__/__/____

(27) Ensemble Methods

Idea: Combine multiple models to improve accuracy, stability and generalization.

- Base learners (individual models) can be same type (homogeneous) or different types (heterogeneous)
- Ensemble reduces variance, bias, both depending on mode

(Same Model Diff Data) 1. Bagging (Bootstrap Algorithm)

- Reduces variance of unstable methods (decision trees)
- Train multiple models on different bootstrapped samples of the dataset & aggregate predictions

Algorithm: 1. Given dataset D of size n
2. Repeat B times (no of base learners)
- Draw a bootstrap sample Dd (sample n points with replacement)
- Train base model Md on D_b.
3. Aggregate predictions { Regression / classification

(Easier to Implement) 2. Committee machines / Voting

- Combine predictions from multiple models without resamp
- The simplest ensemble is majority voting (classificatio

(Diff Model or same / Same/Full Data)

① Hard Voting: Each model votes for a class
Class with majority voting → final pred.

② Soft Voting: Models output probabilities
Avg probabilities: predict class with highest mean prob...

* Reduces variance if models are → uncorrelated.

__/__/____

3. Stacking (Stacked Generalization) (higher model)

- Combine multiple base learners using a meta-learner.
- Base learners learn patterns → meta learners learn how to best combine them.

(High model complexity)

1. Train base models $M_1, M_2 .. M_k$ on training data
2. Use base models to generate predictions on a hold-out dataset (or via cross-validation)
3. Train meta-model on these predictions as input-features.

- Can reduce both bias and variance.
- Requires carefull cross-val to avoid overfitting.

4. Boosting

- Combines multiple weak learners (models that perform slightly better than random guessing) into a single strong learner to achieve higher accuracy and predictive power.
- Boosting trains models sequentially.

1. Initial Model is trained on original dataset
2. Error Identifications: Model predictions are evaluated.
3. Weight Adjustment: Misclassified instances get higher weights while correctly classified instances receive lower weights in the subsequent iterations.
4. Sequential Training
5. Iteration and Combination of all weak learners

- Reduce Bias, variance by a bit.
- Improves Accuracy

__/__/____

(28) ADA BOOST (Adaptive Boosting)

Combines multiple weak learners (usually d.t stumps - shallow tr
to form a strong classifier.

- Each weak learner tries to fix the mistakes of the previous ones by giving more weight to misclassified samples.

Step:

① Initialize Weights: All samples start with equal weight

$$w_i = 1/N \quad i = 1, 2, \ldots N$$

② For each round $M = 1, 2 \ldots M$

Train weak classifier $h_m(x)$ using weighted data.

2.1 Compute weighted classification error:

$$\epsilon_m = \frac{\sum_{i=1}^{N} w_i \cdot I\left(y_i \neq h_m(x_i)\right)}{\sum_{i=1}^{N} w_i}$$

~ Proportion of total weights that is misclassified.

optimization

2.2 Compute learners importance weight:

$$\alpha_m = 1/2 \ln\left(1 - \epsilon_m / \epsilon_m\right)$$

- If error = 0.5, $\alpha = 0$ → weak learner ignored
- If error > 0.5, α = -ve → flip predictions

2.3 Update sample weights (based on classification errors)

$$w_i \leftarrow w_i \cdot e^{-\alpha_m y_i h_m(x_i)}$$

- If correctly classified? weight ↑s : weight ↓s

③ Final Model = $F(x) = \sum_{m=1}^{M} \alpha_m h_m(x)$, $\hat{y} = \text{sign}(F(x))$

weighted vote of all weak learners.

__/__/____

→ Loss function : implicitly minimizes exponential loss

$$L = \sum_{i=1}^{N} e^{-y_i F(x_i)} \qquad y_i \in \{-1, +1\}$$

$$F(x) = \sum_m \alpha_m h_m(x)$$

- penalizes misclassified points exponentially.

Disadvantages: 1. Sensitive to outliers
2. Sequential
3. Overfitting possible if too many weak learners

* Adaptive because : - Data reweighted after each iteration
- α changes adaptively

(29) Gradient Boosting (Boosting + Gradient Descent)

- Builds a sequence of weak learners (usually D.Ts)
- Instead of focusing on misclassified samples, it fits each new tree to the gradient of the loss function ie, the direction of the steepest descent to reduce error.

We start with a simple model $F_0(x)$ and then add trees

$$F_M(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x)$$

where $h_m(x)$ = new weak learner

γ_m = step size or weight for that learner.

__/__/____

Steps:

1. Initialize model, start with mean prediction

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_i L(y_i, \gamma)$$

for MSE loss $L(y, f(x)) = 1/2 (y - F(x))^2$:

$$f_0(x) = \text{mean}(y)$$

2. Compute residuals

Residual = negative gradient of loss w.r.t prediction

$$r_{im} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$$

$$r_{im} = y_i - F_{m-1}(x_i)$$

This is just the error b/w prediction and actual.

3. Fit a weak learner $h_m(x)$

Fit a regression tree to predict the residuals r_{im} from inputs x_i

4. Compute optimal multiplier γ_m

For each leaf of the tree, find the best constant multiplier γ_m that minimizes the loss

$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_i L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

5. Update model,

$$F_m(x) = F_{m-1}(x) + \eta \gamma_m h_m(x)$$

where η is the learning rate

6. Repeat for M iterations,

Keep fitting new trees to the current residuals.

* Each new tree nudges the model in the direction that most reduce loss

Just like Gradient Descent, but in function space.

__/__/____

(30) Random Forests

Problem with Bagging: Bagging reduces variance by averaging multiple trees trained on bootstrapped samples
- But if the trees are highly correlated, then averaging doesn't help much.

→ Random Forest which is an ensemble of decision trees where each tree is trained on a random subset of
- Data samples } Classification: majority vote
- Features. } Regression: Average of predictions

The Goal of Random Forests is to reduce correlation between trees while keeping each tree's variance high enough
→ This is achieved by adding feature randomness on top of data randomness.

Steps:
1. Bootstrap sampling (like Bagging)
2. Random feature selection: instead of considering all p features, consider only a random subset of size mtry
   Classification, mtry = $\sqrt{p}$
   Regression, mtry = $p/3$
3. Build each tree fully
4. Aggregate predictions.

→ Variance reduction.

Variance of the avg of T trees: $Var(\bar{y}) = \rho\sigma^2 + \frac{(1-\rho)\sigma^2}{T}$

If $\rho = 1$ (fully correlated) → no variance deduction
If $\rho = 0$ (independent trees) → variance ↓ drastically

→ Out of Bag (OOB) Error – used as validation for trees.