이용한다.
  3. **expression** : $argmax_w P(w|X,y)$

b **Lasso Regression** :

- L1 Regularization을 통해, 학습 데이터에 과적합 되지 않도록 만든 Linear Regression

1.4.2 Expression

1. **expression** : $\boldsymbol{y = Xw + b}$
2. **loss function**: $L(w) = RSS(w) + \lambda \sum_{i=1}^{n} |w|$

1.4.3 Properties

1. Convex Optimization Program.
2. Degree of L1 regularization(λ) $\propto$ Simplest of Model
   a. The hypothesis of weight can prevent model from over fitting about training data sets.
   b. Too strong the hypothesis of weight can cause under fitting.
3. Feature Selection : $Weight = (w_1, \ldots, w_n)$가 Laplace Distribution을 따른다는 가정이 Optimal Solution일 때, 특정 w_j가 0이 되도록 만든다.

1.4.4 Optimization

- **goal** : Find $argmin_w RSS(w) + \lambda \sum_{i=1}^{n} |w|$
- **Perspective of Probability** :
  - **expression** : $argmax_w P(w|X,y)$
  - **Posterior Probability** : $P(w|X,y) \sim P(y|X,w)P(w)$
  - **Likelyhood Probability** : $P(y|X,w) \sim N(y|\mu, \sigma^2)$
  - **Prior Probability** : $P(w) \sim Laplace(w|0,b)$

- **Properties** :
  - Convex Optimization.
  - This program has a global minimum at $\frac{\mathrm{d}}{\mathrm{d}x}(f(x)) = 0$
- **search method**
  1. $L(w)$는 $w = 0$에서 미분이 불가능하기 때문에 Optimal solution을 직접 찾을 수 없다.
  2. Gradient Descent
     * **gradient** :
       a. **RSS term** : $\frac{\mathrm{d}}{\mathrm{d}w}(RSS(w))$
       b. **Constraint term** : $\frac{\mathrm{d}}{\mathrm{d}w_j}(\lambda \sum_{i=1}^{n} |w|) = \begin{cases} +\lambda & \text{if } w > 0 \\ 0 & \text{if } w = 0 (Subderivative) \\ -\lambda & \text{if } w < 0 \end{cases}$
       c. **gradient** : g = RSS term + Constraint term
- **Implementation** : Use ridge gradient to do gradient descent.

1.4.5 Prediction

Algorithm : Return $< weight, data >$

1.5 Elastic Net Regression

1.5.1 Definition

a **Elastic Net Regression** : L1, L2 Regularization을 동시에 사용하여, Ridge와 LASSO의 장점을 섞은 Linear Regression

1.5.2 Expression

1. **expression** : $\boldsymbol{y = Xw + b}$
2. **loss function**: $L(w) = RSS(w) + L2ratio * \lambda \|w\| + L1ratio * \lambda \sum_{i=1}^{n} |w|$

1.5.3 Properties

1. Convex Optimization Program.
2. L1 and L2 regularization make model to be smooth.
   a. The hypothesis of weight can prevent model from over fitting about training data sets.
   b. Too strong the hypothesis of weight can cause under fitting.
3. Weight Decay와 Feature Selection 특징을 동시에 갖고 있다. Ratio를 조절하여 어떤 특징을 우선할지 선택할 수 있다.

1.5.4 Optimization

- **goal** : Find $argmin_w RSS(w) + \lambda \sum_{i=1}^{n} |w|$
- **Perspective of Probability** :
  - **expression** : $argmax_w P(w|X,y)^2$
  - **Posterior Probability** :

