Statistical
Machine Learning

6주차

담당 : 11기 명재성

Review

- Lagrange Multiplier Theorem

$$
\begin{aligned}
&\min_{\mathbf{x}} && f(\mathbf{x}) \\
&\text{subject to} && g_i(\mathbf{x}) \le 0, \quad \text{for} \quad i = 1, \cdots, m \\
& && h_j(\mathbf{x}) = 0, \quad \text{for} \quad j = 1, \cdots, \mathrm{k}
\end{aligned}
$$

Review

- Lagrange Multiplier Theorem

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) + \sum_{i}^{m} \alpha_i \, g_i(\mathbf{x}) + \sum_{j}^{\mathrm{k}} \gamma_j \, h_i(\mathbf{x})$$

$$\alpha_i \geq 0, \quad \text{for} \quad i = 1, \cdots, m$$

$$\gamma_j \geq 0, \quad \text{for} \quad j = 1, \cdots, \mathrm{k}$$

Review

- KKT conditions

1. $\nabla f(\mathbf{x}) + \sum_i^m \alpha_i \nabla g_i(\mathbf{x}) + \sum_j^{\mathrm{k}} \gamma_j \nabla h_i(\mathbf{x}) = 0$ (Stationary)

2. $\alpha_i g_i(\mathbf{x}) = 0, \quad$ for $\quad i = 1, \cdots, m$ (Complementary Slackness)

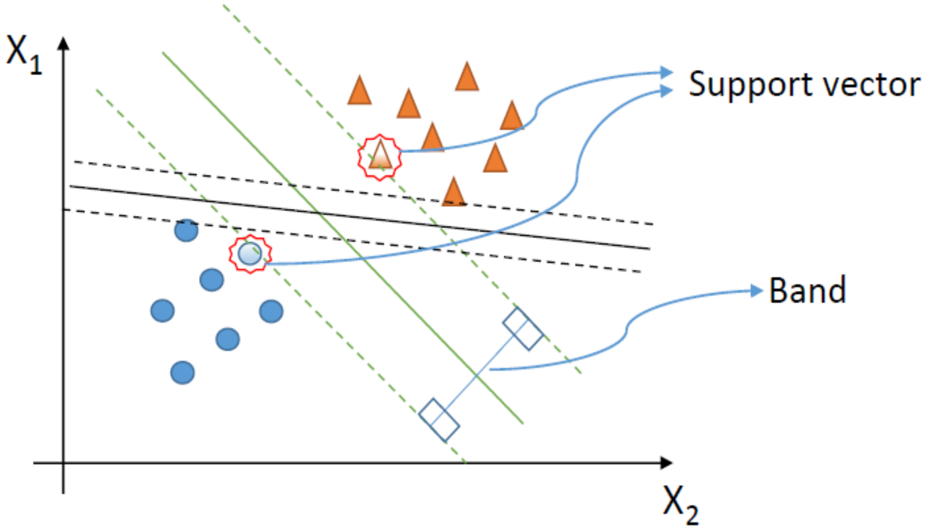
3. $g_i(\mathbf{x}) \leq 0, \quad$ for $\quad i = 1, \cdots, m \quad$ *and* (Primal Feasibility)

   $h_j(\mathbf{x}) = 0, \quad$ for $\quad j = 1, \cdots, \mathrm{k}$

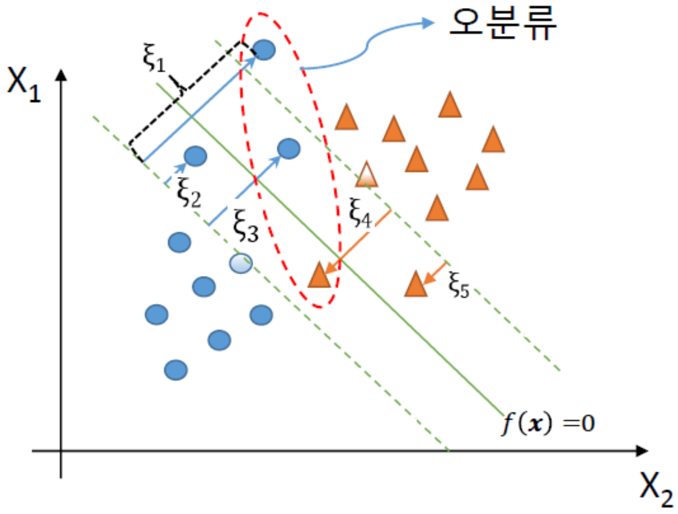
4. $\alpha_i \geq 0, \quad$ for $\quad i = 1, \cdots, m$ (Dual Feasibility)

Review

- Hard Margin Classifier



- Soft Margin Classifier



Review

- We want to maximize the width of the band.

$$\max_{\beta_0, \boldsymbol{\beta}} M \iff \min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2} ||\boldsymbol{\beta}||^2$$

subject to $y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) \geq \mathrm{M}$, for $i = 1, \cdots, \mathrm{n}$ subject to $y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) \geq 1$, for $i = 1, \cdots, \mathrm{n}$

and $||\boldsymbol{\beta}|| = 1$

Review

- Hard Margin Classifier

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2} ||\boldsymbol{\beta}||^2$$

subject to $y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) \geq 1$, for $i = 1, \cdots, \mathrm{n}$

- Soft Margin Classifier

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2} ||\boldsymbol{\beta}||^2$$

subject to $y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) \geq 1 - \zeta_i$

and $\zeta_i \geq 0,$

and $\sum_{\mathrm{i}}^{\mathrm{n}} \zeta_i \leq \tilde{C},$ for $i = 1, \cdots, \mathrm{n}$

Review

- Support Vector Machine solves

$$\min_{\beta_0, \boldsymbol{\beta}, \zeta_i} \ ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{\mathrm{i}}^{\mathrm{n}} \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \gamma_i \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \alpha_i \left[\, y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) - (1 - \zeta_i)\right]$$

