Data Analysis:
Machine Learning and Regression

Mark Hendricks

August Review

UChicago Financial Mathematics

Outline

In-sample

There is a tendency to over-parameterize models to make them fit the sample data too well.

- Are we fitting the sample-specific noise?
- Parameterizing in-sample noise leads to bad out-of-sample (OOS) performance.

Model fit

R-squared will show fit always improves w/ more parameters.

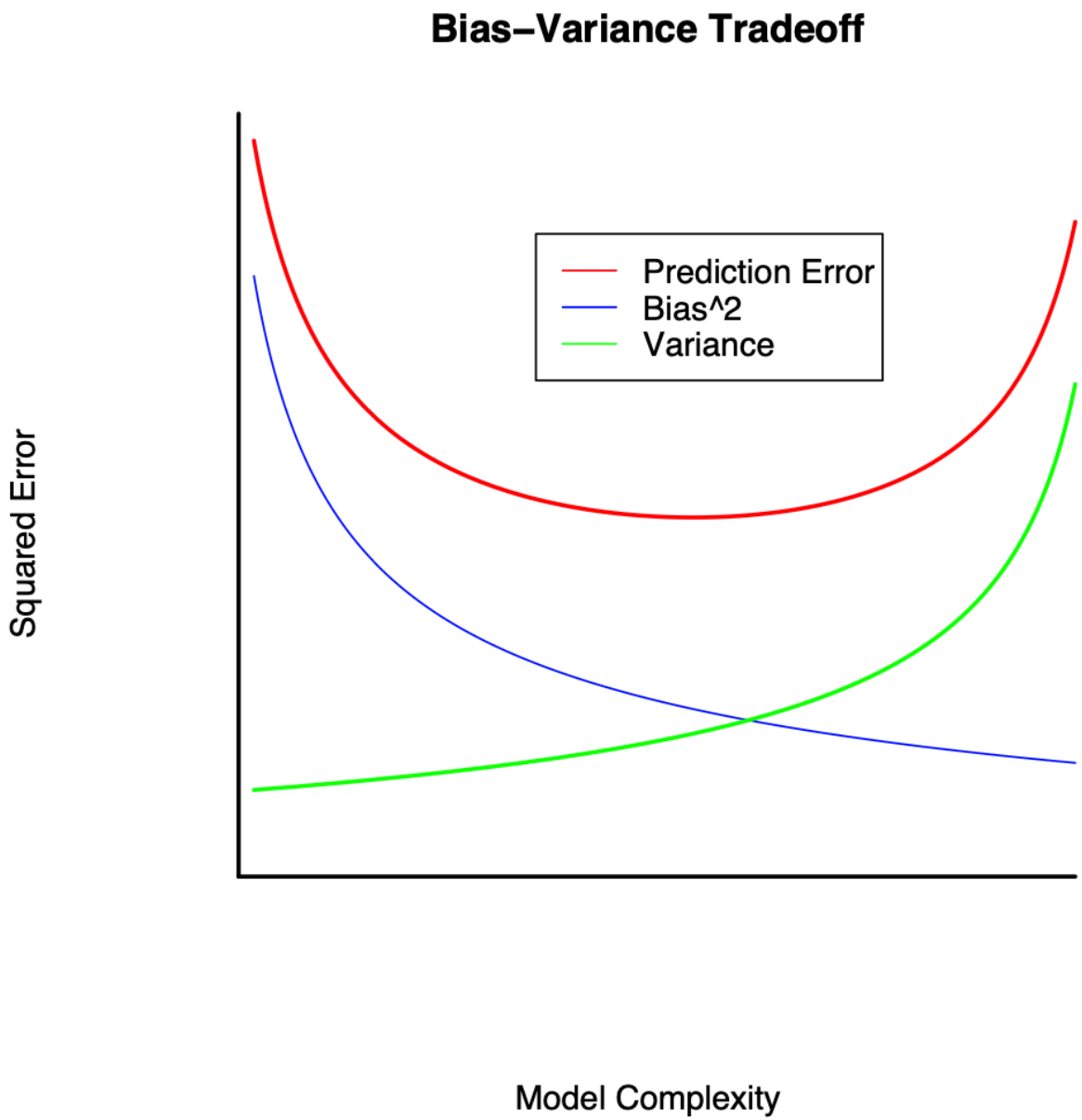
- Adj R-squared popular, but not model specific.
- t-stat dependent on other regressors included.

