Regression metrics optimization

Metrics optimization: our plan

1) **Regression**
   - MSE, (R)MSE, R-squared
   - MAE
   - (R)MSPE, MAPE
   - (R)MSLE
2) **Classification:**
   - Accuracy
   - Logloss
   - AUC
   - Cohen's Kappa

RMSE, MSE, R-squared

Regression계열에서 쉽게 사용

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

How do you optimize them?

Just fit the right model!

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$$

$$R^2 = 1 - \frac{MSE}{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

RMSE, MSE, R-squared

- **Tree-based**

  `XGBoost, LightGBM`

  `sklearn.RandomForestRegressor`

- **Linear models**

  `sklearn.<>Regression`

  `sklearn.SGDRegressor`

  `Vowpal Wabbit` *(quantile loss)*

- **Neural nets**

  `PyTorch, Keras, TF,` etc.

Synonyms: L2 loss

Read the docs!

MAE

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

How do you optimize it?

Again, just run the right model!

MAE

- **Tree-based**

  `~~XGBoost~~, LightGBM`

  MAE is not twice differentiable, not differentiable when the predictions are equal to target

  `sklearn.RandomForestRegressor`

  MSE보다 느림

- **Linear models**

  `~~sklearn.<>Regression~~`

  `~~sklearn.SGDRegressor~~`

  대신 Huber Loss는 제공 (error가 크면 MAE와 비슷)

  `Vowpal Wabbit (quantile loss)`

  quantile loss라는 이름으로 제공 (50% quentile(median) -> MAE)

- **Neural nets**

  `PyTorch, Keras, TF, etc.`

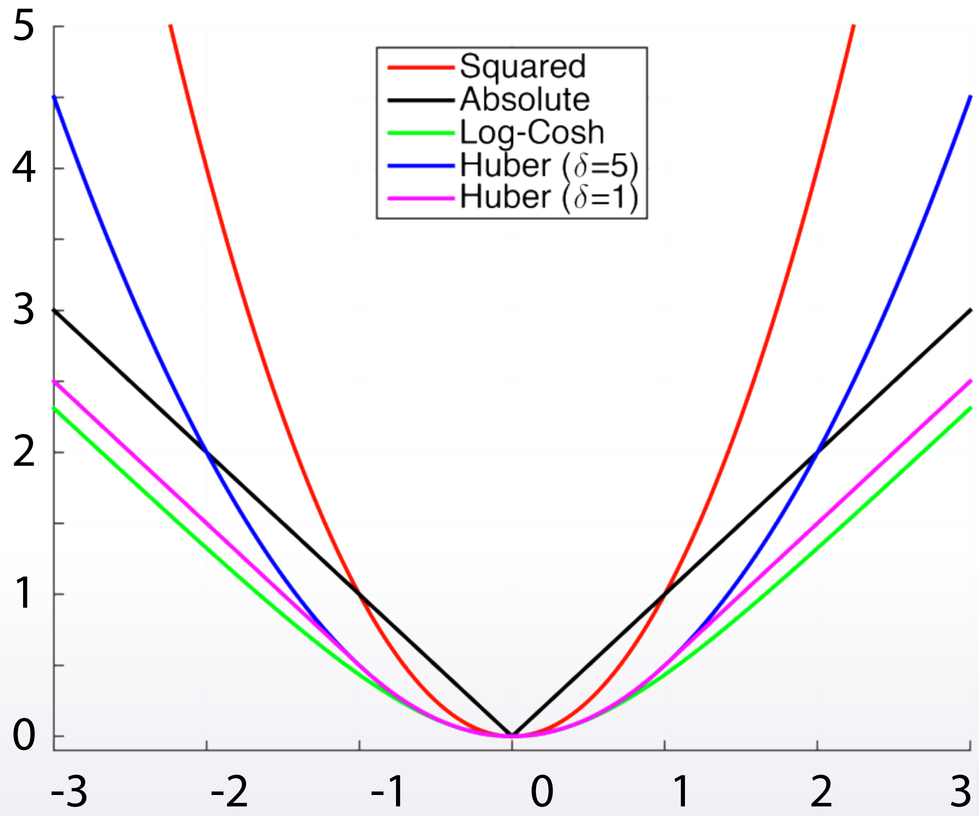
Synonyms: L1, Median regression

Read the docs!

MAE: optimal constant

Huber losss, the most famous way to make MAE smooth
= MSE(for small error, approach zero error) + MAE(for large error, robustness)
https://en.wikipedia.org/wiki/Huber_loss 참고

$$\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$



MSPE and MAPE

$$\text{MSPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2 \qquad \text{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

How do you optimize them?

~~Just run the right model!~~

custom loss, early stopping, sampling 후 MSE를 최적화

MSPE (MAPE) as weighted MSE (MAE)

Sample weights

(분모는 합계가 1이 되도록 가중치 생성한 것, 중요X)

$$\mathrm{MSPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2$$	$$w_i = \frac{1/y_i^2}{\sum_{i=1}^{N} 1/y_i^2}$$
$$\mathrm{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left\|\frac{y_i - \hat{y}_i}{y_i}\right\|$$	$$w_i = \frac{1/y_i}{\sum_{i=1}^{N} 1/y_i}$$

MSPE (MAPE)

- **Use weights for samples (`sample_weights`)**
  - And use MSE (MAE)
  - *Not every library accepts sample weights*
    - `XGBoost, LightGBM` accept
    - Neural nets
      - Easy to implement if not supported
- **Resample the train set**
  - `df.sample(weights=sample_weights)`
  - And use *any* model that optimizes MSE (MAE)
  - Usually need to resample many times and average

RMSLE (Root mean square logarithmic error), MSE와 관련되어 있어서 최적화하기 쉬움

$$\mathrm{RMSLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log(y_i+1) - \log(\hat{y}_i+1)\right)^2} =$$

$$= \sqrt{MSE\left(\log(y_i+1), \log(\hat{y}_i+1)\right)}$$

Train:

1. Transform target:

$$z_i = \log(y_i + 1)$$

log가 0에서 정의되지 않으므로 작은 수를 더함

2. Fit a model with MSE loss

Test:

- Transform predictions back:

$$\hat{y}_i = \exp(\hat{z}_i) - 1$$

Conclusion

1) **Regression**
   - MSE, (R)MSE, R-squared
   - MAE
   - (R)MSPE, MAPE
   - (R)MSLE
2) **Classification:**
   - Accuracy
   - Logloss
   - AUC
   - Cohen's Kappa