Review

- Support Vector Machine solves

$$\min_{\beta_0, \boldsymbol{\beta}, \zeta_i} \; ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{\mathrm{i}}^{\mathrm{n}} \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \gamma_i \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \alpha_i \left[\, y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) - (1 - \zeta_i)\right]$$

$$\Longleftrightarrow \quad \min_{\beta_0, \boldsymbol{\beta}, \zeta_i} \; ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{\mathrm{i}}^{\mathrm{n}} \zeta_i \qquad \text{subject to} \quad y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) \geq 1 - \zeta_i$$

$$\text{and} \quad \zeta_i \geq 0, \quad \text{for } i = 1, \cdots, \mathrm{n}$$

Review

- Support Vector Machine solves

$$\min_{\beta_0, \boldsymbol{\beta}, \zeta_i} \; ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{\mathrm{i}}^{\mathrm{n}} \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \gamma_i \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \alpha_i \left[y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) - (1 - \zeta_i) \right]$$

$$\Longleftrightarrow \quad \min_{\beta_0, \boldsymbol{\beta}, \zeta_i} \; ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{\mathrm{i}}^{\mathrm{n}} \zeta_i \quad \text{subject to} \quad \zeta_i \geq [1 - y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i)]_+$$

$$\text{for } i = 1, \cdots, \mathrm{n}$$

Review

- Support Vector Machine solves

$$\min_{\beta_0, \boldsymbol{\beta}, \zeta_i} \ ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{\mathrm{i}}^{\mathrm{n}} \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \gamma_i \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \alpha_i \left[\, y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) - (1 - \zeta_i)\right]$$

$$\Longleftrightarrow \quad \min_{\beta_0, \boldsymbol{\beta}} \ ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{\mathrm{i}}^{\mathrm{n}} [1 - y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i)]_+$$

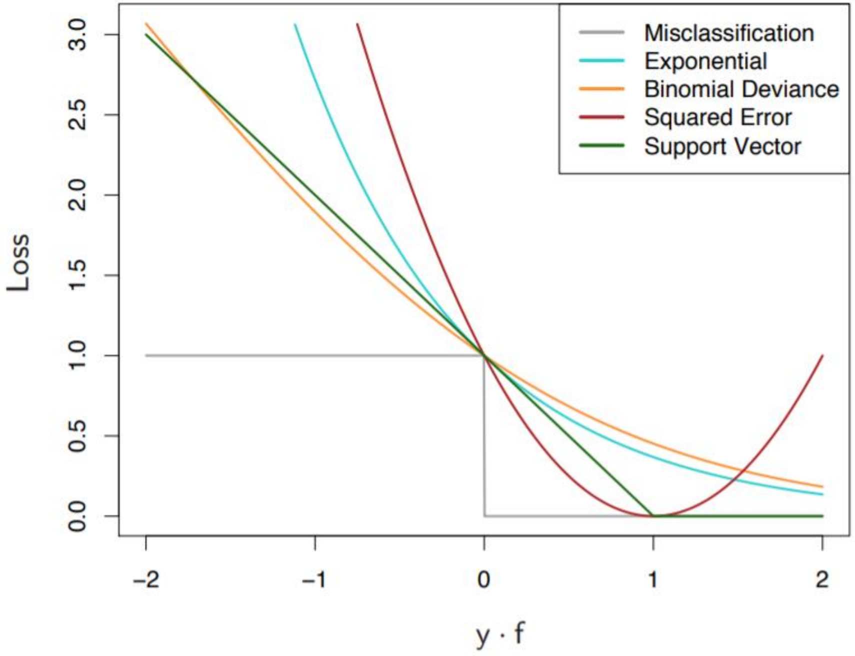
Review

- Support Vector Machine solves

$$\min_{\boldsymbol{\beta}} \sum_{i}^{n}[1 - y_i f(\mathbf{x}_i)]_+ + \lambda||\boldsymbol{\beta}||^2$$

⇨ Expression of " Loss + Penalty "

Review


Misclassification
Exponential
Binomial Deviance
Squared Error
Support Vector
Loss
0.0
0.5
1.0
1.5
2.0
2.5
3.0
−2
−1
0
1
2
y · f

Review

$$\min_{\beta_0, \boldsymbol{\beta}, \zeta_i} \ ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{\mathrm{i}}^{\mathrm{n}} \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \gamma_i \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \alpha_i \left[y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) - (1 - \zeta_i) \right]$$

(Stationary)

$$\begin{cases} \dfrac{\partial}{\partial \beta_0}\mathcal{L}_p: & \displaystyle\sum_{i}^{n} \alpha_i y_i = 0 \\ \dfrac{\partial}{\partial \boldsymbol{\beta}}\mathcal{L}_p: & \boldsymbol{\beta} = \displaystyle\sum_{i}^{n} \alpha_i y_i \mathbf{x}_i \\ \dfrac{\partial}{\partial \zeta_i}\mathcal{L}_p: & \alpha_i = C - \gamma_i \end{cases}$$

(Complementary Slackness)

$$\begin{cases} \alpha_i[\, y_i f(\mathbf{x}_i) - (1 - \zeta_i)] = 0 \\ \gamma_i \zeta_i = 0 \end{cases}$$

Review

$$\min_{\beta_0, \boldsymbol{\beta}, \zeta_i} \ ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{i}^{n} \zeta_i - \sum_{i}^{n} \gamma_i \zeta_i - \sum_{i}^{n} \alpha_i \left[y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) - (1 - \zeta_i) \right]$$

$$\Longleftrightarrow \quad \max_{\alpha_i} \ \sum_{i}^{n} \alpha_i + \frac{1}{2} \sum_{i}^{n} \sum_{j}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j \qquad \text{QP}$$

subject to $0 \le \alpha_i \le C$

and $\sum_{i}^{n} \alpha_i y_i = 0,$ for $i = 1, \cdots, n$

Review

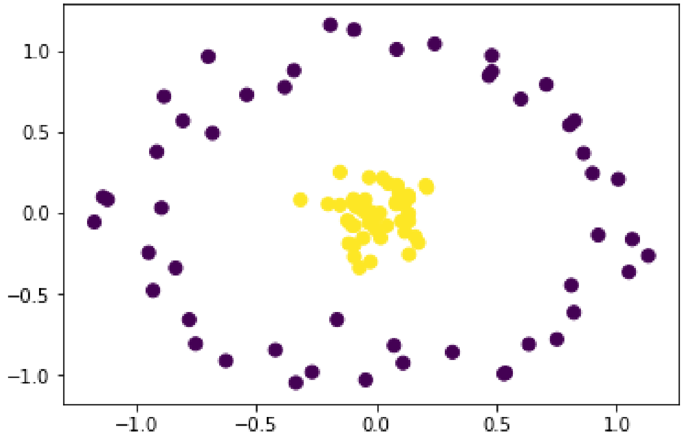
$$\min_{\beta_0, \boldsymbol{\beta}, \zeta_i} \ ||\boldsymbol{\beta}||^2 + \mathrm{C}\sum_{\mathrm{i}}^{\mathrm{n}} \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \gamma_i \zeta_i - \sum_{\mathrm{i}}^{\mathrm{n}} \alpha_i \left[y_i(\beta_0 + \boldsymbol{\beta}^T \mathbf{X}_i) - (1 - \zeta_i) \right]$$

$$\Longrightarrow \quad \widehat{\boldsymbol{\beta}} = \sum_{i}^{n} \widehat{\alpha_i}\, y_i \mathbf{x}_i$$

