* In traditional regression models, n > p
  + N: number of data points (sample size)
  + P: number of parameters
* If p is too high, the model gets overfitted
  + Training data works very well, testing data fails
* Modern models can collect lots of parameters, so p >> n
  + High dimensionality
* Some features contribute to overfitting:
  + Unimportant/irrelevant ones
  + Highly correlated features (multicollinearity)
  + Interaction terms, polynomial terms, etc
* Goal: only keep features that add substance to model
  + Reduces variance
  + Increases interpretability of the model
* Flexibility = number of parameters = p
* Low variance: n >> p (low-dimensional setting)
* N ~= p: high variance, overfitting
* P > n: infinite variance (high-dimensional setting)
* Options for feature selection: subset selection, shrinkage, dimension reduction

Subset selection: run the model with different combinations of variables, pick model with lowest RSS

* Con: very computationally intensive (total of 2^p possible models- infeasible with p >= 40)
* Can lead to overfitting: with a very large search space, it’s easy to find a model that works well for training but not for test
* Pro: easy to understand
* R library: leap
* Function: regsubsets(): automatically picks best model
* Process of subset selection:
  + Start with null model M0 (no features)
  + Find the best M1 (model with 1 predictor)
    - Best model: largest R2 or smallest RSS
  + Continue with M2, M3, M4, etc
  + Choose the best model from M0, M1, M2, …
    - Use cross-validated error, AIC, BIC, or adjusted R2
    - Want smallest test error, not training

Deviance: analogous to RSS for non-linear regression models

Forward subset selection:

* Process:
  + Start with M0: model with no predictors
  + Create M1 by adding variable that gives greatest additional improvement to fit (highest R2/lowest RSS)
  + End up with M0, M1, M2, …
  + Choose best Mk (cross-validation, AIC, BIC, or adjusted R2)
* Forward selection is computationally better than full subset selection
* Not guaranteed to result in best model
* Differs from best subset if there is correlation between the features

Backward stepwise selection: start with all features, eliminate one at a time until the best model is reached

* Pro: more computationally efficient than full selection
* Cons: can’t run if p > n (more features than samples), the final model is not always the best model
* Regsubsets(method= “backward”)

Choosing the best model: can’t use RSS or R2 to select among models with different number of features

* Larger models will always be better than or equal to smaller ones (with training error)
* Goal: estimate test error for a model
  + Indirect approach: compute training error and add some adjustment due to overfitting
  + Direct approach: cross-validation
* Cp, AIC, BIC, adjusted R2: all indirect approaches
* Want to minimize Cp, AIC, BIC
* Cp =
  + D: number of parameters (predictors + intercept)
    - For M3 (3 predictors), d = 4
  + Sigma squared: estimate of the variance of the error associated with each response measurement
  + N: number of observations (rows)
  + Only works when n > p
* AIC = -2log(L) + 2d
  + Linear model: -2log(L) = RSS/(sigma^2)
  + For linear model, AIC is proportional to Cp
* BIC =
  + Heavier penalty on models with more parameters (compared to Cp)
  + More likely to select smaller model
* TSS: total sum of squares =
  + Y bar is the average y response
* R2 =
* Adjusted R2: adjusts for models with different number of parameters
  + Adjusted R2 =
  + Pays a price for unnecessary features in the model
* AIC/cp/BIC/adjusted R2 are hard to do with ridge/lasso: the value of d is not known

Cross-validation: direct approach for choosing the best model

* Validation: divide data into 2 parts, train the data on the training set, evaluate on the testing part
* Cross-validation: divide the data into a number of parts
  + At each part, train the model on all but 1 part, test on the last part
* After doing subset selection, do cross-validation for each model Mk, and then select the model with the lowest cross-validated error
* Often better than cp/AIC/BIC: direct estimate of test error without having to estimate sigma^2
* Applicable to wider range of models
  + No need to estimate sigma^2
  + Don’t need to know d (which might be the case in ridge/lasso)
* Use the one-standard-error rule for cross-validation

Shrinkage (regularization): reduce the beta coefficient for unimportant features

* Previous models use least squares, shrinkage does not

Ridge regression: minimize

* Pay a price for having large coefficients -> shrink coefficients towards 0
* Lambda: tuning parameter (determine with cross-validation)
  + If lambda = 0, it’s just least squares
  + Large lambda: all coefficients go to 0
* L2 norm:
* Common to plot l2 norm(ridge regression)/l2 norm(least squares)
  + Small lambda: ridge = least squares, ratio is 1
  + Large lambda: ridge coefficients = 0, ratio is 0
* Ridge regression is sensitive to scaling
  + Should always standardize features before doing ridge regression
  + Divide by standard deviation

Lasso regression: similar to ridge regression, but the coefficients can be 0

* Minimize
* L1 norm:
* If lambda is large enough, a coefficient can become 0
  + Subset selection
* Usually good to use ridge and lasso together, then pick the best one
* Must use cross-validation to get lambda for ridge and lasso
  + Can’t use AIC/BIC/other indirect methods: we don’t know what d is

Dimension reduction: use least squares on a set of new predictors

* New predictors: linear combinations of the original predictors
* Fit a least squares model to the new predictors Z1, Z2, …Zm
  + Coefficients: theta1, theta2, …theta\_m
  + Constrains the possible space of beta terms
* Similar to ridge/lasso: we’re constraining the beta coefficients, just in a different form
* Can result in low bias and low variance
* Only works if M < P (fewer new features than old features)
* PCR: type of dimension reduction
  + Perform PCA to get linear combinations of variables
  + Apply least squares on new predictors
* First principal component: normalized linear combination of variables with the largest variance
* Second principal component: largest variance that’s not correlated with the first
* Replace correlated original variables with a representation that captures their joint variance
* Adding components increases variance, decreases bias
* MSE = bias^2 + variance
* Use cross-validation to choose number of dimensions
* PCR is unsupervised: the response is not used to help determine the principal component directions
  + No guarantee that the directions that describe the predictors will also predict the response
* Partial least squares (PLS): supervised version
  + Makes sure the directions predict the response
  + Puts highest weight on variables that are most correlated
  + Similar to ridge/PCR: doesn’t tend to work any better
  + Find M different linear combinations (projections) of the features
* Forward stepwise selection: start with no features, add one feature at a time (k) until all features (p) are in the model
  + At each step, add feature that gives greatest improvement to the model
  + Approximately (p^2)/2 possible models- much more computationally efficient than subset selection
  + Con: the final model is not always the best
* Hybrid approach: blend forward and backward
  + Only available on some software
  + More computationally efficient than subset selection
* Choosing the optimal model should be based on test error, not training error (RSS, R^2)
  + Training set MSE is an underestimate of the test MSE
  + Least squares method minimizes training MSE, not testing MSE
  + Can estimate the test error indirectly by adjusting the training error
* C\_p: add a penalty based on the estimated variance of residuals
  + Choose the model with the lowest C\_p
  + d is the number of features used in the model (d <= p)
  + A higher value of d (bigger model)
* AIC, BIC are similar ways of estimating test error with some penalty
* Can estimate test error directly with validation or cross-validation
* One standard error rule: law of parsimony: choose the simplest model (model with smallest p)
  + Calculate standard error of estimated MSE for each p
  + Select smallest model for which test MSE is within 1 standard error of the lowest point on the curve
* Model selection involves selection of features
* Cost function is also called the loss function
  + Should be minimized
* To standardize a feature, must divide it by standard deviation
  + Shows the number of standard deviations your data point is away from the mean
  + A typical datapoint should fall between -3 to 3 standard deviations