AI in Marketing: Overfit and Regularization

Model Selection

- Model selection can include deciding which X variables to include in the model.
- In our example, we used $y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$. We could have also used
  - $y = \beta_0 + \beta_1 X_1$, or
  - $y = \beta_0 + \beta_2 X_2$
- When you have many X variables, choosing some to hold out of the model can actually *improve* your predictions.
- (More generally, models can be made more or less complex. For the linear models we focus on here, adding more X variables makes them more complex. For other types models, there are additional ways to increase the complexity.)

Introduction

Key concepts in supervised machine learning:

- **Overfit**
  - Too many variables in a predictive model => Bad predictions.
  - The optimal collection of variables depends on many factors including:
    - The number of observations in the data set.
    - How correlated the variables are.
    - How much measurement error is in the variables.
  - We'll look at some examples.
- **Regularization**
  - Attempts to "solve" the problem of overfit.
  - Fundamental part of modern machine learning algorithms.
  - Selects variables to be included in the model.
  - Removes variable that will make its predictions worse.
  - Selects a model that generalizes to new data sets from the same source.

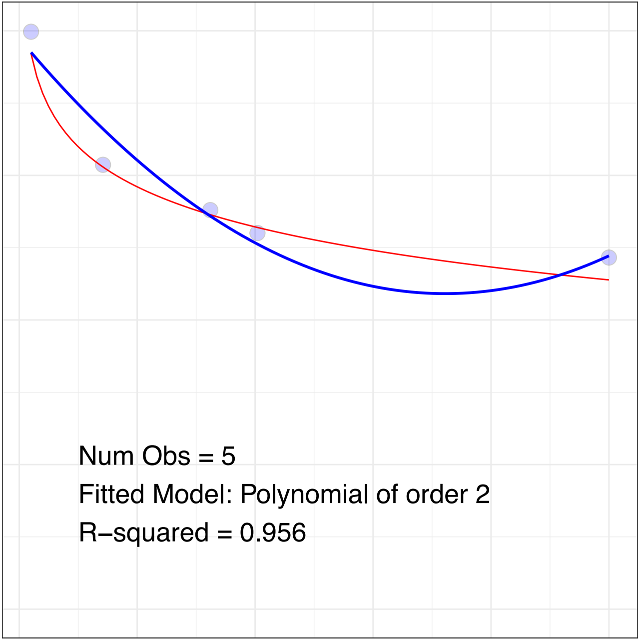
Example Introduction

- In the following example, we will present y as a function of X.
- To increase the complexity of our model, we will add polynomials of X.
  - E.g., X^2, X^3... etc.
- We will look at the following models:
  - $y = \beta_0 + \beta_1 x$
  - $y = \beta_0 + \beta_1 x + \beta_2 x^2$
  - $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$

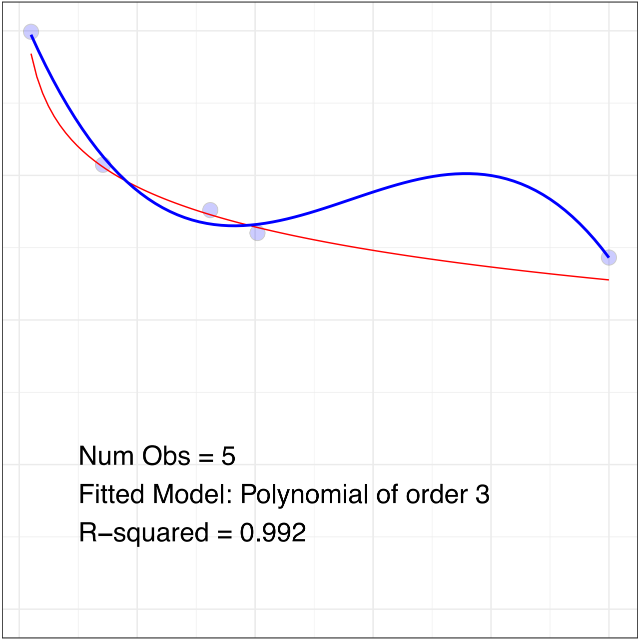
Consider a Function of one variable

Fit with a Linear Model

Fit with a 2nd-order Polynomial



Fit with a 3rd-order Polynomial



Which Fit is Preferred?

- In reality, we rarely know the true model.
  - As a result, we rely on models that can approximate the true model
- R^2 values improved each time we added an extra polynomial term
  - We expect R^2 will continue to increase with more polynomial terms
- But which model will give us the best predictions?

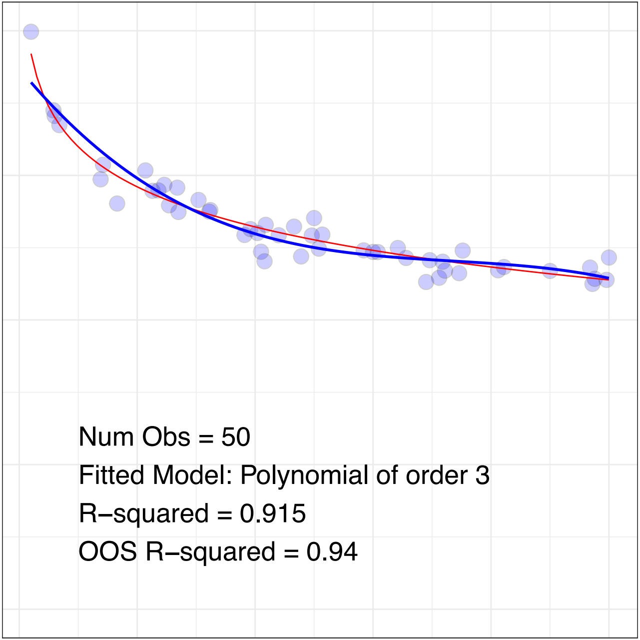
Holdout Predictions

- Imagine we had some additional data that was left out of the plots shown above
  - Importantly, these holdout data weren't used to fit our polynomial models
- Then, we could compare our models' predictions against those holdout data to compute the ***holdout,*** *or* ***out-of-sample (OOS)*** R^2:

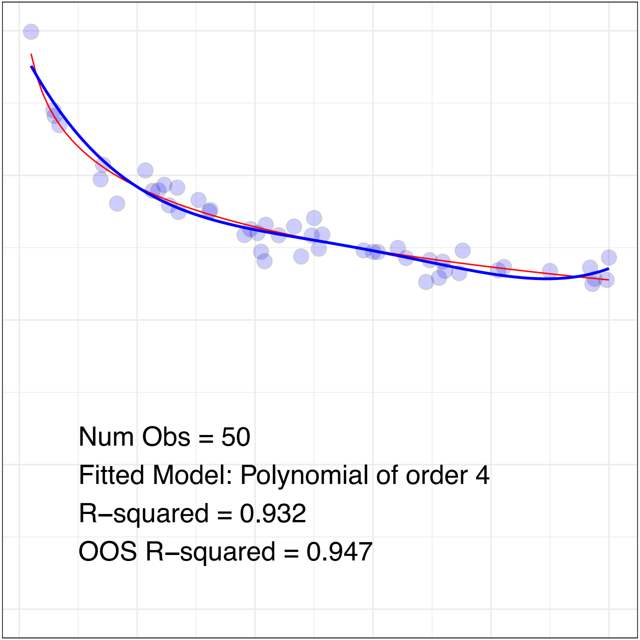
Poly order	R^2	OOS R^2
1	0.701	0.744
2	0.956	0.856
3	0.992	0.515

- ***Overfit*** caused the holdout R^2 to drop when adding the $\beta_3 x^3$ term to the model
  - We had too many variables and too few observations to fit a 3rd-order polynomial
- Now, let's repeat this exercise with 50 observations...

