Lecture 1: Bias-Variance Trade-off, Cross-Validation & Bootstrap

This lecture introduces the fundamental concepts of model performance, including the trade-off between bias and variance, and methods for assessing and improving model generalization, such as cross-validation and the bootstrap.

Existing Exam Questions and Answers

Q: Discuss the Bias-Variance trade-off. How does it differ between LOOCV and 10-fold cross-validation?
A: The bias-variance trade-off is a central concept in statistical learning that involves minimizing two sources of error that prevent models from generalizing well to unseen data.

- **Bias** refers to the error introduced by approximating a real-life problem, which may be very complicated, by a much simpler model. High-bias models are typically too simple and may underfit the data, failing to capture the true relationship between predictors and the response.
- **Variance** refers to the amount by which the model's learned function would change if we estimated it using a different training dataset. High-variance models are overly complex and can overfit the data by learning the noise in the training set.

The total expected error of a model can be decomposed into the sum of its squared bias, its variance, and the irreducible error:

$$\mathrm{Err}(x) = \mathrm{Var}(\hat{f}(x_0)) + \mathrm{Bias}(\hat{f}(x_0))^2 + \mathrm{Var}(\epsilon)$$

More flexible models tend to have low bias but high variance, while less flexible models have high bias and low variance. The trade-off lies in finding an optimal model complexity that minimizes the total error.

Regarding cross-validation methods:

- **Leave-One-Out Cross-Validation (LOOCV)** has less bias than k-fold CV because it uses almost the entire dataset (n-1 observations) for training in each fold, resulting in a model whose performance is a nearly unbiased estimate of the test error. However, LOOCV has higher variance. Because the training sets in each fold are nearly identical, the outputs of each fold are highly correlated, which inflates the variance of the final performance estimate.
- **10-fold Cross-Validation** (or k-fold with k ¡ n) has slightly more bias than LOOCV because each training set is smaller than the full dataset. However, it has lower variance because the training folds are more distinct from one another, leading to less correlated outputs across folds. This often makes 5-fold or 10-fold CV a preferred method in practice as it balances the bias-variance trade-off for estimating test error.

Q: What is bootstrapping, how does it work, and when would you use it instead of CV?
A: Bootstrapping is a resampling technique where repeated samples of the same size are drawn with replacement from the original dataset.

How it works:

1. From a dataset of size n, a new training dataset of size n is created by randomly selecting observations with replacement. This means some observations may appear multiple times, while others may not be selected at all.
2. The observations not selected for the training set (the "out-of-bag" samples) are used as a test set.
3. A model is trained on the bootstrapped training set, and its performance is evaluated on the out-of-bag test set.
4. This process is repeated many times (e.g., hundreds or thousands of "folds") to get a distribution of the model's performance.
5. The final performance estimate is the average of the results from all the folds.

When to use it instead of Cross-Validation (CV): While both bootstrapping and CV are used for model assessment, bootstrapping is most commonly used to assess the uncertainty of an estimator, such as calculating standard errors or confidence intervals for model coefficients. In contrast, CV is primarily used for model assessment (estimating test error) and model selection (choosing the right level of flexibility).

Bootstrapping can provide more accurate measures of both the bias and variance of an estimator compared to basic CV, which makes it particularly useful when the underlying statistical distributions are unknown or violated.

Q: What effects do you see when you cross-validate? Discuss in terms of bias-variance tradeoff. How does it differ in LOOCV or in 10-fold cross-validation?
A: Cross-validation helps in model assessment and model selection by providing a more accurate estimate of a model's performance on unseen data than the training error would suggest. The main effect is that it allows us to choose a model with the right level of flexibility, avoiding overfitting.

In terms of the bias-variance trade-off, cross-validation helps identify the point where increasing model flexibility (and thus decreasing bias) starts to significantly increase variance, leading to a higher total test error. By plotting the cross-validated test error against model complexity, we can find the optimal model that balances this trade-off.

The differences between LOOCV and 10-fold CV in terms of the bias-variance trade-off are:

- **LOOCV:** Provides a nearly unbiased estimate of the test error because each training set is almost the size of the entire dataset. However, it has high variance because the training sets are highly correlated, leading to less stable estimates.
- **10-fold CV:** Has a slight increase in bias compared to LOOCV since the models are trained on smaller subsets of the data. In return, it achieves a lower variance because the training folds are more independent, leading to a more stable and reliable estimate of the test error.

New Exam Questions and Answers

Q: Describe three ways of deciding between a linear or non-linear model. For one of them, discuss the bias-variance trade-off.
A: Here are three methods to decide between a linear and a non-linear model:

1. **Visual Inspection of Data:** The first step in any analysis is to visualize the data. Creating scatter plots of the response variable against each predictor can reveal non-linear patterns. If a scatter plot shows a clear curve, a non-linear model might be more appropriate.

2. **Analysis of Residuals:** After fitting a linear model, you should examine the residuals (the difference between the observed and predicted values). A plot of residuals versus fitted values should show no discernible pattern. If there's a U-shape or other systematic pattern, it suggests that the linear model is not capturing the full relationship, and a non-linear model may be better.

3. **Cross-Validation:** You can fit both a linear and a non-linear model (e.g., a polynomial or a spline) and use cross-validation to estimate the test error for each. The model with the lower cross-validated test error is generally preferred. This approach directly addresses the bias-variance trade-off.

Bias-Variance Discussion for Cross-Validation: A simple linear model has high bias but low variance. A more flexible, non-linear model has lower bias but higher variance. When we use cross-validation, we are trying to estimate the total test error, which is a combination of bias and variance. If the true relationship is non-linear, the linear model's high bias will lead to a high test error. A non-linear model will better capture the true pattern (lower bias), but if it's too flexible, it might overfit (high variance), also leading to a high test error. Cross-validation helps us find the sweet spot where the reduction in bias from using a non-linear model outweighs the increase in variance, resulting in the lowest overall test error.

Lectures 2 & 3: Variable Selection, Regularization, and High Dimensionality

These lectures focus on the challenges of working with many predictors, especially when the number of predictors (p) is close to or greater than the number of observations (n). Methods like subset selection, regularization (Ridge and Lasso), and dimension reduction (PCR) are discussed as solutions.

Existing Exam Questions and Answers

Q: Discuss Lasso, Ridge, and PCR. When to use each, and what are the differences between them?
A: Ridge Regression, Lasso, and Principal Component Regression (PCR) are three methods used to handle issues like multicollinearity and high dimensionality in linear models.

- **Ridge Regression:**
  - **What it is:** Ridge regression is a shrinkage method that adds a penalty term to the RSS, proportional to the sum of the squares of the coefficients ($\lambda \sum_{j=1}^{p} \beta_j^2$).
  - **Effect:** This penalty shrinks the coefficients towards zero but never sets them exactly to zero. It is effective at reducing the variance of the model, especially when predictors are highly correlated.
  - **When to use:** Use Ridge when you believe most of your predictors are relevant to the response, and you want to reduce the impact of multicollinearity.
- **Lasso (Least Absolute Shrinkage and Selection Operator):**
  - **What it is:** Lasso is another shrinkage method that uses a different penalty, proportional to the sum of the absolute values of the coefficients ($\lambda \sum_{j=1}^{p} |\beta_j|$).
  - **Effect:** The key difference from Ridge is that the Lasso penalty can shrink some coefficients to exactly zero, effectively performing variable selection. This results in a simpler, more interpretable model.
  - **When to use:** Use Lasso when you suspect that many of your predictors are irrelevant or redundant, and you want to produce a more parsimonious model.
- **Principal Component Regression (PCR):**
  - **What it is:** PCR is a dimension reduction technique. It first identifies a smaller set of new variables, called principal components, that are linear combinations of the original predictors and capture the most variance in the data. Then, it fits a linear regression model using these components as predictors.
  - **Effect:** By reducing the number of predictors, PCR can reduce the variance of the model. The number of components to use is typically chosen via cross-validation.
  - **When to use:** PCR is useful when you have a large number of correlated predictors and want to reduce the dimensionality of the problem before fitting a model. It works best when the directions of high variance in the predictors are also associated with the response.

Differences Summary:

- **Variable Selection:** Lasso performs automatic variable selection; Ridge and PCR do not (they keep all predictors in the model in some form).
- **Coefficients:** Ridge shrinks coefficients but doesn't zero them out. Lasso can set coefficients to zero. PCR's coefficients are for the principal components, not the original variables directly.
- **Mechanism:** Ridge and Lasso are shrinkage methods that penalize large coefficients. PCR is a two-step dimension reduction method.

Q: What are three considerations you should have when analyzing a dataset with many predictors?
A: When analyzing high-dimensional datasets (where the number of predictors p is large), you should consider the following:

1. **The Curse of Dimensionality:** High dimensionality poses significant challenges. As p increases, the data becomes sparse, and the distance between observations tends to be large. This makes it difficult to make accurate predictions. Furthermore, the risk of overfitting increases dramatically, as a complex model can find spurious patterns in the noise of the training data. When p ¿ n, ordinary least squares regression cannot be used as it has no unique solution.

2. **Collinearity and Interpretation:** In high-dimensional datasets, it's very likely that many predictors are correlated with each other (collinearity). This makes it difficult to disentangle the individual effect of each predictor on the response. A model might select one predictor from a group of correlated ones, but another, equally good model might select a different predictor from the same group. Therefore, you should not overstate the importance of the specific variables selected by a model, as they may not be the "true" or unique predictors.

3. **Appropriate Modeling Techniques:** Standard linear regression is often inappropriate for high-dimensional data. You must use methods specifically designed for this situation. These include:

   - **Subset Selection Methods:** Forward or backward stepwise selection can be used, but they can be computationally intensive and may miss the optimal model.
   - **Regularization/Shrinkage Methods:** Ridge and Lasso are well-suited for high dimensions. They can handle p ¿ n situations and help control variance and overfitting.
   - **Dimension Reduction Methods:** PCR and PLS reduce the number of predictors to a smaller set of components, making the problem more manageable.

New Exam Questions and Answers

Q: Explain why standardization of predictors is important for Ridge and Lasso regression, but not for a single decision tree.
A: Standardization is crucial for Ridge and Lasso because these methods apply a penalty to the size of the coefficients. The penalty term for Ridge is

$$\lambda \sum \beta_j^2$$

and for Lasso is

$$\lambda \sum |\beta_j|$$

. If the predictors are on different scales (e.g., one measured in kilograms and another in meters), the predictor with a larger scale will naturally have a smaller coefficient, and the penalty will unfairly punish predictors with smaller scales. To ensure that the penalty is applied fairly to all predictors, they should be standardized to have a similar scale (e.g., by making them have a standard deviation of 1) before fitting the model.

For a single decision tree, standardization is not necessary. Decision trees work by recursively splitting the data based on the values of the predictors to minimize a cost function like RSS or the Gini index. The splits are chosen based on threshold values (e.g., Age ¡ 50). The choice of the best split point for a predictor is independent of the scale of other predictors. The tree algorithm considers each predictor one at a time, so their relative scales do not influence the tree-building process.

Lecture 4: Beyond Linearity (Splines, GAMs)

This lecture explores methods for modeling non-linear relationships, moving beyond simple linear models to more flexible approaches like splines and Generalized Additive Models (GAMs).

Existing Exam Questions and Answers

Q: Define smoothing splines and cubic splines. How do you control their degrees of freedom?
A:

- **Cubic Spline:** A cubic spline is a function created by fitting piecewise cubic polynomials to different regions of the data, which are separated by points called "knots". To ensure the resulting function is smooth, the pieces are connected such that the function itself, as well as its first and second derivatives, are continuous at the knots. A cubic spline with K knots has K+4 degrees of freedom.

- **Smoothing Spline:** A smoothing spline is a function g(x) that aims to fit the data well without being too "wiggly" or rough. It does this by minimizing a loss function that balances the Residual Sum of Squares (RSS) with a penalty for roughness. The formula is:

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

  The term

$$\int g''(t)^2 dt$$

  is a roughness penalty that is large for functions that change direction rapidly. The result is a natural cubic spline with knots at every unique data point.

Controlling Degrees of Freedom:

- **For Cubic Splines (and Regression Splines in general):** The flexibility, or degrees of freedom, is controlled by the number and placement of the knots. More knots lead to a more flexible function and more degrees of freedom. The number of knots can be chosen using cross-validation.

- **For Smoothing Splines:** The degrees of freedom are controlled by the smoothing parameter, λ.
  - When λ is 0, the spline has n degrees of freedom and perfectly interpolates the data (high variance).
  - As λ approaches infinity, the spline becomes a straight line with 2 degrees of freedom (high bias). The effective degrees of freedom for a given λ can be calculated, and λ is typically chosen using cross-validation, often with an efficient Leave-One-Out (LOOCV) shortcut.

Q: When would you use a GAM instead of a GLM? What are the formulations of GAM in comparison to GLM?
A: You would use a Generalized Additive Model (GAM) instead of a Generalized Linear Model (GLM) when you suspect that the relationship between some of your predictors and the response is non-linear. A GAM allows you to fit a flexible, non-linear function to each predictor while maintaining the additive structure of the model, which can lead to a better fit (lower bias) without sacrificing too much interpretability.

Formulations:

- **GLM Formulation:** A GLM generalizes the linear model by allowing the response to have a distribution other than normal and by using a link function g() to connect the mean of the response to the linear predictor. The formulation is:

  $$g(E(y_i)) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$$

  This model is linear in its parameters.

- **GAM Formulation:** A GAM extends the GLM by replacing each linear term $\beta_j x_{ij}$ with a smooth, non-linear function $f_j(x_{ij})$. The formulation is:

  $$g(E(y_i)) = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip})$$

  Each function f_j can be a spline, a local regression, or any other smoothing function. This allows the model to capture complex, non-linear relationships while still being additive, meaning you can examine the effect of each predictor individually.

New Exam Questions and Answers

Q: What is the main problem with using high-degree polynomials for fitting non-linear data, and how do splines solve this issue?
A: The main problem with using a single, high-degree polynomial to fit non-linear data is that it creates a **global fit**. This means that the shape of the fitted curve in one region of the data is affected by the data in other, more remote regions. This often leads to erratic and wild behavior, especially near the boundaries (tails) of the data, where the function can "flap about madly". This high flexibility in the tails increases the variance of the model and can lead to very poor predictions.

Splines solve this problem by using a **piecewise approach**. Instead of fitting one global function, splines fit separate, low-degree polynomials (typically cubic) in different regions of the data. These polynomial segments are then joined smoothly at points called "knots". This has two key advantages:

1. **Local Fit:** The behavior of the spline in one region does not affect its behavior in distant regions. This allows the function to adapt to the local properties of the data, providing a much more stable and accurate fit.
2. **Controlled Flexibility:** By adding knots, we can increase the flexibility of the function only where it is needed, without introducing the wild oscillations that plague high-degree polynomials. Furthermore, methods like natural splines add extra constraints to force the function to be linear in the tails, which further reduces variance and improves stability.

Lecture 5: Trees, Bagging, and Boosting

This final lecture covers tree-based methods and powerful ensemble techniques like Bagging, Random Forests, and Boosting, which combine multiple models to achieve better predictive performance.

Existing Exam Questions and Answers

Q: What is bagging and its algorithm? What is boosting? Why does Random Forest often outperform bagging?
A:

- **Bagging (Bootstrap Aggregating):**
  - **What it is:** Bagging is an ensemble method that aims to reduce the variance of a statistical learning model, particularly unstable ones like decision trees.
  - **Algorithm:**
    1. Create B new training datasets by sampling with replacement (bootstrapping) from the original dataset.
    2. Train a separate model (e.g., a decision tree) on each of the B bootstrapped datasets.
    3. To make a prediction for a new observation, average the predictions from all B models (for regression) or take a majority vote (for classification).
- **Boosting:**
  - **What it is:** Boosting is another ensemble method that builds a strong model by sequentially combining many "weak" learners. Unlike bagging, boosting is a progressive learning process.
  - **How it works:** It learns slowly. The first model is fit to the original data. Subsequent models are fit to the residuals (the errors) of the previous model. Each new tree focuses on the data points that the previous trees struggled to predict correctly. The final model is an additive combination of all the trees, often with a shrinkage parameter to control the learning rate.

