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**

  Random forest classifier predictions turn out to be quite bad in terms of logloss. But there is a way to make them better, we can calibrate the predictions to better fit logloss.

  `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

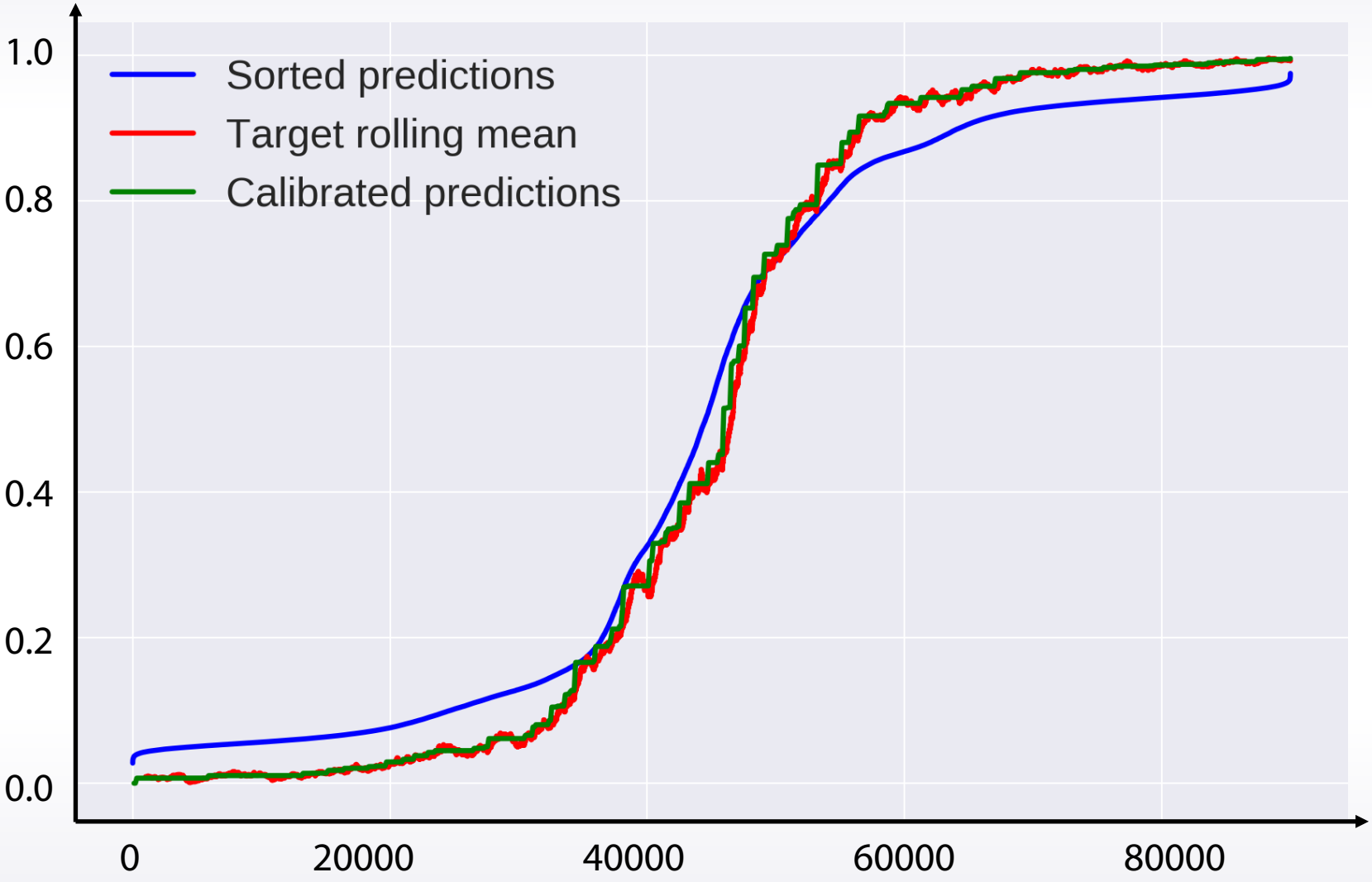
Correct probabilities:

- 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

While finally, we can use stacking, so the idea is, we can fit any classifier. It doesn't need to optimize logloss, it just needs to be good, for example, in terms of AUC.

Accuracy

Logloss was the only metric that is easy to optimize directly.

