Supervised Learning with Tabular Data

Gov 2018

Naijia Liu

February 2024

Generalized Additive Models (GAM)

- Linear regression assumes strong functional form:
  $E[Y_i|X_i] = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_K X_{iK}$

Generalized Additive Models (GAM)

- Linear regression assumes strong functional form: $E[Y_i|X_i] = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_K X_{iK}$
- Consider instead $E[Y_i|X_i] = \beta_0 + f_1(X_{i1}) + \cdots + f_K(X_{iK})$, where $f_k(\cdot)$ is any smoothing function (e.g. LOESS, cubic smoothing splines)

Generalized Additive Models (GAM)

- Linear regression assumes strong functional form: $E[Y_i|X_i] = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_K X_{iK}$
- Consider instead $E[Y_i|X_i] = \beta_0 + f_1(X_{i1}) + \cdots + f_K(X_{iK})$, where $f_k(\cdot)$ is any smoothing function (e.g. LOESS, cubic smoothing splines)
- For example, it can be linear regression: $f(X_{i1}) = \beta_1 X_{i1}$

Generalized Additive Models (GAM)

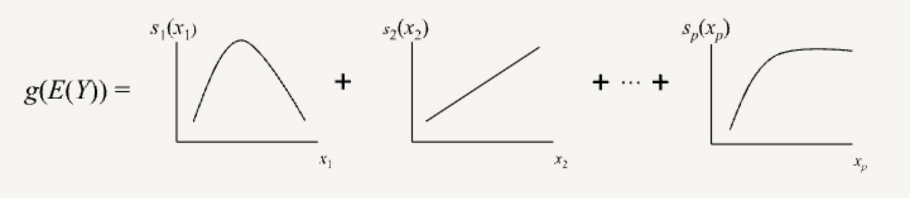
- Linear regression assumes strong functional form: $E[Y_i|X_i] = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_K X_{iK}$
- Consider instead $E[Y_i|X_i] = \beta_0 + f_1(X_{i1}) + \cdots + f_K(X_{iK})$, where $f_k(\cdot)$ is any smoothing function (e.g. LOESS, cubic smoothing splines)
- For example, it can be linear regression: $f(X_{i1}) = \beta_1 X_{i1}$
- Relationships between IV and DV follow some smooth patterns (linear or non-linear).

Generalized Additive Models (GAM)

- Linear regression assumes strong functional form: $E[Y_i|X_i] = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_K X_{iK}$
- Consider instead $E[Y_i|X_i] = \beta_0 + f_1(X_{i1}) + \cdots + f_K(X_{iK})$, where $f_k(\cdot)$ is any smoothing function (e.g. LOESS, cubic smoothing splines)
- For example, it can be linear regression: $f(X_{i1}) = \beta_1 X_{i1}$
- Relationships between IV and DV follow some smooth patterns (linear or non-linear).
- We can estimate these smooth relationships simultaneously and then predict $E(Y|X)$ by simply adding them up.

Example

$$E(Y|X) = a + b \cdot x^2 + c \cdot x + \cdots + \exp(x) + \epsilon$$



Generalized Additive Models (GAM)

- Estimate by iteratively "backfitting": fit $f_1(\cdot)$ with regular (bivariate) smoothing procedure, fit $f_2(\cdot)$ on residuals, ... until convergence

Generalized Additive Models (GAM)

- Estimate by iteratively "backfitting": fit $f_1(\cdot)$ with regular (bivariate) smoothing procedure, fit $f_2(\cdot)$ on residuals, ... until convergence
- Can use a combination of linear and flexible terms (e.g. to estimate constant treatment effect while controlling flexibly for other variables)

Generalized Additive Models (GAM)

- Estimate by iteratively "backfitting": fit $f_1(\cdot)$ with regular (bivariate) smoothing procedure, fit $f_2(\cdot)$ on residuals, ... until convergence
- Can use a combination of linear and flexible terms (e.g. to estimate constant treatment effect while controlling flexibly for other variables)
- Also works for generalized linear models: $E[Y_i] = g^{-1}\left(\beta_0 + 27(X_{i1}) + \cdots + f_K(X_{iK})\right)$, where $g(\cdot)$ is an exponential link function

GAM is a Flexible Model

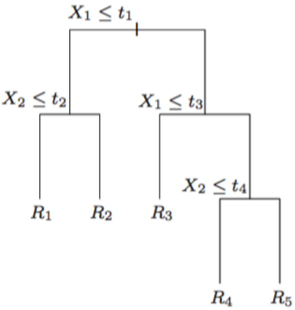
- Lots of freedom for researchers in selecting the smoothing functions.

GAM is a Flexible Model

- Lots of freedom for researchers in selecting the smoothing functions.
- We can penalize the objective function.

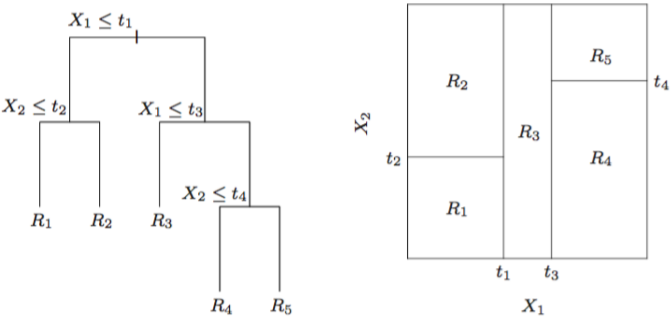
Tree-Based Methods

Classification and Regression Trees (CART) recursively partition the input space



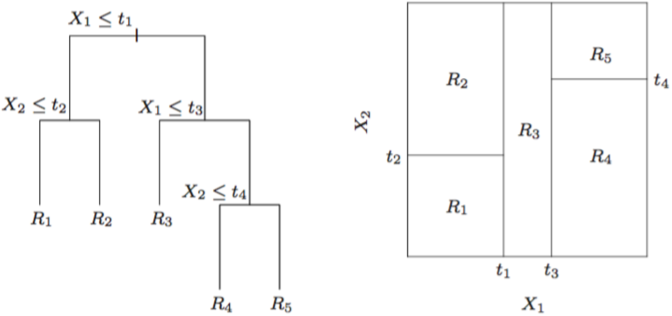
Tree-Based Methods

Classification and Regression Trees (CART) recursively partition the input space



Tree-Based Methods

Classification and Regression Trees (CART) recursively partition the input space



$$f(x) = \mathbb{E}[\mathbf{y}|X] = \sum_{m=1}^{M} \bar{y}_m \mathbb{1}\{x \in R_m\}$$

Regression Trees

- Partition data into subsets by observed covariates

Regression Trees

- Partition data into subsets by observed covariates
- Prediction using the average within each subset

Regression Trees

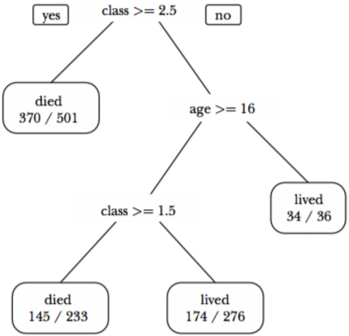
- Partition data into subsets by observed covariates
- Prediction using the average within each subset
- Very interpretable! (with simple average: no modeling!)

Regression Trees

- Partition data into subsets by observed covariates
- Prediction using the average within each subset
- Very interpretable! (with simple average: no modeling!)
  - "average of value with these characteristics"

Regression Trees

- Partition data into subsets by observed covariates
- Prediction using the average within each subset
- Very interpretable! (with simple average: no modeling!)
  - "average of value with these characteristics"
  - Titanic example from Varian (2014, JEP): rich children survived



Greedy Algorithm

- Recursive partition for good in-sample MSE in prediction

Greedy Algorithm

- Recursive partition for good in-sample MSE in prediction
- Finding the optimal partitioning is NP-complete (i.e., computation time grows too quickly as the size of the problem grows)

Greedy Algorithm

- Recursive partition for good in-sample MSE in prediction
- Finding the optimal partitioning is NP-complete (i.e., computation time grows too quickly as the size of the problem grows)
- Greedy algorithm (not thinking ahead, missing high-level interaction): locally optimal choice at each stage (not necessarily leads to a global optimum)

CART Algorithm

- At each stage, axis parallel splits is done by:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

where $R_1(j,s)=\{X|X_j\le s\}$ and $R_2(j,s)=\{X|X_j\ge s\}$, and c_m is the mean of $y_i|i\in R_m(j,s)$.

CART Algorithm

- At each stage, axis parallel splits is done by:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

  where $R_1(j,s)=\{X|X_j\leq s\}$ and $R_2(j,s)=\{X|X_j\geq s\}$, and c_m is the mean of $y_i|i\in R_m(j,s)$.

- CART is very popular and easy to implement (`rpart` package in R)

CART Algorithm

- At each stage, axis parallel splits is done by:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

  where $R_1(j,s)=\{X|X_j\leq s\}$ and $R_2(j,s)=\{X|X_j\geq s\}$, and c_m is the mean of $y_i|i\in R_m(j,s)$.

- CART is very popular and easy to implement (`rpart` package in R)
- However, the greedy algorithm often leads to poor prediction. It also has a problem of over-fitting (pruning is needed), and thus highly variable.

Model Selection

- What variables should we include in a given model?
- Ridge and Lasso regressions: the knowledge of λ value.
- GAM depends on the selection of smoothing function.
- CART requires pruning to achieve optimal performance.
- $\cdots$

1. Regression with Continuous Outcome
    - Generalized Additive Models (GAM)
    - CART
    - Motivation of Model Selection

2. Model Selection and Combination
    - Model Selection
    - Benign Overfitting
    - Model Aggregation
    - Evaluation Metrics, Cross-Validation

Goals

- Model assessment: Estimate generalization error
  - How good will this model be when I use it out-of-sample?
  - Is it better/worse than random choice?
  - Is it better/worse than human coders? (inter-coder reliability)
  - Than a previous approach?

