Ensemble Learning

As simple models may perform poorly, in ensemble learning we combine many models (each only a little better than random guessing) to get a strong meta-model.

Ensemble models work because good models agree on the same prediction, while (uncorrelated) bad models on different ones.

makes several models independently and averages their predictions to reduce overfitting:

- **Bagging** (short for “bootstrap aggregating”) generates $B$ random sets of examples $\{S_i : i = 1, \ldots, B\}$ of size $N$ by from the training data. For each $S_i$, train a decision tree $f_i$. Make a prediction:
  - for classification as the most frequent of the $B$ predictions
  - for regression as the average of the $B$ predictions, $\hat{y} = \hat{f}(x) = \frac{1}{B} \sum_{i=1}^{B} f_i(x)$

- A random forest is a bagging variant that strives for uncorrelated trees by selecting a random subset of for each tree. Hyperparameters include
  - the maximum $d$ of each tree
  - the $B$ of trees
  - the number of to include per subset

This is widely used and effective because it reduces overfitting of the ensemble model by under-emphasizing noise, outliers, and over- or under-represented examples in the data.

**Boosting** builds models to reduce underfitting:

- **Boosting** iteratively creates models such that model $(i + 1)$ is trained to correct model $i$’s errors by training examples to increase the weight of mis-classified examples and decrease the weight of correctly-classified examples. The final ensemble model combines all the models.
• For gradient boosting:

  – For regression:

    * Start with a constant model \( f = f_0(x) = \frac{1}{N} \sum_{i=1}^{N} y_i \).

    * Calculate \( e_i = y_i - f(x_i) \) for \( i = 1, \ldots, N \). Then train a new model \( f_1 \) with the original \( y \) values replaced by the \( e_i \). The boosted model is then \( f = f_0 + \alpha f_1 \), where hyperparameter \( \alpha \) is the learning rate.

    * Repeat by training \( f_2 \) on residuals with respect to \( f_1 \) and get the boosted model \( f = f_0 + \alpha f_1 + \alpha f_2 \), and so on, until making model \( f_M \), where \( M \) is the maximum number of trees.

Model \( i + 1 \) is trained to \( \frac{1}{N} \sum_{i=1}^{N} y_i \) of model \( i \). Recall that in gradient descent we move our parameter vector by step size \( \alpha \) opposite the direction of the gradient toward the value that minimizes an objective function. Gradient boosting uses \( e_i \) as a proxy for the gradient, as they show how to adjust the model to \( e_i \). Again \( \alpha \) limits the amount the model moves in one step. Burkov asserts the overall model \( f \) minimizes MSE.

Hyperparameters include:

* the number ______ of trees
* the learning rate ______
* the _______ of trees

Boosting reduces _______ (where bagging reduced overfitting); _______ the depth and number of trees can help boosting avoid overfitting.

  – There is also gradient boosting for classification and for other tasks.

Gradient boosting is one of the best ML algorithms, as it gives accurate models and can handle _______ data sets. It usually outperforms random forests but is slower to train.

---

1Burkov calls this residual “\( \hat{y}_i \),” which I think must be a typo since we usually call a prediction \( \hat{y} \).
Python

To learn more:

- Reference manual:

  - bagging:
    * regression:
    * classification:
  - random forest:
    * regression:
    * classification:
    * plot decision surface:
  - gradient boosting:
    * classification:
    * regression: