MATH 342W / 642 / RM 742 Spring 2024 HW #5

Professor Adam Kapelner

Sunday 19th May, 2024

Problem 1

These are some questions related to probability estimation modeling and asymmetric cost modeling.

(a) [easy] Why is logistic regression an example of a "generalized linear model" (glm)?

Logistic regression is considered a generalized linear model (GLM) because it combines a linear predictor with a specific link function to model a binomial response variable.

(b) [easy] What is $\mathcal{H}_{pr}$ for the probability estimation algorithm that employs the linear model in the covariates with logistic link function?

The candidate space H for probablity estimation includes all logistic regression models formed from different sets of parameters.

(c) [easy] If logistic regression predicts 3.1415 for a new $\boldsymbol{x}_*$, what is the probability estimate that $y = 1$ for this $\boldsymbol{x}_*$?

In logistic regression, the prediction of a value such as 3.1415 for a new input $\mathbf{x}_*$ corresponds to the linear predictor η. To estimate the probability that $y = 1$ for this input, we need to apply the logistic (sigmoid) function to this linear predictor value. The logistic function is defined as:

$$\sigma(\eta) = \frac{1}{1 + \exp{(() - \eta)}}$$

Given $\eta = 3.1415$, we can calculate the probability estimate as follows:

$$P(y = 1 \mid \mathbf{x}_*) = \sigma(3.1415) = \frac{1}{1 + \exp{(() - 3.1415)}}$$

We compute:

$$\sigma(3.1415) = \frac{1}{1 + \exp{(() - 3.1415)}} \approx \frac{1}{1 + \exp{(() - 3.1415)}} \approx \frac{1}{1 + 0.0432} \approx \frac{1}{1.0432} \approx 0.9586$$

So, the probability estimate that $y = 1$ for $\mathbf{x}_*$ is approximately 0.9586.

(d) [harder] What is $\mathcal{H}_{pr}$ for the probability estimation algorithm that employs the linear model in the covariates with cloglog link function?

The hypothesis space $\mathcal{H}_{pr}$ for a probability estimation algorithm using a linear model in the covariates with a complementary log-log (cloglog) link function consists of all possible models that map input features to probabilities using this link function applied to a linear combination of the input features. The complementary log-log link function is defined as:

$$g(\mu) = \log(-\log(1-\mu))$$

where μ is the probability of the binary outcome.

For a linear predictor η, the relationship between η and the probability μ using the cloglog link function is:

$$\eta = \log(-\log(1-\mu))$$

Solving for μ, we get:

$$\mu = 1 - \exp\left(() - \exp\left(() \, \eta\right)\right)$$

Therefore, the hypothesis space $\mathcal{H}_{pr}$ can be defined as:

$$\mathcal{H}_{pr} = \left\{ h_\beta(\mathbf{x}) \mid h_\beta(\mathbf{x}) = 1 - \exp\left()\left(-\exp\left()\left(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p\right)\right), \ \beta \in \mathbb{R}^{p+1} \right\}$$

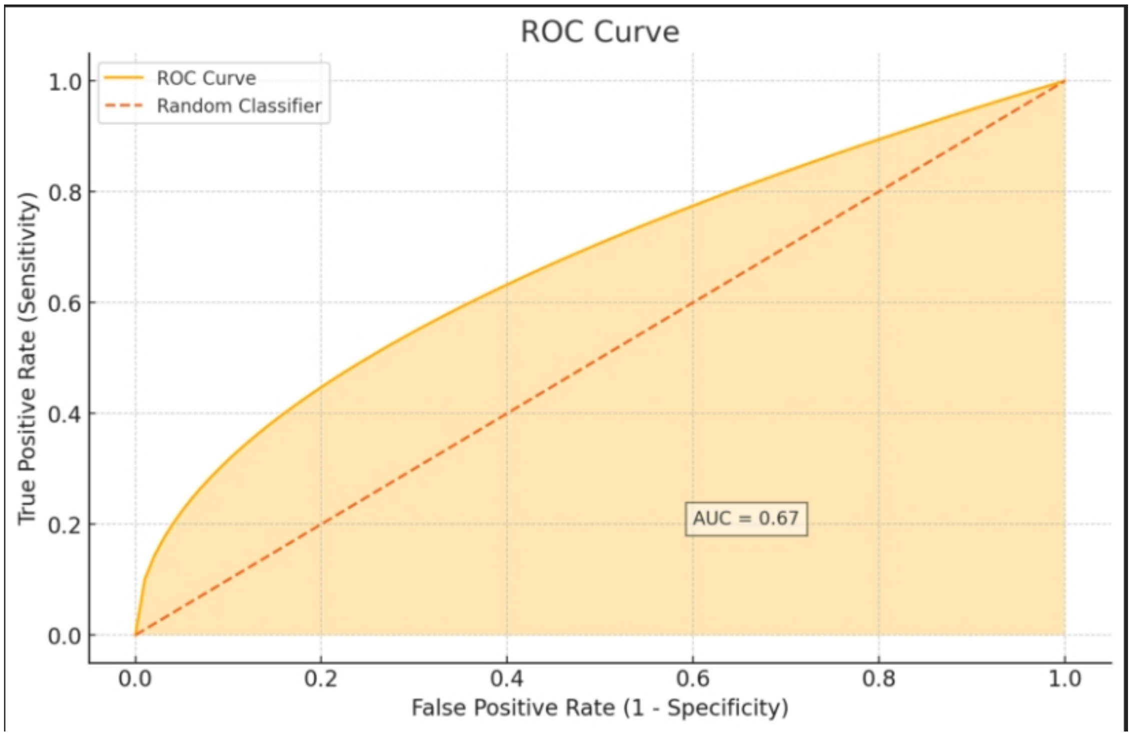
where: - $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ is the vector of input features, - $\beta = (\beta_0, \beta_1, \beta_2, \ldots, \beta_p)$ is the vector of parameters (coefficients) including the intercept term β_0.

In this function space, each function $h_\beta(\mathbf{x})$ represents a model that maps the input features $\mathbf{x}$ to a probability using the cloglog link function applied to the linear predictor.

(e) [difficult] Generalize linear probability estimation to the case where $\mathcal{Y} = \{C_1, C_2, C_3\}$. Use the logistic link function like in logistic regression. Write down the objective function that you would numerically maximize. This objective function is one that is argmax'd over the parameters (you define what these parameters are — that is part of the question).

Once you get the answer you can see how this easily goes to $K > 3$ response categories. The algorithm for general K is known as "multinomial logistic regression", "polytomous LR", "multiclass LR", "softmax regression", "multinomial logit" (mlogit), the "maximum entropy" (MaxEnt) classifier, and the "conditional maximum entropy model". You can inflate your resume with lots of jazz by doing this one question!

(f) [easy] Graph a canonical ROC and label the axes. In your drawing estimate AUC. Explain very clearly what is measured by the x axis and the y axis.



A model I would employ from the curve would be (.8, .4). It would be for a model whose true positive are really important like MRI/CT scan, due to the importance of making a firm and valid diagnosis for someone.

(g) [easy] Pick one point on your ROC curve from the previous question. Explain a situation why you would employ this model.

(h) [harder] Graph a canonical DET curve and label the axes. Explain very clearly what is measured by the x axis and the y axis. Make sure the DET curve's intersections with the axes is correct.

(i) [easy] Pick one point on your DET curve from the previous question. Explain a situation why you would employ this model.

(j) [difficult] [MA] The line of random guessing on the ROC curve is the diagonal line with slope one extending from the origin. What is the corresponding line of random guessing in the DET curve? This is not easy...

Problem 2

These are some questions related to bias-variance decomposition. Assume the two assumptions from the notes about the random variable model that produces the δ values, the error due to ignorance.

(a) [easy] Write down (do not derive) the decomposition of MSE for a given $\boldsymbol{x}_*$ where $\mathbb{D}$ is assumed fixed but the response associated with $\boldsymbol{x}_*$ is assumed random.

The decomposition we discussed in class for MSE for a given g_* is the following.

$$E_x[Bias[g(\bar(x)]^2 + Var[g(\bar(x)]^2] + \sigma^2$$

(b) [easy] Write down (do not derive) the decomposition of MSE for a given $\boldsymbol{x}_*$ where the responses in $\mathbb{D}$ is random but the $\boldsymbol{X}$ matrix is assumed fixed and the response associated with $\boldsymbol{x}_*$ is assumed random like previously.

(c) [easy] Write down (do not derive) the decomposition of MSE for general predictions of a phenomenon where all quantities are considered random.

$$\mathrm{E}\left[(Y-\hat{f}(X))^2\right] = \underbrace{\left(\mathbb{E}[\hat{f}(X)] - f(X)\right)^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}\left[\left(\hat{f}(X) - \mathbb{E}[\hat{f}(X)]\right)^2\right]}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Irreducible Error}}$$

(d) [difficult] Why is it in (a) there is only a "bias" but no "variance" term? Why did the additional source of randomness in (b) spawn the variance term, a new source of error?

(e) [harder] A high bias / low variance algorithm is underfit or overfit?

A high bias / low variance algorithm is underfit. When an algorithm is underfit, it fails to capture the complexity of the data, resulting in poor performance on both the training set and the test set. This is characteristic of high bias, where the model does not fit the training data well and consequently generalizes poorly to new data.

(f) [harder] A low bias / high variance algorithm is underfit or overfit?

A low bias / high variance algorithm is overfit. When an algorithm is overfit, it performs exceptionally well on the training data but poorly on test data because it has learned not only the underlying patterns but also the noise and specific peculiarities of the training data. This results in poor generalization to new data, which is a characteristic of high variance.

(g) [harder] Explain why bagging reduces MSE for "free" regardless of the algorithm employed.

Bagging (Bootstrap Aggregating) reduces the Mean Squared Error (MSE) by decreasing the variance of the model's predictions. It does this by training multiple versions of the model on different bootstrap samples of the data and then averaging their predictions. This averaging process smooths out the predictions and reduces the overall variability, leading to more stable and reliable outcomes. While bagging doesn't significantly affect the bias of the base model, the reduction in variance contributes to a lower overall MSE, making this technique effective regardless of the algorithm used.

(h) [harder] Explain why RF reduces MSE atop bagging M trees and specifically mention the target that it attacks in the MSE decomposition formula and why it's able to reduce that target.

Random Forests reduce the Mean Squared Error (MSE) by targeting the variance component. They achieve this by using bagging, which reduces variance through averaging multiple models, and by further introducing randomness in the feature selection process at each split. This reduces the correlation between trees, leading to less aligned errors and a lower overall variance. The bias remains largely unchanged, but the overall MSE is reduced due to the decreased variance.

(i) [difficult] When can RF lose to bagging M trees? Hint: think hyperparameter choice.

Random Forests can underperform compared to bagging M trees if the hyperparameter controlling the number of features selected at each split is not appropriately set. Too few features lead to weak trees, while too many features reduce the benefit of decorrelation. Proper tuning of this hyperparameter is crucial to harness the full potential of Random Forests

Problem 3

These are some questions related to missingness.

(a) [easy] [MA] What are the three missing data mechanisms? Provide an example when each occurs (i.e., a real world situation). We didn't really cover this in class so I'm making it a MA question only. This concept will NOT be on the exam.

(b) [easy] Why is listwise-deletion a *terrible* idea to employ in your $\mathbb{D}$ when doing supervised learning?

Listwise deletion is a terrible idea in supervised learning because it leads to a loss of valuable information, introduces bias if the missingness is not completely random, wastes resources, and can result in reduced model performance and training instability. More sophisticated methods for handling missing data, such as imputation or model-based approaches, are generally preferred.

(c) [easy] Why is it good practice to augment $\mathbb{D}$ to include missingness dummies? In other words, why would this increase oos predictive accuracy?

Augmenting the dataset with missingness dummies can increase out-of-sample predictive accuracy by retaining information about the pattern of missing data, reducing bias, enhancing the feature set, allowing for conditional handling of missing data, and improving the model's generalization and robustness. This approach helps the model learn more accurately from the available data and make better predictions on new, unseen data that may also contain missing values.

(d) [easy] To impute missing values in $\mathbb{D}$, what is a good default strategy and why?

Mean imputation for numerical data and mode imputation for categorical data are good default strategies because they are simple, easy to implement, preserve the data structure, are computationally efficient, and provide a reasonable baseline for comparison. While they have limitations, they work well in many common scenarios and can be a practical first step in handling missing data.

Problem 4

These are some questions related to gradient boosting. The final gradient boosted model after M iterations is denoted G_M which can be written in a number of equivalent ways (see below). The g_t's denote constituent models and the G_t's denote partial sums of the constituent models up to iteration number t. The constituent models are "steps in functional steps" which have a step size η and a direction component denoted $\tilde{g}_t$. The directional component is the base learner $\mathcal{A}$ fit to the negative gradient of the objective function L which measures how close the current predictions are to the real values of the responses:

$$\begin{aligned}
G_M &= G_{M-1} + g_M \\
&= g_0 + g_1 + \ldots + g_M \\
&= g_0 + \eta\tilde{g}_1 + \ldots + \eta\tilde{g}_M \\
&= g_0 + \eta\mathcal{A}\left(\langle \boldsymbol{X}, -\nabla L(\boldsymbol{y}, \hat{\boldsymbol{y}}_1)\rangle, \mathcal{H}\right) + \ldots + \eta\mathcal{A}\left(\langle \boldsymbol{X}, -\nabla L(\boldsymbol{y}, \hat{\boldsymbol{y}}_M)\rangle, \mathcal{H}\right) \\
&= g_0 + \eta\mathcal{A}\left(\langle \boldsymbol{X}, -\nabla L(\boldsymbol{y}, g_1(\boldsymbol{X}))\rangle, \mathcal{H}\right) + \ldots + \eta\mathcal{A}\left(\langle \boldsymbol{X}, -\nabla L(\boldsymbol{y}, g_M(\boldsymbol{X}))\rangle, \mathcal{H}\right)
\end{aligned}$$

(a) [easy] From a perspective of only multivariable calculus, explain gradient descent and why it's a good idea to find the minimum inputs for an objective function L (in English).