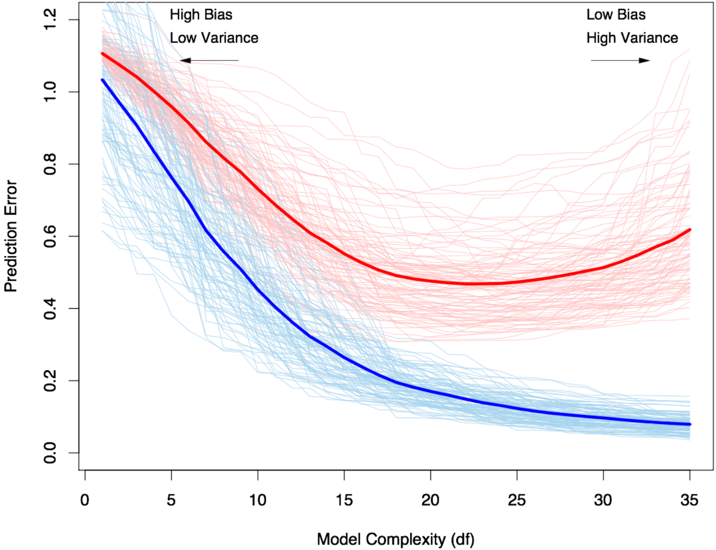
Goals

- Model assessment: Estimate generalization error
  - How good will this model be when I use it out-of-sample?
  - Is it better/worse than random choice?
  - Is it better/worse than human coders? (inter-coder reliability)
  - Than a previous approach?
- Model selection: Compare candidate models to determine best
  - Examples: number of spline knots, polynomial terms, number of terms to drop/retain in step-down variable selection, strength of regularization in LASSO/ridge, number of latent dimensions or clusters in unsupervised models

Goals

- Model assessment: Estimate generalization error
  - How good will this model be when I use it out-of-sample?
  - Is it better/worse than random choice?
  - Is it better/worse than human coders? (inter-coder reliability)
  - Than a previous approach?
- Model selection: Compare candidate models to determine best
  - Examples: number of spline knots, polynomial terms, number of terms to drop/retain in step-down variable selection, strength of regularization in LASSO/ridge, number of latent dimensions or clusters in unsupervised models
- Model aggregation: Combining candidate models into something better

Training Error vs. Test Error



Training Error vs. Test Error

- Beyond a certain point, we need to worry about over-fitting

Training Error vs. Test Error

- Beyond a certain point, we need to worry about over-fitting
- Bias-variance tradeoff!

Training Error vs. Test Error

- Beyond a certain point, we need to worry about over-fitting
- Bias-variance tradeoff!
- Adjusting training error (BIC tends to penalize complex models more heavily)

Training Error vs. Test Error

- Beyond a certain point, we need to worry about over-fitting
- Bias-variance tradeoff!
- Adjusting training error (BIC tends to penalize complex models more heavily)
  - Akaike Information Criterion $= -\frac{2}{N} \cdot \mathrm{loglik} + 2 \cdot \frac{d}{N}$

Training Error vs. Test Error

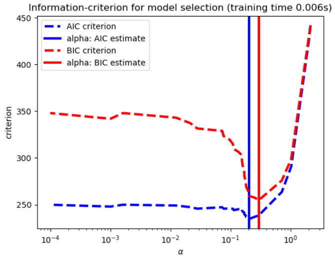
- Beyond a certain point, we need to worry about over-fitting
- Bias-variance tradeoff!
- Adjusting training error (BIC tends to penalize complex models more heavily)
  - Akaike Information Criterion $= -\frac{2}{N} \cdot \mathrm{loglik} + 2 \cdot \frac{d}{N}$
  - Bayesian Information Criterion $= \underbrace{-2 \cdot \mathrm{loglik}}_{\text{Deviance: squared loss}} + (\log N) \cdot d$

Training Error vs. Test Error

- Beyond a certain point, we need to worry about over-fitting
- Bias-variance tradeoff!
- Adjusting training error (BIC tends to penalize complex models more heavily)
  - Akaike Information Criterion $= -\frac{2}{N} \cdot \text{loglik} + 2 \cdot \frac{d}{N}$
  - Bayesian Information Criterion $= \underbrace{-2 \cdot \text{loglik}}_{\text{Deviance: squared loss}} + (\log N) \cdot d$

Training Error vs. Test Error

- Beyond a certain point, we need to worry about over-fitting
- Bias-variance tradeoff!
- Adjusting training error (BIC tends to penalize complex models more heavily)
  - Akaike Information Criterion $= -\frac{2}{N} \cdot \text{loglik} + 2 \cdot \frac{d}{N}$
  - Bayesian Information Criterion $= \underbrace{-2 \cdot \text{loglik}}_{\text{Deviance: squared loss}} + (\log N) \cdot d$



Training Error vs. Test Error

- We don't have a luxury of having a large data as a test set
- Estimating error variance $\hat{\sigma}$ to calculate loglik is problematic with high-dimensional setting ($n < p$)
- Cross-validation: Holding out a subset of the training observations from the training process

Kullback-Leibler Divergence

- Suppose the true probability distribution is $f(x)$
- We are approximating this with $g(x)$

Kullback-Leibler Divergence

- Suppose the true probability distribution is $f(x)$
- We are approximating this with $g(x)$
- One measure of the quality of approximation is KL divergence from f to g,

$$KL(f||g) = \int_{-\infty}^{\infty} f(x) \log \left(\frac{f(x)}{g(x)} \right)$$

- What is the KL divergence if $g(x) = f(x)$?

Kullback-Leibler Divergence

- Suppose the true probability distribution is $f(x)$
- We are approximating this with $g(x)$
- One measure of the quality of approximation is KL divergence from f to g,

$$KL(f||g) = \int_{-\infty}^{\infty} f(x) \log\left(\frac{f(x)}{g(x)}\right)$$

- What is the KL divergence if $g(x) = f(x)$?

$$KL(f||g) = \int_{-\infty}^{\infty} f(x) \log(1) = 0$$

Kullback-Leibler Divergence

- Suppose the true probability distribution is $f(x)$
- We are approximating this with $g(x)$
- One measure of the quality of approximation is KL divergence from f to g,

$$KL(f||g) = \int_{-\infty}^{\infty} f(x) \log\left(\frac{f(x)}{g(x)}\right)$$

- What is the KL divergence if $g(x) = f(x)$?

$$KL(f||g) = \int_{-\infty}^{\infty} f(x) \log(1) = 0$$

- When is this positive?

Kullback-Leibler Divergence

- Suppose the true probability distribution is $f(x)$
- We are approximating this with $g(x)$
- One measure of the quality of approximation is KL divergence from f to g,

$$KL(f||g) = \int_{-\infty}^{\infty} f(x) \log\left(\frac{f(x)}{g(x)}\right)$$

- What is the KL divergence if $g(x) = f(x)$?

$$KL(f||g) = \int_{-\infty}^{\infty} f(x) \log(1) = 0$$

- When is this positive? All possible distributions $g(x) \neq f(x)$.

KL divergence is always positive

- In inequality:

$$\log a \leq a - 1, \forall a > 0$$

KL divergence is always positive

- In inequality:

$$\log a \leq a - 1, \forall a > 0$$

- Proof:

$$\begin{aligned}
-KL(f||g) &= -\int_{-\infty}^{\infty} f(x) \log\left(\frac{f(x)}{g(x)}\right) \\
&= \int_{-\infty}^{\infty} f(x) \log\left(\frac{g(x)}{f(x)}\right) \\
&\leq \int_{-\infty}^{\infty} f(x) \left(\frac{g(x)}{f(x)} - 1\right) \\
&= \int_{-\infty}^{\infty} (g(x) - f(x)) \\
&= 1 - 1 \\
&= 0
\end{aligned}$$

KL divergence is not a proper distance measure

- Kullback-Leibler divergence is not a distance metric!
- Distances must be nonnegative, symmetric, and satisfy triangle inequality: distance from A to C must be less than (A to B) + (B to C)

Kullback-Leibler Divergence

- Kullback-Leibler divergence is not a distance metric!

Kullback-Leibler Divergence

- Kullback-Leibler divergence is not a distance metric!
  - KL is asymmetric: $KL(f||g)$ and $KL(g||f)$ not necessarily the same.

    Bonus Q: what kind of f, g can make them the same?

Kullback-Leibler Divergence

- Kullback-Leibler divergence is not a distance metric!
  - KL is asymmetric: $KL(f||g)$ and $KL(g||f)$ not necessarily the same.

    Bonus Q: what kind of f, g can make them the same?
  - KL does not satisfy triangle inequality

Kullback-Leibler Divergence

- Kullback-Leibler divergence is not a distance metric!
  - KL is asymmetric: $KL(f||g)$ and $KL(g||f)$ not necessarily the same.

    Bonus Q: what kind of f, g can make them the same?
  - KL does not satisfy triangle inequality
- Useful for comparing images (represented as distribution of pixel values in multivariate color space)

Kullback-Leibler Divergence

- Kullback-Leibler divergence is not a distance metric!
  - KL is asymmetric: $KL(f||g)$ and $KL(g||f)$ not necessarily the same.

    Bonus Q: what kind of f, g can make them the same?
  - KL does not satisfy triangle inequality
- Useful for comparing images (represented as distribution of pixel values in multivariate color space)
- Or documents, if they are represented as distributions of word coordinates in a continuous embedding space

  "The limited but real influence of elite rhetoric in the 2009–2010 health care debate" , Hopkins, 2018, Political behavior

Optimality of OOS Likelihood-Based Selection

- True data distribution is $f(x)$

Optimality of OOS Likelihood-Based Selection

- True data distribution is $f(x)$
- You have many candidate models (data distributions): $\hat{f}_1(\cdot), \hat{f}_2(\cdot), \ldots$ (e.g. OLS with all possible βs) How do we normally choose "best"?

Optimality of OOS Likelihood-Based Selection

- True data distribution is $f(x)$
- You have many candidate models (data distributions): $\hat{f}_1(\cdot), \hat{f}_2(\cdot), \ldots$ (e.g. OLS with all possible βs) How do we normally choose "best"?
- Some models achieve MLE for training data (minimized KL divergence to the empirical distribution!) but probably doesn't explain true DGP well

Optimality of OOS Likelihood-Based Selection

- True data distribution is $f(x)$
- You have many candidate models (data distributions): $\hat{f}_1(\cdot), \hat{f}_2(\cdot), \ldots$ (e.g. OLS with all possible βs) How do we normally choose "best"?
- Some models achieve MLE for training data (minimized KL divergence to the empirical distribution!) but probably doesn't explain true DGP well
- One way to choose the best model from among your candidates is the one that produces the closest approximation to the true DGP

Optimality of OOS Likelihood-Based Selection

- True data distribution is $f(x)$
- You have many candidate models (data distributions): $\hat{f}_1(\cdot), \hat{f}_2(\cdot), \ldots$ (e.g. OLS with all possible βs) How do we normally choose "best"?
- Some models achieve MLE for training data (minimized KL divergence to the empirical distribution!) but probably doesn't explain true DGP well
- One way to choose the best model from among your candidates is the one that produces the closest approximation to the true DGP
- That is, the best model minimizes

$$\int f(x) \log \left(\frac{f(x)}{\hat{f}_k(x)} \right) dx$$

Optimality of OOS Likelihood-Based Selection

- True data distribution is $f(x)$
- You have many candidate models (data distributions): $\hat{f}_1(\cdot), \hat{f}_2(\cdot), \ldots$ (e.g. OLS with all possible βs) How do we normally choose "best"?
- Some models achieve MLE for training data (minimized KL divergence to the empirical distribution!) but probably doesn't explain true DGP well
- One way to choose the best model from among your candidates is the one that produces the closest approximation to the true DGP
- That is, the best model minimizes

$$\int f(x) \log \left(\frac{f(x)}{\hat{f}_k(x)} \right) dx$$

- What happens when one of your candidate models is the true model?

Optimality of OOS Likelihood-Based Selection

- True data distribution is $f(x)$
- You have many candidate models (data distributions): $\hat{f}_1(\cdot), \hat{f}_2(\cdot), \ldots$ (e.g. OLS with all possible βs) How do we normally choose "best"?
- Some models achieve MLE for training data (minimized KL divergence to the empirical distribution!) but probably doesn't explain true DGP well
- One way to choose the best model from among your candidates is the one that produces the closest approximation to the true DGP
- That is, the best model minimizes

$$\int f(x) \log\left(\frac{f(x)}{\hat{f}_k(x)}\right) dx$$

- What happens when one of your candidate models is the true model?
- The problem: we don't know the true population distribution

Optimality of Likelihood-Based CV

- Imagine an "oracle" that knows the true model and can pick the candidate that is closest. Can we devise a procedure that asymptotically gives the same answer?

Optimality of Likelihood-Based CV

- Imagine an "oracle" that knows the true model and can pick the candidate that is closest. Can we devise a procedure that asymptotically gives the same answer?

$$
\begin{aligned}
KL(f||\hat{f}) &= \int \log\left(\frac{f(x)}{\hat{f}_k(x)}\right) f(x)dx \\
&= \int \log\left(f(x)\right) f(x)dx - \int \log\left(\hat{f}_k(x)\right) f(x)dx \\
&= C - \int \log\left(\hat{f}_k(x)\right) f(x)dx
\end{aligned}
$$

Optimality of Likelihood-Based CV

- Imagine an "oracle" that knows the true model and can pick the candidate that is closest. Can we devise a procedure that asymptotically gives the same answer?

$$
\begin{aligned}
KL(f||\hat{f}) &= \int \log\left(\frac{f(x)}{\hat{f}_k(x)}\right) f(x)dx \\
&= \int \log\left(f(x)\right) f(x)dx - \int \log\left(\hat{f}_k(x)\right) f(x)dx \\
&= C - \int \log\left(\hat{f}_k(x)\right) f(x)dx
\end{aligned}
$$

- Interpretation: KL divergence is directly related to the expected log likelihood (under the candidate model) of a randomly drawn observation from the true DGP

Optimality of Likelihood-Based CV

- Likelihood based cross validation can give us a model, such that it maximizes the log-likelihood under the true DGP.

Optimality of Likelihood-Based CV

- Likelihood based cross validation can give us a model, such that it maximizes the log-likelihood under the true DGP.
- We have many random draws from true DGP: validation set!

Optimality of Likelihood-Based CV

- Likelihood based cross validation can give us a model, such that it maximizes the log-likelihood under the true DGP.
- We have many random draws from true DGP: validation set!
- As sample size goes to infinity, size of validation set generally also goes to infinity (not true for leave-one-out CV)

Benign Overfitting

- A new statistical phenomenon: good prediction with zero training error (such as a deep learning method) .
- When a perfect fit to training data in linear regression is compatible with accurate prediction.
- Over-parameterization is essential for benign overfitting in some setting: more features than sample size.
- "Benign overfitting in linear regression", Bartlett et al, PNAS, 2020.
- Most of situations we don't have benign overfitting - unless you can justify the choice.
- AI models such as Chat GPT relies on massive over-fitting that are benign.

Model Averaging

- Simplest possible model aggregation:

Model Averaging

- Simplest possible model aggregation:
  - Train K models (logit, SVM, LASSO, tree, etc.)

Model Averaging

- Simplest possible model aggregation:
  - Train K models (logit, SVM, LASSO, tree, etc.)
  - Predict out-of-sample observations with each one

Model Averaging

- Simplest possible model aggregation:
  - Train K models (logit, SVM, LASSO, tree, etc.)
  - Predict out-of-sample observations with each one
  - Use some aggregation rule (e.g. mean, majority vote) to combine

Model Averaging

Bayesian model averaging:

- Define K possible models: $f_1(y|X = x), \ldots, f_K(y|X = x)$

Model Averaging

Bayesian model averaging:

- Define K possible models: $f_1(y|X = x), \ldots, f_K(y|X = x)$
- Place a prior over all models (e.g. uniform $\pi = [1/K, \ldots, 1/K]^\top$)

Model Averaging

Bayesian model averaging:

- Define K possible models: $f_1(y|X = x), \ldots, f_K(y|X = x)$
- Place a prior over all models (e.g. uniform $\pi = [1/K, \ldots, 1/K]^\top$)
- Assumed DGP: first randomly choose a model, then generate all data points from that model

$$Z \sim \text{Cat}(\pi)$$
$$Y_i \sim f_Z(y|X_i = x_i)$$

taking $\mathbf{X}$ as given

Model Averaging

Bayesian model averaging:

- Posterior belief $P(Z = k|\mathbf{Y}, \mathbf{X})$ depends on π_k and how well $f_k()$ explains observed $\mathbf{Y}$

Model Averaging

Bayesian model averaging:

- Posterior belief $P(Z = k|\mathbf{Y}, \mathbf{X})$ depends on π_k and how well $f_k()$ explains observed $\mathbf{Y}$
- Out-of-sample prediction:

Model Averaging

Bayesian model averaging:

- Posterior belief $P(Z = k|\mathbf{Y}, \mathbf{X})$ depends on π_k and how well $f_k()$ explains observed $\mathbf{Y}$
- Out-of-sample prediction:
  - Generate predictions from each model

Model Averaging

Bayesian model averaging:

- Posterior belief $P(Z = k|\mathbf{Y}, \mathbf{X})$ depends on π_k and how well $f_k()$ explains observed $\mathbf{Y}$
- Out-of-sample prediction:
  - Generate predictions from each model
  - Weight each model by posterior belief that it was the chosen one

Model Averaging

Bayesian model averaging:

- Posterior belief $P(Z = k|\mathbf{Y}, \mathbf{X})$ depends on π_k and how well $f_k()$ explains observed $\mathbf{Y}$
- Out-of-sample prediction:
  - Generate predictions from each model
  - Weight each model by posterior belief that it was the chosen one
  - Take weighted average of predictions

Bootstrap Aggregation (Bagging)

- A particular kind of model averaging

Bootstrap Aggregation (Bagging)

- A particular kind of model averaging
- Bootstrap data B times and average results

Bootstrap Aggregation (Bagging)

- A particular kind of model averaging
- Bootstrap data B times and average results
- Reduces variance if bootstrap models aren't highly correlated

Bootstrap Aggregation (Bagging)

- A particular kind of model averaging
- Bootstrap data B times and average results
- Reduces variance if bootstrap models aren't highly correlated
  - If models are highly correlated then little gain

Bootstrap Aggregation (Bagging)

- A particular kind of model averaging
- Bootstrap data B times and average results
- Reduces variance if bootstrap models aren't highly correlated
  - If models are highly correlated then little gain
  - $V(X/2 + Y/2) = V(X)/4 + V(Y)/4 + 2Cov(X, Y)/4$

Bootstrap Aggregation (Bagging)

- A particular kind of model averaging
- Bootstrap data B times and average results
- Reduces variance if bootstrap models aren't highly correlated
  - If models are highly correlated then little gain
  - $V(X/2 + Y/2) = V(X)/4 + V(Y)/4 + 2Cov(X, Y)/4$
  - Can help a lot with extremely unstable models (LASSO, trees)

Bootstrap Aggregation (Bagging)

- A particular kind of model averaging
- Bootstrap data B times and average results
- Reduces variance if bootstrap models aren't highly correlated
  - If models are highly correlated then little gain
  - $V(X/2 + Y/2) = V(X)/4 + V(Y)/4 + 2Cov(X, Y)/4$
  - Can help a lot with extremely unstable models (LASSO, trees)
  - Won't help much when the component models are stable (and can actually hurt)

Bootstrap Aggregation (Bagging)

- A particular kind of model averaging
- Bootstrap data B times and average results
- Reduces variance if bootstrap models aren't highly correlated
  - If models are highly correlated then little gain
  - $V(X/2 + Y/2) = V(X)/4 + V(Y)/4 + 2Cov(X, Y)/4$
  - Can help a lot with extremely unstable models (LASSO, trees)
  - Won't help much when the component models are stable (and can actually hurt)
- However, bagging procedure normally results in highly correlated predictors

Bootstrap Aggregation (Bagging)

- Random forests are a variant of bagged trees

Bootstrap Aggregation (Bagging)

- Random forests are a variant of bagged trees
- Reduce correlation of component models by also randomly choosing subset of features (in addition to bootstrap subsetting/reweighting of data)

Bootstrap Aggregation (Bagging)

- Random forests are a variant of bagged trees
- Reduce correlation of component models by also randomly choosing subset of features (in addition to bootstrap subsetting/reweighting of data)
- Improves variance reduction of bagging by reducing the correlation between models

Bootstrap Aggregation (Bagging)

- Random forests are a variant of bagged trees
- Reduce correlation of component models by also randomly choosing subset of features (in addition to bootstrap subsetting/reweighting of data)
- Improves variance reduction of bagging by reducing the correlation between models
- Easy to implement (`randomForest` package in R) and often very good off-the-shelf performance

Bootstrapping: More Perspectives

Each bootstrap draw is a sample split

- Observation i is in validation set with probability $(1 - 1/N)^N \approx 1 - 1/e \approx 0.632$
- So bootstrap draws each come with a free validation set!

Bootstrapping: More Perspectives

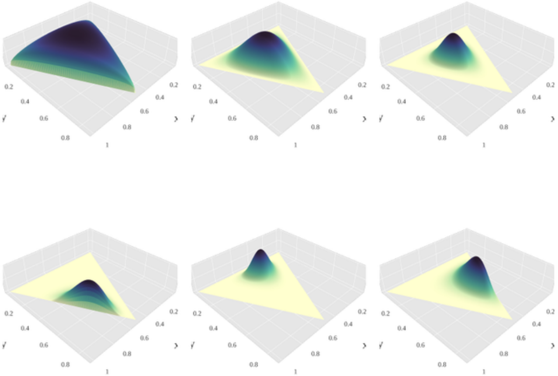
Each bootstrap draw is a sample split

- Observation i is in validation set with probability $(1 - 1/N)^N \approx 1 - 1/e \approx 0.632$
- So bootstrap draws each come with a free validation set!

Each bootstrap draw is a reweighting of the original data

- For each draw, $w_i \in \{0, 1/N, \cdots, 1\})$
- Weights must sum to 1
- In other words, weights are a distribution over the $N - 1$ simplex

Bootstrapping: More Perspectives



Bayesian bootstrap:

- The Dirichlet distribution is also a distribution over the $N-1$ simplex
- $\mathbf{w} = [w_1, \cdots, w_N]$
- $\mathbf{w} \sim \text{Dirichlet}([1, \ldots, 1]^\top)$
- $E[\mathbf{w}] = [1/N, \ldots, 1/N]^\top$
- Procedure: Sample $\mathbf{w}$, then do the original analysis. Repeat.

Super Learner

- Suppose that training is complete and each algorithm has generated a trained model

Super Learner

- Suppose that training is complete and each algorithm has generated a trained model
- Then each model outputs a prediction function

Super Learner

- Suppose that training is complete and each algorithm has generated a trained model
- Then each model outputs a prediction function
- Can think of each prediction function as a basis function (similar to splines)

Super Learner

- Suppose that training is complete and each algorithm has generated a trained model
- Then each model outputs a prediction function
- Can think of each prediction function as a basis function (similar to splines)
- These bases can be combined to form an even better approximation to the true conditional expectation function

Super Learner

- Super Learner algorithm:

Super Learner

- Super Learner algorithm:
  - Fit candidate models to training data

Super Learner

- Super Learner algorithm:
  - Fit candidate models to training data
  - Generate predictions for validation set

Super Learner

- Super Learner algorithm:
  - Fit candidate models to training data
  - Generate predictions for validation set
  - Regress true validation labels on candidate model predictions

$$Y = \beta_1 \text{model } 1 + \beta_2 \text{model } 2 + \beta_3 \text{model } 3 + \cdots$$

Super Learner

- Super Learner algorithm:
  - Fit candidate models to training data
  - Generate predictions for validation set
  - Regress true validation labels on candidate model predictions

$$Y = \beta_1 \text{model } 1 + \beta_2 \text{model } 2 + \beta_3 \text{model } 3 + \cdots$$

  - Use resulting weights

Super Learner

- Super Learner algorithm:
  - Fit candidate models to training data
  - Generate predictions for validation set
  - Regress true validation labels on candidate model predictions

$$Y = \beta_1 \text{model } 1 + \beta_2 \text{model } 2 + \beta_3 \text{model } 3 + \cdots$$

  - Use resulting weights
- Super Learner inherits the asymptotic optimality properties of likelihood-based CV

Super Learner

- Super Learner algorithm:
  - Fit candidate models to training data
  - Generate predictions for validation set
  - Regress true validation labels on candidate model predictions

$$Y = \beta_1 \text{model } 1 + \beta_2 \text{model } 2 + \beta_3 \text{model } 3 + \cdots$$

  - Use resulting weights
- Super Learner inherits the asymptotic optimality properties of likelihood-based CV

Super Learner

- Super Learner algorithm:
  - Fit candidate models to training data
  - Generate predictions for validation set
  - Regress true validation labels on candidate model predictions

$$Y = \beta_1 \text{model } 1 + \beta_2 \text{model } 2 + \beta_3 \text{model } 3 + \cdots$$

  - Use resulting weights
- Super Learner inherits the asymptotic optimality properties of likelihood-based CV (if desired, impose positivity of weights, weights sum to 1, etc.)
- Why?

Super Learner

- Super Learner algorithm:
  - Fit candidate models to training data
  - Generate predictions for validation set
  - Regress true validation labels on candidate model predictions

$$Y = \beta_1 \text{model } 1 + \beta_2 \text{model } 2 + \beta_3 \text{model } 3 + \cdots$$

  - Use resulting weights
- Super Learner inherits the asymptotic optimality properties of likelihood-based CV (if desired, impose positivity of weights, weights sum to 1, etc.)
- Why?

Super Learner

- Super Learner algorithm:
  - Fit candidate models to training data
  - Generate predictions for validation set
  - Regress true validation labels on candidate model predictions

$$Y = \beta_1 \text{model } 1 + \beta_2 \text{model } 2 + \beta_3 \text{model } 3 + \cdots$$

  - Use resulting weights
- Super Learner inherits the asymptotic optimality properties of likelihood-based CV (if desired, impose positivity of weights, weights sum to 1, etc.)
- Why? What happens if the true model is among the candidates?
- Asymptotically at least as accurate as the best possible input

Boosting

- Adaptive boosting (AdaBoost) is another meta-algorithm
- All observations are used for training (no validation set)
- Start by weighting all observations equally, $\mathbf{w}^1 = [w_1, \ldots, w_N]^\top$

Boosting

- Adaptive boosting (AdaBoost) is another meta-algorithm
- All observations are used for training (no validation set)
- Start by weighting all observations equally, $\mathbf{w}^1 = [w_1, \ldots, w_N]^\top$
- Set model index $m = 1$. While model has not converged:
  - Fit weighted model to dataset (according to $\mathbf{w}^m$)
  - Re-predict training data
  - Calculate weighted error (according to $\mathbf{w}^m$)
  - Change $\mathbf{w}$ by upweighting misclassified observations
  - Increment m (increase by 1) and repeat

Boosting

- Adaptive boosting (AdaBoost) is another meta-algorithm
- All observations are used for training (no validation set)
- Start by weighting all observations equally, $\mathbf{w}^1 = [w_1, \ldots, w_N]^\top$
- Set model index $m = 1$. While model has not converged:
  - Fit weighted model to dataset (according to $\mathbf{w}^m$)
  - Re-predict training data
  - Calculate weighted error (according to $\mathbf{w}^m$)
  - Change $\mathbf{w}$ by upweighting misclassified observations
  - Increment m (increase by 1) and repeat
- Now go back and pool the models:
  - Weight model m according to its (weighted) error rate, $\alpha_m = f\left(\mathbf{w}^{m\top}(\mathbf{Y} - \hat{\mathbf{Y})}^\top\right)$

Boosting

- Adaptive boosting (AdaBoost) is another meta-algorithm
- All observations are used for training (no validation set)
- Start by weighting all observations equally, $\mathbf{w}^1 = [w_1, \ldots, w_N]^\top$
- Set model index $m = 1$. While model has not converged:
    - Fit weighted model to dataset (according to $\mathbf{w}^m$)
    - Re-predict training data
    - Calculate weighted error (according to $\mathbf{w}^m$)
    - Change $\mathbf{w}$ by upweighting misclassified observations
    - Increment m (increase by 1) and repeat
- Now go back and pool the models:
    - Weight model m according to its (weighted) error rate, $\alpha_m = f\left(\mathbf{w}^{m\top}(\mathbf{Y} - \hat{\mathbf{Y})}^\top\right)$
    - For out-of-sample data, generate final predictions as $\hat{\mathbf{Y}} = \sum_m \alpha_m \hat{\mathbf{Y}}^m$

Boosting

- Again, the sequential models $m = 1, \dots, M$ create basis functions
- We greedily approximate the true conditional expectation with these bases
- Make the best approximation possible using $m = 1$
- Then fine-tune it with $m = 2, \dots$
- Models evolve in an adaptive way to remove bias
- Later models focus on examples that were misclassified in earlier rounds (hard to classify, e.g. near best decision boundary)

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:
    - Training set ($\mathcal{T}$)

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:
   - Training set ($\mathcal{T}$)
   - Validation (or hold-out) set ($\mathcal{V}$)

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:
   - Training set ($\mathcal{T}$)
   - Validation (or hold-out) set ($\mathcal{V}$)
2. Fit your model on the training set

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:
   - Training set ($\mathcal{T}$)
   - Validation (or hold-out) set ($\mathcal{V}$)
2. Fit your model on the training set
3. Use the fitted model to predict the responses for the observations in the validation set

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:
    - Training set ($\mathcal{T}$)
    - Validation (or hold-out) set ($\mathcal{V}$)
2. Fit your model on the training set
3. Use the fitted model to predict the responses for the observations in the validation set
4. Since you know the "ground truth," you can calculate the generalization performance *from training set to validation set*, e.g.

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:
   - Training set ($\mathcal{T}$)
   - Validation (or hold-out) set ($\mathcal{V}$)
2. Fit your model on the training set
3. Use the fitted model to predict the responses for the observations in the validation set
4. Since you know the "ground truth," you can calculate the generalization performance *from training set to validation set*, e.g.
   - MSE (Mean Squared Errors): $(y_i - \hat{y}_i)^2$

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:
    - Training set ($\mathcal{T}$)
    - Validation (or hold-out) set ($\mathcal{V}$)
2. Fit your model on the training set
3. Use the fitted model to predict the responses for the observations in the validation set
4. Since you know the "ground truth," you can calculate the generalization performance *from training set to validation set*, e.g.
    - MSE (Mean Squared Errors): $(y_i - \hat{y}_i)^2$
    - Misclassification rate (in case of discrete variables)

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:
   - Training set ($\mathcal{T}$)
   - Validation (or hold-out) set ($\mathcal{V}$)
2. Fit your model on the training set
3. Use the fitted model to predict the responses for the observations in the validation set
4. Since you know the "ground truth," you can calculate the generalization performance *from training set to validation set*, e.g.
   - MSE (Mean Squared Errors): $(y_i - \hat{y}_i)^2$
   - Misclassification rate (in case of discrete variables)
5. Then, fit model on all labeled data (training and validation)

Cross-Validation

Creating out-of-sample, in sample.

