Metrics optimization

Lesson overview

In this video:

- **Metrics:**
  - Why there are so many
  - Why should we care about them in competitions

Lesson overview

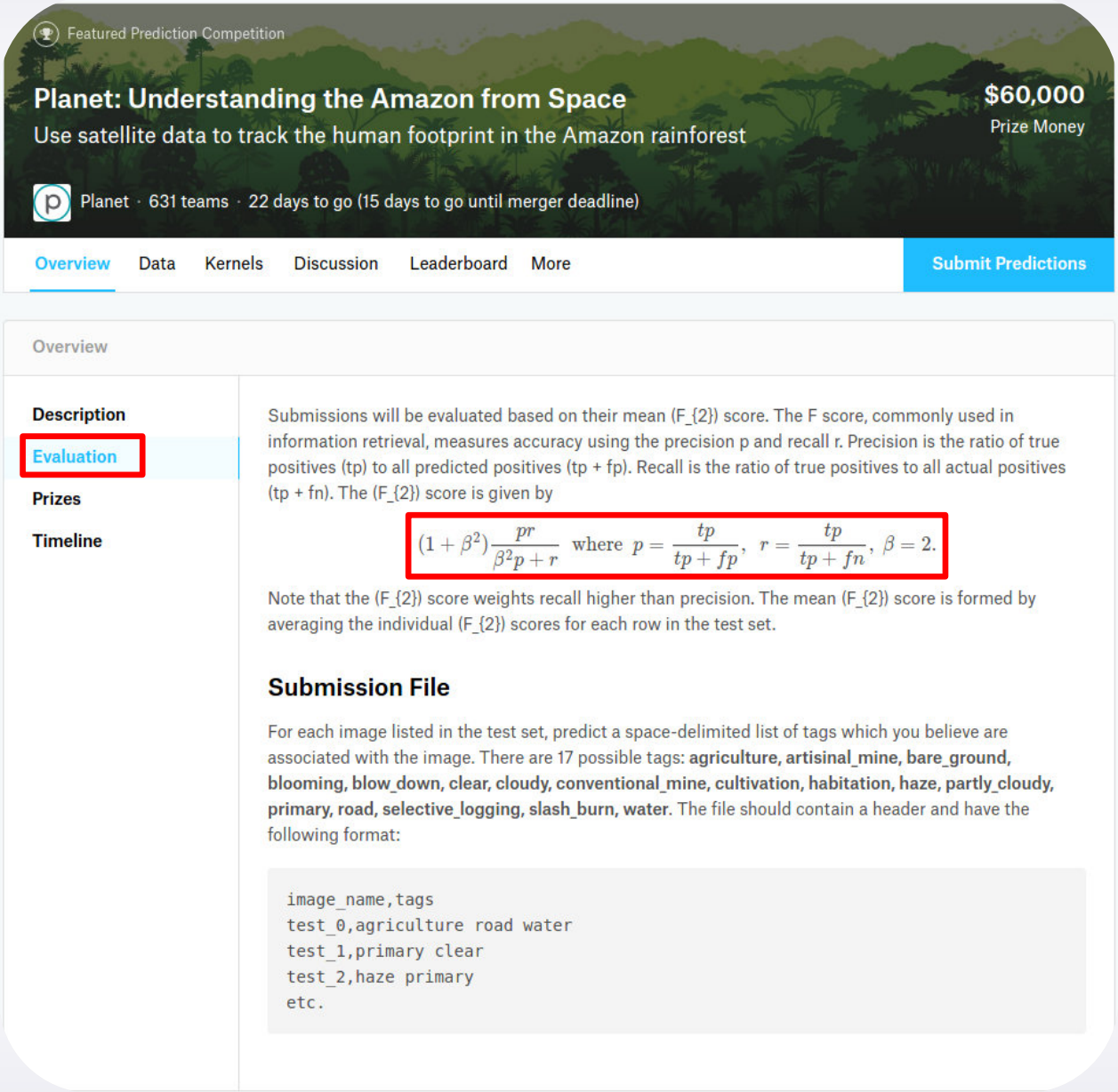
In this video:

- **Metrics:**
  - Why there are so many
  - Why should we care about them in competitions

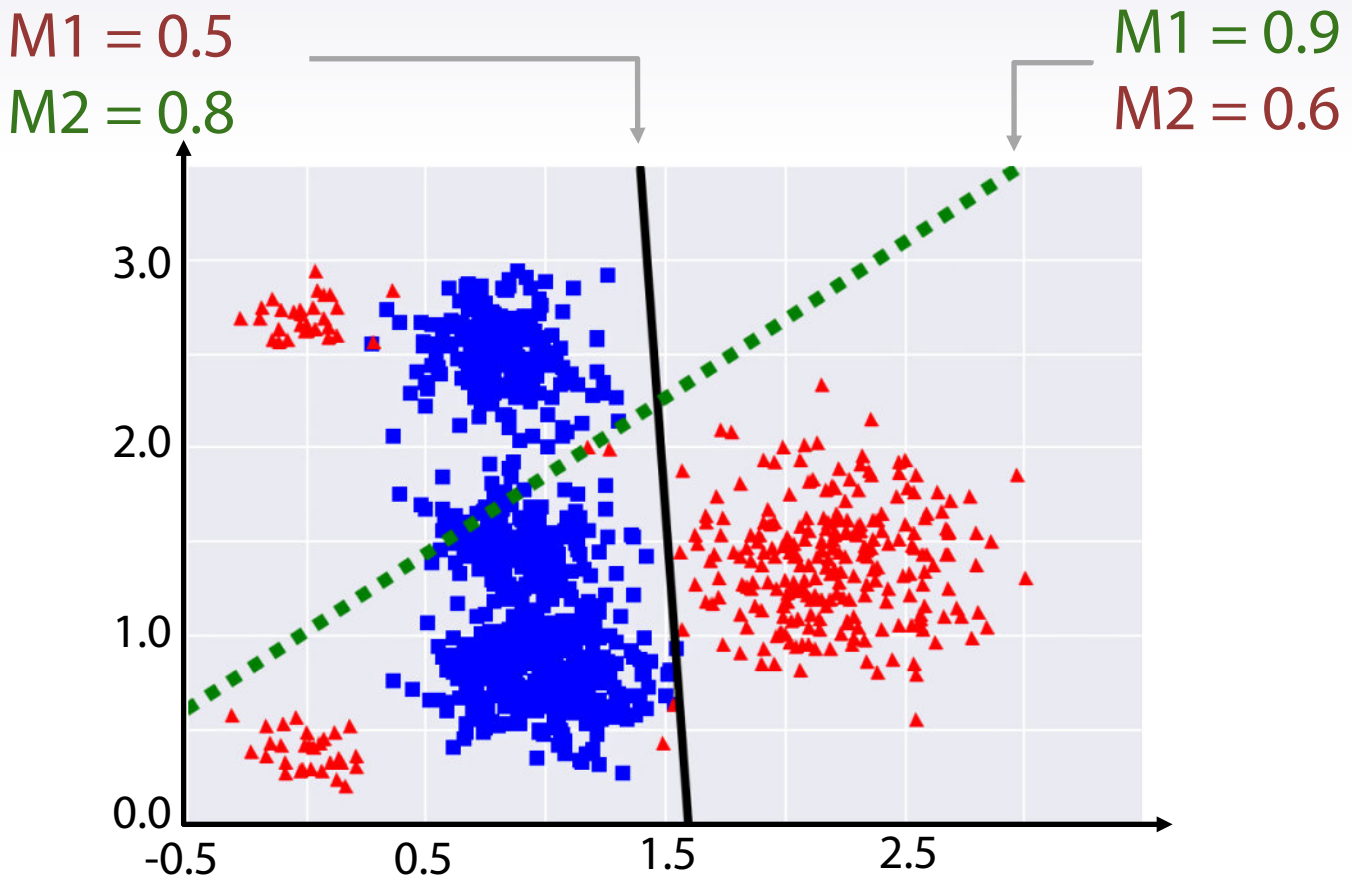
In the following videos:

- **Loss versus metric**
- **Review the most important metrics**
  - For classification and regression tasks
  - Discuss baseline solutions for their optimization
- **Optimization techniques for the metrics**

Metrics


Featured Prediction Competition
Planet: Understanding the Amazon from Space
Use satellite data to track the human footprint in the Amazon rainforest
$60,000
Prize Money
Planet · 631 teams · 22 days to go (15 days to go until merger deadline)
Overview
Data
Kernels
Discussion
Leaderboard
More
Submit Predictions
Overview
Description
Evaluation
Prizes
Timeline
Submissions will be evaluated based on their mean (F_{2}) score. The F score, commonly used in information retrieval, measures accuracy using the precision p and recall r. Precision is the ratio of true positives (tp) to all predicted positives (tp + fp). Recall is the ratio of true positives to all actual positives (tp + fn). The (F_{2}) score is given by
$$(1+\beta^2)\frac{pr}{\beta^2 p + r} \text{ where } p = \frac{tp}{tp+fp}, \ r = \frac{tp}{tp+fn}, \ \beta = 2.$$
Note that the (F_{2}) score weights recall higher than precision. The mean (F_{2}) score is formed by averaging the individual (F_{2}) scores for each row in the test set.
Submission File
For each image listed in the test set, predict a space-delimited list of tags which you believe are associated with the image. There are 17 possible tags: agriculture, artisinal_mine, bare_ground, blooming, blow_down, clear, cloudy, conventional_mine, cultivation, habitation, haze, partly_cloudy, primary, road, selective_logging, slash_burn, water. The file should contain a header and have the following format:
image_name,tags
test_0,agriculture road water
test_1,primary clear
test_2,haze primary
etc.

Motivation



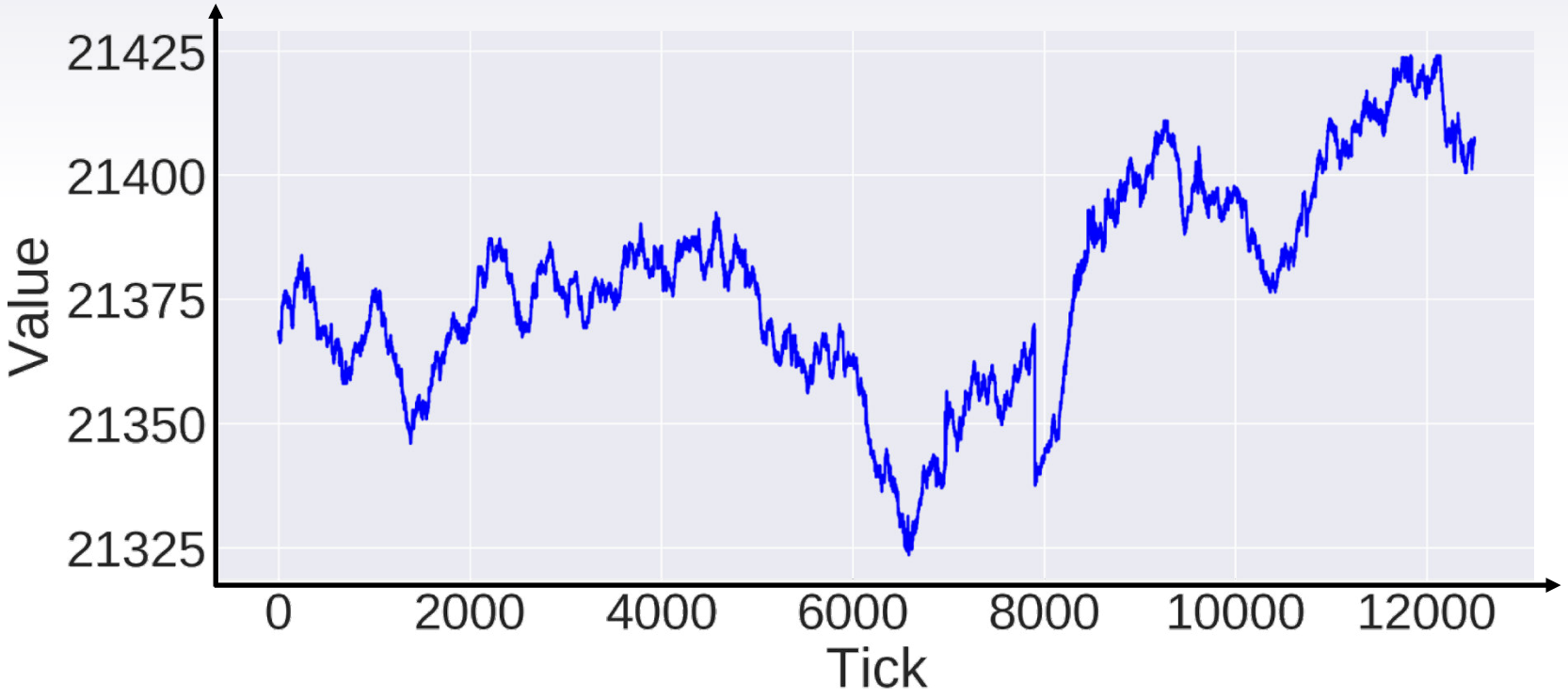
Chosen metric determines optimal decision boundary

Image from "Scalable Learning of Non-Decomposable Objectives", Eban et al. AISTATS, 2017

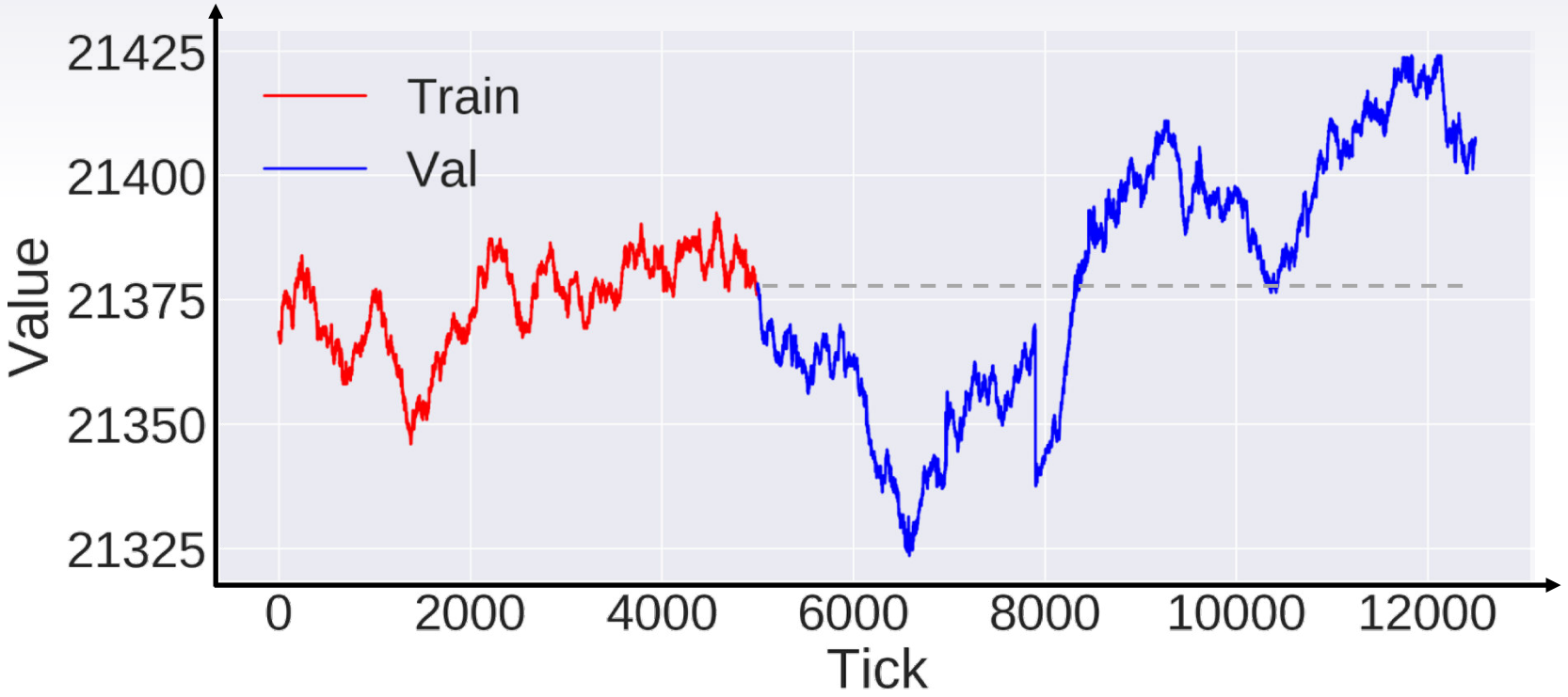
Take-away point

If your model is scored with some metric, you get best results by optimizing exactly that metric

