Ensembles

Tim Lawson

November 19, 2023

Bias and variance

A classification error may occur when:

- Feature vectors/per-class distributions *overlap*
- The model has *high bias* (is not expressive enough)
- The model has *high variance*

(Flach 2012, p. 338–9)

Ensembles

- Learn multiple models from different versions of the data
  - Resample the instances and/or features
  - Reweight the instances
- Aggregate the models' predictions
  - Average the scores or probabilities
  - Choose the majority prediction
- Or, learn multiple types of model

Bagging methods

A bagging method learns multiple instances of a model from random subsets of the data and aggregates the instances' predictions.

- *Pasting*: random subsets of the instances are sampled without replacement (Breiman 1999)
- *Bagging*: random subsets of the instances are sampled with replacement (Breiman 1996)
- *Random subspaces*: random subsets of the features are sampled (Ho 1998)
- *Random patches*: random subsets of the instances and features are sampled (Louppe and Geurts 2012)

(*1.11. Ensembles* n.d., sec. 1.11.3)

Boosting methods

A boosting method learns multiple instances of a model from weighted versions of the data and aggregates the instances' predictions.

- *AdaBoost*: weights are updated based on the error of the previous model (Freund and Schapire 1997)
- *Gradient-boosted trees*: weights are updated based on the gradient of the loss function, e.g., LightGBM (Ke et al. 2017), XGBoost (Chen and Guestrin 2016)

(*1.11. Ensembles* n.d., sec. 1.11.1)

See also, e.g., *arcing* ("adaptively resample and combine") and random forests (Breiman 1998; Breiman 2001).

Boosting methods

Code

```python
def get_model_weight(weighted_error: float) -> float:
    """Get the weight of a model."""
    raise NotImplementedError


def update_weight(
    weight: float,
    label: int,
    prediction: int,
    weighted_error: float,
) -> float:
    """Update the weight of an instance."""
    raise NotImplementedError
```

Multiple types of model

The predictions of different types of models can be aggregated by:

- *Voting* or *averaging* (soft voting)
- *Stacked generalization* (stacking): using the predictions as the feature of a meta-model
- *Meta-learning*: learn a model that predicts whether a model will perform well on a given task and data

References

1.11. Ensembles (n.d.). *1.11. Ensembles: Gradient Boosting, Random Forests, Bagging, Voting, Stacking*.

Breiman, Leo (1996). "Bagging Predictors". In: *Machine Learning* 24.2, pp. 123–140.

– (1998). "Arcing Classifier (with Discussion and a Rejoinder by the Author)". In: *The Annals of Statistics* 26.3, pp. 801–849.

– (1999). "Pasting Small Votes for Classification in Large Databases and On-Line". In: *Machine Learning* 36.1, pp. 85–103.

– (2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32.

Chen, Tianqi and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794.

Flach, Peter (2012). *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. 1st ed. Cambridge University Press.

Freund, Yoav and Robert E Schapire (1997). "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1, pp. 119–139.

Ho, Tin Kam (1998). "The Random Subspace Method for Constructing Decision Forests". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8, pp. 832–844.

References

Ke, Guolin et al. (2017). "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc.

Louppe, Gilles and Pierre Geurts (2012). "Ensembles on Random Patches". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Peter A. Flach, Tijl De Bie, and Nello Cristianini. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 346–361.

Boosting

Definition

- $\{x_i \mid i \in 1..n\},\ \vec{x}_i \in \mathbb{R}^{n_x}$ are instances
- $\vec{y}_i \in \{0,1\}^{n_y}$ is a label (one-hot vector)
- $f^{(t)} : \mathbb{R}^{n_x} \to \{0,1\}^{n_y}$ is a model
- $\vec{\hat{y}}_i^{(t)} = f^{(t)}(\vec{x}_i) \in \{0,1\}^{n_y}$ is a prediction (one-hot vector)
- $w_i^{(t)} \in \mathbb{R},\ w_i^{(0)} = \frac{1}{n},\ \sum_{i=1}^{n} w_i^{(t)} = 1$ is an instance weight
- $\epsilon^{(t)} = \sum_{i:\vec{\hat{y}}_i^{(t)} \neq \vec{y}_i} w_i^{(t)} \in \mathbb{R}$ is the weighted error of model $f^{(t)}$
- $\alpha^{(t)} = f_\alpha(\epsilon^{(t)}) \in \mathbb{R}$ is the weight of model $f^{(t)}$
- $w_i^{(t+1)} = f_w(w_i^{(t)},\ \vec{y}_i,\ \vec{\hat{y}}_i^{(t)},\ \epsilon^{(t)})$ is the updated instance weight
- $\vec{\hat{y}}_i = \sum_{t=1}^{T} \alpha^{(t)} f^{(t)}(x_i) \in \{0,1\}^{n_y}$ is the ensemble model prediction