1. Randomly divide the labeled data into two parts:
    - Training set ($\mathcal{T}$)
    - Validation (or hold-out) set ($\mathcal{V}$)
2. Fit your model on the training set
3. Use the fitted model to predict the responses for the observations in the validation set
4. Since you know the "ground truth," you can calculate the generalization performance *from training set to validation set*, e.g.
    - MSE (Mean Squared Errors): $(y_i - \hat{y}_i)^2$
    - Misclassification rate (in case of discrete variables)
5. Then, fit model on all labeled data (training and validation)
6. Presumably you have a good proxy for generalization performance from "in-sample" (all labeled data) data to "out-of-sample" (test data, new observations to predict, or true DGP)

Cross-Validation

- Validation set can also be used to compare multiple candidate models

Cross-Validation

- Validation set can also be used to compare multiple candidate models
  - Fit K models on training set

Cross-Validation

- Validation set can also be used to compare multiple candidate models
  - Fit K models on training set
  - Select the one that performs best on validation set

Cross-Validation

- Validation set can also be used to compare multiple candidate models
  - Fit K models on training set
  - Select the one that performs best on validation set
  - Fit that model on all labeled data and use it out-of-sample

Cross-Validation

- Validation set can also be used to compare multiple candidate models
  - Fit K models on training set
  - Select the one that performs best on validation set
  - Fit that model on all labeled data and use it out-of-sample
  - Will validation performance be a good estimate of out-of-sample performance?

Cross-Validation

- Validation set can also be used to compare multiple candidate models
  - Fit K models on training set
  - Select the one that performs best on validation set
  - Fit that model on all labeled data and use it out-of-sample
  - Will validation performance be a good estimate of out-of-sample performance?
  - Analogous to multiple testing: one model might perform better by chance, then regress to the mean in OOS

Cross-Validation

- Validation set can also be used to compare multiple candidate models
  - Fit K models on training set
  - Select the one that performs best on validation set
  - Fit that model on all labeled data and use it out-of-sample
  - Will validation performance be a good estimate of out-of-sample performance?
  - Analogous to multiple testing: one model might perform better by chance, then regress to the mean in OOS
- This simple framework doesn't make efficient use of the data

Cross-Validation

- Validation set can also be used to compare multiple candidate models
  - Fit K models on training set
  - Select the one that performs best on validation set
  - Fit that model on all labeled data and use it out-of-sample
  - Will validation performance be a good estimate of out-of-sample performance?
  - Analogous to multiple testing: one model might perform better by chance, then regress to the mean in OOS
- This simple framework doesn't make efficient use of the data
  - Validation set doesn't contribute to estimating model at all

Cross-Validation

- Validation set can also be used to compare multiple candidate models
  - Fit K models on training set
  - Select the one that performs best on validation set
  - Fit that model on all labeled data and use it out-of-sample
  - Will validation performance be a good estimate of out-of-sample performance?
  - Analogous to multiple testing: one model might perform better by chance, then regress to the mean in OOS
- This simple framework doesn't make efficient use of the data
  - Validation set doesn't contribute to estimating model at all
  - Training outcomes don't contribute directly to estimating performance

Cross-Validation

- Validation set can also be used to compare multiple candidate models
  - Fit K models on training set
  - Select the one that performs best on validation set
  - Fit that model on all labeled data and use it out-of-sample
  - Will validation performance be a good estimate of out-of-sample performance?
  - Analogous to multiple testing: one model might perform better by chance, then regress to the mean in OOS
- This simple framework doesn't make efficient use of the data
  - Validation set doesn't contribute to estimating model at all
  - Training outcomes don't contribute directly to estimating performance
  - Solution: K-fold cross-validation

Cross-Validation

- Caveat: Validation performance may be worse than test performance

Cross-Validation

- Caveat: Validation performance may be worse than test performance
  - Remember that goal is to understand how good your model is *conditional on all labeled data*

Cross-Validation

- Caveat: Validation performance may be worse than test performance
  - Remember that goal is to understand how good your model is *conditional on all labeled data*
  - Cross-validation tells you how good your model is conditional of *some* labeled data your training set is less than N

Cross-Validation

- Caveat: Validation performance may be worse than test performance
  - Remember that goal is to understand how good your model is *conditional on all labeled data*
  - Cross-validation tells you how good your model is conditional of *some* labeled data your training set is less than N
  - With fewer observations, you fit the true data distribution worse

Cross-Validation

- Caveat: Validation performance may be worse than test performance
  - Remember that goal is to understand how good your model is *conditional on all labeled data*
  - Cross-validation tells you how good your model is conditional of *some* labeled data your training set is less than N
  - With fewer observations, you fit the true data distribution worse
  - True out-of-sample performance is somewhere between your validation performance and in-sample (possibly overfit) performance

Cross-Validation

- Caveat: Validation performance may be worse than test performance
  - Remember that goal is to understand how good your model is *conditional on all labeled data*
  - Cross-validation tells you how good your model is conditional of *some* labeled data your training set is less than N
  - With fewer observations, you fit the true data distribution worse
  - True out-of-sample performance is somewhere between your validation performance and in-sample (possibly overfit) performance
  - Under some assumptions these two measures (e.g. MSE, error rate) can be interpolated

Cross-Validation

- Caveat: Validation performance may be worse than test performance
    - Remember that goal is to understand how good your model is *conditional on all labeled data*
    - Cross-validation tells you how good your model is conditional of *some* labeled data your training set is less than N
    - With fewer observations, you fit the true data distribution worse
    - True out-of-sample performance is somewhere between your validation performance and in-sample (possibly overfit) performance
    - Under some assumptions these two measures (e.g. MSE, error rate) can be interpolated
- Tradeoff between training and validation size:

Cross-Validation

- Caveat: Validation performance may be worse than test performance
  - Remember that goal is to understand how good your model is *conditional on all labeled data*
  - Cross-validation tells you how good your model is conditional of *some* labeled data your training set is less than N
  - With fewer observations, you fit the true data distribution worse
  - True out-of-sample performance is somewhere between your validation performance and in-sample (possibly overfit) performance
  - Under some assumptions these two measures (e.g. MSE, error rate) can be interpolated
- Tradeoff between training and validation size:
  - With larger training set, the training model quality is closer to quality of the eventual model fit on all observations

Cross-Validation

- Caveat: Validation performance may be worse than test performance
  - Remember that goal is to understand how good your model is *conditional on all labeled data*
  - Cross-validation tells you how good your model is conditional of *some* labeled data your training set is less than N
  - With fewer observations, you fit the true data distribution worse
  - True out-of-sample performance is somewhere between your validation performance and in-sample (possibly overfit) performance
  - Under some assumptions these two measures (e.g. MSE, error rate) can be interpolated
- Tradeoff between training and validation size:
  - With larger training set, the training model quality is closer to quality of the eventual model fit on all observations
  - On the other hand, your estimate of validation performance becomes noisier

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish
- **An invalid procedure:** First do variable selection to find features that are individually highly predictive of the outcome, using both training and test set, then train (validate) on training (validation) set alone

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish
- **An invalid procedure:** First do variable selection to find features that are individually highly predictive of the outcome, using both training and test set, then train (validate) on training (validation) set alone
- In the above example, cross-validated performance will be an extremely poor measure of generalization (test) performance

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish
- **An invalid procedure:** First do variable selection to find features that are individually highly predictive of the outcome, using both training and test set, then train (validate) on training (validation) set alone
- In the above example, cross-validated performance will be an extremely poor measure of generalization (test) performance
- However, joint preprocessing of $[\mathbf{X}_{\text{train}}^{\top}, \mathbf{X}_{\text{test}}^{\top}]^{\top}$ together isn't subject to this issue

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish
- **An invalid procedure:** First do variable selection to find features that are individually highly predictive of the outcome, using both training and test set, then train (validate) on training (validation) set alone
- In the above example, cross-validated performance will be an extremely poor measure of generalization (test) performance
- However, joint preprocessing of $[\mathbf{X}_{\text{train}}^\top, \mathbf{X}_{\text{test}}^\top]^\top$ together isn't subject to this issue
  - Standardizing features

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish
- **An invalid procedure:** First do variable selection to find features that are individually highly predictive of the outcome, using both training and test set, then train (validate) on training (validation) set alone
- In the above example, cross-validated performance will be an extremely poor measure of generalization (test) performance
- However, joint preprocessing of $[\mathbf{X}_{\text{train}}^\top, \mathbf{X}_{\text{test}}^\top]^\top$ together isn't subject to this issue
    - Standardizing features
    - Dimension reduction

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish
- **An invalid procedure:** First do variable selection to find features that are individually highly predictive of the outcome, using both training and test set, then train (validate) on training (validation) set alone
- In the above example, cross-validated performance will be an extremely poor measure of generalization (test) performance
- However, joint preprocessing of $[\mathbf{X}_{\text{train}}^\top, \mathbf{X}_{\text{test}}^\top]^\top$ together isn't subject to this issue
  - Standardizing features
  - Dimension reduction
  - Discarding rare tokens/words in a document-term matrix

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish
- **An invalid procedure:** First do variable selection to find features that are individually highly predictive of the outcome, using both training and test set, then train (validate) on training (validation) set alone
- In the above example, cross-validated performance will be an extremely poor measure of generalization (test) performance
- However, joint preprocessing of $[\mathbf{X}_{\text{train}}^\top, \mathbf{X}_{\text{test}}^\top]^\top$ together isn't subject to this issue
  - Standardizing features
  - Dimension reduction
  - Discarding rare tokens/words in a document-term matrix
- Why?

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish
- **An invalid procedure:** First do variable selection to find features that are individually highly predictive of the outcome, using both training and test set, then train (validate) on training (validation) set alone
- In the above example, cross-validated performance will be an extremely poor measure of generalization (test) performance
- However, joint preprocessing of $[\mathbf{X}_{\text{train}}^{\top}, \mathbf{X}_{\text{test}}^{\top}]^{\top}$ together isn't subject to this issue
  - Standardizing features
  - Dimension reduction
  - Discarding rare tokens/words in a document-term matrix
- Why?