Why Random Forest Outperforms Bagging: Random Forests improve on bagged trees by introducing an element of randomness that decorrelates the trees. In bagging, if there is one very strong predictor, it will likely be chosen as the top split in most of the bagged trees. This makes the trees highly correlated, and averaging highly correlated quantities does not lead to a large reduction in variance. Random Forests solve this by, at each split in each tree, considering only a random subset of predictors as candidates for the split. This forces the trees to use a more diverse set of predictors, making them less correlated with each other. Averaging many uncorrelated models leads to a more significant reduction in variance, which is why Random Forests often achieve better predictive performance than bagging.

Q: Talk about bagging and boosting, what are they, and compare them with GAM in terms of flexibility.
A:

- **Bagging** is an ensemble method that creates multiple independent models on bootstrapped samples of the data and averages their predictions to reduce variance.
- **Boosting** is a sequential ensemble method where each new model is trained to correct the errors of the previous ones, building a strong predictor from many weak ones.

Comparison with GAM in terms of Flexibility:

- **GAMs:** A GAM offers flexibility by modeling non-linear relationships for each predictor, but it is fundamentally **additive**. This means it does not inherently model interactions between variables unless they are explicitly added to the model. Its flexibility is high in capturing non-linear effects but limited in capturing complex interactions.
- **Bagging and Boosting (with Trees):** These methods, especially when used with decision trees, are highly flexible and can naturally capture complex, high-order interactions between predictors. The tree structure itself is a way of modeling interactions (e.g., the effect of Age might depend on Gender). Because they can model these interactions automatically, they are generally considered **more flexible** than GAMs. However, this high flexibility comes at the cost of interpretability; it is much harder to understand the relationships in a boosted model or a random forest than in a GAM.

Q: Rank (lasso, gam, linear model, random forest) by sensitivity to outliers.
A: Ranking models by their sensitivity to outliers involves thinking about how much a single extreme data point can influence the overall fit.

1. **Linear Model (Most Sensitive):** A standard linear model fit by least squares is highly sensitive to outliers. Because it minimizes the sum of squared errors, a large outlier creates a very large squared error, which will pull the entire regression line towards it.
2. **GAM:** A GAM is also quite sensitive, as it is often based on local least squares fits (like LOESS) or penalized least squares (like splines). An outlier can still distort the smooth function in its local neighborhood.
3. **Lasso:** Lasso regression is generally more robust to outliers than a standard linear model. While it still uses a least squares loss function, the L1 penalty shrinks coefficients. This shrinkage can dampen the effect of an outlier, preventing it from having an overly large influence on the coefficients of other variables.
4. **Random Forest (Least Sensitive):** Random Forests are highly robust to outliers. They are based on decision trees, which partition the data based on splits, not on minimizing a squared distance. An outlier might end up in its own leaf, but it will have little to no effect on the overall structure of the splits in the rest of the tree. When hundreds of such trees are averaged, the effect of any single outlier is almost completely washed out.

Rank (Most to Least Sensitive):

1. Linear Model
2. GAM
3. Lasso
4. Random Forest

New Exam Questions and Answers

Q: Explain the pruning process for a decision tree and why it is generally preferred over stopping the tree growth early.

A: Pruning is the process of reducing the size of a large, complex decision tree by removing branches that provide little predictive power. The standard approach is called **cost complexity pruning**.

How it works:

1. First, a very large tree (T_0) is grown on the training data, often until each terminal node has very few observations.
2. Then, we consider a sequence of pruned subtrees, obtained by removing branches. This process is guided by a tuning parameter, α, which balances the RSS of the tree against its complexity (the number of terminal nodes, $|T|$). The goal is to minimize:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

3. For each value of α, there is a best subtree. Cross-validation is then used to find the optimal value of α, and thus the best pruned tree.

Why Pruning is Preferred over Stopping Early: Stopping the tree growth early (a "greedy" approach) is often sub-optimal. A seemingly weak split early on (one that doesn't cause a large drop in RSS) might be followed by a very good split later down the branch. By stopping the growth too soon, we might miss these more valuable, deeper interactions in the data.

The "grow then prune" strategy allows the tree to explore these deeper structures first and then intelligently decide which branches are not contributing enough to justify their complexity. This makes it more likely to find a tree with better predictive accuracy than one found by simply stopping the growth based on a simple criterion.