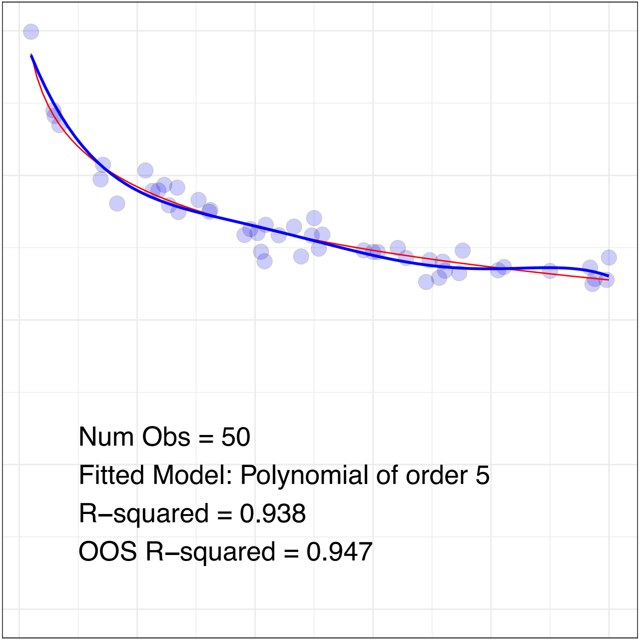
50 Observations and 3rd-order Poly



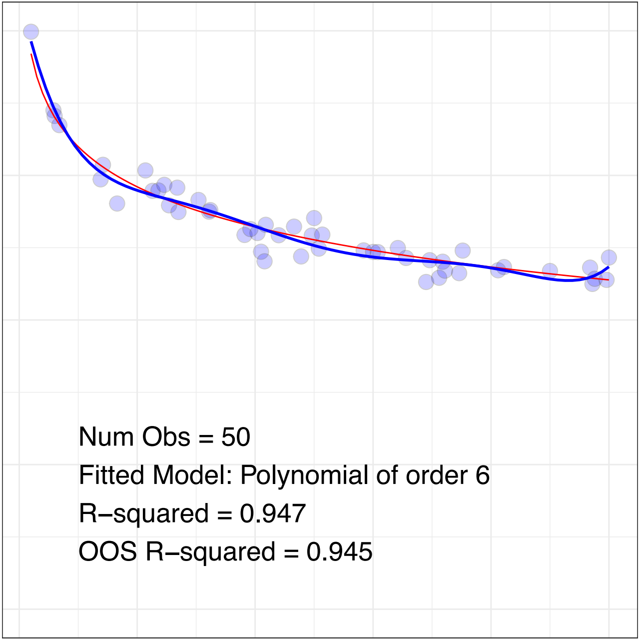
50 Observations and 4th-order Poly



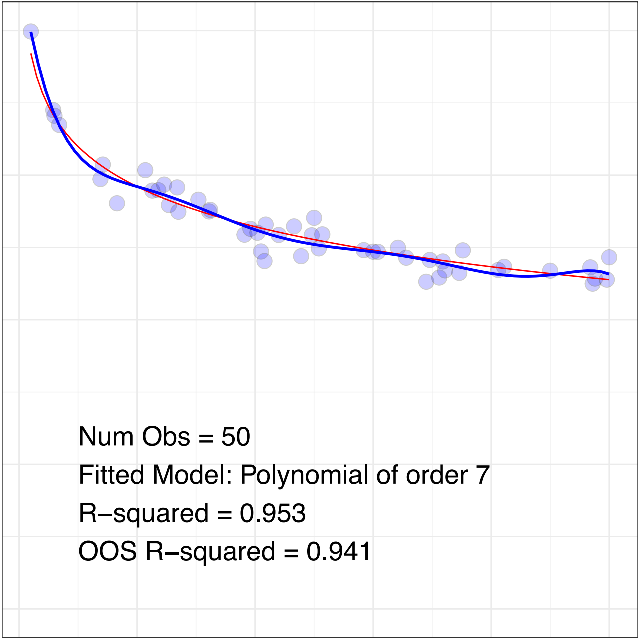
50 Observations and 5th-order Poly



50 Observations and 6th-order Poly



50 Observations and 7th-order Poly



50 Observation study Review

Poly order	R^2	OOS R^2
3	0.915	0.94
4	0.932	0.947
5	0.938	0.947
6	0.947	0.945
7	0.953	0.941

- This time, the 3rd-order polynomial could be improved upon
  - We had enough data to improve the OOS R^2 with $\beta_3 x^3$, $\beta_4 x^4$ and terms
  - The $\beta_5 x^5$ term didn't change the OOS R^2
  - However, the addition of the $\beta_6 x^6$ caused the OOS R^2 to drop due to **overfit**
- Now, what if y is a function of more than one x variable?
  - Algorithms can find the optimal model for us

Overfit and Regularization Recap

- All of the steps we have shown here can be automated by a computer.
- While there are wide variety of Machine Learning methods, they generally do some version of what we have done here:
    1. **Select a Model**: Try out different inputs (e.g, X variables) to include in the model.
    2. **Calibrate the Model**: estimate the $\hat{\beta}$ values using the training data.
    3. **Validate the Models**: measure the quality of the predictions on the validation sample.
    4. **Repeat**: the model that makes the best predictions can be put into use for when don't have y values.

What Causes Overfit?

- No data set perfectly represents its source.
- All data contain some random noise.
- A complex model can “learn” about this noise and predict it in new data sets.
- Predictions based on the noisy features of the original data set don’t generalize to new data sets.
- Models that try to predict the noise unique to one data set suffer from **overfit**.
  - More complex model have more opportunities to learn the noise.