Boosting

Derivation

- What should the weights of the models f_α/get_model_weights be?
- What should the weight updates f_w/update_weight be?

Assume that the weight updates f_w are:

$$w_i^{(t+1)} = \frac{w_i^{(t)}}{Z^{(t)}} \times \begin{cases} e^{-\alpha^{(t)}} & \text{if } \vec{\hat{y}}_i^{(t)} = \vec{y}_i \\ e^{\alpha^{(t)}} & \text{otherwise} \end{cases}$$

This can be simplified with:

$$\delta(\vec{y}_i, \vec{\hat{y}}_i^{(t)}) = \begin{cases} 1 & \text{if } \vec{\hat{y}}_i^{(t)} = \vec{y}_i \\ -1 & \text{otherwise} \end{cases}$$

Boosting

Derivation

The weight updates are:

$$w_i^{(t+1)} = w_i^{(t)} \frac{\exp(-\alpha^{(t)} \delta(\vec{y}_i, \hat{\vec{y}}_i^{(t)}))}{Z^{(t)}}$$

Each update is multiplicative:

$$w_i^{(T)} = w_i^{(0)} \prod_{t=1}^{T} \frac{\exp(-\alpha^{(t)} \delta(\vec{y}_i, \hat{\vec{y}}_i^{(t)}))}{Z^{(t)}} = \frac{1}{n} \frac{\exp(-\delta(\vec{y}_i, \hat{\vec{y}}_i^{(t)}))}{\prod_{t=1}^{T} Z^{(t)}}$$

Each set of instance weights sums to 1:

$$1 = \sum_{i=1}^{n} w_i^{(t)} = \sum_{i=1}^{n} \frac{1}{n} \frac{\exp(-\delta(\vec{y}_i, \hat{\vec{y}}_i^{(t)}))}{\prod_{t=1}^{T} Z^{(t)}}$$

$$\prod_{t=1}^{T} Z^{(t)} = \frac{1}{n} \sum_{i=1}^{n} \exp(-\delta(\vec{y}_i, \hat{\vec{y}}_i^{(t)}))$$

Boosting
Derivation

$\exp(-\delta(\vec{y}_i, \hat{\vec{y}}_i^{(t)})) \geq 1$ if x_i is misclassified by the ensemble, so $\prod_{t=1}^{T} Z^{(t)}$ is an upper bound on the ensemble error.

$\prod_{t=1}^{T} Z^{(t)}$ could be minimized by minimizing the model error $(n)Z^{(t)}$:

$$nZ^{(t)} = \sum_{i=1}^{n} w_i^{(t)} \exp(-\alpha^{(t)}\delta(\vec{y}_i, \hat{\vec{y}}_i^{(t)}))$$

By the definitions of $\epsilon^{(t)}$ and $\delta(\vec{y}_i, \hat{\vec{y}}_i^{(t)})$:

$$nZ^{(t)} = \epsilon^{(t)} \exp(\alpha^{(t)}) + (1 - \epsilon^{(t)}) \exp(-\alpha^{(t)})$$

Boosting

Derivation

Therefore, $Z^{(t)}$ is minimized when:

$$\frac{\partial Z^{(t)}}{\partial \alpha^{(t)}} = \epsilon^{(t)} \exp(\alpha^{(t)}) - (1 - \epsilon^{(t)}) \exp(-\alpha^{(t)}) = 0$$

$$\exp(2\alpha^{(t)}) = \frac{1 - \epsilon^{(t)}}{\epsilon^{(t)}}$$

That is:

$$\alpha^{(t)} = \frac{1}{2} \ln \left(\frac{1 - \epsilon^{(t)}}{\epsilon^{(t)}} \right), \quad Z^{(t)} = 2\sqrt{\epsilon^{(t)}(1 - \epsilon^{(t)})}$$