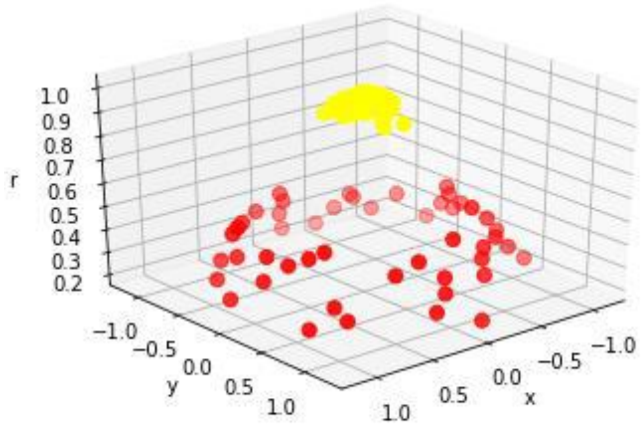
$$\widehat{\beta_0} = y_i - \widehat{\boldsymbol{\beta}}^T \mathbf{x}_\mathrm{k} \quad \text{for any support vector } \mathbf{x}_\mathrm{k}$$

$$\widehat{f(\mathbf{x}_i)} = \widehat{\beta_0} + \widehat{\boldsymbol{\beta}}^T \mathbf{x}_i$$

Review



(x_1, x_2)



$(x_1, x_2, \exp(-(x_1{}^2 + x_2{}^2))$

Review

```python
### Grid search에 의한 초모수 결정 (SVM) ###
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
pipe_svc = make_pipeline(StandardScaler(), SVC(random_state=1))
param_range = [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]
param_grid = [{'svc__C': param_range, 'svc__kernel': ['linear']},
              {'svc__C': param_range, 'svc__gamma': param_range,
               'svc__kernel': ['rbf']},
              {'svc__C': param_range, 'svc__degree': [2,3,4,5],
               'svc__kernel': ['poly']}]
gs = GridSearchCV(estimator=pipe_svc, param_grid=param_grid,
                  scoring='accuracy', cv=10)
gs = gs.fit(X_train, y_train)
print(gs.best_score_)
print(gs.best_params_)

clf = gs.best_estimator_
clf.fit(X_train, y_train)
clf.score(X_train,y_train)
clf.score(X_test, y_test)
```

Review

- Support Vector Regression solves

$$\underset{\boldsymbol{\beta}}{min} \sum_{i}^{n} L_{\epsilon}[y_i - f(\mathbf{x}_i)] + \lambda||\boldsymbol{\beta}||^2_{\mathcal{H}_K}$$

Review

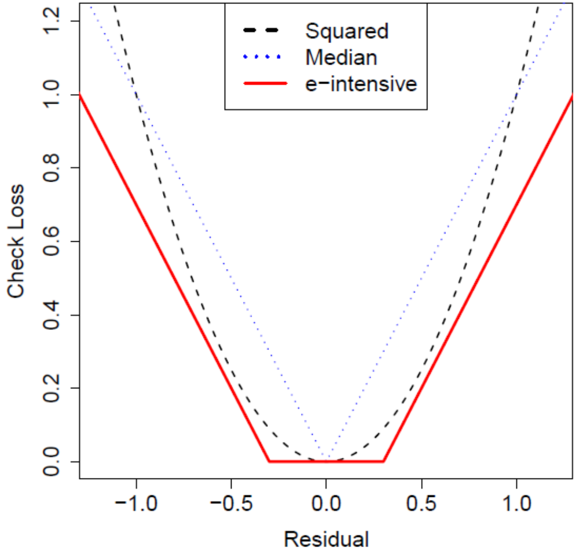


Figure: ϵ-intensive loss for SVR.

Jackknife Estimator

- Let $\hat{\theta}_{[i]}$ denotes the "Leave-One-Out" estimator

- Jackknife pseudo-values are defined by

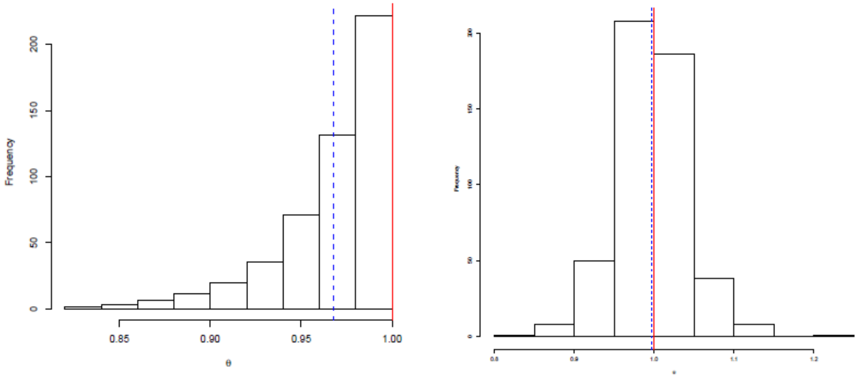
$$\hat{\theta}_{ps,i} = n\hat{\theta} - (n-1)\hat{\theta}_{[i]}$$

- Bias-adjusted Jackknife estimator is

$$\hat{\theta}_J = \frac{1}{n}\sum\hat{\theta}_{ps,i} = \hat{\theta} - (n-1)(\overline{\theta}_{[\mathrm{n}]} - \hat{\theta})$$

Jackknife Estimator

- Illustration of the bias corrected version of the sample maximum $\hat{\theta}$ for $U_i \overset{iid}{\sim} (0, 1)$. (i.e. $\theta = 1$)



(a) $\hat{\theta} = U_{(n)}$ (b) Bias-Corrected $\hat{\theta}$, $\hat{\theta}_J$

Bootstrap

- Bootstrap is a general technique for estimating unknown quantities associated with sampling distribution of estimators such as

  - Standard Errors
  - Confidence Intervals
  - p-values

Bootstrap

- Suppose $F(x)$ is the true population distribution.
- We estimate the functional of F based on the sample $X_1, \ldots, X_\mathrm{n}$.