Technical Note on R^2

- In the overfit examples using polynomials, I used R^2 as a measure of prediction quality for continuous outcomes.
- Another common measure used in these applications is Mean Squared Error (MSE)
  - $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ or Root Mean Squared Error (RMSE), $\text{RMSE} = \sqrt{\text{MSE}}$
  - y_i are the observed outcomes
  - $\hat{y}_i$ are the predicted outcomes
- R^2 and RMSE are very closely related, differing only by a linear transformation, e.g. $R^2 = a + b \times \text{RMSE}$.
- I find interpreting R^2 more intuitive, but RMSE is commonly used for the same purpose.
  - R^2 measure how good the fit is.
  - RMSE measure how bad the fit is.

Extension to Holdout Sampling: Cross-Validation

Cross Validation Procedure:

- Divide randomly divide the entire data set into k "folds."
  - k is typically in the 4–10 range.
  - k can be larger for small data sets, e.g. < 100 observations.
- Assign one of the folds as the holdout sample. Train using data across all of the other folds.
- Repeat with each fold taking a turn as the holdout sample.

Cross Validation Benefit:

- Instead of just one measure of prediction quality, we have K measures.
- The average of these measures may be more accurate
  - Just one holdout sample could be (un)lucky.
  - Harder for luck to drive the result with multiple holdout samples.
- The variance of prediction quality is informative
  - Tells us about the range of possible outcomes on new data.
- Particularly useful when there are a small number of observations.

Extension to Holdout Sampling: Cross-Validation

Cross Validation Procedure:

- Divide randomly divide the entire data set into k "folds."
  - k is typically in the 4–10 range.
  - k can be larger for small data sets, e.g. < 100 observations.
- Assign one of the folds as the holdout sample. Train using data across all of the other folds.
- Repeat with each fold taking a turn as the holdout sample.

Cross Validation Benefit:

- Instead of just one measure of prediction quality, we have K measures.
- The average of these measures may be more accurate
  - Just one holdout sample could be (un)lucky.
  - Harder for luck to drive the result with multiple holdout samples.
- The variance of prediction quality is informative
  - Tells us about the range of possible outcomes on new data.
- Particularly useful when there are a small number of observations.

Cross-Validation

4-fold validation (k=4)

LASSO Regression

How to Automate Model Selection?

- For the standard linear model, we find the β values by minimizing the sum of squares:

$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{k=1}^{p} x_{ik}\beta_k \right)^2$$

- Adding more x variables will always improve the in-sample R^2
- But, as we saw above, the OOS R^2 will get worse if we add too many
  - Solution: add another term that **penalizes** adding more x variables:

$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{k=1}^{p} x_{ik}\beta_k \right)^2 + \lambda \sum_{k=1}^{p} | \beta_k |$$

- The $\lambda \geq 0$ is a tuning parameter that determines the optimal number of x terms.
  - λ allows for **regularization**
  - The optimal λ can be found by repeatedly performing cross-validation
- Intuition for penalty term: "Let's not get too excited about any observed correlation between x and y. It might be spurious and lead to overfit. It's better to bias the β estimates toward zero."

How to Automate Model Selection?

- For the standard linear model, we find the β values by minimizing the sum of squares:

$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{k=1}^{p} x_{ik} \beta_k \right)^2$$

- Adding more x variables will always improve the in-sample R^2
- But, as we saw above, the OOS R^2 will get worse if we add too many
  - Solution: add another term that **penalizes** adding more x variables:

$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{k=1}^{p} x_{ik} \beta_k \right)^2 + \lambda \sum_{k=1}^{p} | \beta_k |$$

- The $\lambda \geq 0$ is a tuning parameter that determines the optimal number of x terms.
  - λ allows for **regularization**
  - The optimal λ can be found by repeatedly performing cross-validation
- Intuition for penalty term: "Let's not get too excited about any observed correlation between x and y. It might be spurious and lead to overfit. It's better to bias the β estimates toward zero."

How to Automate Model Selection?

$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{k=1}^{p} x_{ik}\beta_k \right)^2 + \lambda \sum_{k=1}^{p} \mid \beta_k \mid$$