Motivation 2

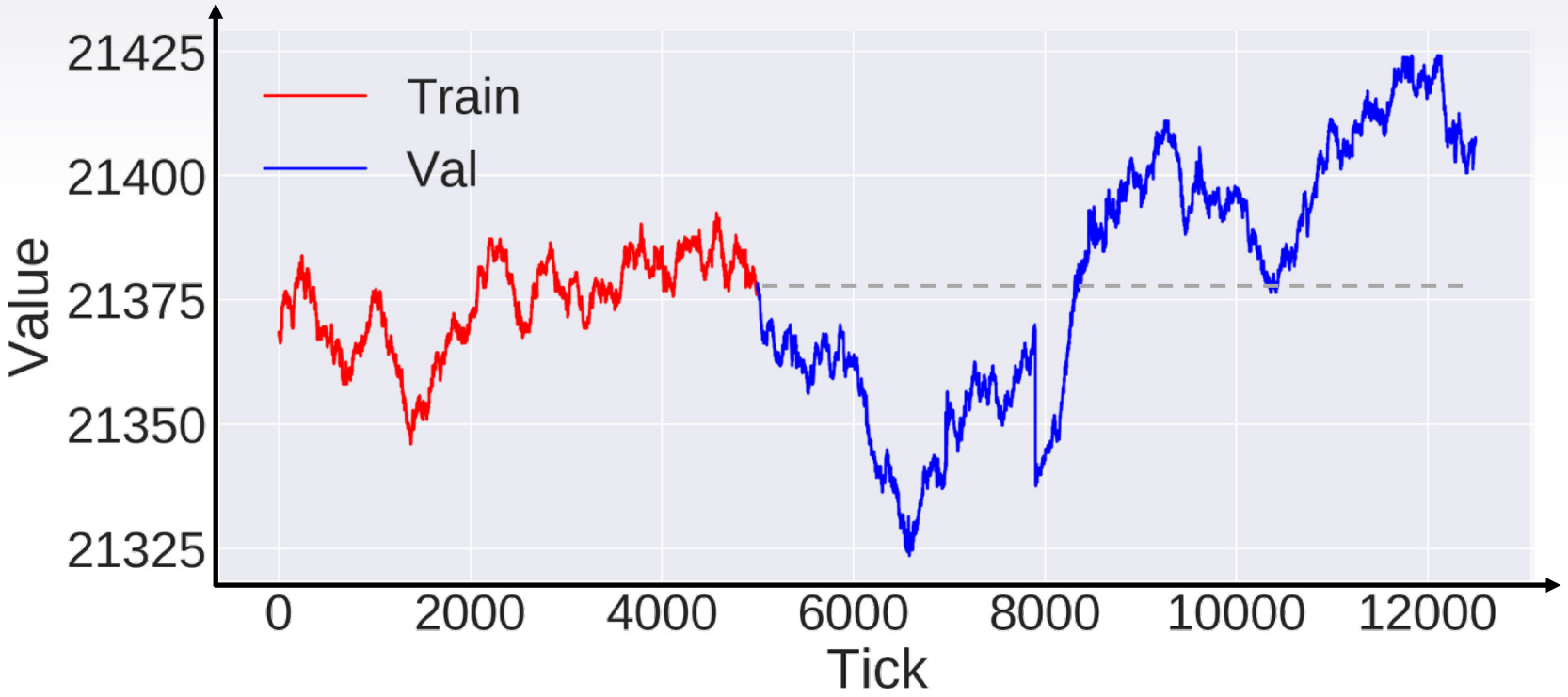


Motivation 2



$$Loss(\hat{y}_i; y_i) = \begin{cases} |y_i - \hat{y}_i|, & \text{if trend predicted correctly} \\ (y_i - \hat{y}_i)^2, & \text{if trend predicted incorrectly} \end{cases}$$

Motivation 2



$$Loss(\hat{y}_i; y_i) = \begin{cases} |y_i - \hat{y}_i|, & \text{if trend predicted correctly} \\ (y_i - \hat{y}_i)^2, & \text{if trend predicted incorrectly} \end{cases}$$

Predict *trend* instead of the values:

Predict $y_{last} + 10^{-6}$

or $y_{last} - 10^{-6}$

Conclusion

- **Why there are so many metrics?**
  - Different metrics for different problems

- **Why should we care about metric in competitions?**
  - It is how the competitors are ranked!

Regression metrics:
(R)MSE, R-squared, MAE

Plan for the video

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

Plan for the video

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

Notation

- N – number of objects

- $y \in \mathbb{R}^N$ – target values

  $\hat{y} \in \mathbb{R}^N$ – predictions

- $\hat{y}_i \in \mathbb{R}$ – prediction for i-th object

  $y_i \in \mathbb{R}$ – target for i-th object

MSE: Mean Square Error

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

MSE: Mean Square Error

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

Data:

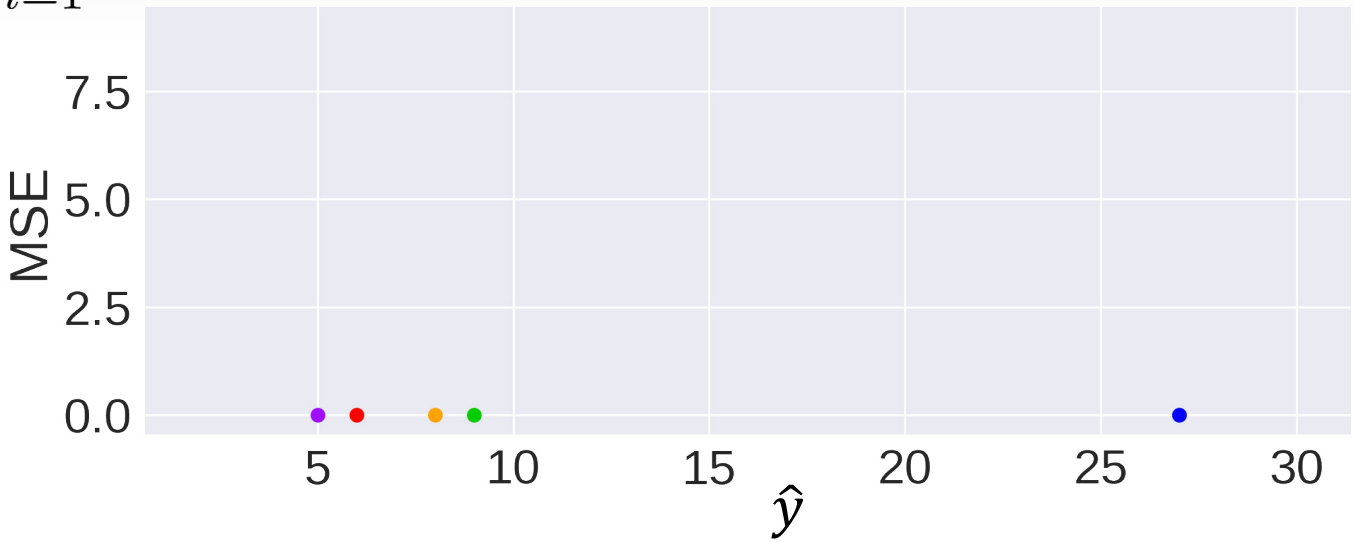
X	Y
...	5
...	9
...	8
...	6
...	27

MSE: Mean Square Error

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

Data:

X	Y
...	5
...	9
...	8
...	6
...	27

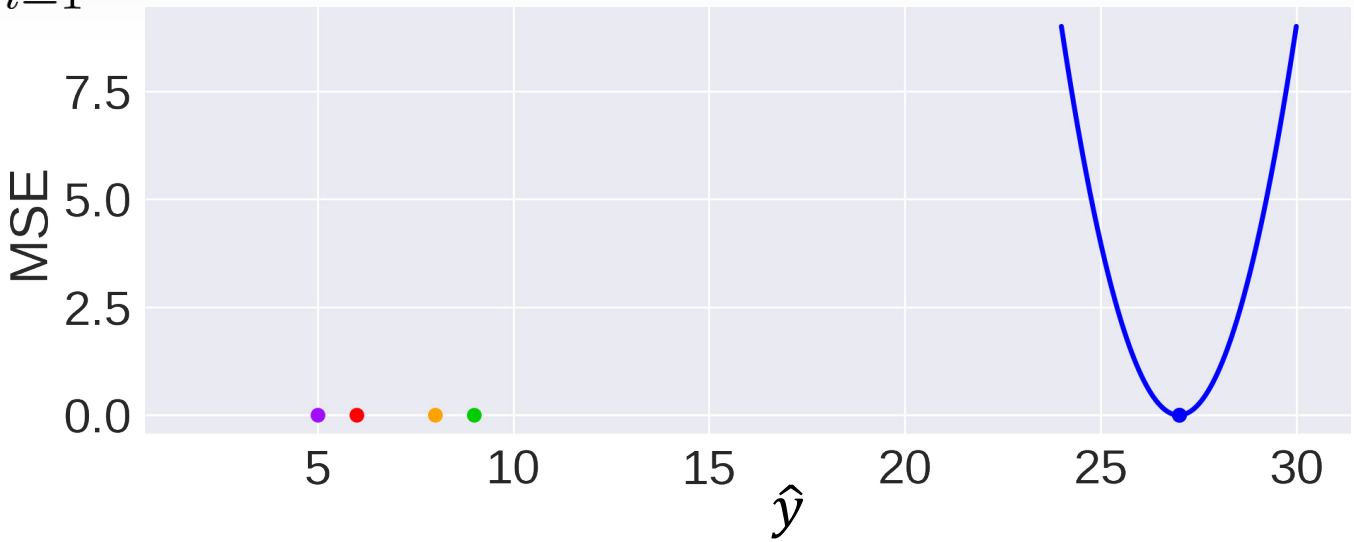


MSE: Mean Square Error

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

Data:

X	Y
...	5
...	9
...	8
...	6
...	27

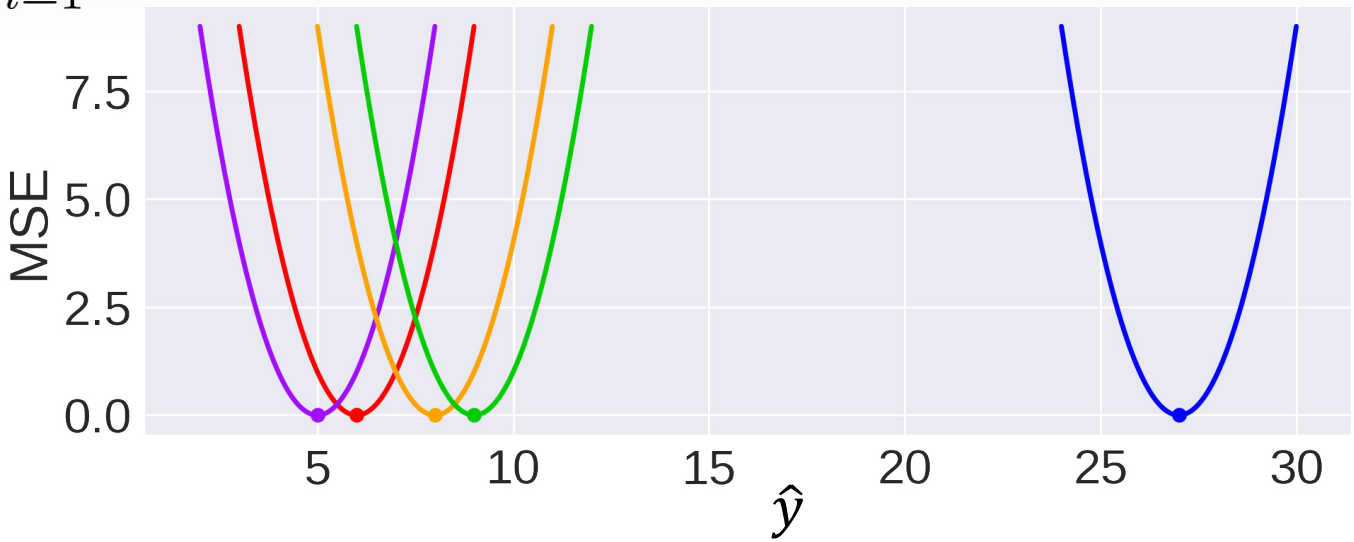


MSE: Mean Square Error

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

Data:

X	Y
...	5
...	9
...	8
...	6
...	27



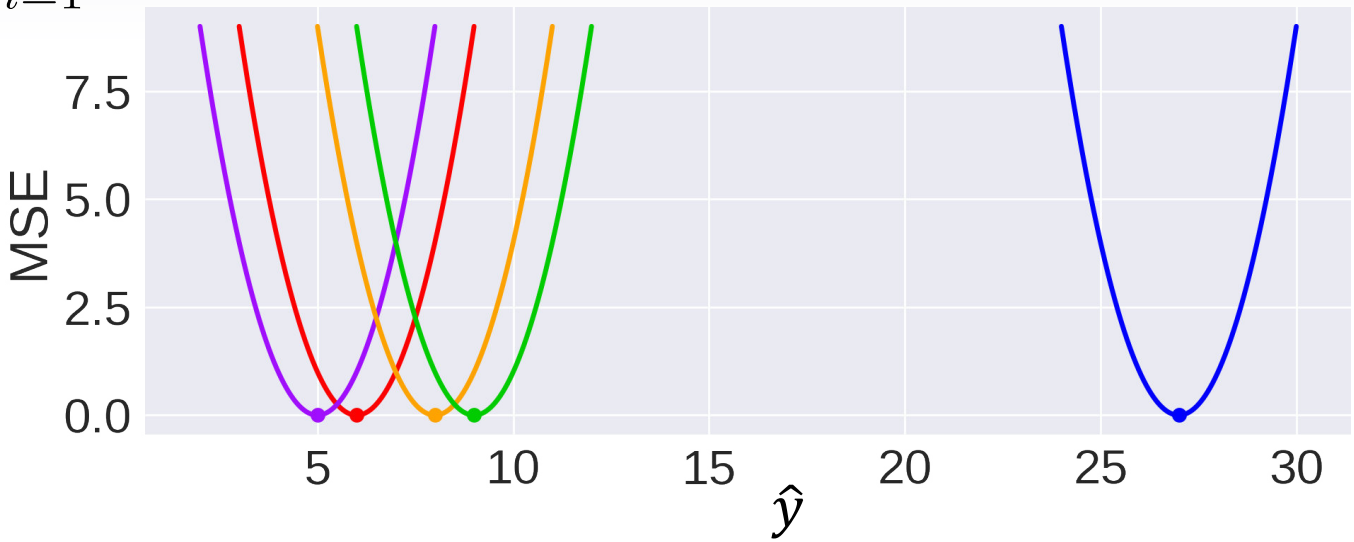
MSE: optimal constant

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \alpha)^2$$

Best constant: ?

Data:

X	Y
...	5
...	9
...	8
...	6
...	27



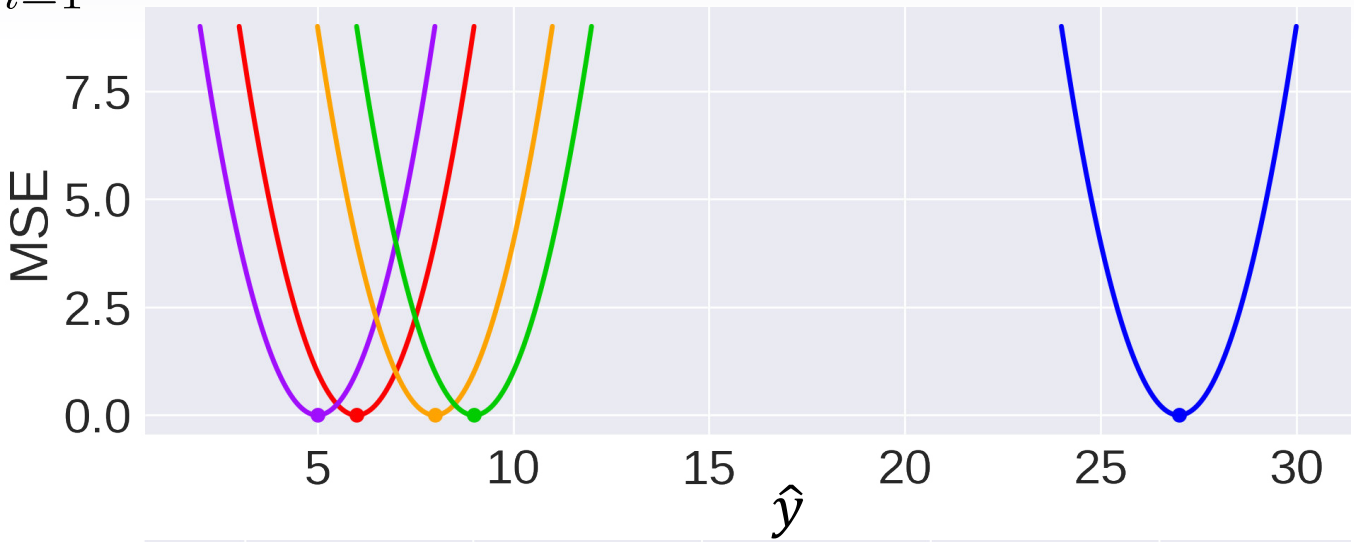
MSE: optimal constant

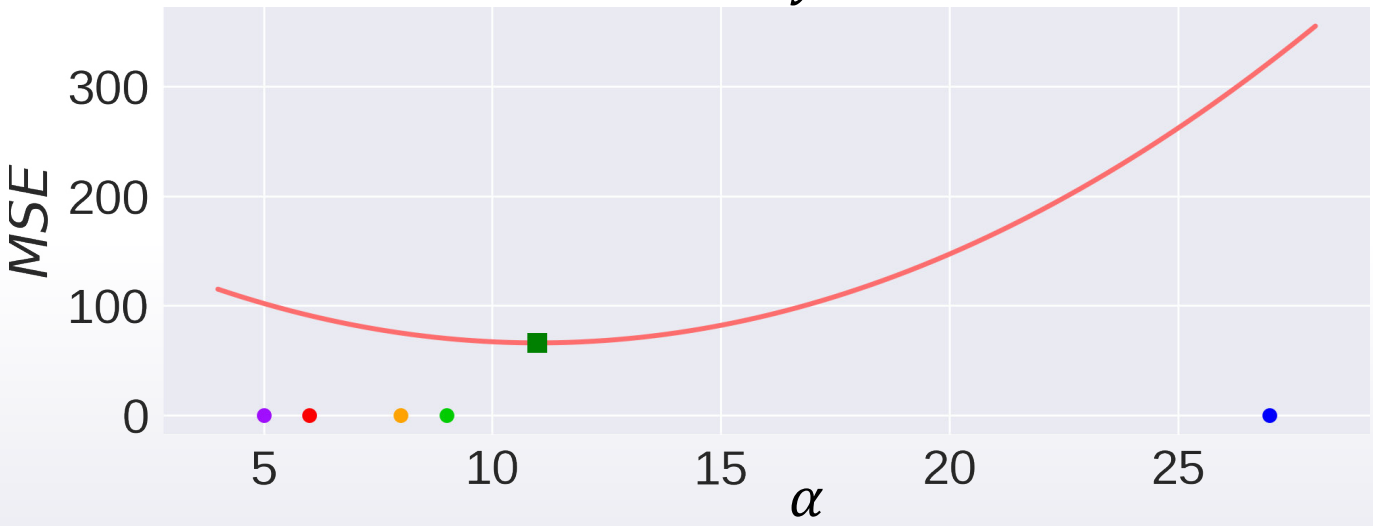
$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \alpha)^2$$