Ex) Population expectation

$$\mu = E[X] = \int x f(x)\, dx \quad \left(= \int x\, dF(x)\right)$$

$$\hat{\mu} = \overline{X}_n = \frac{1}{n} \sum X_i \quad \left(= \int x\, dF_n(x)\right)$$

Bootstrap

- $F_n(x)$ denotes the empirical distribution of $(X_1, \dots, X_\mathrm{n})$.

$$F_n(x) = \frac{1}{n}\sum_{i}^{n} I\,(x \le X_i)$$

- Underlying fundamentals of this idea is

$$F_n(x) \quad \rightarrow \quad F(x)$$

Bootstrap

- Uncertainty / Randomness comes from

$$F(x) - F_n(x)$$

- Uncertainty quantification is not trivial since we only have a single $F_n(x)$ for unknown $F(x)$

Bootstrap

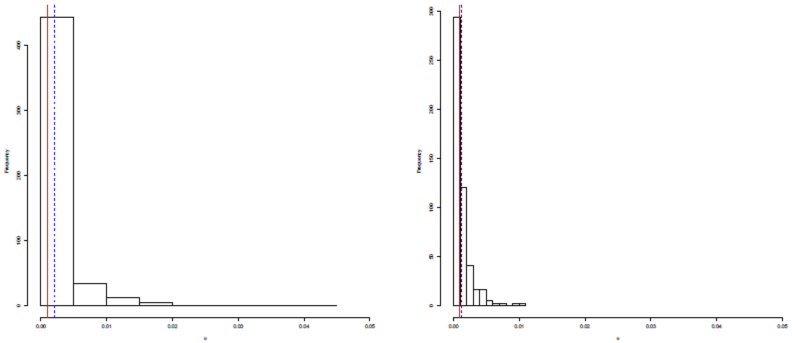
- Given a set of sample $(X_1, \ldots, X_\mathrm{n})$, a bootstrap sample denoted by $(X_1^*, \ldots, X_\mathrm{n}^*)$ is a random drawing samples **with replacement** from $(X_1, \ldots, X_\mathrm{n})$.

- The idea of bootstrap is

$$F_n^*(x) \quad \rightarrow \quad F_n(x) \qquad \approx \qquad F_n(x) \quad \rightarrow \quad F(x)$$

Bootstrap

- Comparison of variance estimator for sample maximum $\hat{\theta}$ for $U_i \overset{iid}{\sim} (0, 1)$. (i.e. $\theta = 1$)



(a) Jacknife (b) Bootstrap

Figure: Histogram of variance estimator for 500 independent repetitions: Monte Carlo MSE is .00903 for the jackknife estimator and .00108 for the bootstrap estimator.

For Imbalanced Data

- Undersampling

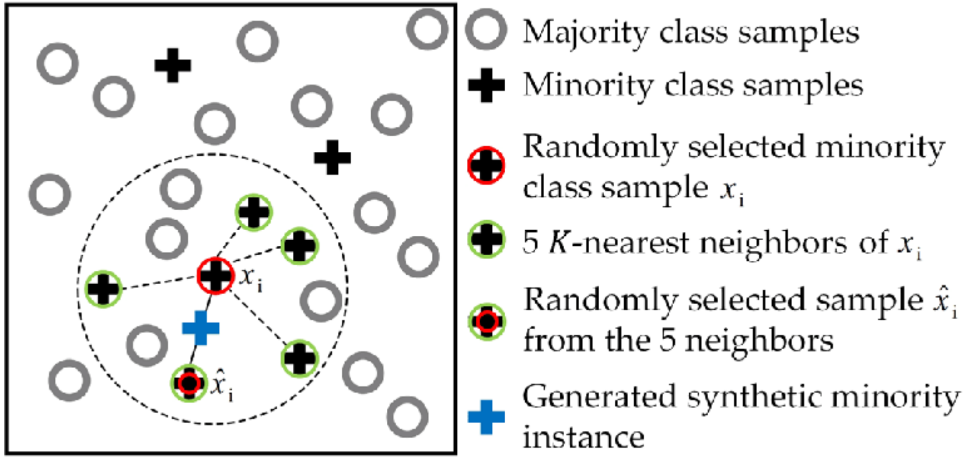
- Oversampling

  - SMOTE

  - ADASYN

SMOTE and ADASYN

- Synthetic Minority Oversampling TEchnique



SMOTE and ADASYN

- Synthetic Minority Oversampling TEchnique

$$\mathbf{x}_{syn} = \mathbf{x}_i + \lambda(\mathbf{x}_k - \mathbf{x}_i) \qquad \text{where} \quad \mathbf{x}_k \in S_k$$

- ADaptive SYNthetic sampling method

SMOTE and ADASYN

```
[ ] from collections import Counter
    from sklearn.datasets import make_classification
    from imblearn.over_sampling import SMOTE, ADASYN
    X, y = make_classification(n_classes=3, weights=[0.03, 0.07,0.9],n_features=10,
                               n_clusters_per_class=1, n_samples=2000, random_state=10)
    print('Original dataset shape %s' % Counter(y))
```

```
Original dataset shape Counter({2: 1795, 1: 141, 0: 64})
```

```
[ ] sm = SMOTE(random_state=42)
    X_res, y_res = sm.fit_resample(X, y)
    print('Resampled dataset shape %s' % Counter(y_res))
```

```
Resampled dataset shape Counter({2: 1795, 1: 1795, 0: 1795})
```

```
[ ] ada=ADASYN(random_state=42)
    X_syn,y_syn=ada.fit_resample(X,y)
    print('Resampled dataset shape from ADASYN %s' % Counter(y_syn))
```

```
Resampled dataset shape from ADASYN Counter({1: 1805, 2: 1795, 0: 1795})
```