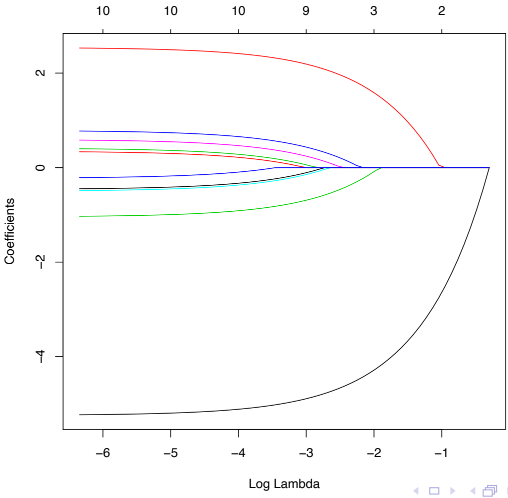
- This expression is for a **LASSO estimator** (Least Absolute Selection and Shrinkage Operator)
  - $\lambda = 0$ means there is no penalty for adding x terms (same as a standard linear model)
  - As λ increases, the penalty of adding x terms increases
  - As $\lambda \to \infty$ the price of adding x terms will get too high, and none will be included in the model
- Note: The LASSO regression should be applied to standardized inputs:

$$x'_{ik} = \frac{x_{ik}}{\mathrm{sd}(x_k)}$$

  - This makes the magnitudes and the cost of all inputs comparable
  - Most packages will have an option to do this for you

Algorithm steps I

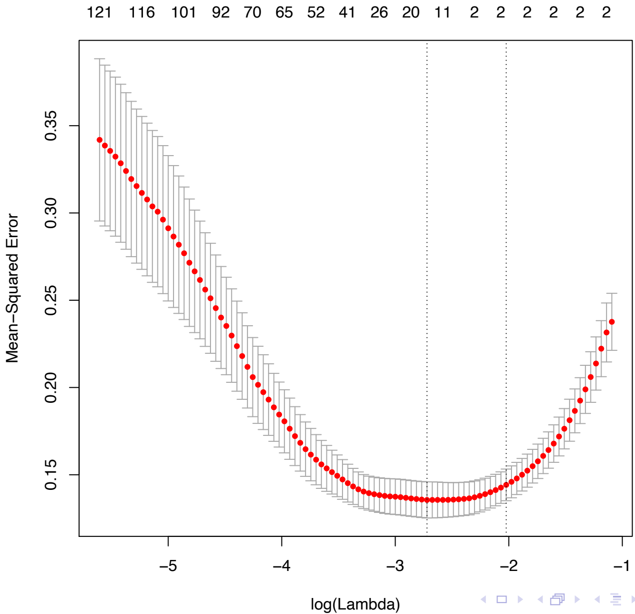
1. Set λ high so that no x values are included in the model
2. Lower it gradually until more x variables are included
3. Track their corresponding β estimates. E.g.,



Algorithm Steps II: Cross-validation

1. Take the training data and randomly split them into K **folds**
    1. Most commonly used: $K = 10$
2. Choose some value for the tuning parameter, λ (perhaps from algorithm in the last slide)
3. Pick one of the folds, k, and set it aside as a validation data set (or "leave out").
4. Estimate the model using the data in the other folds: $1, ..., k-1, k+1, ..., 10$
5. Predict the output y_i in fold k based on the model estimates and record the MSE
6. Repeat for all folds, k, and compute the average MSE over all folds
    1. This gives an **out-of-sample prediction error** for the chosen tuning parameter, λ.

Cross-validation error curve

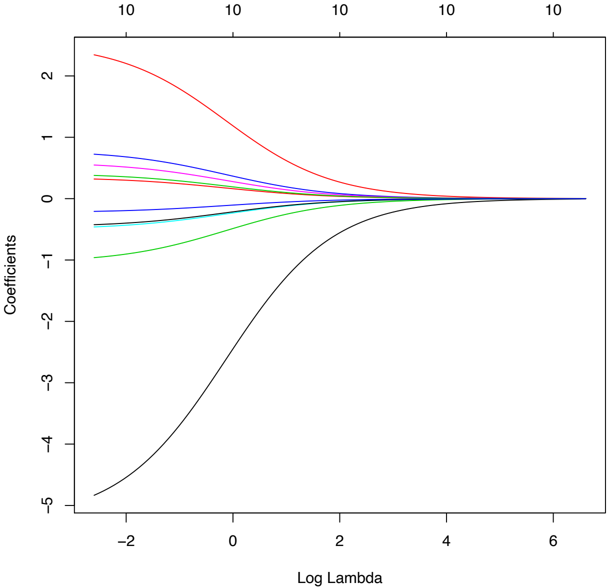


Variable Shrinkage: Ridge Regression

- **LASSO** choses which variables to include in the regression and which to drop
- Alternative: **Ridge Regression**.
- **Ridge regression** typically admits all variables (though some have very small coefficients)
  - small coefficient values also help to reduce overfit

$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{k=1}^{p} x_{ik}\beta_k \right)^2 + \lambda \sum_{k=1}^{p} \beta_k^2$$

Ridge Coefficients



Ridge Cross-validation error curve