Cross-Validation

- A word of warning: the training algorithm should never see validation-set outcomes (directly or indirectly)
- Each run of the training algorithm should run entirely independently from start to finish
- **An invalid procedure:** First do variable selection to find features that are individually highly predictive of the outcome, using both training and test set, then train (validate) on training (validation) set alone
- In the above example, cross-validated performance will be an extremely poor measure of generalization (test) performance
- However, joint preprocessing of $[\mathbf{X}_{\text{train}}^\top, \mathbf{X}_{\text{test}}^\top]^\top$ together isn't subject to this issue
  - Standardizing features
  - Dimension reduction
  - Discarding rare tokens/words in a document-term matrix
- Why? These steps doesn't involve access to $\mathbf{Y}_{\text{test}}$

K-fold Cross-Validation

- Randomly divide the set of observations into K groups (folds)

K-fold Cross-Validation

- Randomly divide the set of observations into K groups (folds)
- Start with folds 1 to $(K-1)$ as training set and fold K as validation (held-out fold), $\mathcal{V}_K$, alternate through held-out folds

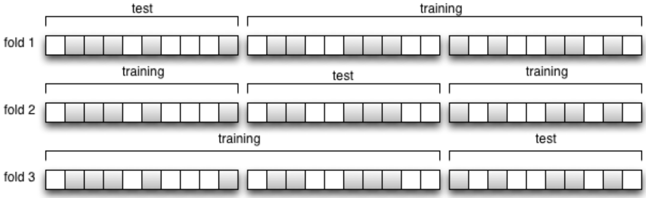
$$\text{CV estimate of MSE } = \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in\mathcal{V}_k}(y_i-\hat{y}_i)^2$$

K-fold Cross-Validation

- Randomly divide the set of observations into K groups (folds)
- Start with folds 1 to $(K-1)$ as training set and fold K as validation (held-out fold), $\mathcal{V}_K$, alternate through held-out folds

$$\text{CV estimate of MSE } = \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in\mathcal{V}_k}(y_i - \hat{y}_i)^2$$

- Three-fold cross-validation example:

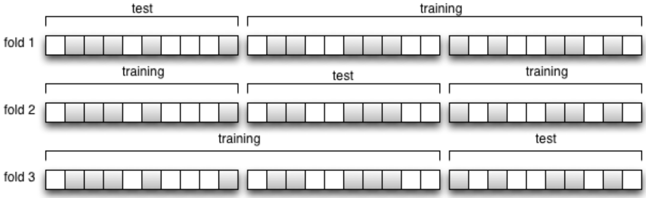


K-fold Cross-Validation

- Randomly divide the set of observations into K groups (folds)
- Start with folds 1 to $(K-1)$ as training set and fold K as validation (held-out fold), $\mathcal{V}_K$, alternate through held-out folds

$$\text{CV estimate of MSE } = \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in\mathcal{V}_k}(y_i-\hat{y}_i)^2$$

- Three-fold cross-validation example:

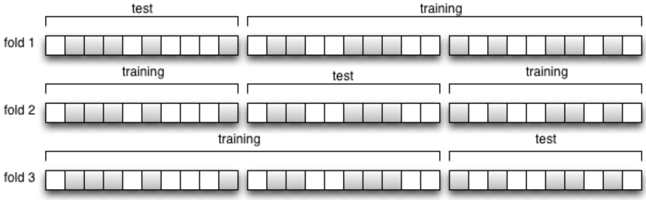


K-fold Cross-Validation

- Randomly divide the set of observations into K groups (folds)
- Start with folds 1 to $(K-1)$ as training set and fold K as validation (held-out fold), $\mathcal{V}_K$, alternate through held-out folds

$$\text{CV estimate of MSE } = \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in\mathcal{V}_k}(y_i-\hat{y}_i)^2$$

- Three-fold cross-validation example:



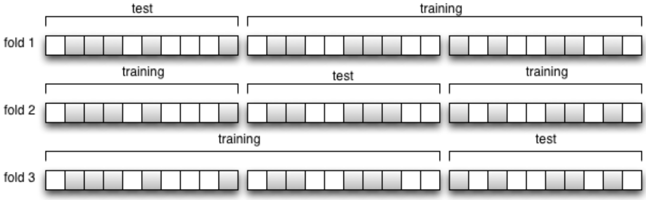
- In practice, K=5 or 10 is often used and works well

K-fold Cross-Validation

- Randomly divide the set of observations into K groups (folds)
- Start with folds 1 to $(K-1)$ as training set and fold K as validation (held-out fold), $\mathcal{V}_K$, alternate through held-out folds

$$\text{CV estimate of MSE } = \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in\mathcal{V}_k}(y_i-\hat{y}_i)^2$$

- Three-fold cross-validation example:



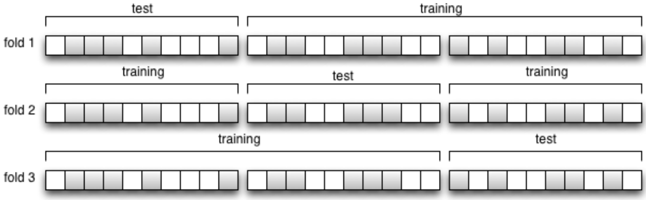
- In practice, K=5 or 10 is often used and works well
- $K=N$ (leave one out): low bias in generalization error

K-fold Cross-Validation

- Randomly divide the set of observations into K groups (folds)
- Start with folds 1 to $(K-1)$ as training set and fold K as validation (held-out fold), $\mathcal{V}_K$, alternate through held-out folds

$$\text{CV estimate of MSE } = \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in\mathcal{V}_k}(y_i - \hat{y}_i)^2$$

- Three-fold cross-validation example:



- In practice, K=5 or 10 is often used and works well
- $K = N$ (leave one out): low bias in generalization error
- Computationally expensive

Evaluation Metrics for Classification

- Infinite number of "scoring rules" to evaluate probabilistic predictions

Evaluation Metrics for Classification

- Infinite number of "scoring rules" to evaluate probabilistic predictions
- "Proper" scoring rules are those that incentivize honest reporting of believed probabilities

Evaluation Metrics for Classification

- Infinite number of "scoring rules" to evaluate probabilistic predictions
- "Proper" scoring rules are those that incentivize honest reporting of believed probabilities
  - Logarithmic (essentially a likelihood-based approach): $\sum_{i=1}^{N} \log()$

Evaluation Metrics for Classification

- Infinite number of "scoring rules" to evaluate probabilistic predictions
- "Proper" scoring rules are those that incentivize honest reporting of believed probabilities
  - Logarithmic (essentially a likelihood-based approach): $\sum_{i=1}^{N} \log()$
  - Quadratic/Brier score for probabilistic predictions, $\sum_{i=1}^{N} \sum_{k=1}^{K} \left(\hat{P}(Y_i = k|\mathbf{X}_i) - \mathbf{1}(Y_i = k)\right)^2$

Evaluation Metrics for Classification

- Infinite number of "scoring rules" to evaluate probabilistic predictions
- "Proper" scoring rules are those that incentivize honest reporting of believed probabilities
  - Logarithmic (essentially a likelihood-based approach): $\sum_{i=1}^{N} \log()$
  - Quadratic/Brier score for probabilistic predictions, $\sum_{i=1}^{N} \sum_{k=1}^{K} \left(\hat{P}(Y_i = k | \mathbf{X}_i) - \mathbf{1}(Y_i = k) \right)^2$
  - Many others

Evaluation Metrics for Classification

- Infinite number of "scoring rules" to evaluate probabilistic predictions
- "Proper" scoring rules are those that incentivize honest reporting of believed probabilities
  - Logarithmic (essentially a likelihood-based approach): $\sum_{i=1}^{N} \log()$
  - Quadratic/Brier score for probabilistic predictions, $\sum_{i=1}^{N} \sum_{k=1}^{K} \left(\hat{P}(Y_i = k|\mathbf{X}_i) - \mathbf{1}(Y_i = k)\right)^2$
  - Many others
- Equivalently, maximizing with a proper scoring rule encourages well-calibrated predictions

Evaluation Metrics for Classification

- Infinite number of "scoring rules" to evaluate probabilistic predictions
- "Proper" scoring rules are those that incentivize honest reporting of believed probabilities
  - Logarithmic (essentially a likelihood-based approach): $\sum_{i=1}^{N} \log()$
  - Quadratic/Brier score for probabilistic predictions, $\sum_{i=1}^{N} \sum_{k=1}^{K} \left(\hat{P}(Y_i = k | \mathbf{X}_i) - \mathbf{1}(Y_i = k) \right)^2$
  - Many others
- Equivalently, maximizing with a proper scoring rule encourages well-calibrated predictions
- Improper scoring rules:

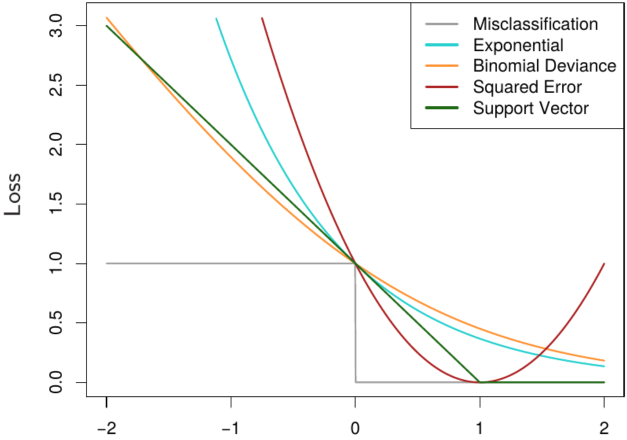
Evaluation Metrics for Classification

- Infinite number of "scoring rules" to evaluate probabilistic predictions
- "Proper" scoring rules are those that incentivize honest reporting of believed probabilities
  - Logarithmic (essentially a likelihood-based approach): $\sum_{i=1}^{N} \log()$
  - Quadratic/Brier score for probabilistic predictions, $\sum_{i=1}^{N} \sum_{k=1}^{K} \left(\hat{P}(Y_i = k|\mathbf{X}_i) - \mathbf{1}(Y_i = k)\right)^2$
  - Many others
- Equivalently, maximizing with a proper scoring rule encourages well-calibrated predictions
- Improper scoring rules:
  - Accuracy (e.g. with imbalanced classes, can often be maximized by always predicting dominant class)

Evaluation Metrics for Classification

- Infinite number of "scoring rules" to evaluate probabilistic predictions
- "Proper" scoring rules are those that incentivize honest reporting of believed probabilities
  - Logarithmic (essentially a likelihood-based approach): $\sum_{i=1}^{N} \log()$
  - Quadratic/Brier score for probabilistic predictions, $\sum_{i=1}^{N} \sum_{k=1}^{K} \left(\hat{P}(Y_i = k | \mathbf{X}_i) - \mathbf{1}(Y_i = k) \right)^2$
  - Many others
- Equivalently, maximizing with a proper scoring rule encourages well-calibrated predictions
- Improper scoring rules:
  - Accuracy (e.g. with imbalanced classes, can often be maximized by always predicting dominant class)
  - Any objective functions that assigns higher loss to misclassification (e.g. is more willing to overpredict cancer than miss it)

Evaluation Metrics for Classification



- Loss for observation with $y_i = +1$ (vs -1). x-axis is $y \cdot f(x)$. This is also related to Hinge Loss.