Bias-variance tradeoff

Most of our machine-learning techniques are dealing with this tradeoff.

$$\underbrace{\text{var}\left[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{b})^2\right]}_{\text{data variance}} = \sigma_\epsilon^2 + \underbrace{(\boldsymbol{X}(\mathbb{E}[\boldsymbol{b}]-\boldsymbol{\beta}))^2}_{\text{bias squared}} + \underbrace{\mathbb{E}\left[(\boldsymbol{X}(\boldsymbol{b}-\mathbb{E}[\boldsymbol{b}]))^2\right]}_{\text{estimator variance}}$$

Where all moments are conditional on $\boldsymbol{X}$.


Bias–Variance Tradeoff
Prediction Error
Bias^2
Variance
Squared Error
Model Complexity

OOS

Out-of-sample (OOS) fit judges the model by sample points excluded from estimation.

- Suppose you have N points in your sample.
- You estimate $\boldsymbol{b}$ from the first n points.

Define,

$$\mathcal{L}^{IS}(\boldsymbol{b}) = \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i'\boldsymbol{b}\right)^2$$

$$\mathcal{L}^{OOS}(\boldsymbol{b}) = \sum_{i=n+1}^{N} \left(y_i - \boldsymbol{x}_i'\boldsymbol{b}\right)^2$$

OOS R-squared

The OOS-R-squared, $\mathcal{R}^2_{oos}$ is the R-squared calculation based on the IS estimation of the model, applied to OOS data.

$$\mathcal{R}^2_{oos} = 1 - \frac{\mathcal{L}^{OOS}(\boldsymbol{b})}{\mathcal{L}^{OOS}(\boldsymbol{b} = 0)}$$

where $\mathcal{L}^{OOS}(\boldsymbol{b} = 0)$ is simply the null model. For instance, our usual $\mathcal{R}^2$ can be written as[1],

$$\mathcal{R}^2_{is} = 1 - \frac{\mathcal{L}^{IS}(\boldsymbol{b})}{\mathcal{L}^{IS}(\boldsymbol{b} = 0)} = 1 - \frac{\sum_{i=1}^n e_i^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

[1] This assumes the regression model has an intercept.

OOS R-squared

If the training data is not representative, it may not give optimal out-of-sample performance.

$$\mathcal{R}^2_{is} = 1 - \frac{\mathcal{L}^{IS}(\boldsymbol{b})}{\mathcal{L}^{IS}(\boldsymbol{b}=0)} = 1 - \frac{\sum_{i=1}^n e_i^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

$$\mathcal{R}^2_{oos} = 1 - \frac{\mathcal{L}^{OOS}(\boldsymbol{b})}{\mathcal{L}^{OOS}(\boldsymbol{b}=0)}$$

Selecting $\boldsymbol{b}$ to maximize $\mathcal{R}^2_{is}$ does not mean it will maximize $\mathcal{R}^2_{oos}$.

Interpreting $\mathcal{R}^2_{oos}$

OOS R-squared is much different than the usual in-sample stat.

- $\mathcal{R}^2_{is} \in [0, 1]$ for models with an intercept (or de-meaned data.)
- $\mathcal{R}^2_{oos}$ can be (and often is) negative, if the model does worse than no model.
- Models with higher $\mathcal{R}^2_{is}$ often have lower $\mathcal{R}^2_{oos}$.

IS estimation by construction improves IS R-squared, but if it is fitting IS noise, then OOS R-squared gets worse.

Getting OOS data

How do we get the OOS data?

- One can hold out the final portion of the sample, but then results are impacted by specifics of this one period.
- A more general approach is to use K-folds, which just means multiple subsamples.

K-Folds

- Randomly split the data into K subsamples.
- For each subsample, estimate the model excluding it and treating it as the OOS data.
- Using completely randomized subsamples can be a problem if the data has serial correlation. In that case, one could use sequential subsamples.

Outline

OLS

OLS solves

$$\boldsymbol{b} = \arg\min_{\boldsymbol{b}_o} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i' \boldsymbol{b}_o\right)^2$$

- This is an unconstrained optimization.
- $\boldsymbol{b}$ minimizes the sum of squared errors.

Question

How would you solve a regression model where the vector, $\boldsymbol{b}$, must satisfy an equality constraint?