Ensemble Learning

- Voting Classifier

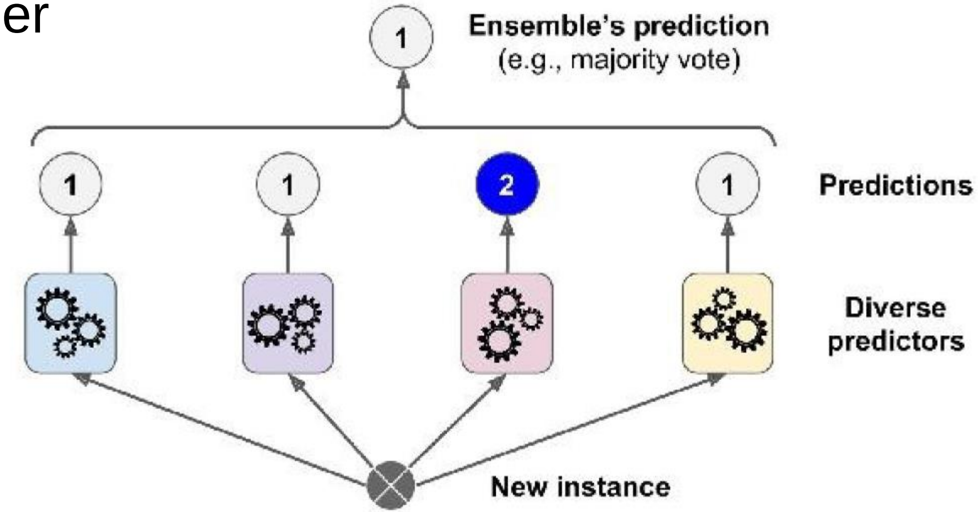


Figure 7-2. Hard voting classifier predictions

Ensemble Learning

- Bagging (Bootstrap Aggregating)

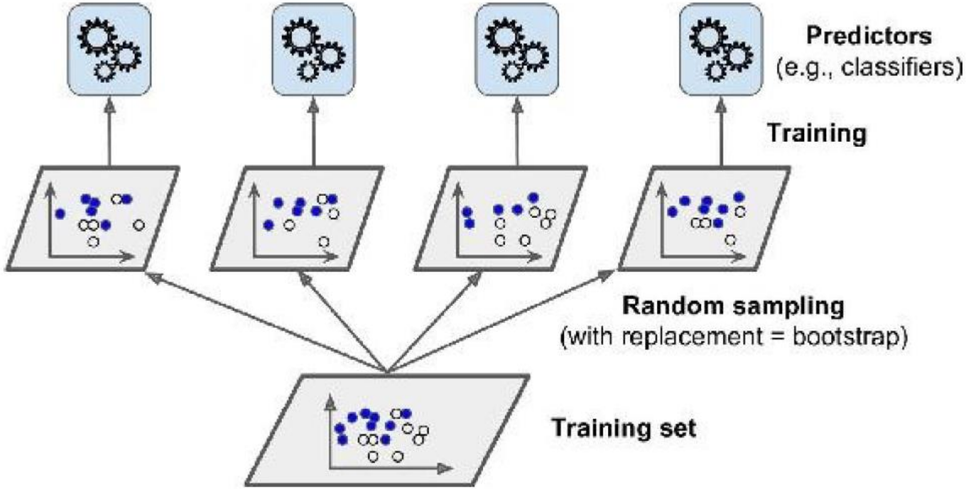
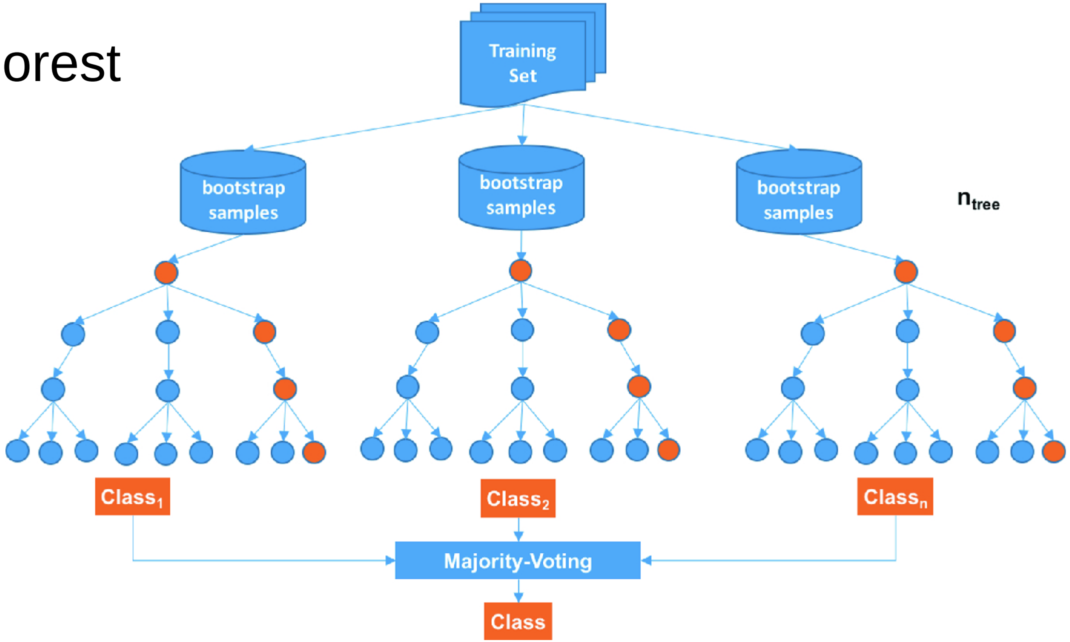


Figure 7-4. Pasting/bagging training set sampling and training

Ensemble Learning

- Random Forest



Ensemble Learning

- Random Forest

1. From $\mathbf{X}_{n \times p}$, obtain $\mathbf{X}^*_{n \times p}$ bootstrap samples.

2. For $\mathbf{X}^*_{n \times p}$, fit a decision tree by using randomly selected k ($\leq p$) features.

   In general, $k = \sqrt{p}$.

3. Repeat 1-2 M times. (M = # of trees)

Ensemble Learning

특성	로지스틱	KNN	LDA	SVM	의사결정 나무	최소제곱 선형모형	Neural network
자료 type 민감성	상	상	상	상	하	상	상
결측 자료 영향	상	중	상	상	하	상	상
이상치 민감성	상	하	상	상	하	상	상
표준화	선택	선택	선택	선택	불필요	불필요	필요
해석의 용이성	용이	난해	난해	난해	용이	용이	매우 난해
성능	중간	중간	중간	중간	중간	중간	높음