Evaluation Metrics for Classification

- Confusion matrix: contingency table of true and predicted classes (in validation or test set)

	Actually -1	Actually $+1$
Predicted -1	A	B
Predicted $+1$	C	D

Evaluation Metrics for Classification

- Confusion matrix: contingency table of true and predicted classes (in validation or test set)

	Actually -1	Actually $+1$
Predicted -1	A	B
Predicted $+1$	C	D

- True positive rate: Among observations that are actually positive, proportion predicted to be positive, $TPR = D/(B + D)$

Evaluation Metrics for Classification

- Confusion matrix: contingency table of true and predicted classes (in validation or test set)

	Actually -1	Actually $+1$
Predicted -1	A	B
Predicted $+1$	C	D

- True positive rate: Among observations that are actually positive, proportion predicted to be positive, $TPR = D/(B + D)$
- True negative rate: $TNR = A/(A + C)$

Evaluation Metrics for Classification

- Confusion matrix: contingency table of true and predicted classes (in validation or test set)

	Actually -1	Actually $+1$
Predicted -1	A	B
Predicted $+1$	C	D

- False positive rate: Among observations that are actually negative, proportion predicted to be positive, $FPR = C/(A + C) = 1 - TNR$
- False negative rate: $FNR = B/(B + D) = 1 - TPR$

Evaluation Metrics for Classification

- Confusion matrix: contingency table of true and predicted classes (in validation or test set)

	Actually -1	Actually $+1$
Predicted -1	A	B
Predicted $+1$	C	D

- False positive rate: Among observations that are actually negative, proportion predicted to be positive, $FPR = C/(A + C) = 1 - TNR$
- False negative rate: $FNR = B/(B + D) = 1 - TPR$

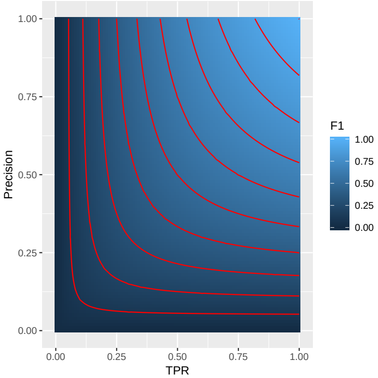
Evaluation Metrics for Classification

- Confusion matrix: contingency table of true and predicted classes (in validation or test set)

	Actually -1	Actually $+1$
Predicted -1	A	B
Predicted $+1$	C	D

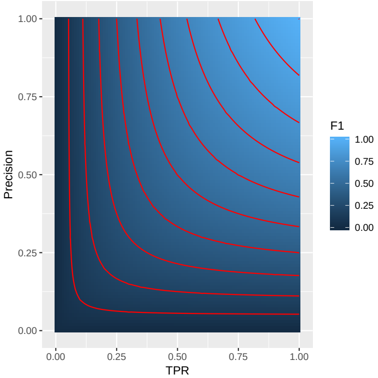
- False positive rate: Among observations that are actually negative, proportion predicted to be positive, $FPR = C/(A + C) = 1 - TNR$
- False negative rate: $FNR = B/(B + D) = 1 - TPR$
- Precision: Among observations that are predicted to be positive, proportion that are actually positive: $D/(C + D)$
- Accuracy: $(A + D)/(A + B + C + D)$

Evaluation Metrics for Classification



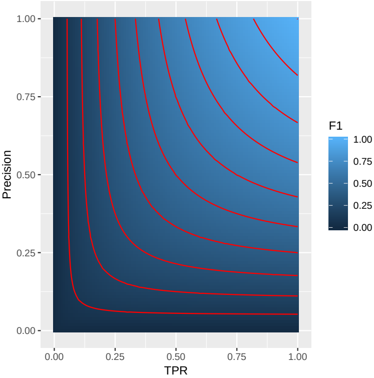
- F_1 score: An ad-hoc objective rewarding both TPR and precision

Evaluation Metrics for Classification



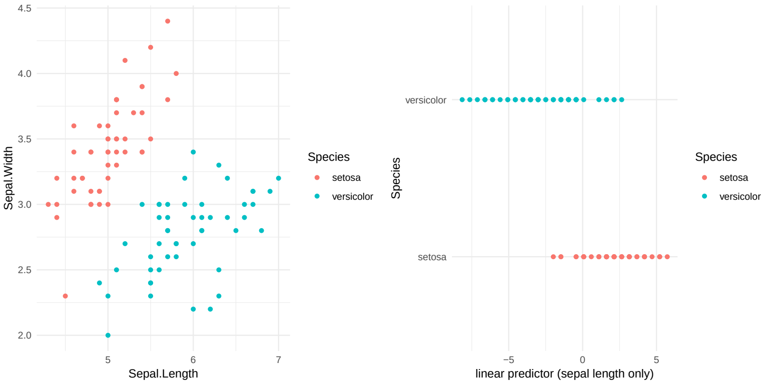
- F_1 score: An ad-hoc objective rewarding both TPR and precision
- Harmonic mean: $F_1 = \left(\frac{TPR^{-1} + \text{Precision}^{-1}}{2}\right)^{-1}$

Evaluation Metrics for Classification



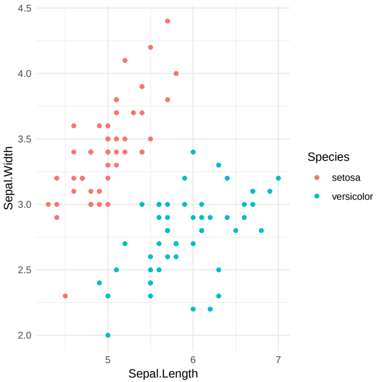
- F_1 score: An ad-hoc objective rewarding both TPR and precision
- Harmonic mean: $F_1 = \left(\frac{TPR^{-1} + \text{Precision}^{-1}}{2}\right)^{-1}$
- Greatest gain can be made by improving whichever performance measure is currently worst

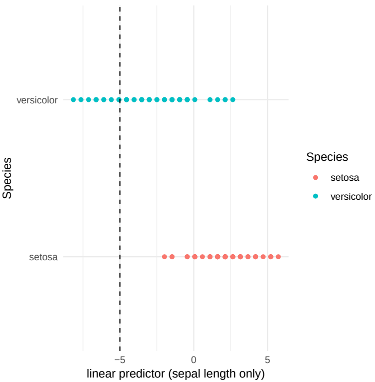
Evaluation Metrics for Classification



- Receiver operating characteristic curve: any threshold for the predictor value, θ, leads to a $FPR(\theta)$ and $TPR(\theta)$
- We can try different threshold values.

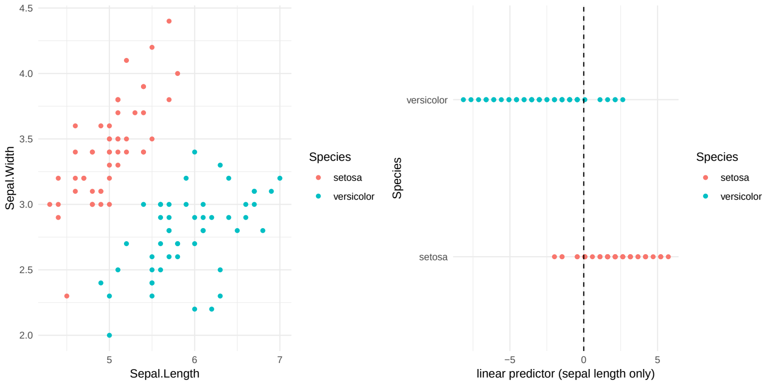
Evaluation Metrics for Classification





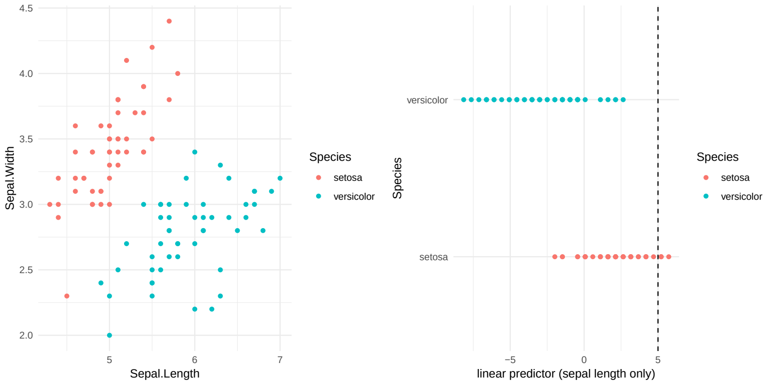
- Receiver operating characteristic curve: any threshold for the predictor value, θ, leads to a $FPR(\theta)$ and $TPR(\theta)$
- We can try different threshold values.

Evaluation Metrics for Classification



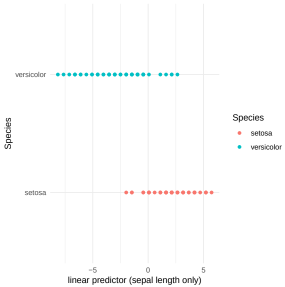
- Receiver operating characteristic curve: any threshold for the predictor value, θ, leads to a $FPR(\theta)$ and $TPR(\theta)$
- We can try different threshold values.

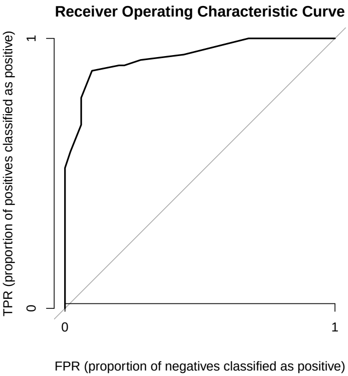
Evaluation Metrics for Classification



- Receiver operating characteristic curve: any threshold for the predictor value, θ, leads to a $FPR(\theta)$ and $TPR(\theta)$
- We can try different threshold values.