Best constant: target mean

Data:

X	Y
...	5
...	9
...	8
...	6
...	27





MSE notes: RMSE

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

MSE notes: RMSE

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

Root mean square error

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

MSE notes: RMSE

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

Root mean square error

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

- $$\text{MSE}(a) > \text{MSE}(b) \iff \text{RMSE}(a) > \text{RMSE}(b)$$

MSE notes: RMSE

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

Root mean square error

- $$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} = \sqrt{\mathrm{MSE}}$$
- $$\mathrm{MSE}(a) > \mathrm{MSE}(b) \iff \mathrm{RMSE}(a) > \mathrm{RMSE}(b)$$
- $$\frac{\partial \mathrm{RMSE}}{\partial \hat{y}_i} = \frac{1}{2\sqrt{MSE}}\frac{\partial \mathrm{MSE}}{\partial \hat{y}_i}$$

MSE notes: RMSE

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

MSE notes: RMSE

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

R-squared:

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

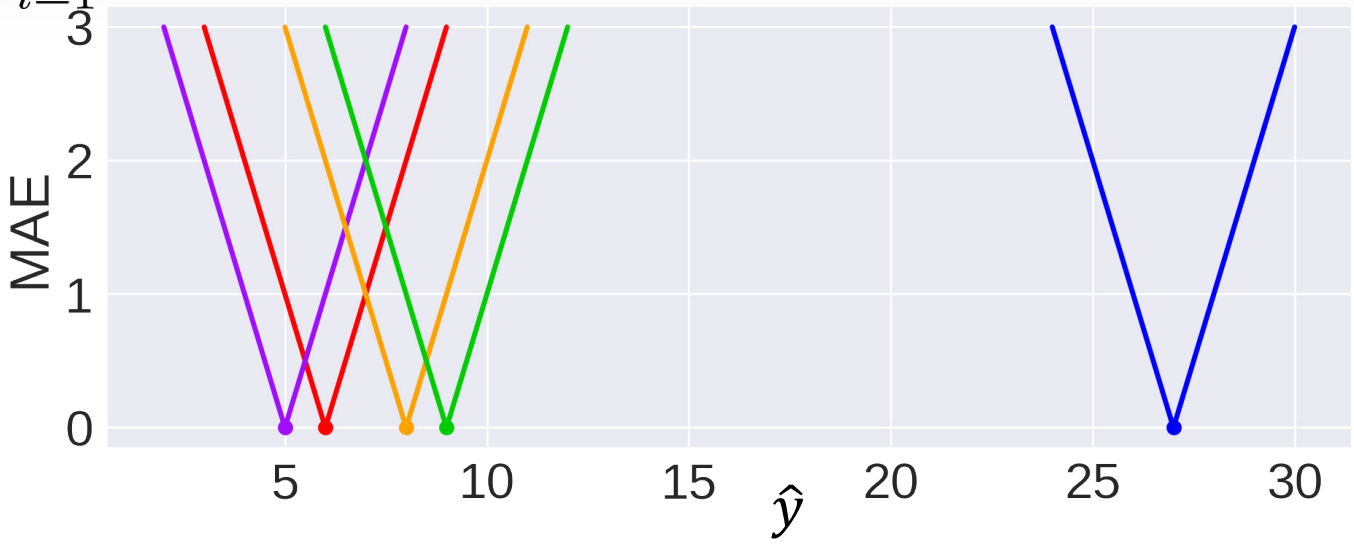
$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

MAE: Mean Absolute Error

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

Data:

X	Y
-1	5
1	9
-2	8
3	6
3	27



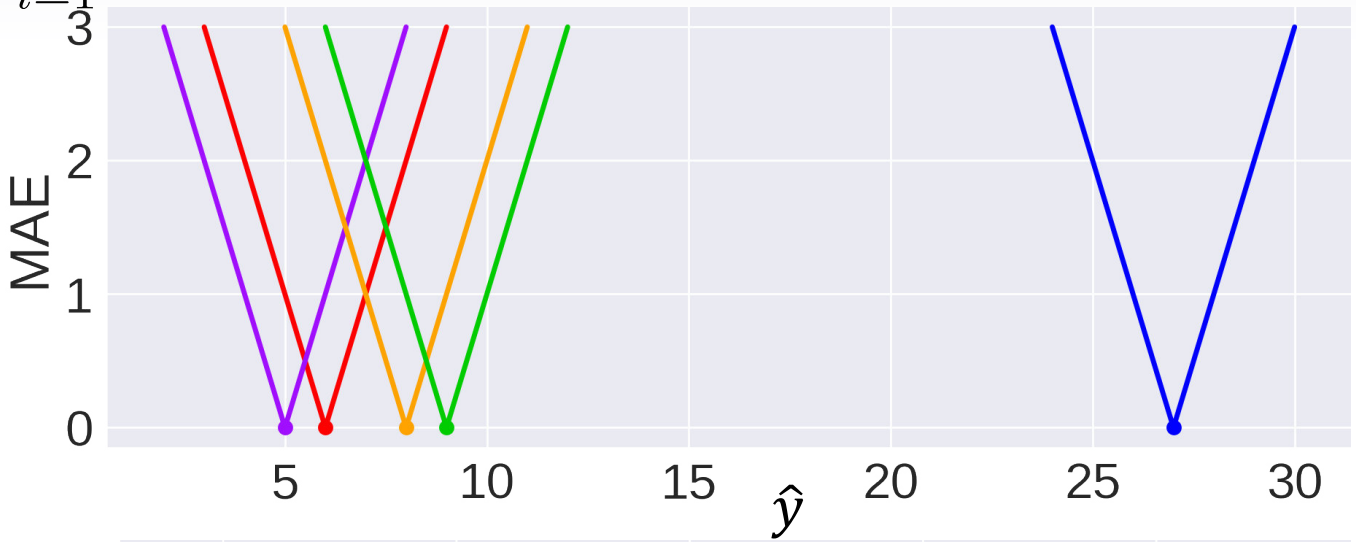
MAE: optimal constant

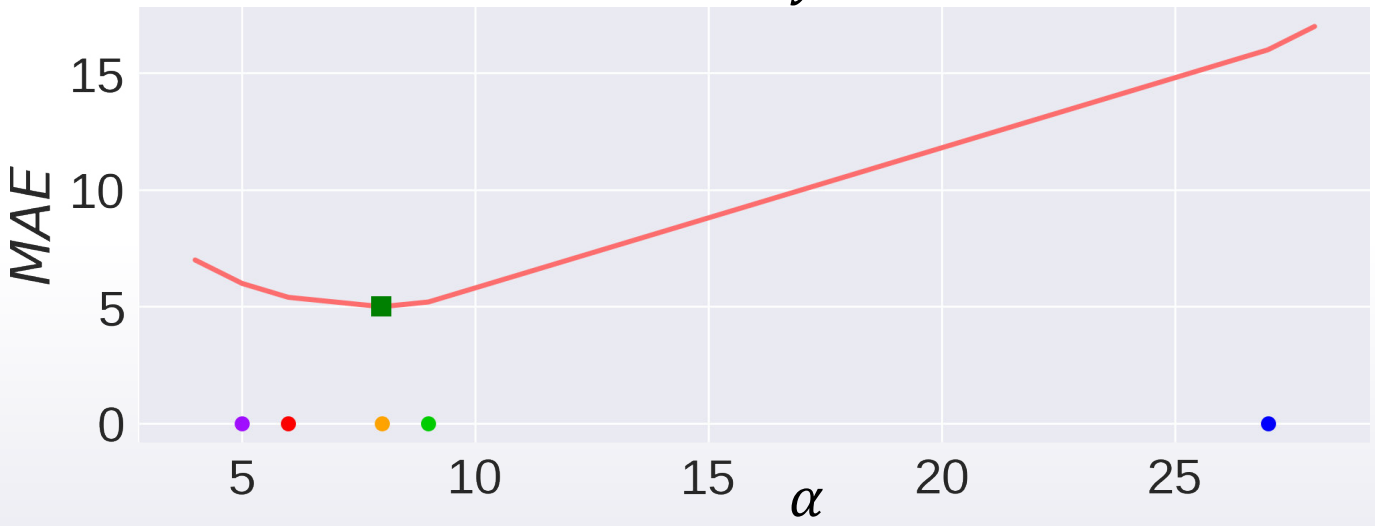
$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \alpha|$$

Best constant: target median

Data:

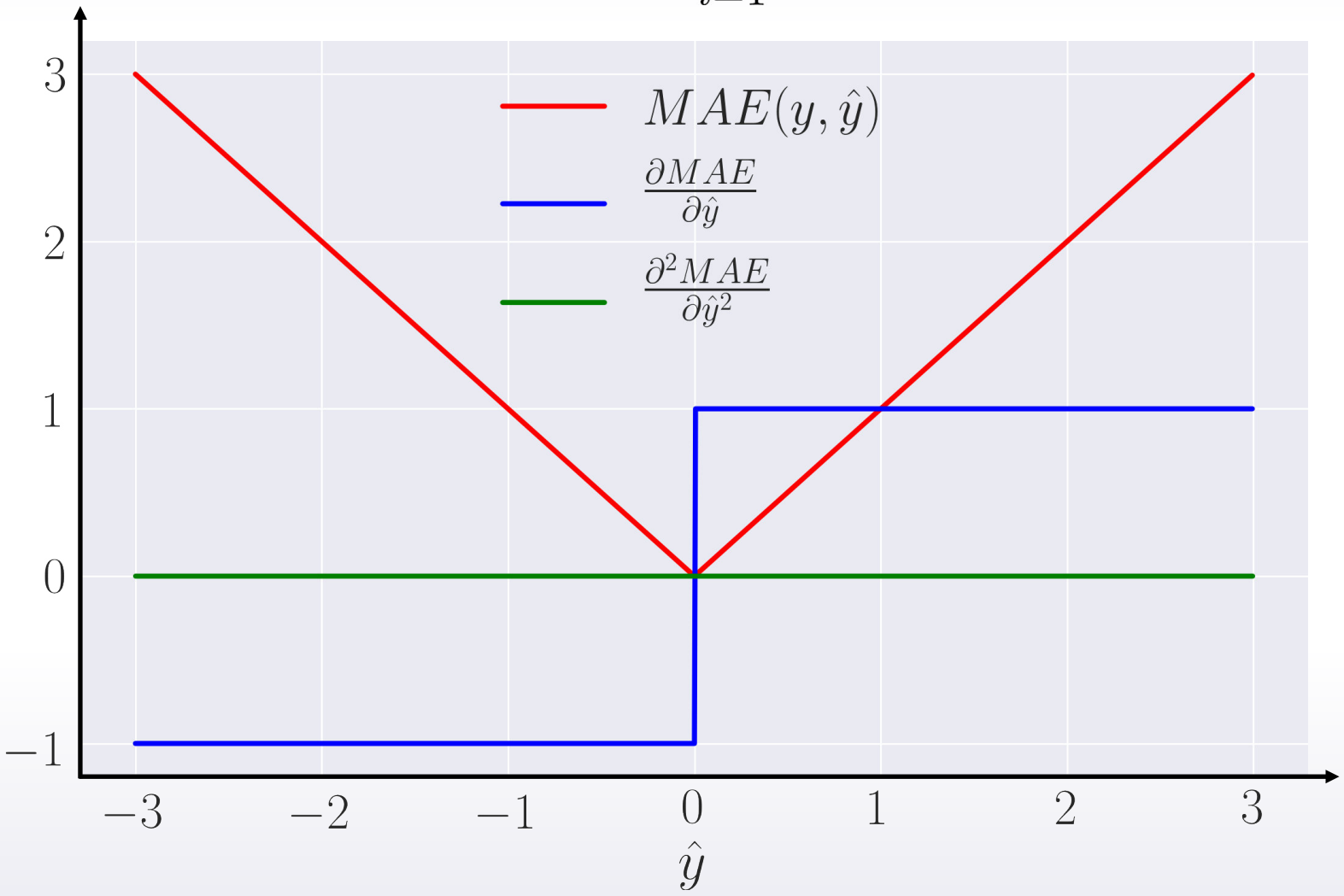
X	Y
-1	5
1	9
-2	8
3	6
3	27





MAE: derivatives

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



MAE vs MSE

- **Do you have outliers in the data?**
  - Use MAE
- **Are you sure they are outliers?**
  - Use MAE
- **Or they are just unexpected values we should still care about?**
  - Use MSE

Conclusion

- Discussed the following metrics:
  - **MSE, RMSE, R-squared**
    - They are the same from optimization perspective
  - **MAE**
    - Robust to outliers

Regression metrics: (R)MSPE, MAPE, (R)MSLE

Plan for the video

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

From MSE and MAE to MSPE and MAPE

- **Shop 1**: predicted 9, sold 10, MSE = 1
- **Shop 2**: predicted 999, sold 1000, MSE = 1

From MSE and MAE to MSPE and MAPE

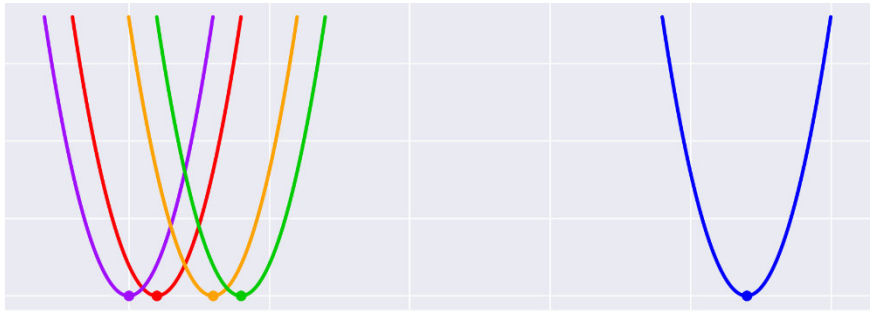
- **Shop 1**: predicted 9, sold 10, MSE = 1
- **Shop 2**: predicted 999, sold 1000, MSE = 1

- **Shop 1**: predicted 9, sold 10, MSE = 1
- **Shop 2**: predicted 900, sold 1000, MSE = 10000

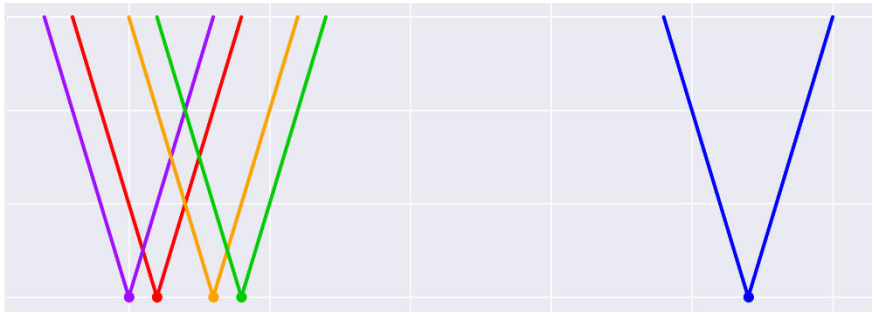
- **Shop 1**: predicted 9, sold 10, relative_metric = 1
- **Shop 2**: predicted 900, sold 1000, relative_metric = 1

From MSE and MAE to MSPE and MAPE

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

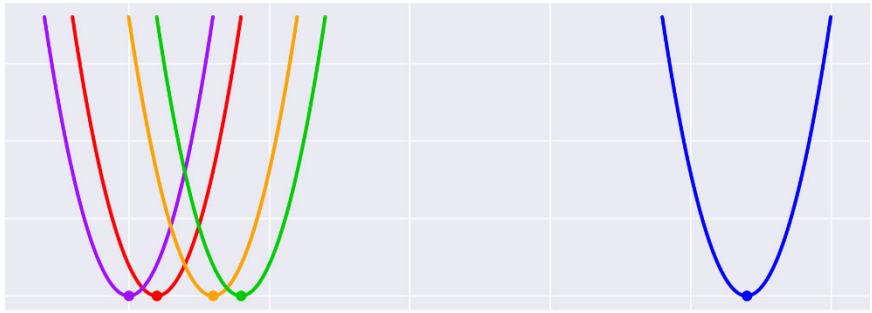


$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

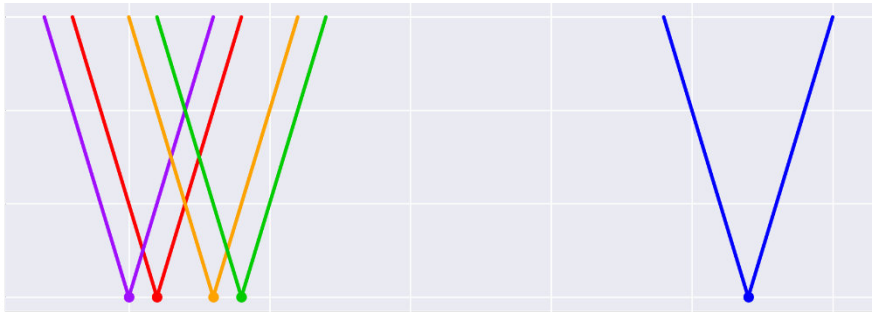


From MSE and MAE to MSPE and MAPE

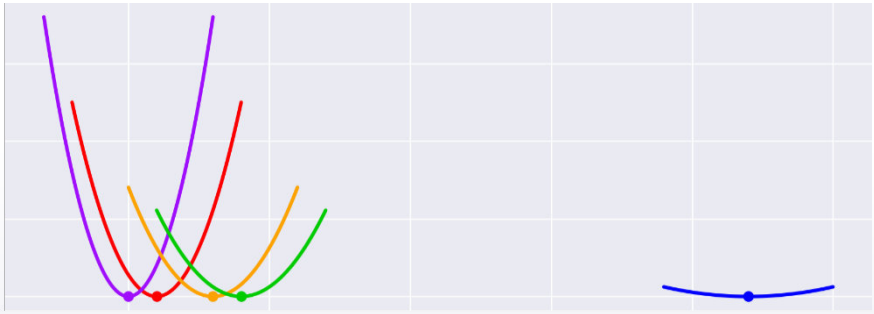
$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$