- We could add constraints explicitly.
- Or we could add a penalty to the objective function.

Regularized regression

Suppose we have k regressors,

$$\boldsymbol{b} = \arg\min_{\boldsymbol{b}_o} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i' \boldsymbol{b}_o\right)^2 + \lambda \sum_{i=1}^{k} c\left(b_o^i\right)$$

where

- b_o^i is the i-th element of the candidate solution vector, $\boldsymbol{b}_o$.
- $c(\cdot)$ is the regularizing function for non-zero elements of $\boldsymbol{b}$.
- We assume c is a non-negative function with $c(0) = 0$.
- λ is a data-specific parameter.

Question

- If we believe the data are explained by a linear (in β) model,

$$y = \alpha + \boldsymbol{x}'\boldsymbol{\beta} + \epsilon$$

  then how do we justify estimating it with the regularized, (constrained,) model?

- Suppose the classic OLS assumptions hold. Is there any justification for using regularized regressions?

- Is the regularized/constrained estimator biased/consistent? Does the regularized/constrained estimator reduce estimator variance?

Regularizing functions

There are several popular regularizing functions.

Stepwise	L_0	$c(\boldsymbol{b}) = \mathbb{1}_{\boldsymbol{b} \neq 0}$
LASSO	L_1	$c(\boldsymbol{b}) = \lvert\boldsymbol{b}\rvert$
Ridge	L_2	$c(\boldsymbol{b}) = \lvert\boldsymbol{b}\rvert^2$
Elastic Net		$c(\boldsymbol{b}) = \alpha\lvert\boldsymbol{b}\rvert + \frac{1-\alpha}{2}\lvert\boldsymbol{b}\rvert^2$
Log		$c(\boldsymbol{b}) = \log(1 + \lvert\boldsymbol{b}\rvert)$