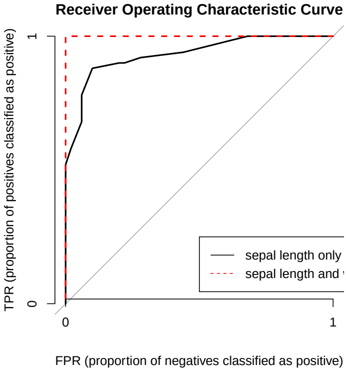
Evaluation Metrics for Classification





- Receiver operating characteristic curve: any threshold for the predictor value, θ, leads to a $FPR(\theta)$ and $TPR(\theta)$
- We can try different threshold values.

Evaluation Metrics for Classification



- Receiver operating characteristic curve: any threshold for the predictor value, θ, leads to a $FPR(\theta)$ and $TPR(\theta)$
- We can try different threshold values.
- ROC will be better if we use more variables.

Evaluation Metrics for Classification

- The area under the curve is often used as a measure of classifier quality $AUC = \int_{-\infty}^{\infty} TPR(\theta) \left(\frac{d}{d\theta} FPR(\theta) \right) d\theta$

Evaluation Metrics for Classification

- The area under the curve is often used as a measure of classifier quality $AUC = \int_{-\infty}^{\infty} TPR(\theta) \left(\frac{d}{d\theta} FPR(\theta)\right) d\theta$
- Define $F(\cdot)$ as the CDF of the predicted scores: $F(\theta) = P(\hat{Y}_i \leq \theta)$

Evaluation Metrics for Classification

- The area under the curve is often used as a measure of classifier quality $AUC = \int_{-\infty}^{\infty} TPR(\theta) \left(\frac{d}{d\theta} FPR(\theta) \right) d\theta$
- Define $F(\cdot)$ as the CDF of the predicted scores: $F(\theta) = P(\hat{Y}_i \leq \theta)$
- Now note that $FPR(\theta) = P(\hat{Y}_i > \theta | Y_i = -1) = 1 - F(\theta | Y_i = -1)$

Evaluation Metrics for Classification

- The area under the curve is often used as a measure of classifier quality $AUC = \int_{-\infty}^{\infty} TPR(\theta) \left(\frac{d}{d\theta} FPR(\theta) \right) d\theta$
- Define $F(\cdot)$ as the CDF of the predicted scores: $F(\theta) = P(\hat{Y}_i \leq \theta)$
- Now note that $FPR(\theta) = P(\hat{Y}_i > \theta | Y_i = -1) = 1 - F(\theta | Y_i = -1)$
- Similarly $TPR(\theta) = P(\hat{Y}_i > \theta | Y_i = +1) = 1 - F(\theta | Y_i = +1)$

Evaluation Metrics for Classification

Note that:

$$\frac{d}{d\theta}FPR(\theta) = -f(\theta|Y_i = -1)$$

Evaluation Metrics for Classification

Note that:

$$\frac{d}{d\theta}FPR(\theta) = -f(\theta|Y_i = -1)$$

$$\begin{aligned} AUC &= \int_{-\infty}^{\infty} P(\hat{Y}_i > \theta|Y_i = +1) \cdot (-f(\theta|Y_i = -1))d\theta \\ &= \int_{\infty}^{-\infty} P(\hat{Y}_i > \theta|Y_i = +1) \cdot f(\theta|Y_i = -1)d\theta \\ &= E[Z > Z'] \end{aligned}$$

Where Z (Z') is randomly drawn predictions from positive (negative) class and reversing sign is because increasing θ decreases FPR

Evaluation Metrics for Classification

- "Hard" (discretized) confusion matrix with K classes:

Evaluation Metrics for Classification

- "Hard" (discretized) confusion matrix with K classes:
  - Encode each unit's outcome as a "one-hot" vector

Evaluation Metrics for Classification

- "Hard" (discretized) confusion matrix with K classes:
  - Encode each unit's outcome as a "one-hot" vector
  - if $K = 3$, then $Y_i = 2 \ \rightarrow \mathbf{Y}_i = [0, 1, 0]^\top$

Evaluation Metrics for Classification

- "Hard" (discretized) confusion matrix with K classes:
  - Encode each unit's outcome as a "one-hot" vector
  - if $K = 3$, then $Y_i = 2 \;\rightarrow \mathbf{Y}_i = [0, 1, 0]^\top$
  - Collect these in $N \times K$ matrix, $\mathbf{Y} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_N]^\top$

Evaluation Metrics for Classification

- "Hard" (discretized) confusion matrix with K classes:
  - Encode each unit's outcome as a "one-hot" vector
  - if $K = 3$, then $Y_i = 2 \rightarrow \mathbf{Y}_i = [0, 1, 0]^\top$
  - Collect these in $N \times K$ matrix, $\mathbf{Y} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_N]^\top$

Evaluation Metrics for Classification

- "Hard" (discretized) confusion matrix with K classes:
  - Encode each unit's outcome as a "one-hot" vector
  - if $K = 3$, then $Y_i = 2 \; \rightarrow \mathbf{Y}_i = [0, 1, 0]^\top$
  - Collect these in $N \times K$ matrix, $\mathbf{Y} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_N]^\top$
  - Similarly, encode each hard prediction as one-hot $\hat{\mathbf{Y}}_i$

Evaluation Metrics for Classification

- "Hard" (discretized) confusion matrix with K classes:
  - Encode each unit's outcome as a "one-hot" vector
  - if $K = 3$, then $Y_i = 2 \ \rightarrow \mathbf{Y}_i = [0, 1, 0]^\top$
  - Collect these in $N \times K$ matrix, $\mathbf{Y} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_N]^\top$
  - Similarly, encode each hard prediction as one-hot $\hat{\mathbf{Y}}_i$
  - Collect these predictions in $\hat{\mathbf{Y}}$

Evaluation Metrics for Classification

- "Hard" (discretized) confusion matrix with K classes:
  - Encode each unit's outcome as a "one-hot" vector
  - if $K = 3$, then $Y_i = 2 \rightarrow \mathbf{Y}_i = [0, 1, 0]^\top$
  - Collect these in $N \times K$ matrix, $\mathbf{Y} = [\mathbf{Y}_1, \cdots, \mathbf{Y}_N]^\top$
  - Similarly, encode each hard prediction as one-hot $\hat{\mathbf{Y}}_i$
  - Collect these predictions in $\hat{\mathbf{Y}}$
  - Then the hard confusion matrix is simply $\hat{\mathbf{Y}}^\top \mathbf{Y}$

Evaluation Metrics for Classification

- Generalizes to "soft" confusion matrix with probabilistic predictions

Evaluation Metrics for Classification

- Generalizes to "soft" confusion matrix with probabilistic predictions
- Represent each prediction with the stochastic vector $\hat{\mathbf{Y}}_i = [\hat{P}(Y_i = 1|\mathbf{X}_i), \cdots, \hat{P}(Y_i = K|\mathbf{X}_i)]^\top$

Evaluation Metrics for Classification

- Generalizes to "soft" confusion matrix with probabilistic predictions
- Represent each prediction with the stochastic vector $\hat{\mathbf{Y}}_i = [\hat{P}(Y_i = 1|\mathbf{X}_i), \cdots, \hat{P}(Y_i = K|\mathbf{X}_i)]^\top$
- Collect predictions in a $N \times K$ row-stochastic matrix, $\hat{\mathbf{Y}}$

Evaluation Metrics for Classification

- Generalizes to "soft" confusion matrix with probabilistic predictions
- Represent each prediction with the stochastic vector $\hat{\mathbf{Y}}_i = [\hat{P}(Y_i = 1|\mathbf{X}_i), \cdots, \hat{P}(Y_i = K|\mathbf{X}_i)]^\top$
- Collect predictions in a $N \times K$ row-stochastic matrix, $\hat{\mathbf{Y}}$
- Then soft confusion matrix is $\hat{\mathbf{Y}}$

Inference for Cross-Validated Evaluation

- First, determine the evaluation measure of interest (e.g. MSE)

Inference for Cross-Validated Evaluation

- First, determine the evaluation measure of interest (e.g. MSE)
- For each held-out fold, calculate the measure

Inference for Cross-Validated Evaluation

- First, determine the evaluation measure of interest (e.g. MSE)
- For each held-out fold, calculate the measure
- Each fold, you obtain an estimate of generalization MSE (somewhat conservatively biased due to training on $N(K-1)/K$ observations instead of N)

Inference for Cross-Validated Evaluation

- First, determine the evaluation measure of interest (e.g. MSE)
- For each held-out fold, calculate the measure
- Each fold, you obtain an estimate of generalization MSE (somewhat conservatively biased due to training on $N(K-1)/K$ observations instead of N)
- Averaging across folds, you obtain an aggregated estimate of generalization MSE

Inference for Cross-Validated Evaluation

- Uncertainty not straightforward: All your foldwise MSE estimates are correlated!

Inference for Cross-Validated Evaluation

- Uncertainty not straightforward: All your foldwise MSE estimates are correlated!

Inference for Cross-Validated Evaluation

- Uncertainty not straightforward: All your foldwise MSE estimates are correlated! Based on overlapping training sets
- In general, options are unsatisfactory:

Inference for Cross-Validated Evaluation

- Uncertainty not straightforward: All your foldwise MSE estimates are correlated! Based on overlapping training sets
- In general, options are unsatisfactory:
  - Try to estimate and correct for the correlations

Inference for Cross-Validated Evaluation

- Uncertainty not straightforward: All your foldwise MSE estimates are correlated! Based on overlapping training sets
- In general, options are unsatisfactory:
  - Try to estimate and correct for the correlations
  - Alternatively, make conservative estimate of variance

Inference for Cross-Validated Evaluation

- Uncertainty not straightforward: All your foldwise MSE estimates are correlated! Based on overlapping training sets
- In general, options are unsatisfactory:
  - Try to estimate and correct for the correlations
  - Alternatively, make conservative estimate of variance
  - For more, see Nadeau, Claude and Yoshua Bengio. 2003. "Inference for the Generalization Error." *Machine Learning*, 53(3).