Ensemble Learning

- Boosting

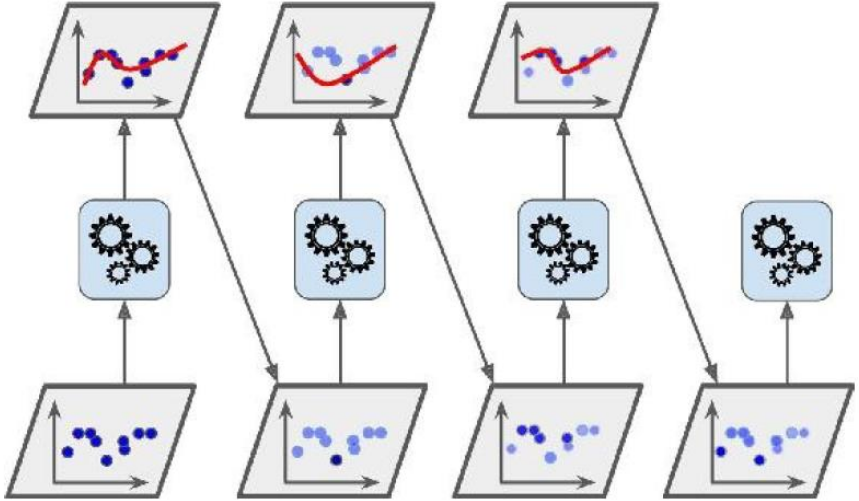


Figure 7-7. AdaBoost sequential training with instance weight updates

Ensemble Learning

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

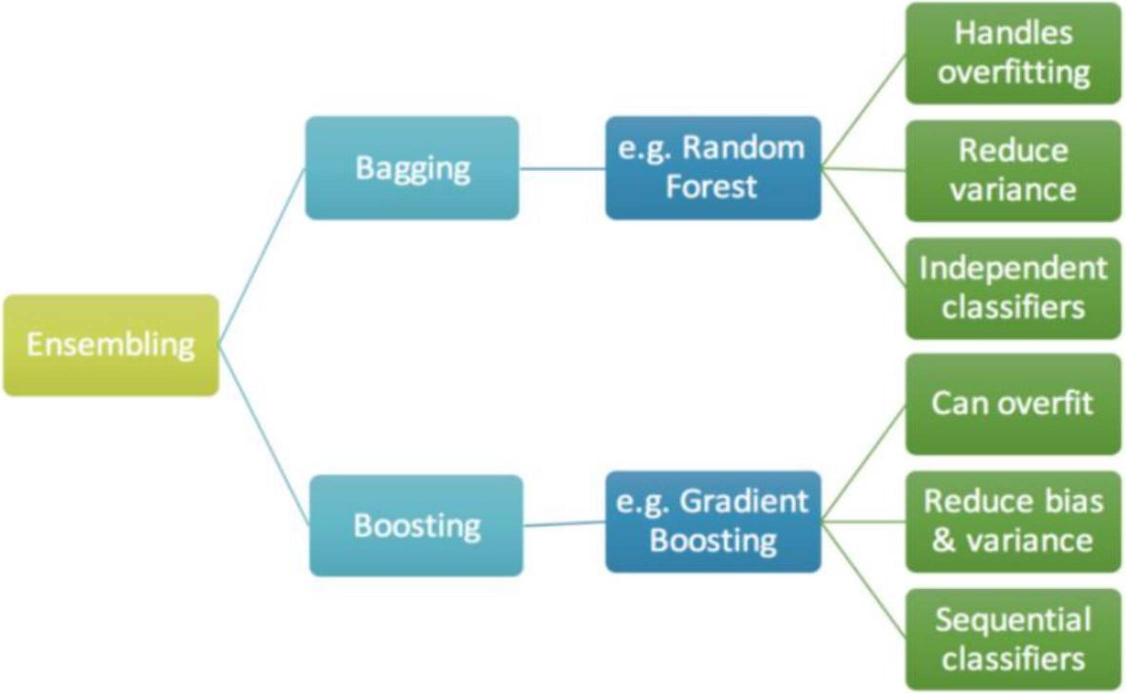
1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:

   (a) Compute

   $$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Ensemble Learning

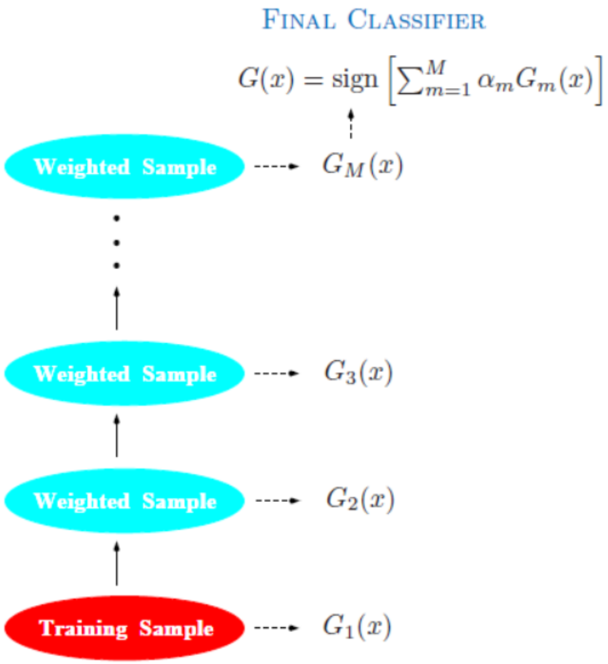


- 병렬적 모델결합
- 독립적으로 모델 구성
- 매 sampling마다 동일 가중치 부여

- 직렬적 모델결합
- 이전 모델의 오류를 바탕으로 새 모델 구성
- 학습오류 큰 데이터에 가중치 부여
- 단일 모델의 성능 낮을 경우

Ensemble Learning (AdaBoost)

- AdaBoost (Adaptive Boosting)



Ensemble Learning (AdaBoost)

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.