    Expression : $P(w|X,y)^2 \sim P(y|X,w)P(w|0,\sigma^2) * P(y|X,w)P(w|0,b)$

    Mean : weight가 Gaussian Prior를 가짐과 동시에, Laplace Prior를 가진다.
  - **Likelyhood Probability** : $P(y|X,w) \sim N(y|\mu,\sigma^2)$
  - **Prior Probability** :
    - $p(w|0,\sigma^2) \sim N(w|0,\sigma^2)$
    - $P(w|0,b) \sim Laplace(w|0,b)$
- **Properties** :
  - Convex Optimization.
  - This program has a global minimum at $\frac{\mathrm{d}}{\mathrm{d}x}(f(x)) = 0$
- **search method**
  1. Laplace Prior를 가지기 때문에 Optimal solution을 직접 찾을 수 없다.
  2. Gradient Descent
     - **gradient** :

       a. **RSS term** : $\frac{\mathrm{d}}{\mathrm{d}w}(RSS(w))$

       b. **Ridge term** : L2 ratio * $\lambda * 2 * w)$

       c. **Lasso term** : $\frac{\mathrm{d}}{\mathrm{d}w_j}(\lambda \sum_{i=1}^{n} |w|) = \begin{cases} +\lambda & \text{if } w > 0 \\ 0 & \text{if } w = 0 (Subderivative) \\ -\lambda & \text{if } w < 0 \end{cases}$

       c. **gradient** : g = RSS term + 0.5*(Ridge term + Lasso term)
- **Implementation** : Use an elastic net gradient to do gradient descent.

1.5.5 Prediction

Algorithm : Return $< weight, data >$

1.6 K-Nearest-Neighbors Regression

1.6.1 Definition

- Definition : A Regression that use the nearest n data of target to predict target value.

1.6.2 Expression

- Expression : $y = \sum_{i=1}^{n} P_i * y_i$
  - $P_i = \frac{K(x,x_i)}{S}$
  - $K(x_i, x) = \|x_i - x\|^2$
  - $S = \sum_{j=1}^{n} K(x_j, x)$
- Perspective of Probability :
  1. $\forall x \in Neighbors$, Find the weight(P_i) by using norm
  2. P_i is a probability since $\sum_{i=1}^{n} P_i = 1$
  3. y_{target} = weighted average of neighbor's labels

1.6.3 Properties

1. Non-Parameteric Model, Lazy Model
2. Whenever predicting target, KNN creates new model by using neighbors.
3. It can predict targets that have non-linear relation better than linear regression.
4. 거리 측정 방식에 따라 정확도가 달라진다.

Norm	r2-score
L1 norm	0.41
L2 norm	0.28

Table 1: r2-score(neighbor = 2)

1.6.4 Optimization

- **goal** : Save training data in model.
- **Properties** :
  - KNN Regression doesn't make parameters s.t weights.

1.6.5 Prediction

Algorithm :

1. Calculate Distances between target and all train data.
2. Find n Neighbors.
3. Calculate $P_j = 1 - dist_j / \sum_{i=1}^{n} dist_i$
4. return $< P, label >$ where $P = (P_1, \ldots, P_n)$

1.7 Decision Tree Regression

1.7.1 Definition

a **Decision Tree** : edge를 결정하는 규칙과 결과를 tree structure로 표현한 것

b **Decision Tree Regression** : training data로 Decision Tree를 생성하여, target을 예측하는 Model

1.7.2 Expression

1. **expression** : $(x_1, x_2, \ldots, x_n, Y)$

2. **loss function**:

1.7.3 Properties

1. 예측 결과 분석이 쉽다.

2. 여러 종류의 features가 섞여 있어도, 어느정도 결과를 보장한다.

3. big data set에 좋다.

1.7.4 Optimization

- **goal** : Find Optimal edges and leaf

- **search method**

  1. : 매 분기마다 모든 변수를 고려하여 standard deviation이 작은 변수를 기준으로 분할한다.
  2. : 매 분기마다 target과의 상관관계가 큰 순서대로 변수를 정하여 그것을 기준으로 분할한다.

1.7.5 Implementation

1. **Model** :

   - Recursion을 이용한 Decision Tree
   - DecisionTreeRegressor와 node class를 구현
   - DecisionTreeRegreesor 안에서 node의 recursion으로 tree를 fitting했다.