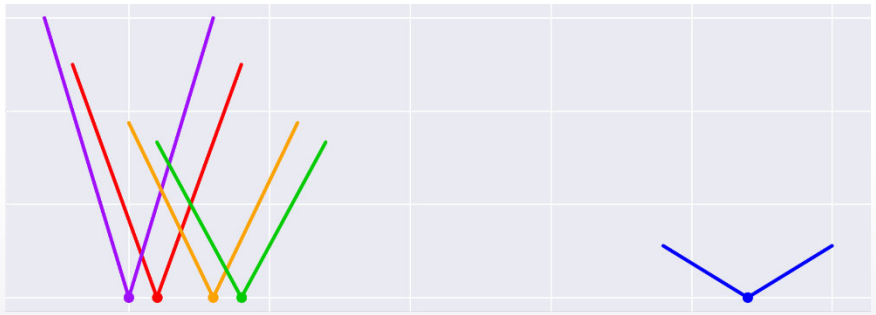
$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$



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



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



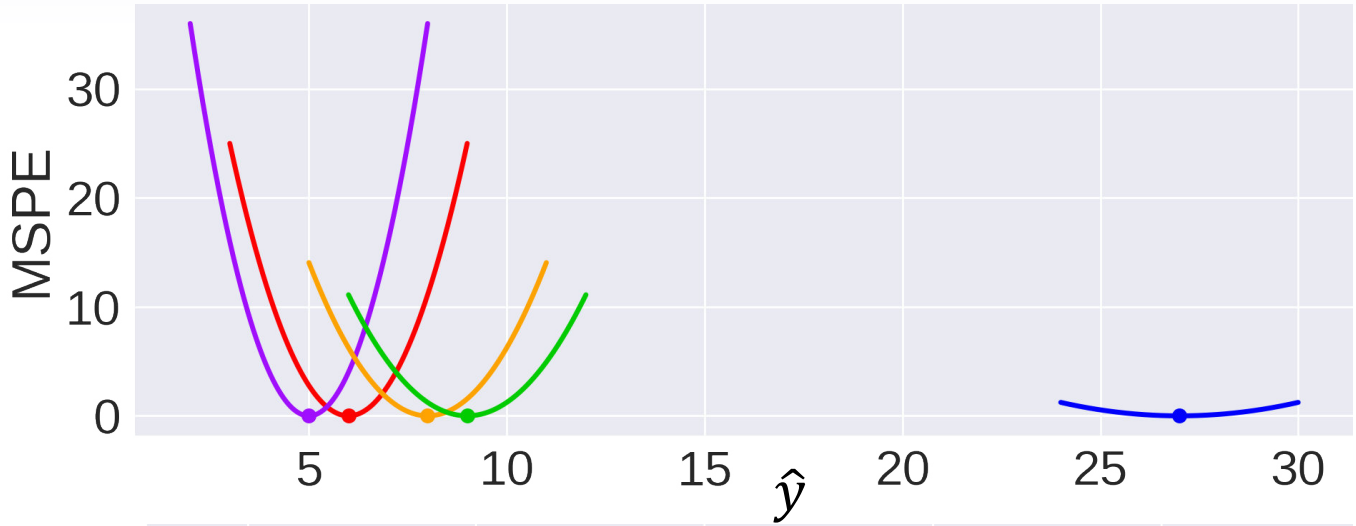
MSPE: constant

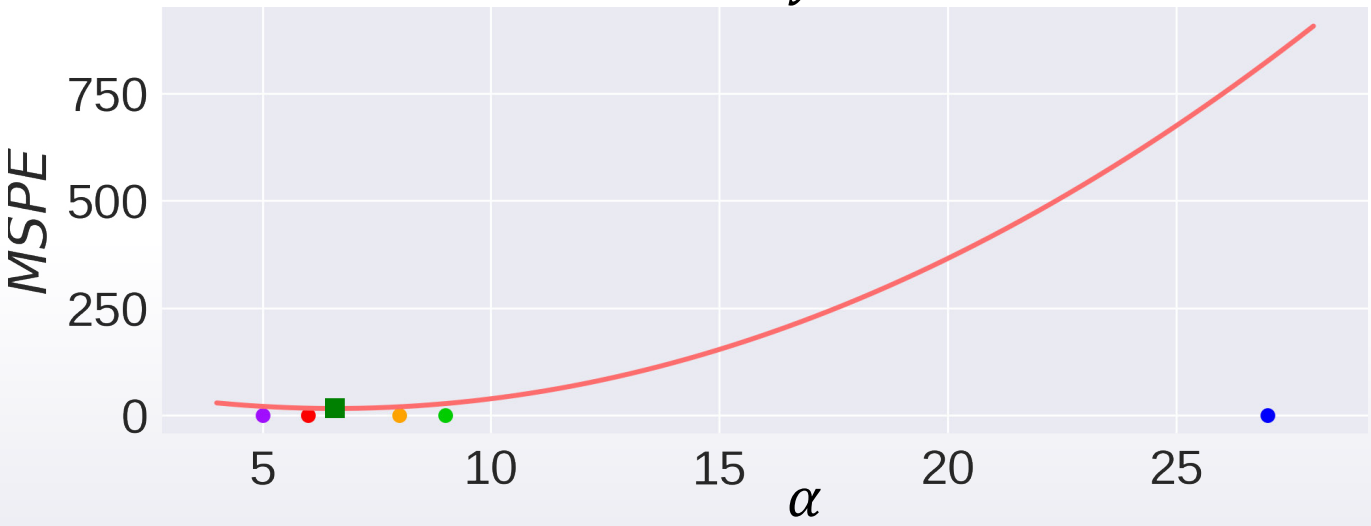
$$\text{MSPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left(\frac{y_i - \alpha}{y_i}\right)^2$$

Best constant:
weighted target mean

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25





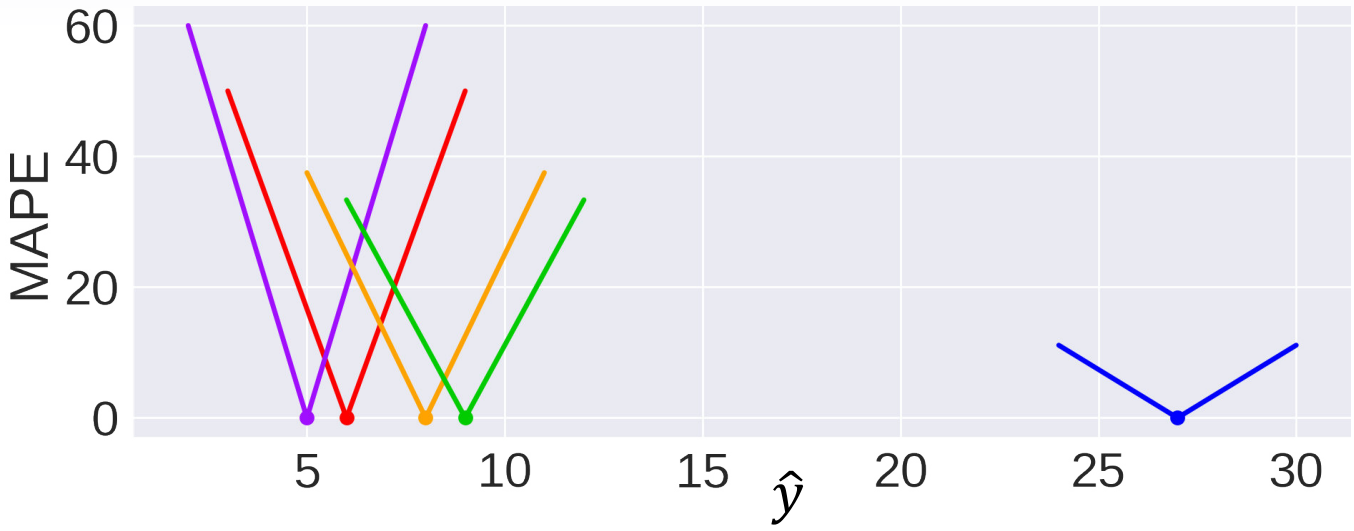
MAPE: constant

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^{N} \left| \frac{y_i - \alpha}{y_i} \right|$$

Best constant:
?

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25



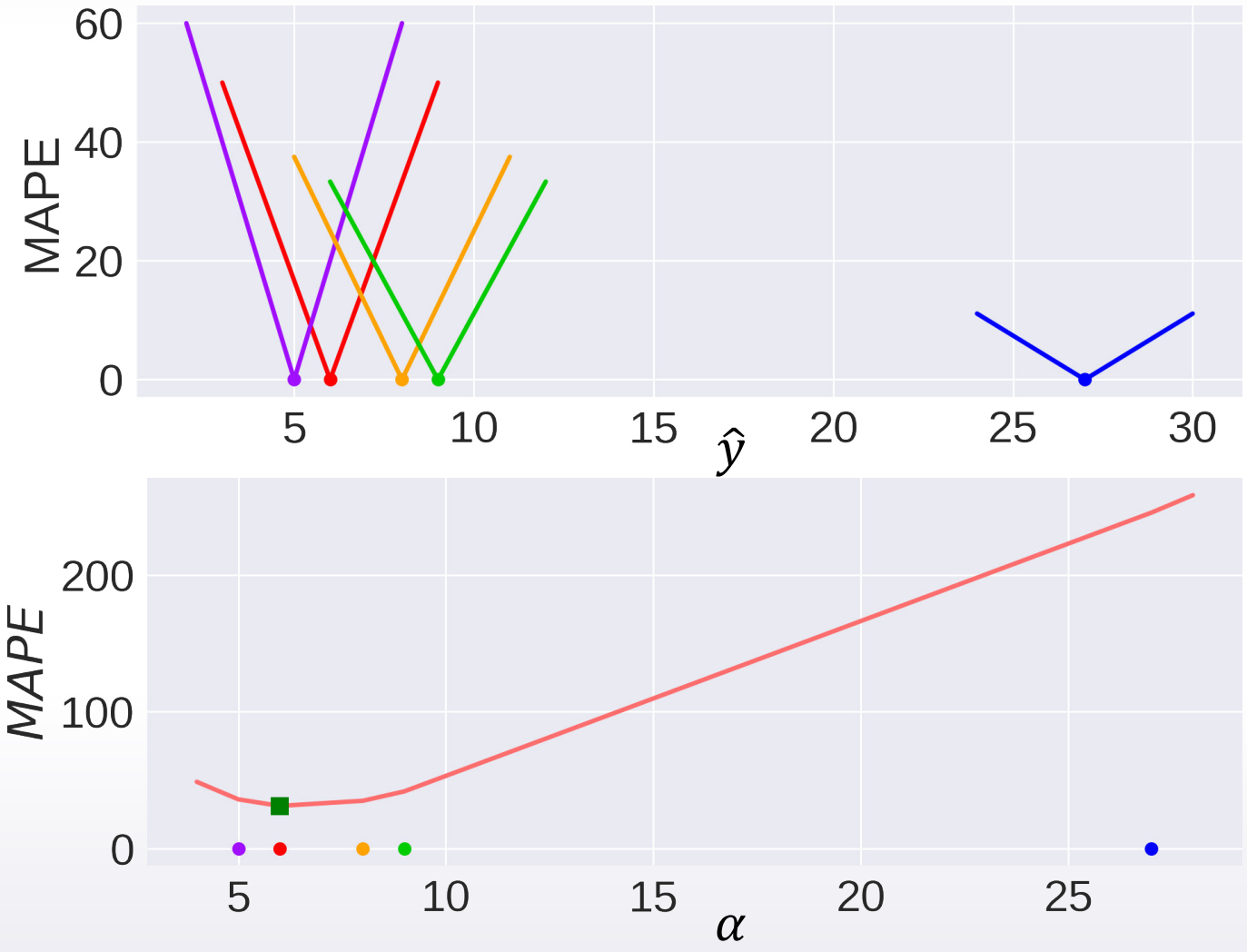
MAPE: constant

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^{N} \left| \frac{y_i - \alpha}{y_i} \right|$$

Best constant:
weighted target median

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25



(R)MSLE: Root Mean Square Logarithmic Error

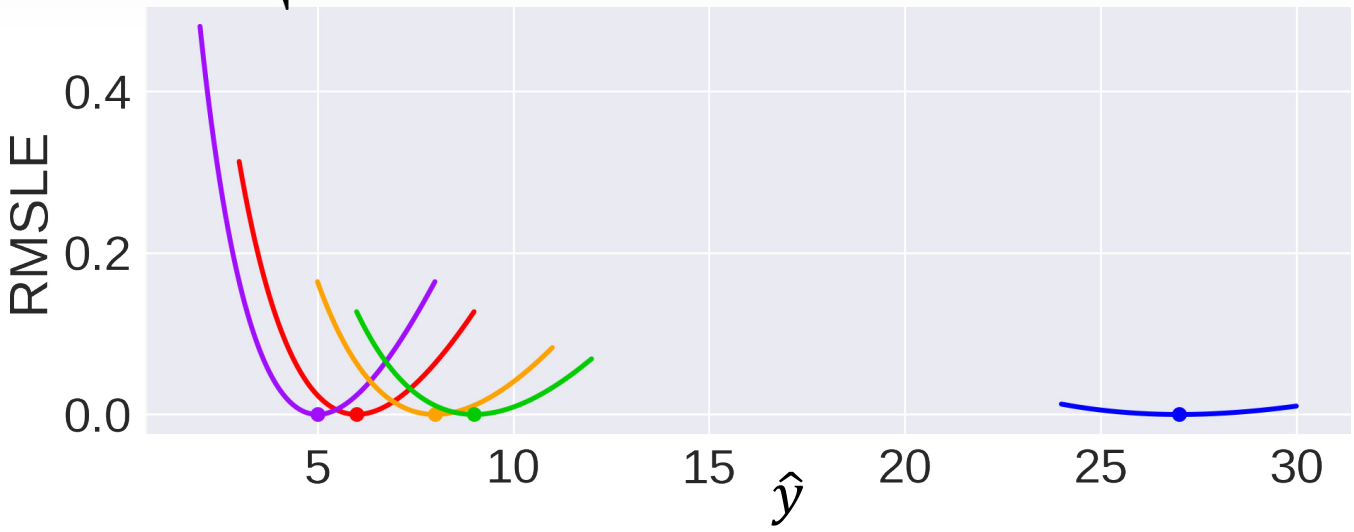
$$\begin{aligned}\text{RMSLE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log(y_i+1)-\log(\hat{y}_i+1)\right)^2} = \\ &= RMSE\left(\log(y_i+1),\log(\hat{y}_i+1)\right) = \\ &= \sqrt{MSE\left(\log(y_i+1),\log(\hat{y}_i+1)\right)}\end{aligned}$$

(R)MSLE: Root Mean Square Logarithmic Error

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

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25



(R)MSLE: constant

$$\text{RMSLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log(y_i+1)-\log(\alpha+1)\right)^2} =$$

$$= RMSE\left(\log(y_i+1), \log(\alpha+1)\right) =$$

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

(R)MSLE: constant

$$\text{RMSLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log(y_i+1) - \log(\alpha+1)\right)^2} =$$
$$= RMSE\left(\log(y_i+1), \log(\alpha+1)\right) =$$
$$= \sqrt{MSE\left(\log(y_i+1), \log(\alpha+1)\right)}$$

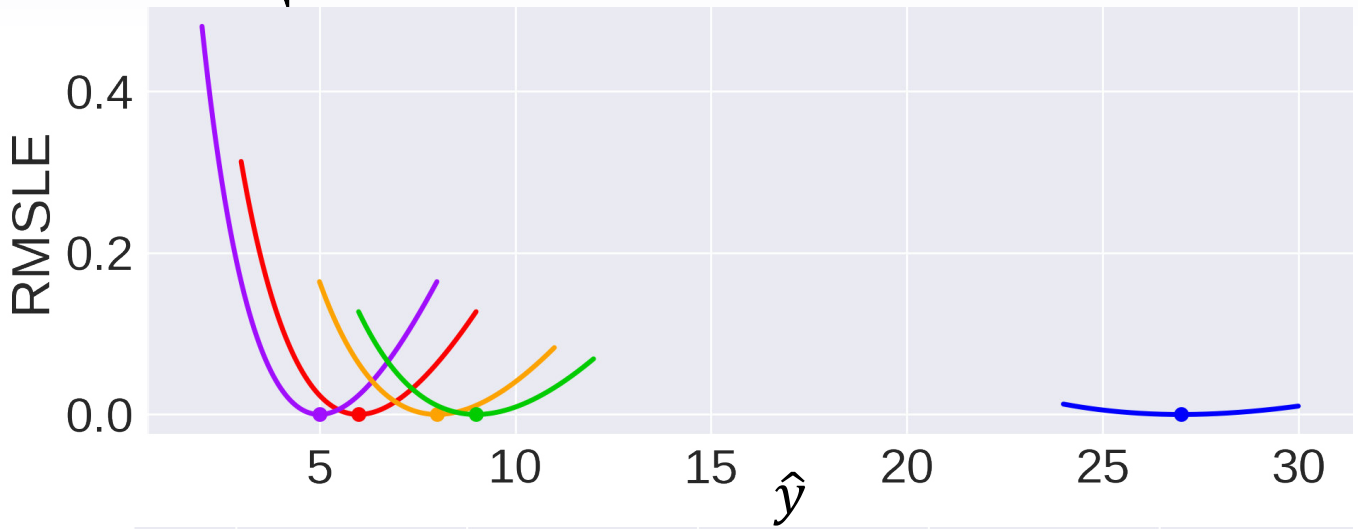
- Best constant in log space is a mean target value
- We need to exponentiate it to get an answer

