sTHEORETICAL ASSIGNMENT SOLUTION

True or False

(1) **True** (Pruning helps to prevent overfitting, which generally leads to better generalization on unseen data, thus reducing test error.)

(2) **False (**In k-fold cross-validation, you split the data into k subsets (folds). Each fold is used once as a test set while the remaining k-1 folds are used for training. This process is repeated k times, and the performance is averaged over the k folds.)

(3) **True (**Gradient Boosting builds models sequentially, where each new model is trained to correct the errors of the previous ones, effectively performing gradient descent in function space.)

(4) **True.** Random splitting ensures that both train and test sets are representative of the overall data distribution, reducing bias and providing a more accurate estimate of model performance.

(5) **False.** A classifier can overfit to the training data and perform perfectly on it (0% training error) but perform very poorly on unseen data, potentially resulting in a very high test error, including 100%

(6) **False.** Increasing regularization typically increases bias but reduces variance. Regularization constrains the model, making it simpler and less flexible, which often leads to higher bias.

(7) **False.** The ID3 algorithm aims to create a tree that fits the training data perfectly, which may not be the most compact tree. It does not inherently minimize the size of the tree.

(8) **False.** All classifiers make some assumptions about the data. These assumptions are what allow them to generalize from the training data to unseen data.

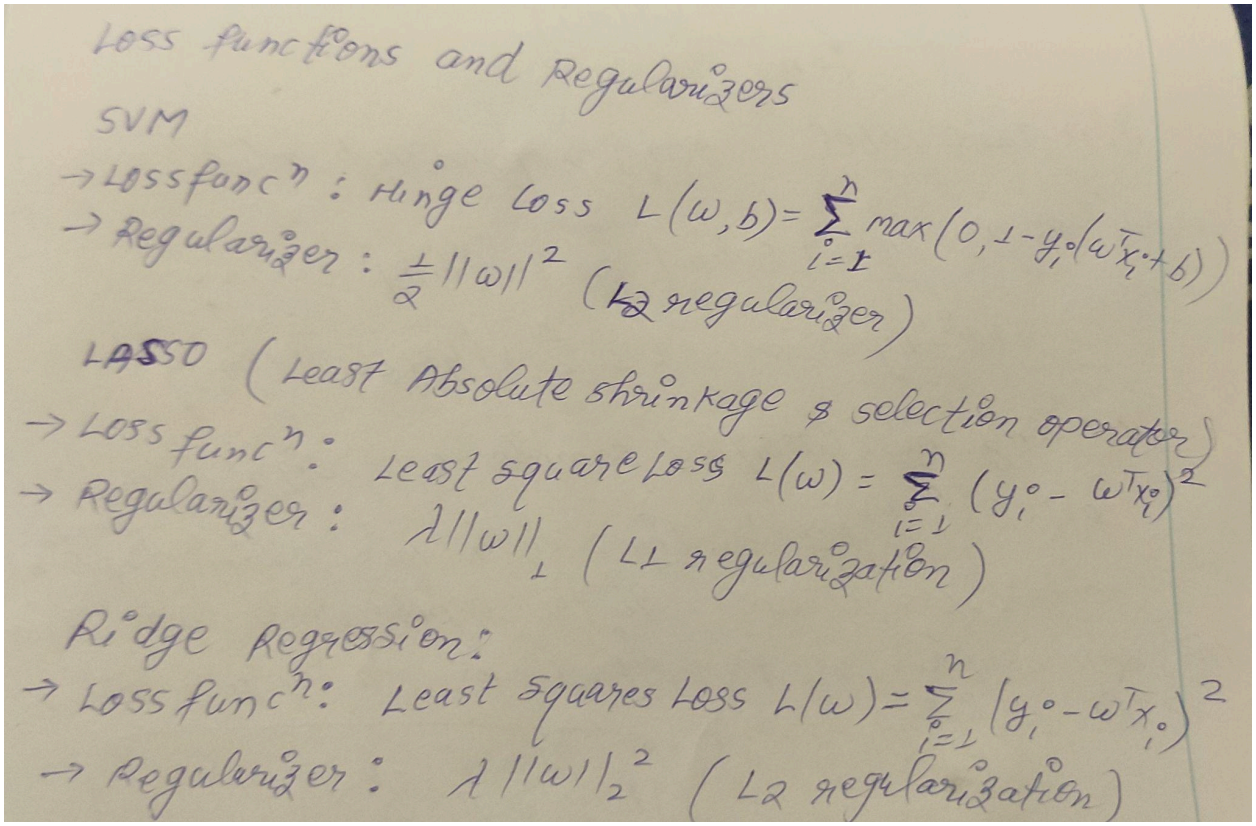
(9) **True.** Random Forests use Bagging (Bootstrap Aggregating) to train multiple decision trees on different subsets of the data and then average their predictions, reducing the variance.

(10) **True.** This is a known result in the theory of k-nearest neighbor classifiers, where the asymptotic error rate is bounded by twice the Bayes error.

(11) **False.** Squared loss regression trees typically have a time complexity of O(nlogn)O(n \log n)O(nlogn) per split due to the need to sort the data at each split.

(12) **False.** The Bayes optimal error represents the lowest possible error rate given the inherent noise in the data, not assuming noise-free conditions. It accounts for the irreducible error due to the stochastic nature of the problem.