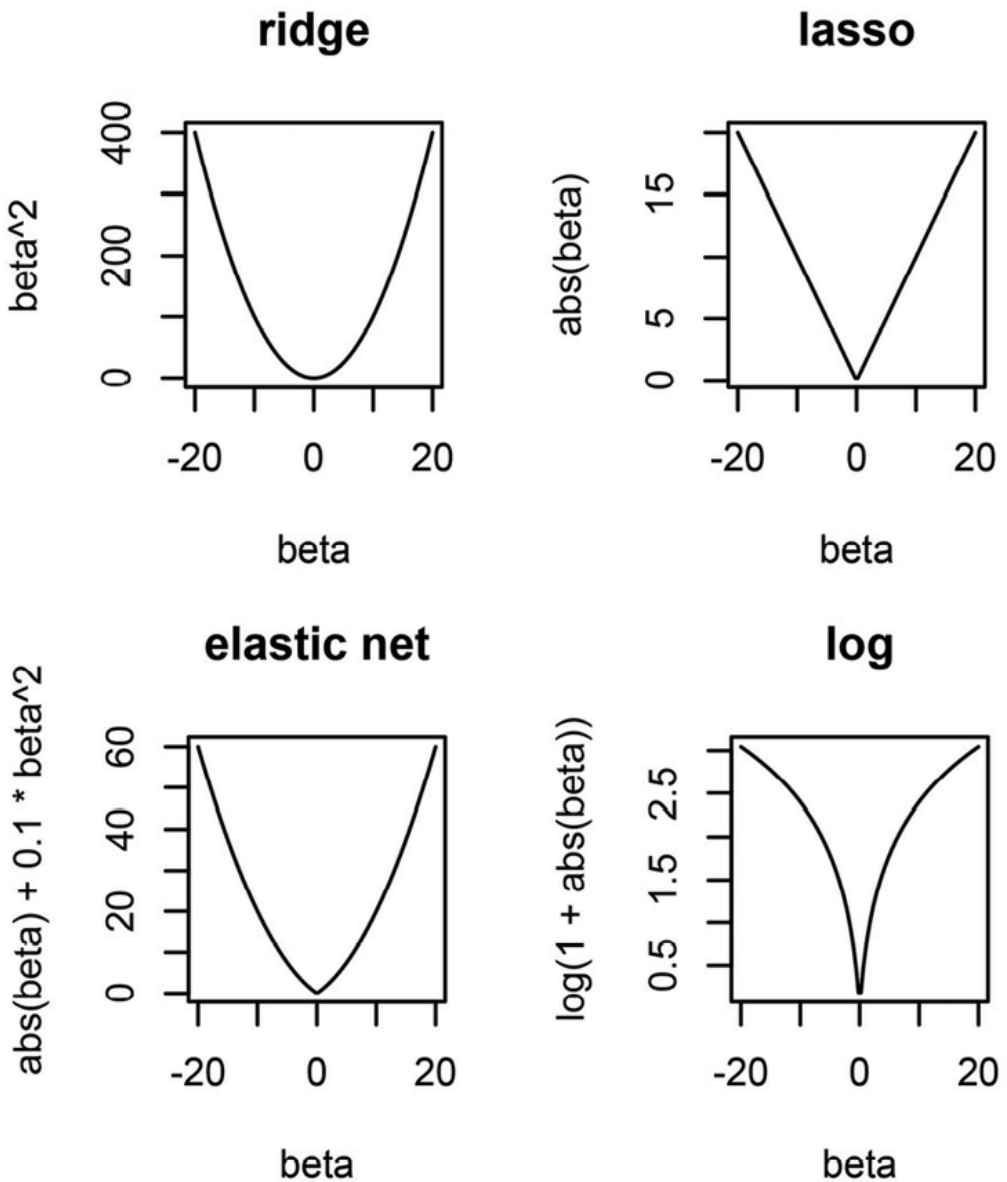


Figure: Source: Taddy, 2019

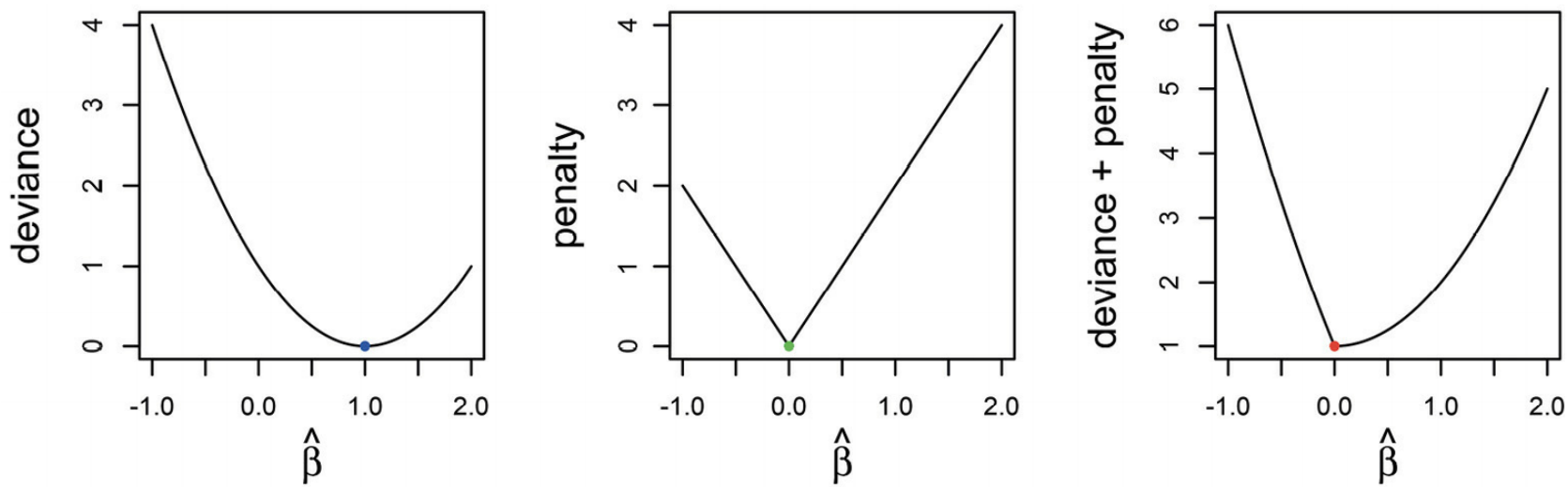


Figure: Source: Taddy, 2019

Question

- Both Ridge and LASSO are in this family of estimators. What is the difference in their specification?
- Which estimator leads to sparse models?

Choosing the regularizing function

Choose based on what you want to penalize.