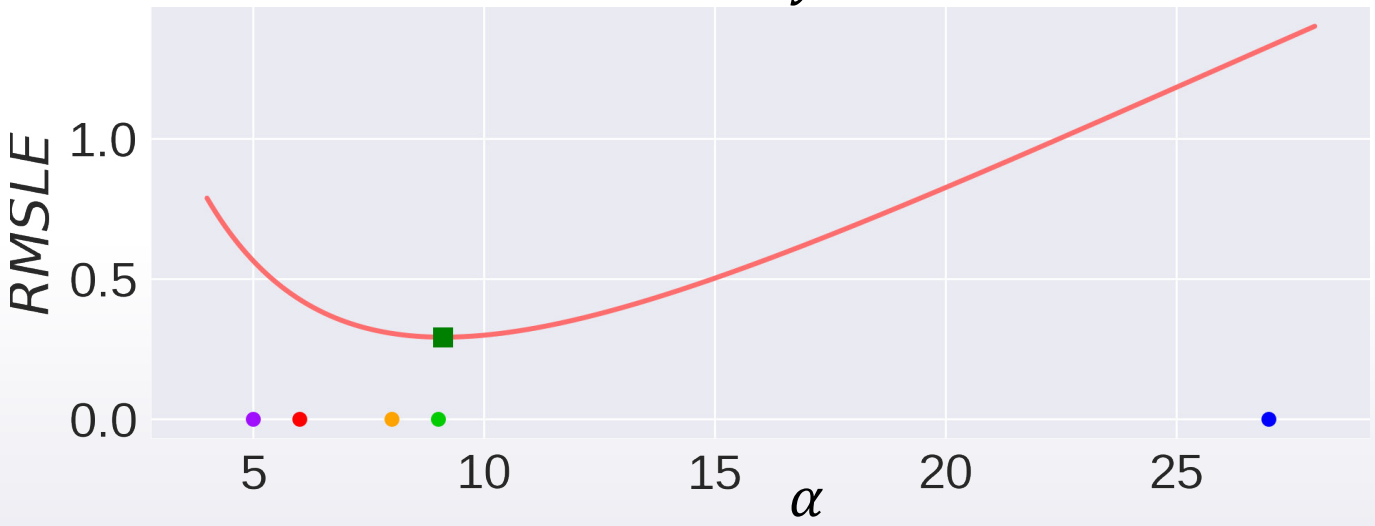
(R)MSLE: constant

$$\text{RMSLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log(y_i+1) - \log(\alpha+1)\right)^2}$$

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25





Compare the constants

Metric	Constant
MSE	11
RMSLE	9.11
MAE	8
MSPE	6.6
MAPE	6

Conclusion

Discussed the metrics, sensitive to relative errors:

- **(R)MSPE**
  - Weighted version of MSE
- **MAPE**
  - Weighted version of MAE
- **(R)MSLE**
  - MSE in log space

Classification metrics

Plan for the video

- Accuracy
- Logarithmic loss
- Area under ROC curve
- (Quadratic weighted) Kappa

Notation

- N – is number of objects
- L – is number of classes

- y – ground truth
- $\hat{y}$ – predictions
- $[a = b]$ – indicator function

- Soft labels (soft predictions) are classifier's scores

- Hard labels (hard predictions):
  - $\arg\max_i f_i(x)$
  - $[f(x) > b]$, b – threshold

Accuracy score

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}[\hat{y}_i = y_i]$$

- How frequently our class prediction is correct.

Accuracy score

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}[\alpha = y_i]$$

- How frequently our class prediction is correct.
- Best constant:
  - **predict the most frequent class.**

Accuracy score

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}[\alpha = y_i]$$

- How frequently our class prediction is correct.
- Best constant:
  - **predict the most frequent class.**

- Dataset:
  - 10 cats
  - 90 dogs

Predict always dog:

Accuracy = **0.9**!

Logarithmic loss (logloss)

- Binary:

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

$$y_i \in \mathbb{R}, \quad \hat{y}_i \in \mathbb{R}$$

Logarithmic loss (logloss)

- Binary:

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

$$y_i \in \mathbb{R}, \quad \hat{y}_i \in \mathbb{R}$$

- Multiclass:

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{L} y_{il} \log(\hat{y}_{il})$$

$$y_i \in \mathbb{R}^L, \quad \hat{y}_i \in \mathbb{R}^L$$

Logarithmic loss (logloss)

- Binary:

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

$$y_i \in \mathbb{R}, \quad \hat{y}_i \in \mathbb{R}$$

- Multiclass:

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{L} y_{il} \log(\hat{y}_{il})$$

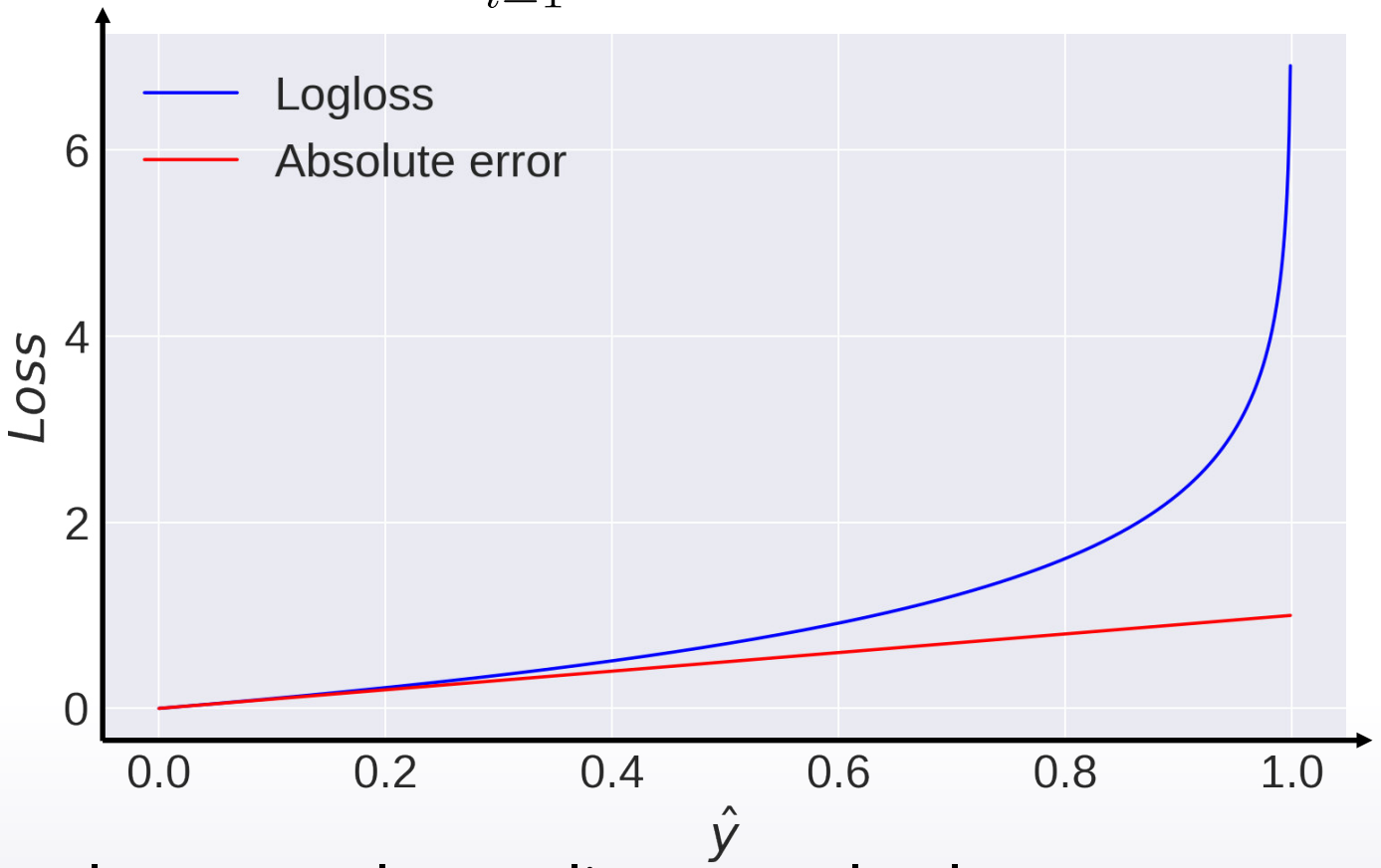
$$y_i \in \mathbb{R}^L, \quad \hat{y}_i \in \mathbb{R}^L$$

- In practice:

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{L} y_{il} \log(\min(\max(\hat{y}_{il}, 10^{-15}), 1 - 10^{-15}))$$

Logarithmic loss (logloss)

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$



- Logloss strongly penalizes completely wrong answers

Logarithmic loss (logloss)

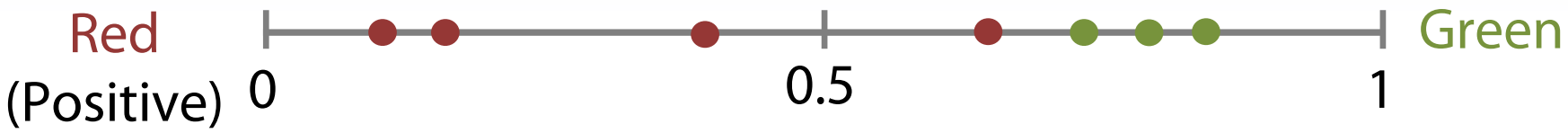
$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\alpha) + (1 - y_i) \log(1 - \alpha)$$

- Best constant:
  - **set α_i to frequency of i-th class.**

- Dataset:
  - 10 cats
  - 90 dogs

$$\alpha = [0.1, 0.9]$$

Area Under Curve (AUC ROC)



$$\text{Accuracy}([\hat{y} > 0.5]) = \frac{6}{7}$$

Area Under Curve (AUC ROC)

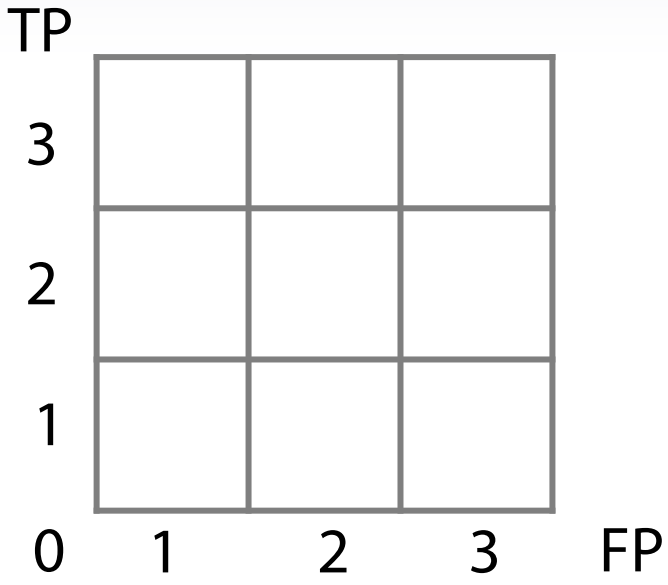
$$\text{Accuracy}([\hat{y} > 0.7]) = 1$$

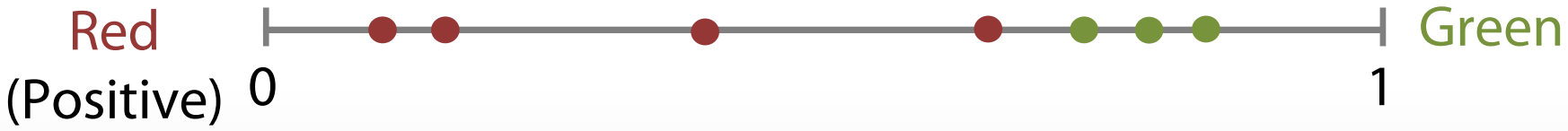
Area Under Curve (AUC ROC)

$$\text{Accuracy}([\hat{y} > 0.7]) = 1$$

- Only for binary tasks
- Depends only on ordering of the predictions, not on absolute values
- **Several explanations**
  1) Area under curve
  2) Pairs ordering

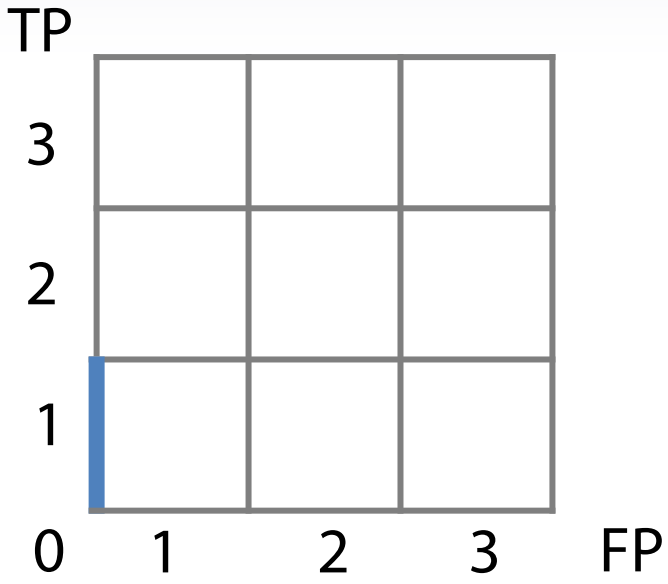
Area Under Curve (AUC ROC)

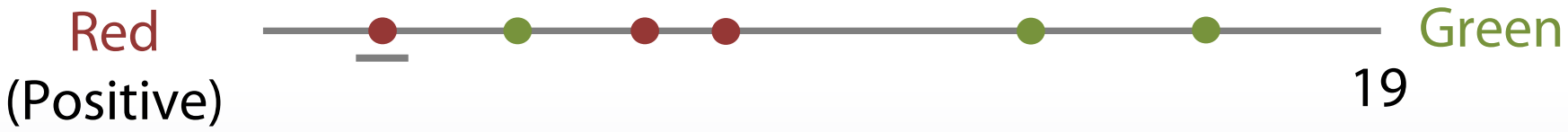




TP – true positives, **FP** – false positives

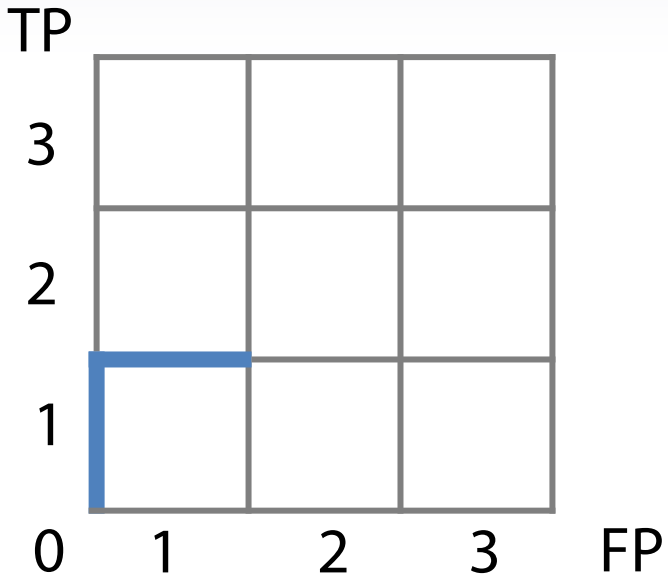
Area Under Curve (AUC ROC)

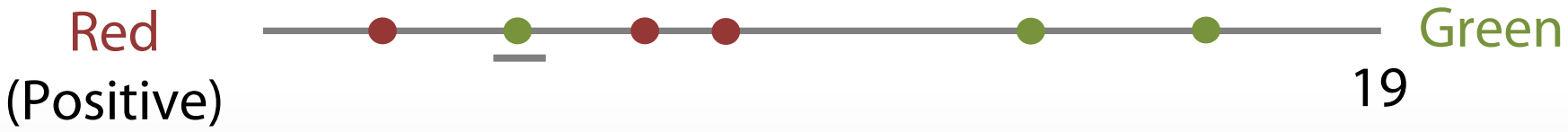




TP – true positives, **FP** – false positives

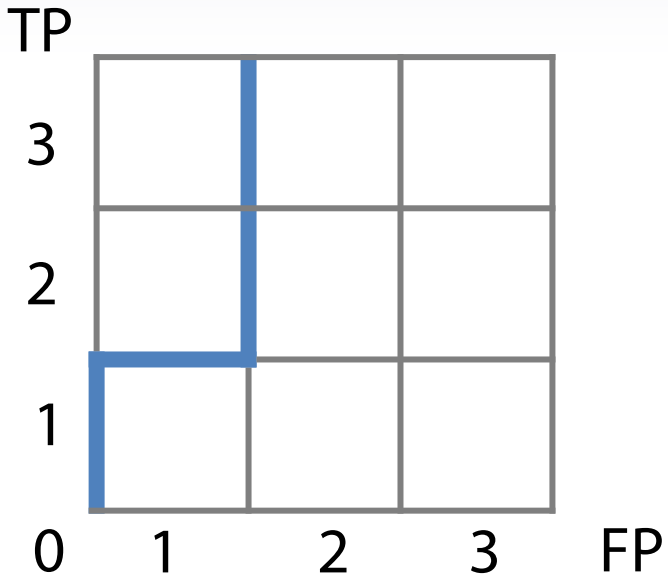
Area Under Curve (AUC ROC)





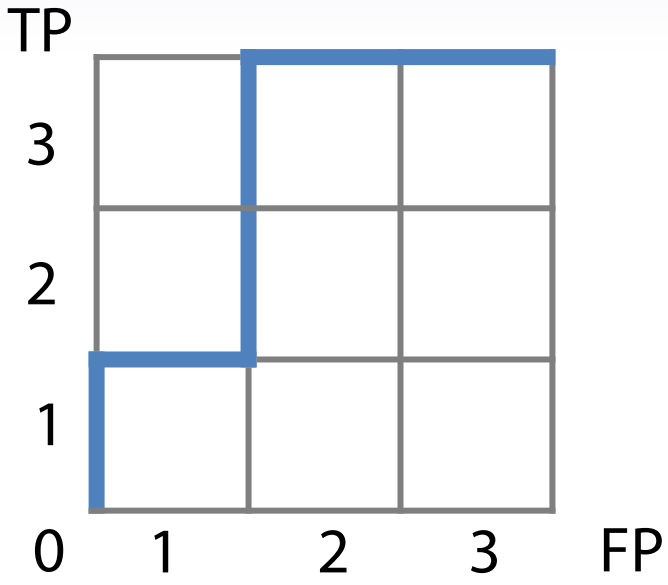
TP – true positives, **FP** – false positives