Gradient decent attepmts to utilize the usefulness of gradients my making micro adjustments to better guide the model to a minima in the complex function space It iteratively adjusts parameters to minimize an objective function by following the negative gradient, ensuring efficient and guided optimization. Its simplicity, computational efficiency, adaptability, and scalability make it an ideal choice for finding minimum inputs of objective functions, ultimately enhancing model performance and predictive accuracy.

(b) [easy] Write the mathematical steps of gradient boosting for supervised learning below. Use L for the objective function to keep the procedure general. Use notation found in the problem header.

Let's denote the final gradient boosted model after M iterations as G_M. The objective function is L, which measures the discrepancy between the predicted values and the actual values. The gradient boosting procedure involves iteratively adding new models to correct the errors of the existing ensemble.

Initialization: 1. **Initial Model**: - Start with an initial model g_0, which could be a simple model like the mean of the target variable.

$$g_0(x) = \arg\min \{\} c \sum_{i=1}^{n} L(y_i, c)$$

Iterative Process for $t = 1$ to M: 2. **Compute Pseudo-Residuals**: - Calculate the negative gradient (pseudo-residuals) of the objective function L with respect to the current model G_{t-1}:

$$r_i^{(t)} = -\left[\frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}\right]_{\hat{y}_i = G_{t-1}(x_i)}, \quad \text{for } i = 1, 2, \ldots, n$$

3. **Fit Base Learner**: - Fit a base learner $\tilde{g}_t$ to the pseudo-residuals $r_i^{(t)}$:

$$\tilde{g}_t = A(\{(x_i, r_i^{(t)})\}_{i=1}^{n})$$

Here, A is the base learning algorithm.

4. **Scale the Base Learner**: - Scale the fitted base learner by a learning rate η:

$$g_t = \eta \tilde{g}_t$$

5. **Update the Ensemble Model**: - Update the current model by adding the scaled base learner:

$$G_t(x) = G_{t-1}(x) + g_t(x)$$

Final Model: 6. **Final Model Expression**: - After M iterations, the final gradient boosted model is:

$$G_M(x) = g_0(x) + \sum_{t=1}^{M} \eta \tilde{g}_t(x)$$

Mathematical Steps of Gradient Boosting for Supervised Learning

Let's denote the final gradient boosted model after M iterations as G_M. The objective function is L, which measures the discrepancy between the predicted values and the actual values. The gradient boosting procedure involves iteratively adding new models to correct the errors of the existing ensemble.

*Initialization: 1. **Initial Model**: - Start with an initial model g_0, which could be a simple model like the mean of the target variable.*

$$g_0(x) = \arg \mathit{min} \{\} c \sum_{i=1}^{n} L(y_i, c)$$

*Iterative Process for $t = 1$ to M: 2. **Compute Pseudo-Residuals**: - Calculate the negative gradient (pseudo-residuals) of the objective function L with respect to the current model G_{t-1}:*

$$r_i^{(t)} = -\left[\frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}\right]_{\hat{y}_i = G_{t-1}(x_i)}, \quad \mathit{for}\ i = 1, 2, \ldots, n$$

*3. **Fit Base Learner**: - Fit a base learner $\tilde{g}_t$ to the pseudo-residuals $r_i^{(t)}$:*

$$\tilde{g}_t = A(\{(x_i, r_i^{(t)})\}_{i=1}^n)$$

Here, A is the base learning algorithm.

*4. **Scale the Base Learner**: - Scale the fitted base learner by a learning rate η:*

$$g_t = \eta \tilde{g}_t$$

*5. **Update the Ensemble Model**: - Update the current model by adding the scaled base learner:*

$$G_t(x) = G_{t-1}(x) + g_t(x)$$

*Final Model: 6. **Final Model Expression**: - After M iterations, the final gradient boosted model is:*

$$G_M(x) = g_0(x) + \sum_{t=1}^{M} \eta \tilde{g}_t(x)$$

(c) *[easy] For regression, what is $g_0(\boldsymbol{x})$?*

In gradient boosting for regression, the initial model g(x) is typically the mean of the target variable y across the training data. This choice provides a simple and effective starting point for minimizing the mean squared error and guiding the subsequent iterations of the boosting process.