1. 노드 분할 기준

   a **The number of least leaf** : 4 30개 중, accurancy가 좋은 개를 선택했다.

   b **Criteria variable for edges dividing**

* The data of our project is multivariate data. So we discussed how to dividing edges from the point of view depth and variable.
  1. **All possible conditions** : 각 node에서, 모든 변수를 고려하여 standard deviation이 작은 변수를 분할의 기준으로 삼는 방법이다. 좋은 Accuracy를 얻었으나, fitting time이 오래 걸리는 단점을 관측했다.
  2. **Correlation conditions** : target과의 상관관계가 큰 순서대로 기준을 정하는 방법이다. **1**보다 Accuracy는 감소했지만, fitting time이 급격히 줄어들었으며, 상관관계로 정렬하지 않은 것보단 Accuracy가 증가함을 관측했다.

c **Implementation for Regression** : To use decision tree for regression, $y_{predict}$ = mean of leaf y_{train}으로 구현했다.

1.8 모델 비교

Model	r2-score
Linear	0.68
Lasso(alpha = 1)	0.66
ElasticNet(alpha = 1, L1$_{r}atio$ = 0.5)	0.60
DicisionTree(All possible conditions)	0.54
Ridge(alpha = 1)	0.48
KNN(n = 2, L1 norm)	0.41
DicisionTree(Correlation conditions)	0.28
KNN(n = 2, L2 norm)	0.28

Table 2: r2-score(model)

Model	Degree 1	Degree 2
Linear	0.68	0.76
Lasso(alpha = 1)	0.66	0.25
ElasticNet(alpha = 1, L1$_{r}atio$ = 0.5)	0.60	0.24
Ridge(alpha = 1)	0.48	0.20

Table 3: r2-score(poly + model)

1.9 The Hard part of Implementation

1.9.1 Gradient Descent

Overview : Optimal Solution을 찾지 못하거나, 심지어 INF value가 나오는 등 많은 문제가 있었다. 그래서 여러 번 알고리즘을 수정하여 Optimal Solution을 찾도록 만들었다.

시도 1 :

1. Update rule : w ¡= w - step * gradient
2. Result : Can't find an optimal solution.
3. Gain :
   a Normalize의 중요성 : normalize를 하지 않았을 때, 1.0e-300 정도의 r2-score를 얻었고, normalize를 했을 때, 0.32 정도의 r2-score를 얻었다.
   b 해를 찾는 과정에서 종종 Overflow Error가 발생하는 것을 관측했다.

시도 2 :

1. Update rule :
   * 복수의 steps을 두어 여러 weight를 찾았다. $weights_{next}$
   * $W_{next} = argmin_w(gradient(weights_{next}))$
2. Result : Better than 1. But can't find an optimal solution.
3. Gain :
   a step을 여러 개 설정하여 complexity을 조금 희새했지만 더 좋은 결과를 얻었다.
   b convergence 조건을 $\|g\|^2 = 0$로 하여 계산량을 줄였다.

시도 3 : Use Momentum Method

1. Update rule :
   * momentum method를 도입하여, 이전 update 결과를 반영했다.
   * $W_{next} = argmin_w(gradient(weights_{next} + gravity * prvupdate))$
2. Result : Better than 2. But can't find an optimal solution.
3. Gain :
   a Unit Vector : update 방향 vector를 unit vector로 하자, overflow error가 사라졌다.
   b max iter를 올렸을 때, 정확도가 증가했지만, 너무 오래 걸리는 단점이 있었고, optimal solution을 찾을 수 없었다.

시도 4 : Use Momentum Method

1. Update rule :
   * Multiple steps and gravities
   * $W_{next} = argmin_w(gradient(next_weights)$
2. Result : Find Optimal solutions.
3. Gain :
   a 여러 스텝을 두어 Complexity가 증가했지만, Optimal Solution을 찾을 수 있었다.
   b max iter를 크게 두지 않아도 적절한 solution을 찾을 수 있었다.