Ensemble Learning (AdaBoost)

- Change little bit…

(c) $as_m = \frac{1}{2}\log\left(\frac{1-err_m}{err_m}\right)$ → amount of say

(d) $w_i^{(m+1)} = \begin{cases} e^{-as_m} & \text{if } y_i = G_{m-1}(x_i) \\ e^{as_m} & \text{if } y_i \neq G_{m-1}(x_i) \end{cases}$ → $\sum w_i^{(m)} \neq 1$

Ensemble Learning (AdaBoost)

관측치	1	2	3	4	5	6	7	8
x	5	10	15	20	25	30	35	40
y	-1	-1	1	1	1	-1	-1	1
가중치	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$

<표 11.1> adaboost를 위한 학습데이터

$$G_1 = 2I(x \geq 12.5) - 1, \quad err_1 = \frac{2}{8} = 0.25, \quad \mathrm{as}_1 = \frac{1}{2} log(\frac{1 - err_1}{err_1}) = 0.55$$

$$G(x) = sign[\mathrm{as}_1 \cdot G_1] = sign[0.55 \cdot (2I(x \geq 12.5) - 1)]$$

Ensemble Learning (AdaBoost)

관측치	1	2	3	4	5	6	7	8
x	5	10	15	20	25	30	35	40
y	-1	-1	1	1	1	-1	-1	1
가중치	0.577	0.577	0.577	0.577	0.577	1.733	1.733	0.577
조정가중치	0.083	0.083	0.083	0.083	0.083	0.251	0.251	0.083

<표 11.2> 아다부스트를 위한 조정된 가중치의 계산

$$w_i^{(2)} = \begin{cases} e^{-as_1} = 0.577 & \text{if } G_1(x) = y_i \\ e^{as_1} = 1.733 & \text{if } G_1(x) \neq y_i \end{cases}$$

Ensemble Learning (AdaBoost)

관측치	1	2	3	4	5	6	7	8
x	5	10	20	30	30	35	35	40
y	-1	-1	1	-1	-1	-1	-1	1
가중치	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$

<표 11.3> 두 번째 tree stump를 위한 데이터셋

$\rightarrow$ reset $w_i^{(2)}$

$$G_2 = 2I(x \geq 37.5) - 1, \quad err_2 = \frac{1}{8} = 0.125, \quad \text{as}_2 = \frac{1}{2} log(\frac{1 - err_2}{err_2}) = 0.97$$

$$G(x) = sign[\text{as}_1 \cdot G_1 + \text{as}_2 \cdot G_2]$$

$$= sign[0.55 \cdot (2I(x \geq 12.5) - 1) + 0.97 \cdot (2I(x \geq 37.5) - 1)]$$

Ensemble Learning (AdaBoost)

관측치	1	2	3	4	5	6	7	8
x	5	10	15	20	25	30	35	40
y	-1	-1	1	1	1	-1	-1	1
가중치	0.379	0.379	2.638	2.638	2.638	0.379	0.379	0.379
조정가중치	0.038	0.038	0.270	0.270	0.270	0.038	0.038	0.038

<표 11.2> 아다부스트를 위한 조정된 가중치의 계산

$$w_i^{(3)} = \begin{cases} e^{-as_2} = 0.379 & \text{if } G_2(x) = y_i \\ e^{as_2} = 2.638 & \text{if } G_2(x) \neq y_i \end{cases}$$

Exponential Loss and AdaBoost

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:

   (a) Compute

   $$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Exponential Loss and AdaBoost

- Our Final model is

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

- If we use squared error loss : $L(y, f(x)) = (y - f(x))^2$

$$L[y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)] = (y_i - f_{m-1}(x_i) - \beta b(x_i; \gamma))^2$$

$$= (r_{im} - \beta b(x_i; \gamma))^2$$

Exponential Loss and AdaBoost

- If we use exponential loss : $L(y, f(x)) = \exp(-yf(x))$

$$(\beta_m, G_m) = \underset{\beta, G}{argmin} \sum_{i=1}^{n} \exp\left[-y_i(f_{m-1}(x_i) + \beta G(x_i))\right]$$

$$= \underset{\beta, G}{argmin} \sum_{i=1}^{n} \exp\left(-y_i f_{m-1}(x_i)\right) \exp(-\beta y_i G(x_i))$$

$$= \underset{\beta, G}{argmin} \sum_{i=1}^{n} w_i^{(m)} \exp(-\beta y_i G(x_i))$$

Exponential Loss and AdaBoost

$$\sum_{i=1}^{n} w_i^{(m)} \exp(-\beta y_i G(x_i)) = \sum_{y_i = G(x_i)} w_i^{(m)} e^{-\beta} + \sum_{y_i \neq G(x_i)} w_i^{(m)} e^{\beta}$$

$$= (e^{\beta} - e^{-\beta}) \sum_{i=1}^{n} w_i^{(m)} I(y_i \neq G(x_i)) + e^{-\beta} \sum_{i=1}^{n} w_i^{(m)}$$

$$G_m = \underset{G}{argmin} \sum_{i=1}^{n} w_i^{(m)} I(y_i \neq G(x_i))$$ $\rightarrow$ tree stump using Impurity

Exponential Loss and AdaBoost

$$\frac{\partial}{\partial \beta}\left[(e^{\beta} - e^{-\beta})\sum_{i=1}^{n} w_i^{(m)} I(y_i \neq G(x_i)) + e^{-\beta}\sum_{i=1}^{n} w_i^{(m)}\right]$$

$$= \beta\left(e^{\beta} + e^{-\beta}\right)\sum_{i=1}^{N} w_i^{(m)} I\left(y_i \neq G_m(x_i)\right) - \beta e^{-\beta} \cdot \sum_{i=1}^{N} w_i^{(m)} \overset{set}{=} 0$$

$(e^{\beta} + e^{-\beta}) \cdot err_m = e^{-\beta},$ where $err_m = \dfrac{\sum_{i=1}^{\mathrm{n}} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{n} w_i^{(m)}}$

Exponential Loss and AdaBoost

$$(e^{\beta} + e^{-\beta}) \cdot err_m = e^{-\beta}, \quad \text{where} \quad err_m = \frac{\sum_{i=1}^{\mathrm{n}} w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{n} w_i^{(m)}}$$

$$e^{\beta} \cdot err_m = e^{-\beta}(1 - err_m)$$

$$\beta_m = \frac{1}{2} \log\left(\frac{1 - err_m}{err_m}\right)$$

$\rightarrow$ amount of say

Exponential Loss and AdaBoost

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x)$$

$$w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$$

$$w_i^{(m+1)} = \exp(-y_i f_m(x_i))$$

$$= \exp(-y_i[f_{m-1}(x) + \beta_m G_m(x)])$$

$$= \exp(-y_i f_{m-1}(x)) \cdot \exp(-y_i \beta_m G_m(x_i))$$

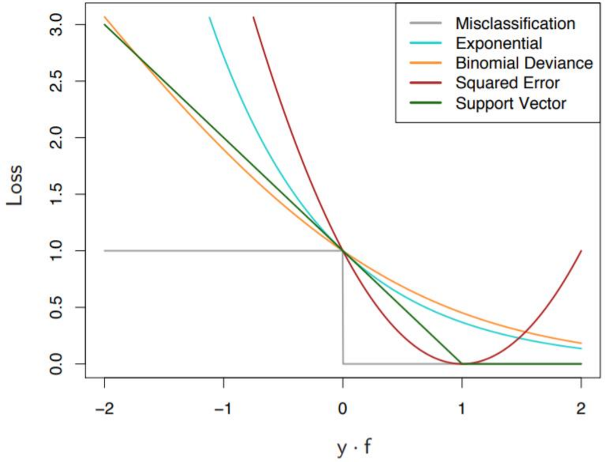
$$= w_i^{(m)} \exp(-\beta_m y_i G_m(x_i)) \quad \rightarrow w_i^{(m+1)} = \begin{cases} e^{-as_m} & \text{if } y_i = G_m(x_i) \\ e^{as_m} & \text{if } y_i \neq G_m(x_i) \end{cases}$$