1. Loss Functions and Regularizers



2. Matching the Loss to the Figure

Let's analyze each plot in the figure and match them with the loss functions:

- **A (Red dashed line):** Likely corresponds to the hinge loss (SVM) as it shows a linear increase after a certain point.
- **B (Cyan line with circles):** Could correspond to a quadratic loss, like the least squares loss used in Ridge Regression.
- **C (Black dashed line):** Appears similar to LASSO as it shows an absolute value type of behavior.
- **D (Green solid line):** This line fits an exponential decay which suggests a squared loss function without regularization.
- **E (Blue dotted line):** Could be a constant value, which does not typically correspond to standard loss functions in regression or SVM.

Matching the losses:

- **Hinge Loss (SVM):** A
- **Least Squares Loss (LASSO):** C
- **Least Squares Loss (Ridge Regression):** B

3. Minimizing Loss Functions with Gradient Descent and Newton's Method

Gradient Descent:

- **Least Squares Loss (LASSO and Ridge Regression):** These loss functions are differentiable and convex, making them suitable for minimization with gradient descent.
- **Hinge Loss (SVM):** Can be minimized with gradient descent as it is also convex.

Newton's Method:

- **Least Squares Loss (Ridge Regression):** The loss function is differentiable and has a well-defined second derivative (Hessian), making it suitable for Newton's method.
- **Hinge Loss (SVM):** Newton's method can be applied, but it's more complex due to the non-differentiability at the hinge point. However, techniques like the sub-gradient method can be used.
- **Least Squares Loss (LASSO):** The L1 regularizer introduces non-differentiability, making Newton's method less straightforward to apply.

(3)

1. High Training and Testing Errors with Limited Depth Decision Trees

- **High Bias:** Model is too simple (underfitting).
- **Inadequate Features/Data Quality:** Features are not predictive or data is noisy.

2. High Training and Testing Errors with Unlimited Depth Decision Trees

- **Overfitting:** Model captures noise, leading to high training and testing errors.
- **Poor Data/Features:** Features are insufficient or data quality is poor.

3. Bagging and Variance Reduction

- **Simulation:** Creates multiple models from different data subsets.
- **Variance Reduction:** Averages out errors from different models, stabilizing predictions.

4. Boosting with Limited Depth Decision Trees

- **Bias Reduction:** Decreases bias by combining weak learners.
- **Variance:** Initially low, but may increase with more iterations due to increased model complexity.

(4)

1. Speeding up kNN Classifier

- **Use Dimensionality Reduction:** Apply techniques like PCA to reduce the number of dimensions.
- **Use Approximate Nearest Neighbor (ANN) Search:** Implement algorithms like KD-trees or Locality-Sensitive Hashing (LSH).

2. Squared Distance in kNN

a) Accuracy Impact:

- **No Impact on Accuracy:** The order of distances (and thus nearest neighbors) remains the same whether using Euclidean or squared Euclidean distance.

b) Validity of Speed-Up Recommendation:

- **Still Valid:** Dimensionality reduction or ANN search will still speed up the classifier, as they reduce the computational complexity regardless of the distance metric used.

3. Curse of Dimensionality Concern

- **Yes, Be Worried:** High dimensionality (d=1,000,000) with relatively few data points (n=5000) can lead to overfitting and poor generalization due to the curse of dimensionality.

4. Neighborhood Size kkk Effects

- **Bias:** Larger kkk increases bias (smoother decision boundary, less sensitive to noise).
- **Variance:** Larger kkk decreases variance (more stable predictions).

5. kNN vs. Linear SVM

kNN:

- **Use Case:** When the decision boundary is non-linear and the training data is well-represented in the feature space.

Linear SVM:

- **Use Case:** When the decision boundary is approximately linear and the data has many irrelevant features or noise.

(5)

1. Prediction Value at a Leaf of a Regression Tree (with Squared-Loss Impurity)

- **Prediction Value:** The mean of the target values in that leaf.
- **Proof of Optimality:** Minimizing squared loss (mean squared error) is achieved by using the mean of the values in that subset because the mean minimizes the sum of squared deviations from each point.

2. Gini Index Maximization and Minimization for 3 Classes

- **Maximized:** When the classes are equally distributed (i.e., p1=p2=p3=1/3).
- **Minimized:** When all items belong to a single class (i.e., one pk=1 and the others are 0).

3. Decision Trees are "Myopic"

- **Explanation:** Decision trees make decisions based on the best split at the current node without considering the impact of this split on future nodes. This greedy approach can lead to suboptimal overall tree structures.

4. Methods to Prevent Overfitting in Decision Trees

- **Pruning:** Remove branches that have little importance after the tree is fully grown.
- **Setting Minimum Samples:** Require a minimum number of samples in a leaf node or for a split.

(6)

1. Training a Random Forest without Validation or Test Data

- **False.**
- **Justification:**
    - **Model Evaluation:** Random forests use bagging (bootstrap aggregating) which involves creating multiple training sets via resampling. However, to evaluate the model's performance and select hyperparameters, it's essential to have a separate validation set to ensure the model generalizes well and avoids

overfitting.

Adaboost Loss function

→ Adaboost minimizes the exponential loss function

$$L = \sum_{i=n}^{n} e^{(-y_i \cdot f(x_i))}$$

y_i → True Label $f(x_i)$ ⇒ predicted value

→ proof as an upper bound on Training Error:

Training Error $E = \sum_{i=1}^{n} 1(y_i \neq \text{sign}(f(x_i)))$

→ For any misclassified sample $y_i \neq \text{sign}(f(x_i))$

$e^{(-y_i \cdot (f(x_i)))} \geq 1$. Thus each misclassified sample points contributes atleast 1 to the exponential loss. Therefore, the exponential loss function L is an upper bound on the training error E:

$$E \leq \sum_{i=1}^{n} e^{(-y_i f(x_i))}$$