(d) *[easy] For probability estimation for binary response, what is $g_0(\boldsymbol{x})$?*

In gradient boosting for probability estimation with a binary response, the initial model g(x) is typically set to the log-odds of the mean of the target variable y across the training data. This provides a baseline prediction of the probability of the positive class and is a natural starting point for logistic regression-based gradient boosting algorithms.

(e) *[harder] What are all the hyperparameters of gradient boosting? There are more than just two.*

In gradient boosting, the max depth and learning rate are crucial hyperparameters that significantly impact model performance. The max depth controls the complexity of each tree; deeper trees can capture more intricate patterns but are more prone to overfitting, while shallower trees are less likely to overfit and tend to generalize better. The learning rate (n) determines the step size at each iteration, balancing the contribution of each tree to the final model. Lower learning rates reduce the risk of overfitting by making smaller adjustments, but they require more trees (iterations) to achieve optimal performance. Proper tuning of these two hyperparameters is essential for finding the right balance between model complexity and generalization, leading to improved predictive accuracy and robustness.

(f) *[easy] For regression, rederive the negative gradient of the objective function L.*

(g) *[easy] For probability estimation for binary response, rederive the negative gradient of the objective function L.*

(h) *[difficult] For probability estimation for binary response scenarios, what is the unit of the output $G_M(\boldsymbol{x}_\star)$?*

In probability estimation for binary response scenarios, the unit of the output $G_M(x_*)$ from the final gradient boosted model is log-odds (logits). This output can be converted to a probability using the sigmoid function, which maps the log-odds to a value between 0 and 1, representing the estimated probability of the positive class

(i) *[easy] For the base learner algorithm $\mathcal{A}$, why is it a good idea to use shallow CART (which is the recommended default)?*

Using shallow CART as the base learner in gradient boosting is advantageous because shallow trees act as simple, interpretable basic learners that prevent overfitting, reduce variance, and ensure computational efficiency. Their limited depth helps maintain controlled model complexity and incremental improvements, which align with the principles of boosting to create a strong, generalizable model from many weak learners.

(j) *[difficult] For the base learner algorithm $\mathcal{A}$, why is it a bad idea to use deep CART?*

Using deep CART as the base learner in gradient boosting is a bad idea because deep trees tend to overfit the training data, resulting in poor generalization to new data. They have high variance, increasing model instability, and are computationally expensive to train. Deep trees also undermine the incremental improvement principle of boosting and reduce the interpretability of the final model. Shallow trees, in contrast, maintain controlled complexity, improve computational efficiency, and enhance the generalizability of the boosted ensemble.

(k) *[difficult] For the base learner algorithm $\mathcal{A}$, why is it a bad idea to use OLS for regression (or logistic regression for probability estimation for binary response)?*

Using OLS for regression or logistic regression for probability estimation as base learners in gradient boosting is a bad idea because these linear models have limited flexibility, high bias, and do not introduce sufficient diversity in the ensemble. They struggle to capture complex, non-linear patterns and interactions between features, leading to suboptimal corrections of residuals and potentially inefficient boosting iterations. Non-linear models like shallow CART are better suited for boosting as they provide the necessary flexibility and diversity to effectively reduce errors and improve model performance.

(l) *[difficult] If M is very, very large, what is the risk in using gradient boosting even using shallow CART as the base learner (the recommended default)?*

Using a very large number of iterations M in gradient boosting, even with shallow CART as the base learner, risks overfitting, increased computational cost, diminishing returns, reduced model interpretability, and hyperparameter sensitivity. While shallow CART trees help mitigate some risks, the sheer number of iterations can still lead to models that are overly complex and less generalizable to new data. Proper early stopping criteria and regularization techniques are essential to prevent these issues.

(m) *[difficult] If η is very, very large but M reasonably correctly chosen, what is the risk in using gradient boosting even using shallow CART as the base learner (the recommended default)?*

Using a very large learning rate (n) in gradient boosting, even with a correctly chosen number of iterations (M) and shallow CART as the base learner, risks overfitting, instability, loss of incremental improvement, and increased variance. Large steps cause the model to make aggressive updates, capturing noise and disrupting the gradual refinement process that is central to boosting. This can lead to poor generalization and erratic model performance. Proper tuning of the learning rate is essential to maintain the stability and effectiveness of the gradient boosting algorithm.