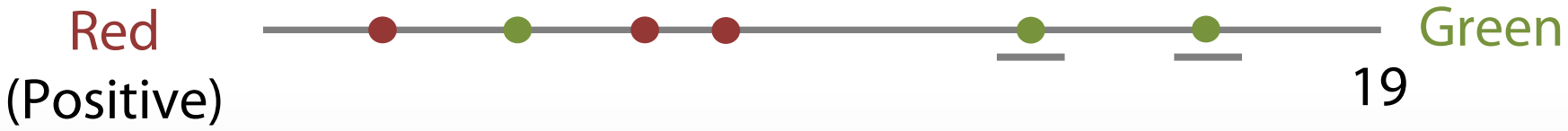
Area Under Curve (AUC ROC)



TP – true positives, **FP** – false positives

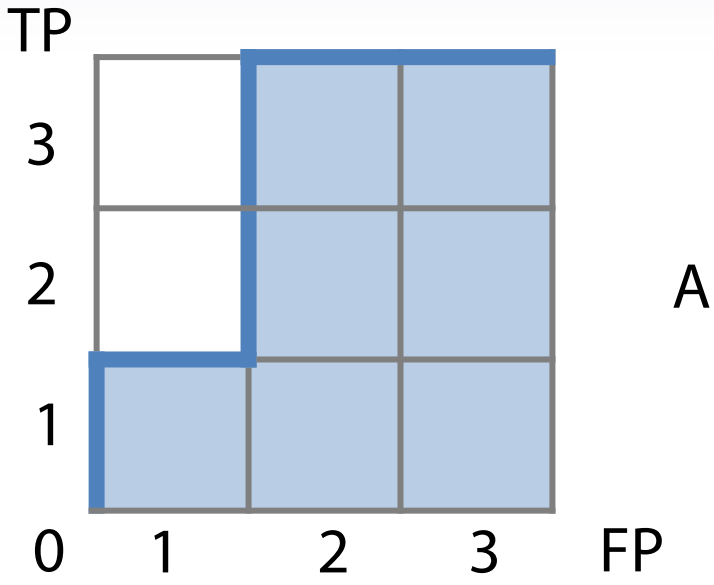
Area Under Curve (AUC ROC)



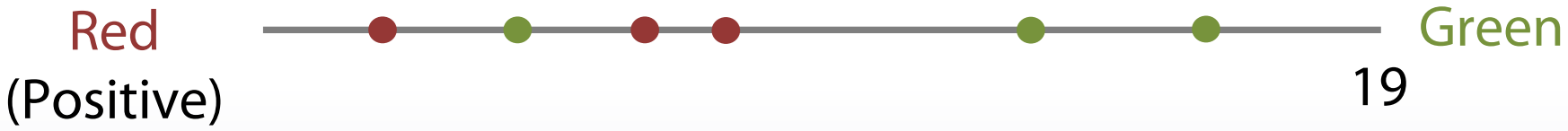


TP – true positives, **FP** – false positives

Area Under Curve (AUC ROC)

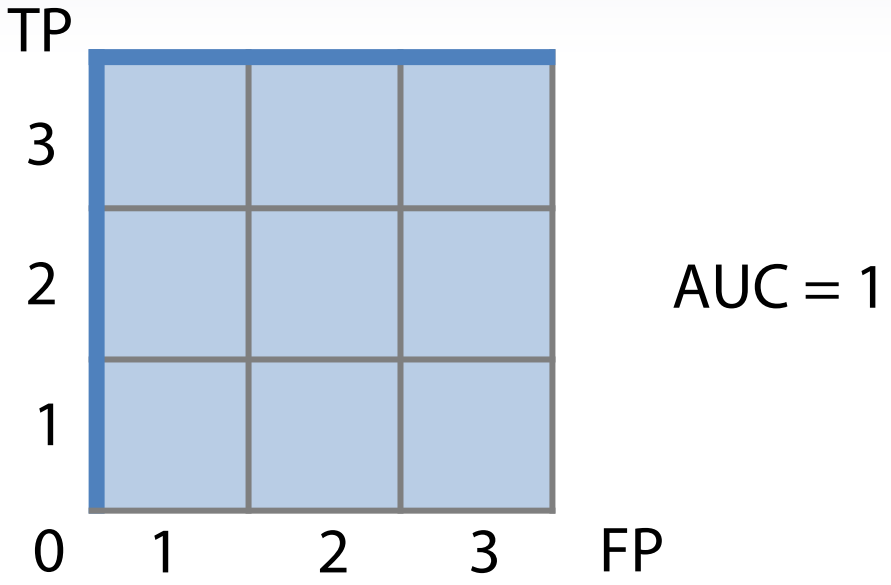


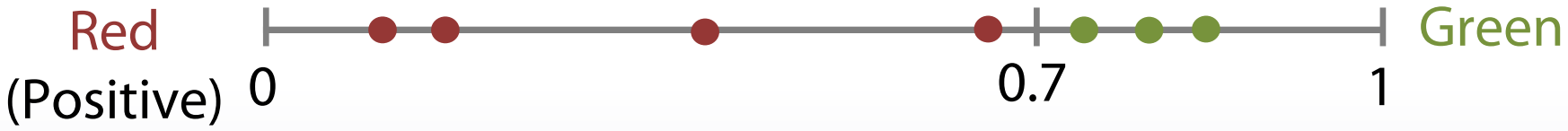
AUC = 7/9



TP – true positives, **FP** – false positives

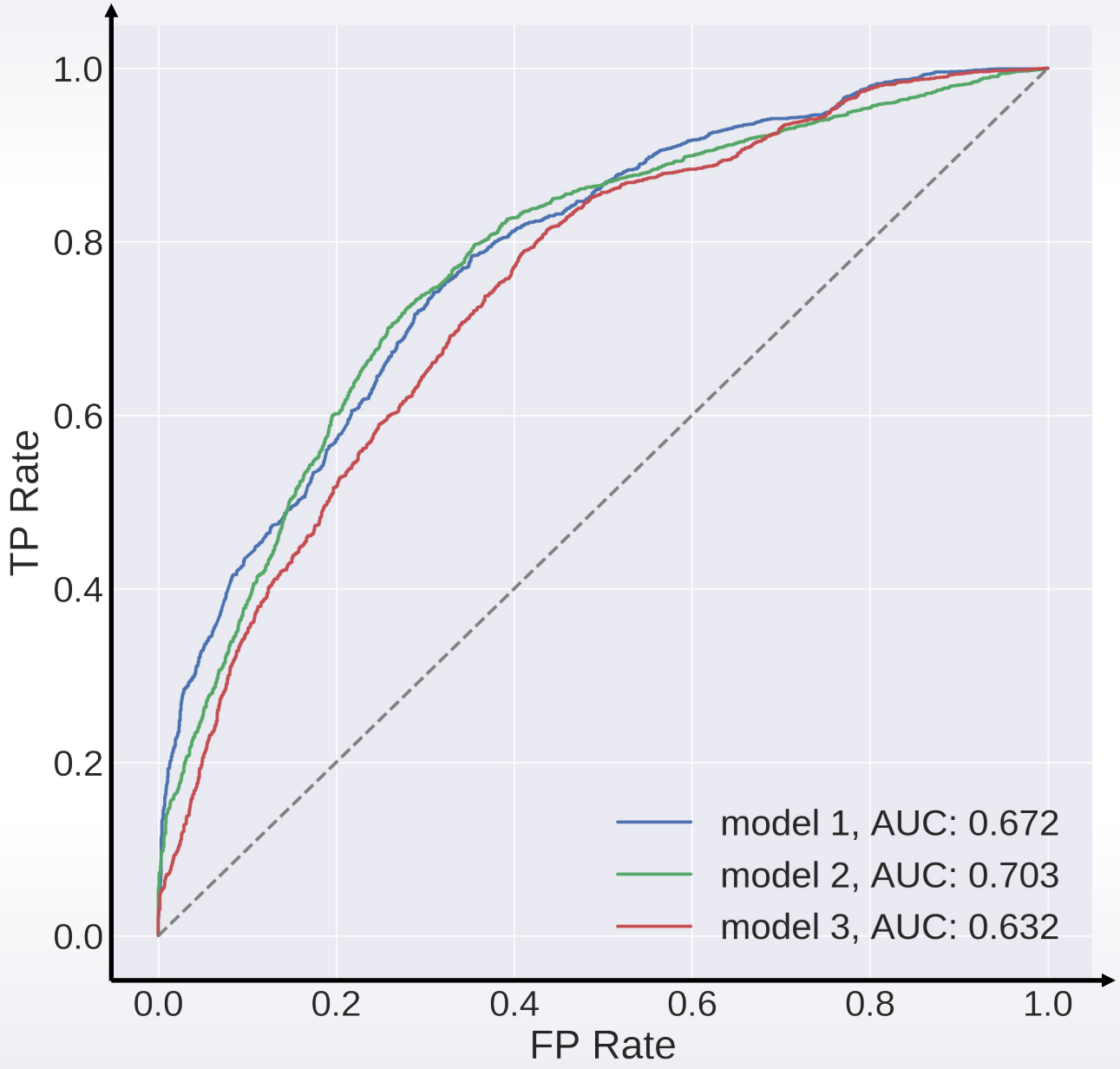
Area Under Curve (AUC ROC)





TP – true positives, **FP** – false positives

Area Under Curve (AUC ROC)



Area Under Curve (AUC ROC)

Area Under Curve (AUC ROC)

$$\text{AUC} = \frac{\#\text{ correctly ordered pairs}}{\text{total number of pairs}} =$$

$$= 1 - \frac{\#\text{ incorrectly ordered pairs}}{\text{total number of pairs}}$$

pair = (red object, green object)

Area Under Curve (AUC ROC)

- **Best constant:**
  - All constants give same score

- **Random predictions lead to AUC = 0.5**

Cohen's Kappa motivation

Dataset:

- 10 cats
- 90 dogs

Baseline accuracy = 0.9

$$\text{my\_score} = 1 - \frac{1 - \text{accuracy}}{1 - \text{baseline}}$$

- accuracy = 1 ⟹ my_score = 1
- accuracy = 0.9 ⟹ my_score = 0

Cohen's Kappa motivation

Dataset:

- 10 cats
- 90 dogs

Predict *20 cats* and *80 dogs* at random: *accuracy* ~ 0.74

0.2*0.1 + 0.8*0.9 = 0.74

$$\text{Cohen's Kappa} = 1 - \frac{1 - \text{accuracy}}{1 - p_e}$$

p_e – what accuracy would be on average, if we randomly permute our predictions

$$p_e = \frac{1}{N^2} \sum_k n_{k1} n_{k2}$$

Cohen's Kappa motivation

Dataset:

- 10 cats
- 90 dogs

Predict *20 cats* and *80 dogs* at random: *accuracy* ~ 0.74

error ~ 0.26

$$\text{Cohen's Kappa} = 1 - \frac{\text{error}}{\text{baseline error}}$$

Weighted error

Dataset:

- 10 cats
- 90 dogs
- 20 tigers

Error weight matrix

pred\ true	cat	dog	tiger
cat	0	1	10
dog	1	0	10
tiger	1	1	0

Weighted error and weighted Kappa

Confusion matrix *C*

pred\ true	cat	dog	tiger
cat	4	2	3
dog	2	88	5
tiger	4	10	12

Weight matrix *W*

pred\ true	cat	dog	tiger
cat	0	1	10
dog	1	0	10
tiger	1	1	0

$$\text{weighted error} = \frac{1}{const} \sum_{i,j} C_{ij} W_{ij}$$

Weighted error and weighted Kappa

Confusion matrix *C*

pred\ true	cat	dog	tiger
cat	4	2	3
dog	2	88	4
tiger	4	10	12

Weight matrix *W*

pred\ true	cat	dog	tiger
cat	0	1	10
dog	1	0	10
tiger	1	1	0

$$\text{weighted error} = \frac{1}{const} \sum_{i,j} C_{ij} W_{ij}$$

$$\text{weighted kappa} = 1 - \frac{\text{weighted error}}{\text{weighted baseline error}}$$

Quadratic and Linear Weighted Kappa

Linear weights

pred\ true	1	2	3
1	0	1	2
2	1	0	1
3	2	1	0

Quadratic weights

pred\ true	1	2	3
1	0	1	4
2	1	0	1
3	4	1	0

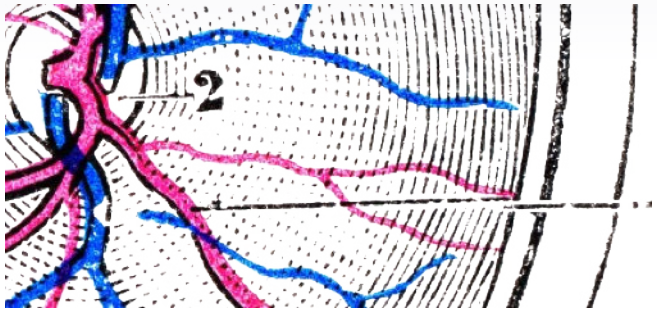
$$w_{ij} = |i - j|$$

$$w_{ij} = (i - j)^2$$

$$\text{weighted kappa} = 1 - \frac{\text{weighted error}}{\text{weighted baseline error}}$$

Quadratic Weighted Kappa

CrowdFlower Search Results Relevance



Diabetic Retinopathy Detection

Prudential Life Insurance Assessment

ASAP

Automated Student Assessment Prize

Phase One: Automated Essay Scoring

The Hewlett Foundation: Automated Essay Scoring

Conclusion

- Accuracy
- Logloss
- AUC (ROC)
- (Quadratic weighted) Kappa

General approaches for metrics optimization

Overview

- Loss vs metric
- Approaches to metrics optimization in general

Loss and metric

Loss and metric

- **Target metric** is what *we* want to optimize

Loss and metric

- **Target metric** is what *we* want to optimize
- **Optimization loss** is what *model* optimizes

Loss and metric

- **Target metric** is what *we* want to optimize
- **Optimization loss** is what *model* optimizes

Loss and metric

- **Target metric** is what *we* want to optimize
- **Optimization loss** is what *model* optimizes

Synonyms: loss, cost, objective

Approaches for target metric optimization

Approaches for target metric optimization

- **Just run the right model!**
  - MSE, Logloss

Approaches for target metric optimization

- **Just run the right model!**
  - MSE, Logloss

- **Preprocess train and optimize another metric**
  - MSPE, MAPE, RMSLE, ...

Approaches for target metric optimization

- **Just run the right model!**
  - MSE, Logloss
- **Preprocess train and optimize another metric**
  - MSPE, MAPE, RMSLE, ...
- **Optimize another metric, postprocess predictions**
  - Accuracy, Kappa

Approaches for target metric optimization

- **Just run the right model!**
  - MSE, Logloss
- **Preprocess train and optimize another metric**
  - MSPE, MAPE, RMSLE, ...
- **Optimize another metric, postprocess predictions**
  - Accuracy, Kappa
- **Write custom loss function**
  - Any, if you can

Custom loss for XGBoost

- **Define an 'objective':**
  - function that computes *first and second order derivatives* w.r.t. predictions.

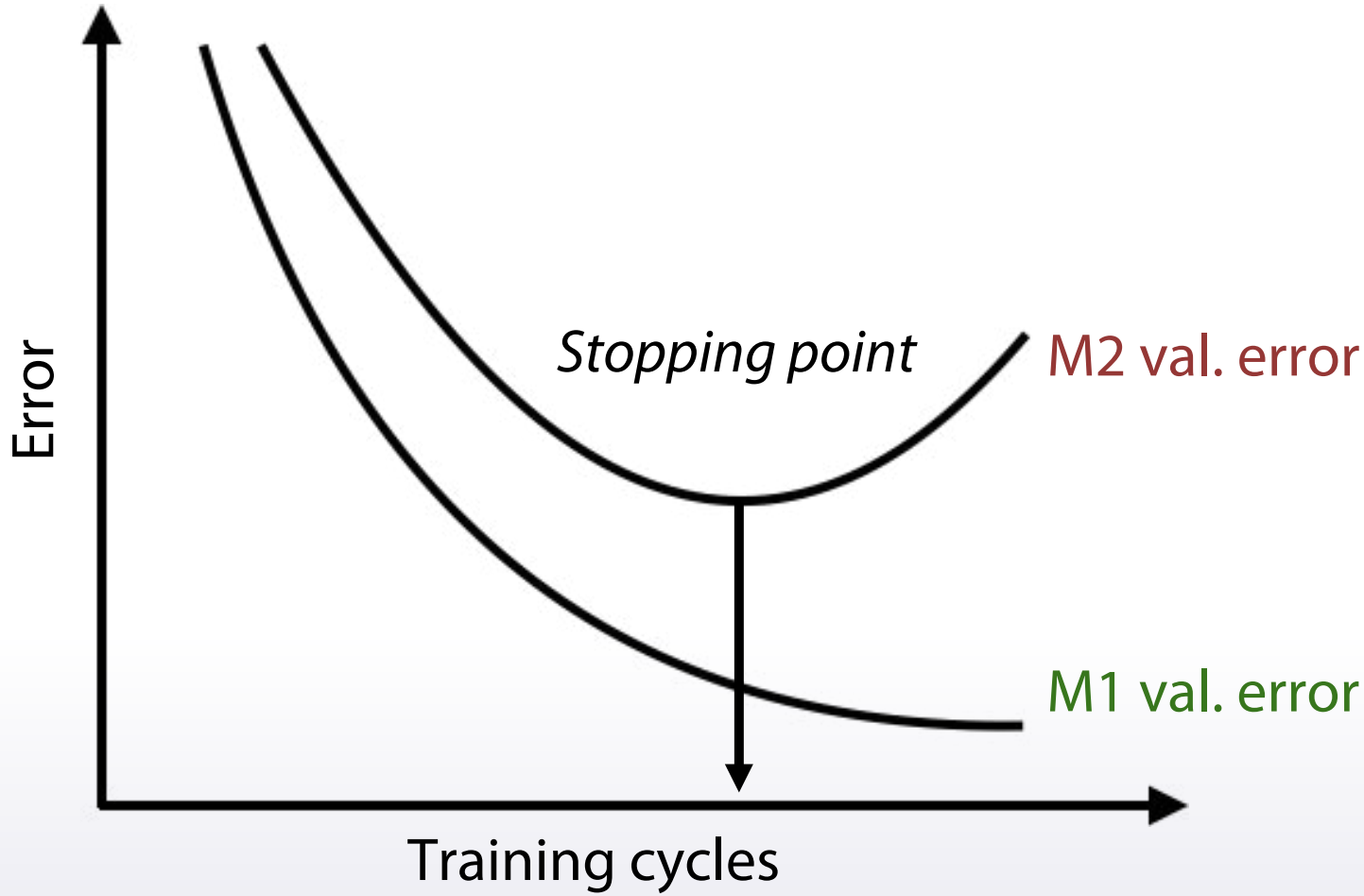
```
def logregobj(preds, dtrain):
    labels = dtrain.get_label()
    preds = 1.0 / (1.0 + np.exp(-preds))
    grad = preds - labels
    hess = preds * (1.0-preds)
    return grad, hess
```

Approaches for target metric optimization

- **Just run the right model!**
  - MSE, Logloss
- **Preprocess train and optimize another metric**
  - MSPE, MAPE, RMSLE, ...
- **Optimize another metric, postprocess predictions**
  - Accuracy, Kappa
- **Write custom loss function**
  - Any, if you can
- **Optimize another metric, use early stopping**
  - Any

Early stopping

- Optimize metric M1, monitor metric M2
  - Stop when M2 score is the best



Conclusion

- **Loss vs metric**
- **Approaches in general:**
  - Just run the right model
  - Preprocess train and optimize another metric
  - Optimize another metric, postprocess predictions
  - Write a custom loss function
  - Optimize another metric, use early stopping

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

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

How do you optimize them?