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

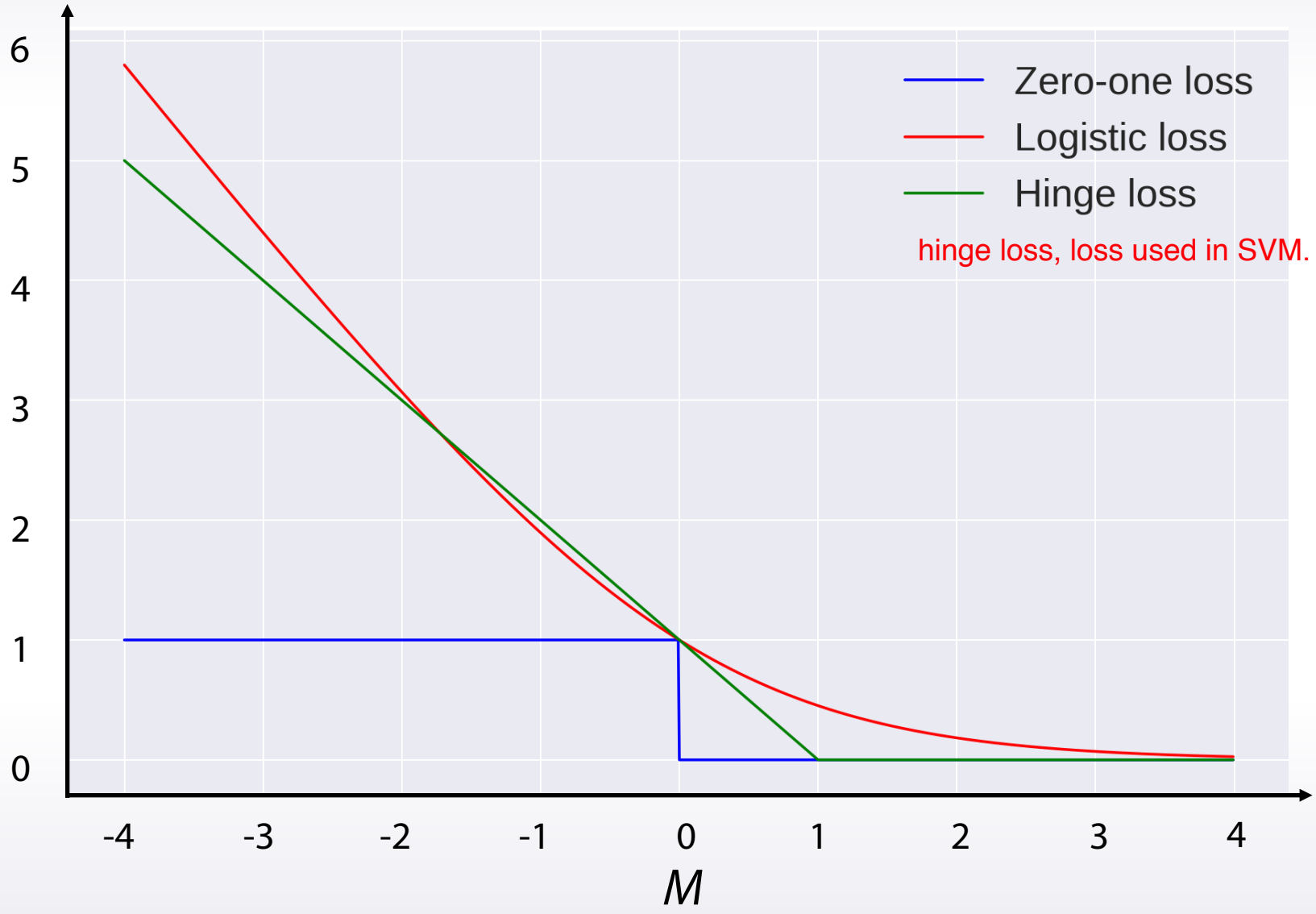
How do you optimize it?

Fit any metric and tune treshold!

if it is a binary classification task, fit any metric, and tune with the binarization threshold. For multi-class tasks, fit any metric and tune parameters comparing the models by their accuracy score, not by the metric that the models were really optimizing.

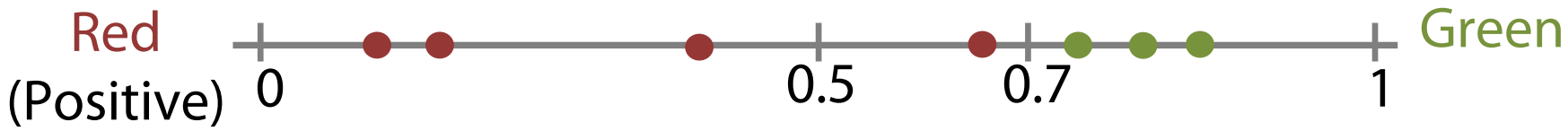
Accuracy

Just to get an intuition why accuracy is hard to optimize, let's look at this plot.



Accuracy

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



If our task is binary and soft predictions sum up to 1, argmax is equivalent to threshold function. Output 1 when the predictions for the class one is higher than 0.5, and output 0 when the prediction's lower.
So we've already seen this example where threshold 0.5 is not optimal

$$b = 0.5 \quad \Rightarrow \quad \text{Accuracy} = \frac{6}{7}$$
$$b = 0.7 \quad \Rightarrow \quad \text{Accuracy} = 1$$

We can tune the threshold we apply, we can do it with a simple grid search implemented with a for loop. Well, it means that we can basically fit any sufficiently powerful model.

Conclusion

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