- LASSO produces many 0 values.
- Ridge allows many non-zero values but severely punishes large estimated effects.
- Elastic net is a mix.
- Log penalizes strongly for non-zero elements, but penalizes weakly for large values.

Question

- Under what circumstances might Ridge be useful?

Ridge Regression

$$\boldsymbol{b}^{\text{ridge}} = \left(\boldsymbol{X}'\boldsymbol{X} + \lambda \mathcal{I}_k\right)^{-1} \left(\boldsymbol{X}'Y\right)$$

- Deals with multicolinearity and ill-conditioning.
- Often is derived from a Bayesian model, related to Black-Litterman.
- Will not reduce dimensionality—non-zero components will still be non-zero.

Question

- Under what circumstances might LASSO be useful?

LASSO Regression

- The constraint does not vanish for small components of $\boldsymbol{b}$, so it forces components to zero.
- Essentially LASSO docks each component of $\boldsymbol{b}$ by a fixed amount, so it still allows very large values.
- LASSO is the most widespread regularized regression.

Nonlinearity

With regularized regression, we lose linearity.

- Scaling of X now matters, because it can't just be offset by rescaling β!
- Try using standard-deviation deflated X.
- Or change regularization function to $\tilde{c}(b^i) = \sigma^i c(b^i)$.[2]

This re-scaling ensures that the regularization applies less to X data that varies little, and thus requires larger β.

[2] Many packages have a setting to do this automatically.

Tuning parameters

The parameter λ is estimated from the data.

- Find the minimum λ such that the estimate is regularized to 0: $\boldsymbol{b} = 0$.
- Estimate the model for a sequence of diminishing λ until getting to $\lambda = 0$, which produces the OLS estimates.

To choose among λ, use

- Information criteria: Akaike (AIC), Bayesian (BIC), etc.
- Cross validation on K-folds.

K-Fold Cross-validation

Use Cross-Validation (CV) on K-folds.

- Split the data into K random subsets.
- For $k = 1...K$,
- Use all data except subsample k to estimate the model.
- Calculate the OOS errors using subsample k.

Regularized regression w/ CV

Use K-fold CV to optimize λ.

- Select a sequence of λ_i.
- Divide into K-folds.
- For each λ_i, estimate the model using K-fold CV to obtain K sets of OOS loss, (such as sum of squared errors.)
- Select the λ_i that minimizes the simple average of the (K sets of) OOS loss values.

Outline

Dimension reduction

Suppose we have a $k \times 1$ vector of data $\boldsymbol{x}_t$, with covariance matrix, Σ. Suppose k is large.

- We may seek a smaller set of variables $p << k$ that explains most of the variation explained by $\boldsymbol{x}$.
- Consider these p factors, $\boldsymbol{z}^1, \boldsymbol{z}^2, ..., \boldsymbol{z}^p$.

Maximizing variation explained

Find the vector, $\boldsymbol{w}^1$ that maximizes

$$\arg\max \boldsymbol{w}'\Sigma\boldsymbol{w}$$
$$s.t. \boldsymbol{w}'\boldsymbol{w} = 1$$

This says $\boldsymbol{w}^1$ is a linear combination of $\boldsymbol{x}_t$ that maximizes the variation explained, rescaled.

Iterating

The full set of factors, $\boldsymbol{w}^i$ could then be developed iteratively...

$$\begin{aligned} &\arg\max \boldsymbol{w}'\Sigma\boldsymbol{w} \\ s.t.\ &\left(\boldsymbol{w}^i\right)' \boldsymbol{w}^j = 0, i \neq j \\ &\left(\boldsymbol{w}^i\right)' \boldsymbol{w}^j = 1, i = j \end{aligned}$$

Factor set

We can iterate to obtain K vectors which will exactly span the space spanned by $\boldsymbol{x}_t$.

- Or we can stop at p, obtaining a reduced dimensionality.
- This dimensionality will maximally explain the space spanned by $\boldsymbol{x}_t$.
- The new space has an orthogonal basis.

The principal components are the factors constructed,

$$\boldsymbol{z}_t^i = \left(\boldsymbol{w}^i\right) \boldsymbol{x}_t$$

Eigenvectors

We do not need to solve the optimization above!

- One can show that $\boldsymbol{w}^i$ is the i-th eigenvector of Σ.
- The i-th eigenvalue gives the total variation explained by the i-th vector.

Thus, PCA is obtained via eigenvector decomposition!