Just fit the right model!

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

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

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

How do you optimize it?

Again, just run the right model!

MAE

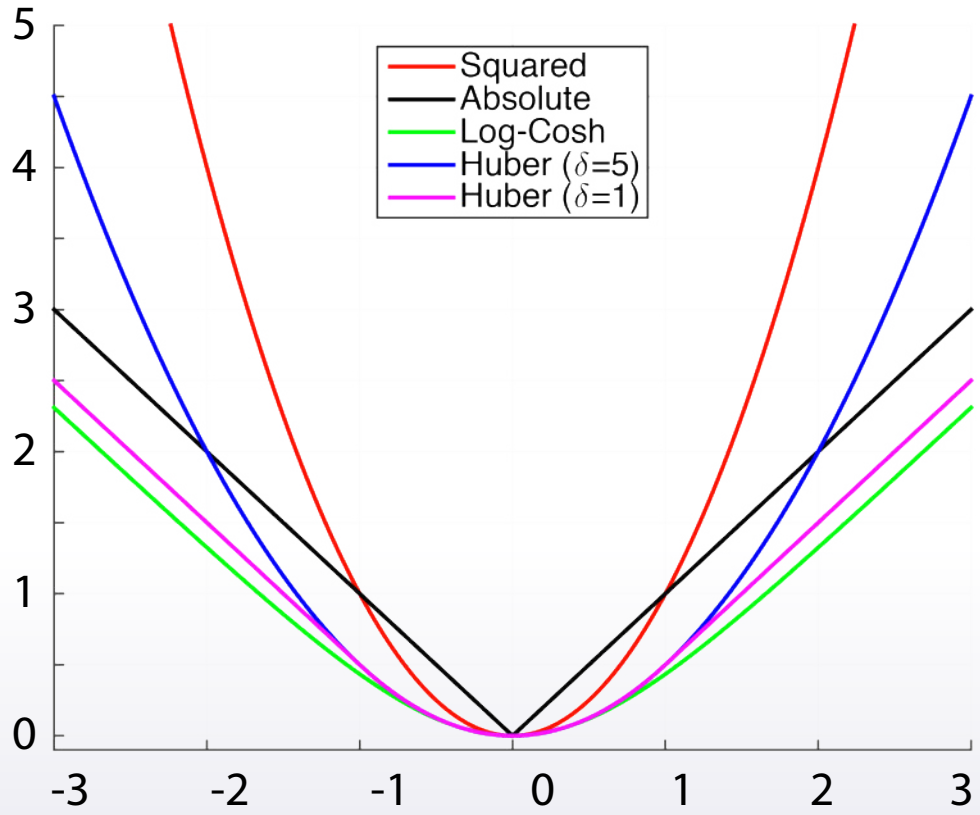
- **Tree-based**
  - ~~`XGBoost`~~`, LightGBM`
  - `sklearn.RandomForestRegressor`
- **Linear models**
  - ~~`sklearn.<>Regression`~~
  - ~~`sklearn.SGDRegressor`~~
  - `Vowpal Wabbit` *`(quantile loss)`*
- **Neural nets**
  - `PyTorch, Keras, TF,` etc.

Synonyms: L1, Median regression

Read the docs!

MAE: optimal constant

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



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!~~

MSPE (MAPE) as weighted MSE (MAE)

Sample weights

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

MSPE (MAPE)

- **Use weights for samples (`sample_weights`)**
  - And use MSE (MAE)

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

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)

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

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

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

Train:

1. Transform target:

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

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

Classification metrics optimization: Logloss and accuracy

Classification metrics optimization

- Logloss
- Accuracy
- AUC
- (Quadratic weighted) Kappa

Logloss

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

How do you optimize it?

Just run the right model!
(or calibrate others)

Logloss

- **Tree-based**
  - `XGBoost, LightGBM`
  - ~~`sklearn.RandomForestClassifier`~~
- **Linear models**
  - `sklearn.<>Regression`
  - `sklearn.SGDRegressor`
  - `Vowpal Wabbit`
- **Neural nets**
  - `PyTorch, Keras, TF,` etc.

Synonyms: Logistic loss

Read the docs!

Logloss

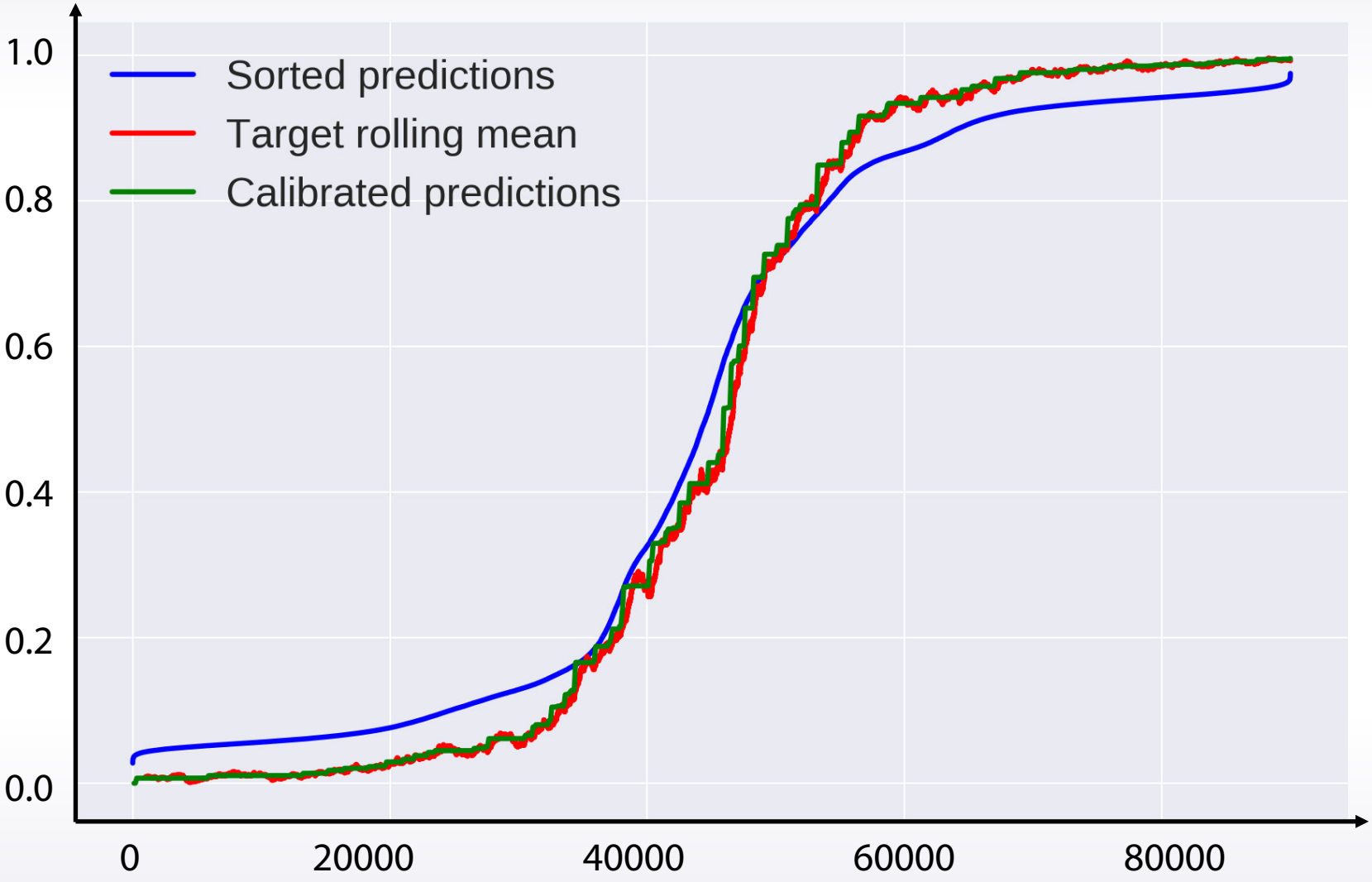
Correct probabilities:

- Take all objects with score e.g. ~ 0.8
  - 80% of them of class 1
  - 20% of them class 0

Incorrect probabilities:

- Take all objects with score e.g. ~ 0.8
  - 50% of them of class 1
  - 50% of them of class 0

Probability calibration



Probability calibration

- Platt scaling
  - Just fit Logistic Regression to your predictions (like in stacking)
- Isotonic regression
  - Just fit Isotonic Regression to your predictions (like in stacking)
- Stacking
  - Just fit XGBoost or neural net to your predictions

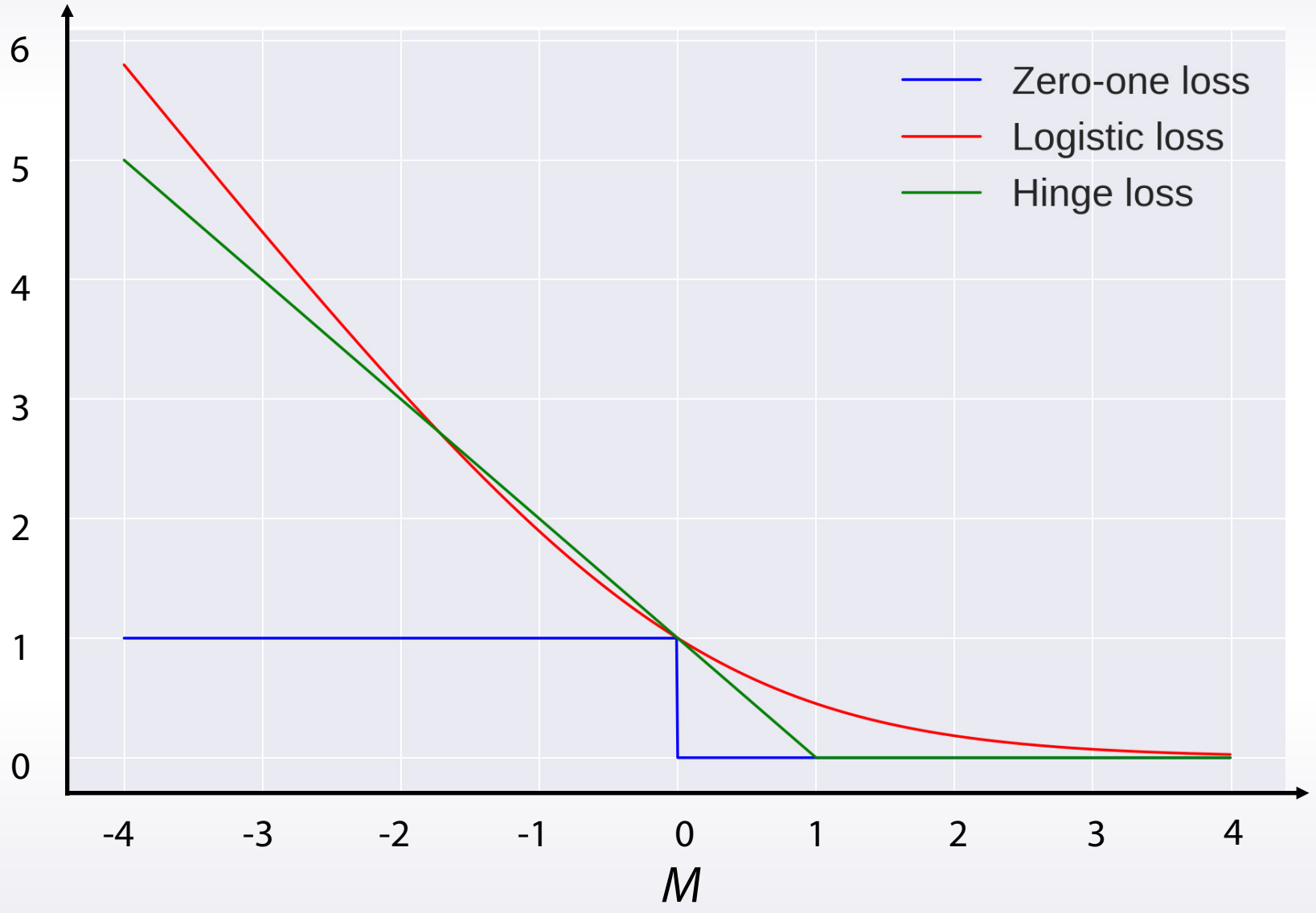
Accuracy

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}[\hat{y}_i = y_i]$$

How do you optimize it?

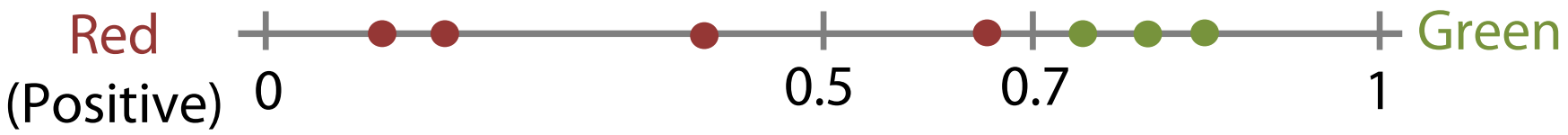
Fit any metric and tune treshold!

Accuracy



Accuracy

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}[[f(x) > b] = y_i]$$



$$b = 0.5 \quad \Rightarrow \quad \text{Accuracy} = \frac{6}{7}$$

$$b = 0.7 \quad \Rightarrow \quad \text{Accuracy} = 1$$

Conclusion

- Logloss
- Accuracy
- AUC
- (Quadratic weighted) Kappa

Classification metrics optimization: AUC and Kappa

In this video

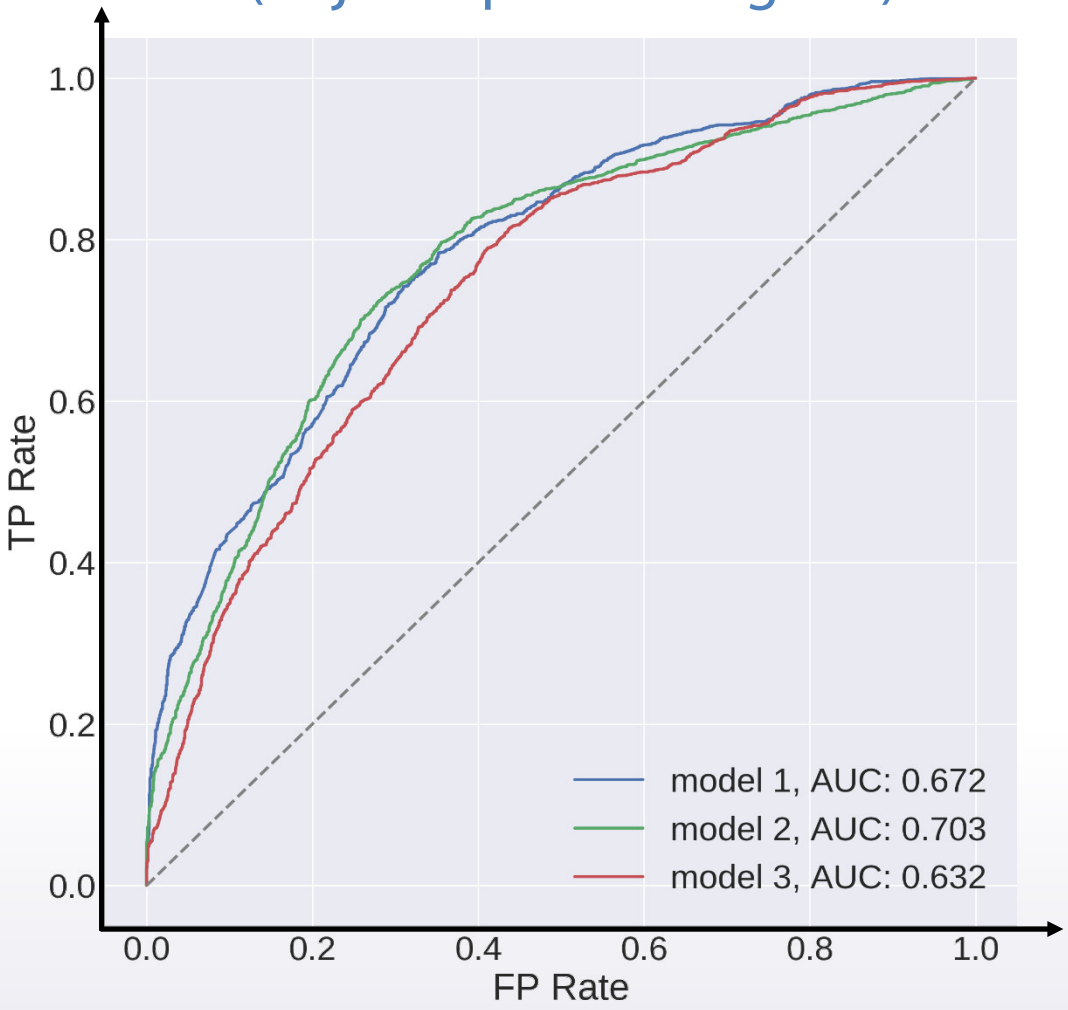
- Logloss
- Accuracy
- AUC
- (Quadratic weighted) Kappa

AUC (ROC)

How do you optimize it?