Exponential Loss and AdaBoost

$$
\begin{aligned}
w_i^{(m+1)} &= w_i^{(m)} \exp(-\beta_m y_i G_m(x_i)) \\
&= w_i^{(m)} \exp(\beta_m (2I(y_i \neq G_m(x_i)) - 1)) \\
&= w_i^{(m)} \exp(2\beta_m \cdot I(y_i \neq G_m(x_i)) - \beta_m) \\
&= w_i^{(m)} \exp(2\beta_m \cdot I(y_i \neq G_m(x_i))) \cdot \exp(-\beta_m) \\
&= w_i^{(m)} \exp(\alpha_m \cdot I(y_i \neq G_m(x_i))) \cdot \exp(-\beta_m) \qquad \rightarrow \quad \text{slide 42}
\end{aligned}
$$

Exponential Loss and AdaBoost

- AdaBoost is a special case of Forward Stagewise Additive Modeling (=Boosting) when we use Exponential Loss!



Ensemble Learning (Gradient Boosting Method)

- What if we want to use another loss (ex: absolute error, cross entropy, hinge, …) or another model $b(x; \gamma_m)$ instead of tree stump?

$$G_m = \underset{G}{argmin} \sum_{i=1}^{n} w_i^{(m)} I(y_i \neq G(x_i))$$ → tree stump using Impurity

→ There exists no general simple fast algorithms for these more general loss criteria.

Ensemble Learning (Gradient Boosting Method)

- Instead of tweaking the instance weights at every iteration like AdaBoost does, Gradient Boosting method tries to fit the new predictor to the residual errors made by the previous predictor.

Ensemble Learning (Gradient Boosting Method)

- Gradient Boosting은 임의의 differentiable loss function에 대해 Forward Stagewise Additive Model의 최적화 문제를 근사적으로 해결하는 알고리즘이다.

$$\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i))$$

$$f_m(\boldsymbol{x}_i) = f_{m-1}(\boldsymbol{x}_i) - \eta_m \frac{\partial L(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)} \Big|_{f(\boldsymbol{x}_i = f_{m-1}(\boldsymbol{x}_i))}$$

$$= f_{m-1}(\underline{x_i}) - \eta_m g_{im}$$

Ensemble Learning (Gradient Boosting Method)

- Gradient Boosting은 임의의 differentiable loss function에 대해 Forward Stagewise Additive Model의 최적화 문제를 근사적으로 해결하는 알고리즘이다.

$$\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i))$$

$$\sum_{i=1}^{n} L(y_i, f_{m-1}(\underline{x_i}) - \eta_m g_{im})$$

$$\beta_m = \eta_m, \quad b(\boldsymbol{x}_i; \boldsymbol{\gamma}_m) = g_{im}$$

$$\sum_{i=1}^{n} L[y_i, f_{m-1}(\boldsymbol{x}_i) + \beta b(\boldsymbol{x}_i; \boldsymbol{\gamma})]$$

Ensemble Learning (Gradient Boosting Method)

이제 boosted tree model의 각 step에서의 우리의 최적화 문제는 아래와 같다.

$$\tilde{\Theta}_m = \arg\min_{\Theta_m} \|-\mathbf{g}_m - \mathbf{t}_m\|_2^2$$

$$= \arg\min_{\Theta_m} \sum_{i=1}^{N} (-g_{im} - T(x_i; \Theta_m))^2$$

$$\text{where} \quad g_{im} = i\text{th coordinate of } \mathbf{g}_m = \left[\frac{\partial L(\mathbf{f})}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}$$

$$= \left[\frac{\partial}{\partial f(x_i)} \sum_{k=1}^{N} L(y_k, f(x_k))\right]_{f(x_i)=f_{m-1}(x_i)} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}$$

$$\mathbf{t}_m = \begin{bmatrix} T(x_1; \Theta_m) \\ \vdots \\ T(x_N; \Theta_m) \end{bmatrix} = \text{a vector of predicted values at training points}$$

...

Ensemble Learning (Gradient Boosting Method)

- For regression..

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$
   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

Ensemble Learning (Gradient Boosting Method)

- For classification..

Algorithm 10.4 *Gradient Boosting for K-class Classification.*

1. Initialize $f_{k0}(x) = 0,\ k = 1, 2, \ldots, K$.
2. For m=1 to M:
   (a) Set
   $$p_k(x) = \frac{e^{f_k(x)}}{\sum_{\ell=1}^{K} e^{f_\ell(x)}},\ k = 1, 2, \ldots, K.$$
   (b) For $k = 1$ to K:
      i. Compute $r_{ikm} = y_{ik} - p_k(x_i),\ i = 1, 2, \ldots, N$.
      ii. Fit a regression tree to the targets $r_{ikm},\ i = 1, 2, \ldots, N$, giving terminal regions $R_{jkm},\ j = 1, 2, \ldots, J_m$.
      iii. Compute
      $$\gamma_{jkm} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{jkm}} r_{ikm}}{\sum_{x_i \in R_{jkm}} |r_{ikm}|(1 - |r_{ikm}|)},\ j = 1, 2, \ldots, J_m.$$
      iv. Update $f_{km}(x) = f_{k,m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jkm} I(x \in R_{jkm})$.
3. Output $\hat{f}_k(x) = f_{kM}(x),\ k = 1, 2, \ldots, K$.

reference

자료

19-2 STAT424 통계적 머신러닝 - 박유성 교수님

교재

파이썬을 이용한 통계적 머신러닝 (2020) – 박유성

ISLR (2013) - G. James, D. Witten, T. Hastie, R. Tibshirani

The elements of Statistical Learning (2001) - J. Friedman, T. Hastie, R. Tibshirani

Hands on Machine Learning (2017) - Aurelien Geron