Question: PCA and R-squared

- We have one time-series to explain, $\boldsymbol{Y}$.
- There are 10 explanatory time-series, collected in $\boldsymbol{X}$.
- We calculate the first three Principal Components, $\boldsymbol{Z}$, using standard methods.

Do these principal components maximize the R-squared relative to any other 3-factors extracted from $\boldsymbol{X}$?

Question

- Suppose a bond trader wishes to trade only the 10 most important bonds out of a set of 100 bonds. Will PCA be useful for this selection?

Identification

- PCA reduces the dimensionality—but in a rotated vector space.
- In terms of the original space, any one PC is a combination of every original variable.
- Use LASSO or other regularized regressions to get sparsity.
- PCA is useful for state-space reduction for statistical and mathematical reasons—not for interpretability.

Outline

Stepwise regression

When deciding on the number of regressors, it is common to see the following:

- Include all regressors under consideration.
- Check the t-stats (p-values) of each regressor.
- Re-run the regression on only the statistically significant regressors.

This is known as backward stepwise regression. (BSR).

Problems with BSR

Though common, BSR should be avoided.

- If regressors are strongly correlated, neither may show significance even though one is.
- The p-values are being generated on an overfit model and may have substantial small-sample bias.

Forward Step Regression

Forward Step Regression (FSR) builds up the model instead of cutting it down.

- Suppose you have J regressors under consideration.
- Fit a univariate regression on each of the J regressors.
- Choose the regressor, (call it j,) with the highest in-sample fit (R-squared?).
- Estimate all bivariate regressions that include regressor j.
- Continue until hitting a limit of regressors, a threshold R-squared, or using a selection rule such as AIC/BIC.

Marginal regression

Marginal regression returns a principal factor based on covariation in $\boldsymbol{x}$, but also impact on y.

- For each component of $\boldsymbol{x}$, (column of $\boldsymbol{X}$,) run a univariate regression, (assume we have de-meaned the data,)

$$y = \boldsymbol{\phi}^j x^j + u$$

- Stack up these univariate regression coefficients, ϕ^j into a vector, $\boldsymbol{\phi}$.
- Calculate the factor,

$$z^1 = \boldsymbol{x}'\boldsymbol{\phi}$$

- Use z^1 to analyze/predict y via OLS:

$$y = z^1\beta^1 + \epsilon$$

Partial Least Squares

Instead of stopping and analyzing y given z^1, repeat the marginal regression.

- Namely, take the sample residuals e from the regression of y on z^1. Do a marginal regression on them to obtain z^2.
- Use z^1 and z^2 to analyze/predict y with OLS.
- Keep repeating for more factors.

Question

- What would it mean to apply "boosting" to marginal regression?
- In the bias-variance tradeoff, what is boosting meant to help?

Boosting

Boosting refers to iteratively using a (typically simple model.)

- For instance, we can "boost" the marginal regression model by re-running it on the residuals from the final predictions.
- Thus, boosting builds new predictions in a sequence from simpler predictions.
- PLS is simply "boosted" MLS.
- Boosting is useful if we have a model with low variance but substantial bias.

Regression trees

Classification and Regression Trees (CART) are a supervised learning tool.

- For $\boldsymbol{x}$ and y, the CART groups y according to thresholds of the associated x data.
- There are an impossible number of potential trees / splits.
- CART starts at the top and seeks the component of x, and its threshold value, that minimizes,

$$\sum_{i \in \text{left}} \left(y^i - \bar{y}^{\text{left}} \right)^2 + \sum_{i \in \text{right}} \left(y^i - \bar{y}^{\text{right}} \right)^2$$

- CART can model complex relationships but is prone to overfitting.

Question

- How do we get random forests from regression trees?
- Why are random forests useful?
- Does a forest improve the bias or the variance of an individual tree?

Bagging

Bootstrap aggregating, or "Bagging", is useful when an estimator is unbiased but highly variable.

- Run the model over many bootstrapped samples.
- This yields many observations from the variable, unbiased estimator distribution.
- Take the average, and you have a less variable, unbiased estimator.

Supervised vs unsupervised

- Regression models are supervised. They model x based on feedback from associated y data.
- PCA, k-means clustering, etc. are unsupervised. They organize x without any feedback from y.
- Partial Least Squares (PLS) takes a PCA-type factorization of x based on information in y.