Run the right model

(or just optimize logloss)

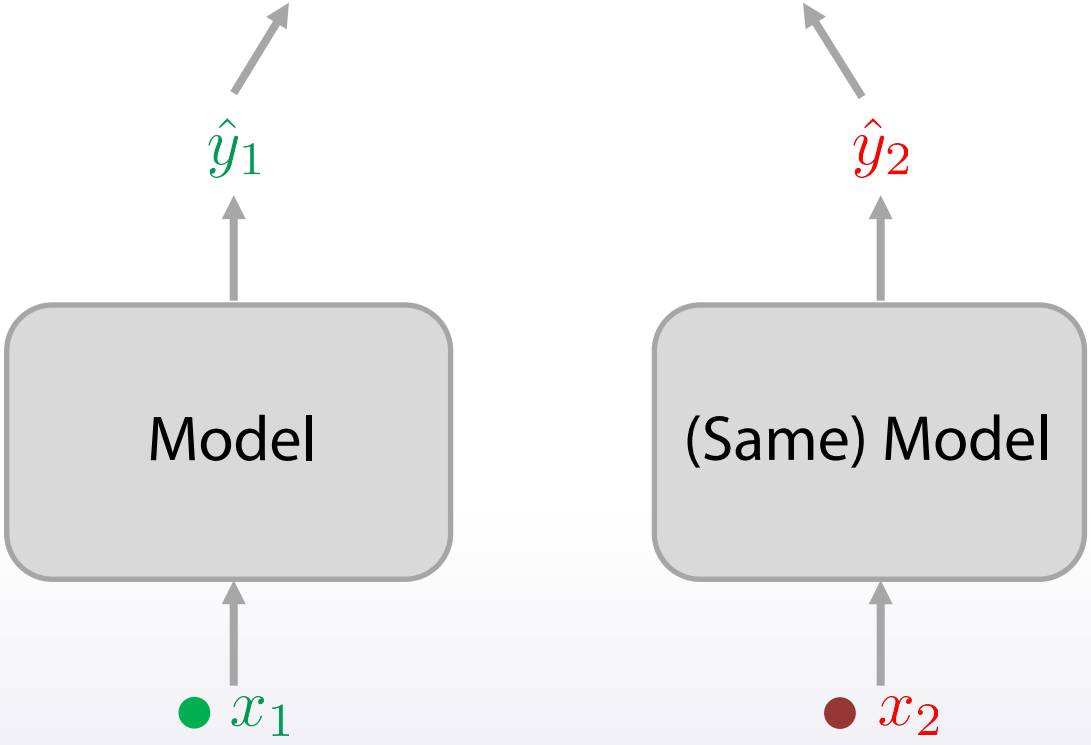


Pointwise loss

$$\min \sum_{i}^{N} l_{point}(\hat{y}_i; y_i)$$

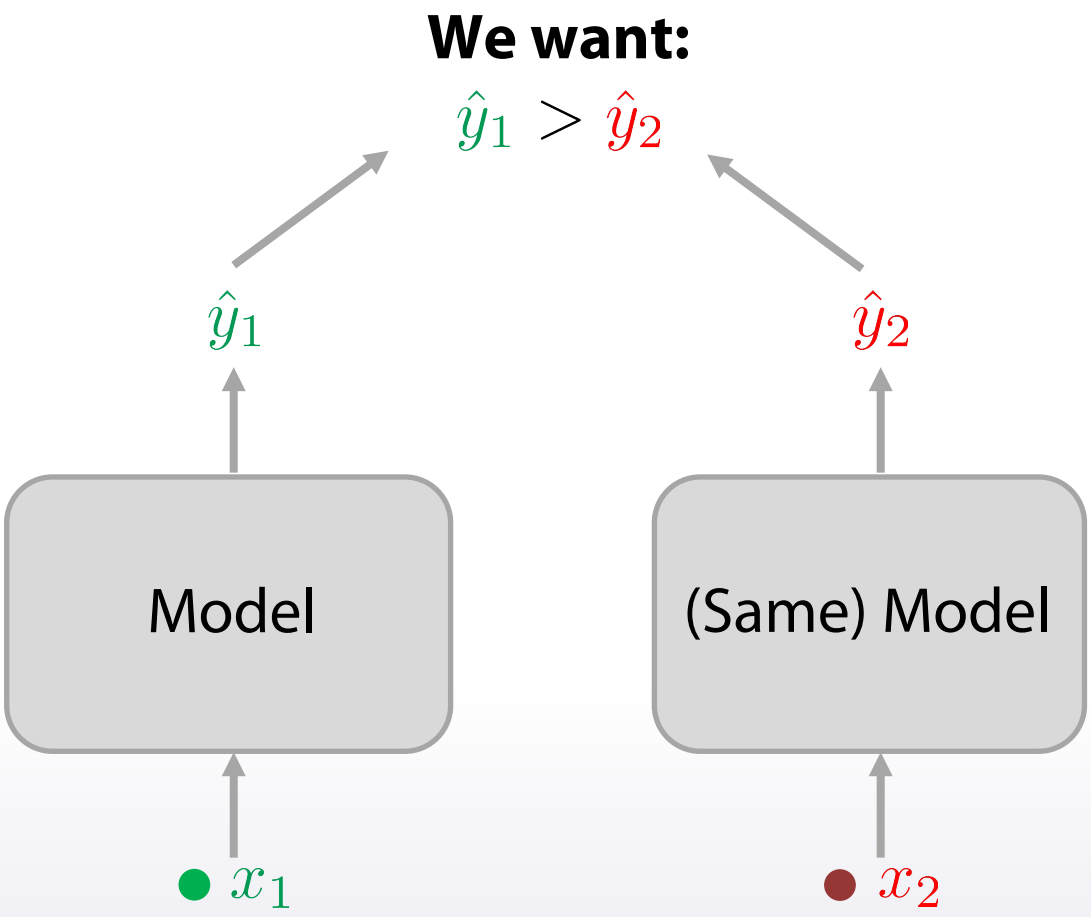
We want:

$\hat{y}_1 = 1$ $\hat{y}_2 = 0$



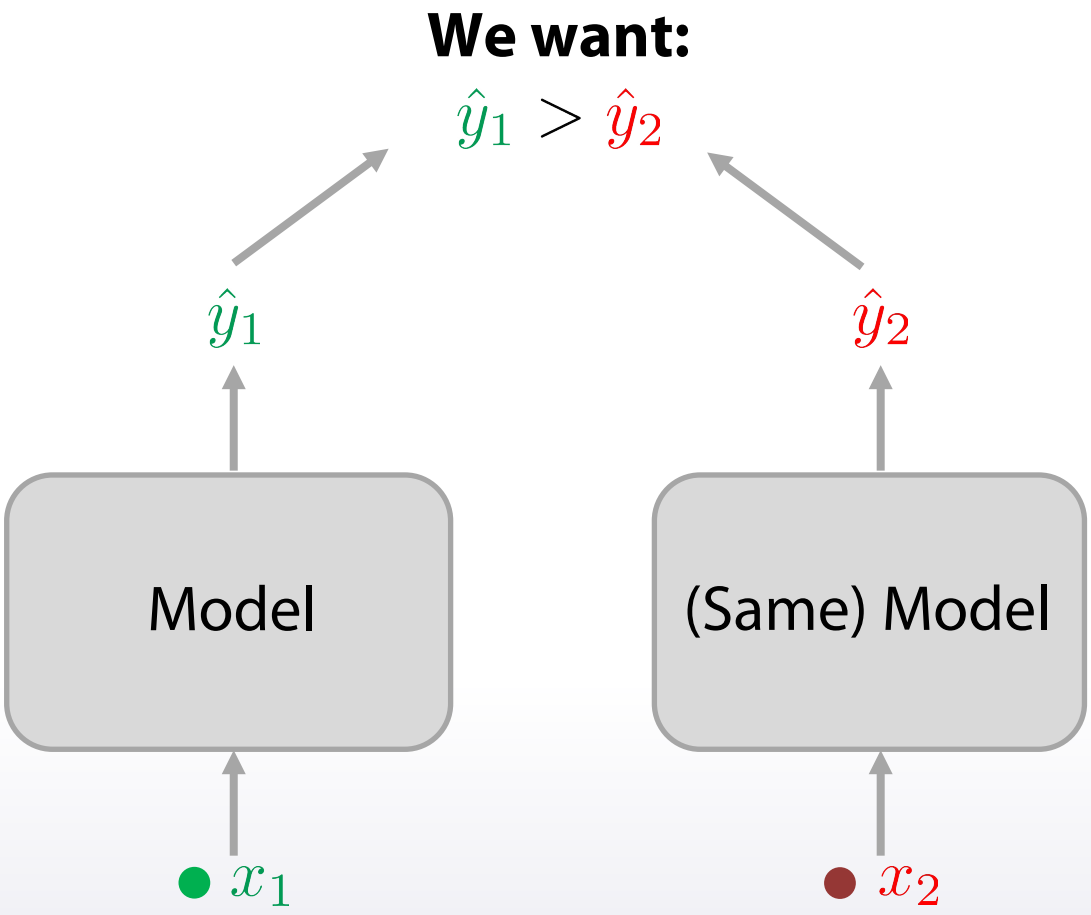
Pairwise loss

$$\min \sum_{i}^{N} \sum_{j}^{N} l_{pair}(\hat{y}_i, \hat{y}_j; y_i, y_j)$$



Pairwise loss

$$\text{Loss} = -\frac{1}{N_0 N_2} \sum_{j:y_j=1}^{N_1} \sum_{i:y_i=0}^{N_0} \log(\text{prob}(\hat{y}_j - \hat{y}_i))$$



AUC

- **Tree-based**
  - `XGBoost, LightGBM`
  - ~~`sklearn.RandomForestClassifier`~~
- **Linear models**
  - ~~`sklearn. LogisticRegression`~~
  - ~~`sklearn.SGDRegressor`~~
  - ~~`Vowpal Wabbit`~~
- **Neural nets**
  - `PyTorch, Keras, TF` – not out of the box

Read the docs!

Quadratic weighted Kappa

How do you optimize it?

- **Optimize *MSE* and find right *thresholds***
  - *Simple*
- **Custom smooth loss for GBDT or neural nets**
  - Harder

MSE + tresholds

1. **Optimize MSE**

$$\text{Kappa}(y, \hat{y}) \approx 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\text{hard to deal with part}} =$$

$$= 1 - \frac{\text{MSE}(y, \hat{y})}{\text{hard to deal with part}}$$

MSE + tresholds

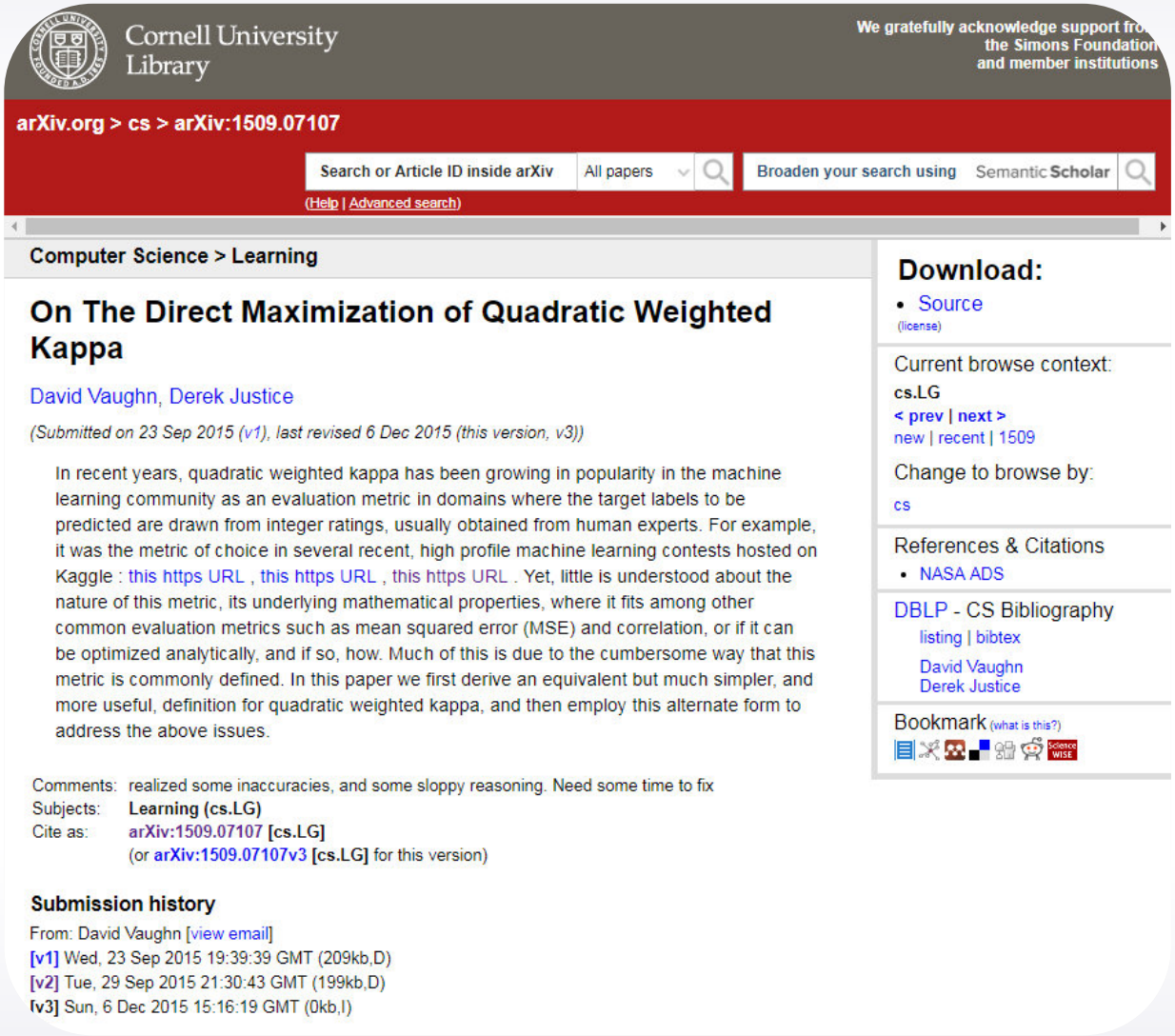
1. **Optimize MSE**

$$\text{Kappa}(y, \hat{y}) \approx 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\text{hard to deal with part}} =$$

$$= 1 - \frac{\text{MSE}(y, \hat{y})}{\text{hard to deal with part}}$$

2. **Find right *thresholds***
   - Bad: `np.round(predictions)`
   - Better: optimize thresholds

Smooth loss



Cornell University Library

We gratefully acknowledge support fro
the Simons Foundation
and member institutions

arXiv.org > cs > arXiv:1509.07107

Search or Article ID inside arXiv | All papers | Broaden your search using Semantic Scholar

(Help | Advanced search)

Computer Science > Learning

On The Direct Maximization of Quadratic Weighted Kappa

David Vaughn, Derek Justice

(Submitted on 23 Sep 2015 (v1), last revised 6 Dec 2015 (this version, v3))

In recent years, quadratic weighted kappa has been growing in popularity in the machine learning community as an evaluation metric in domains where the target labels to be predicted are drawn from integer ratings, usually obtained from human experts. For example, it was the metric of choice in several recent, high profile machine learning contests hosted on Kaggle : this https URL , this https URL , this https URL . Yet, little is understood about the nature of this metric, its underlying mathematical properties, where it fits among other common evaluation metrics such as mean squared error (MSE) and correlation, or if it can be optimized analytically, and if so, how. Much of this is due to the cumbersome way that this metric is commonly defined. In this paper we first derive an equivalent but much simpler, and more useful, definition for quadratic weighted kappa, and then employ this alternate form to address the above issues.

Comments: realized some inaccuracies, and some sloppy reasoning. Need some time to fix
Subjects: **Learning (cs.LG)**
Cite as: **arXiv:1509.07107 [cs.LG]**
(or **arXiv:1509.07107v3 [cs.LG]** for this version)

Submission history

From: David Vaughn [view email]
[v1] Wed, 23 Sep 2015 19:39:39 GMT (209kb,D)
[v2] Tue, 29 Sep 2015 21:30:43 GMT (199kb,D)
[v3] Sun, 6 Dec 2015 15:16:19 GMT (0kb,I)

Download:

- Source
(license)

Current browse context:
cs.LG
< prev | next >
new | recent | 1509

Change to browse by:
cs

References & Citations
- NASA ADS

DBLP - CS Bibliography
listing | bibtex
David Vaughn
Derek Justice

Bookmark (what is this?)

Lesson conclusion

- **Target metric is how competitors are scored**
- **Target metric VS optimization loss**
- **Regression metrics**
  - MSE, RMSE, R-squared
  - MAE
  - MSPE, MAPE
  - RMSLE
- **Classification metrics**
  - Accuracy
  - Logloss
  - AUC
  - (Quadratic weighted) Kappa