7 Problems and Solutions (Part 3 of 3)

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 \( N \) 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 $\hat{y}_i$. 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 residuals 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 $\hat{y}_i$ as a proxy for the gradient, as they show how to adjust the model to $\nabla L_f$. 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.

— For classification (in the binary case with $M$ regression decision trees):

As in logistic regression, use the sigmoid function $P(y = 1|x, f) = \frac{1}{1 + e^{-f(x)}}$, but now $f(x) = \sum_{m=1}^{M} f_m(x)$ where $f_m$ is a decision tree.

Choose $f$ to maximize log likelihood, $L_f = \sum_{i=1}^{N} \ln P(y_i = 1|x_i, f)$:

* Start with a constant model $f = f_0 = \frac{p}{1 - p}$ (odds of $y = 1$), where $p = \frac{1}{N} \sum_{i=1}^{N} y_i$.

* At each step $m$, add a new tree to the model. To find the best $f_m$, find for each $i = 1, \ldots, N$ the $\nabla L_f$: $g_i = \frac{\partial L_f}{\partial f(x_i)} = \frac{1}{e^f(x_i) + 1}$, where $f$ is the ensemble model built at step $m - 1$. Now $g = (g_1, \ldots, g_N)$ is the gradient $\nabla L_f(x)$.

* Transform training data by replacing label $y_i$ with ______ and build a new tree $f_m$.

* Find the optimal update step $\rho_m$ as $\rho_m \leftarrow \arg \max \rho L_f + \rho f_m$ and update the ensemble model by adding the new tree $f_m$ as $f \leftarrow f + \alpha \rho_m f_m$ (where $\alpha$ is the learning rate).

* Iterate until $m = M$ and return $f$.

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.

---

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

2 I added the $(x_i)$ to Burkov’s terse notation in the partial derivative denominator.
Python

To learn more:

- Reference manual:
  - bagging:
  - random forest:
    - plot decision surface: [https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_iris.html](https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_iris.html)
  - gradient boosting: