FOR INSTRUCTOR PURPOSES ONLY

# INSTRUCTOR NOTES

- Insert Text Here

FOR INSTRUCTOR PURPOSES ONLY

# MATERIALS

- Insert Text Here

# FOR INSTRUCTOR PURPOSES ONLY

## PRE-WORK

- Insert Text Here

GENERAL ASSEMBLY

# EVALUATING MODEL FIT

*Chris Connell*

*Data Scientist*

# LEARNING OBJECTIVES

After this lesson, you should be able to:

- How to conduct linear regression modeling
- Use interaction effects and dummy categorical variables
- Understand model complexity, underfitting, right fit, and overfitting
- Define regularization and error metrics for regression problems

COURSE

# PRE-WORK

# PRE-WORK REVIEW

- Understand goodness of fit (r-squared)
- Measure statistical significance of features
- Recall what a residual is
- Implement a sklearn estimator to predict a target variable

# Review

# REVIEW

- Simple and Multiple Linear Regressions
- Common regression assumptions; how to check for them
- OLS (Ordinary Least Squares)
- How to interpret the model's parameters
- Variable Transformations
- Inference, Fit, $R^2$ (r-square), and $R^{2}_{adj}$ (adjusted $R^2$)
- Multicollinearity

OPENING

# R-SQUARED AND RESIDUALS

# WHAT IS R-SQUARED?  WHAT IS A RESIDUAL?

- R-squared,  the central metric introduced for linear regression
- Which model performed better, one with an r-squared of 0.79 or 0.81?
- R-squared measures explain variance.
- But does it tell the magnitude or scale of error?
- We’ll explore loss functions and find ways to refine our model.

# ADJUSTED R-SQUARED

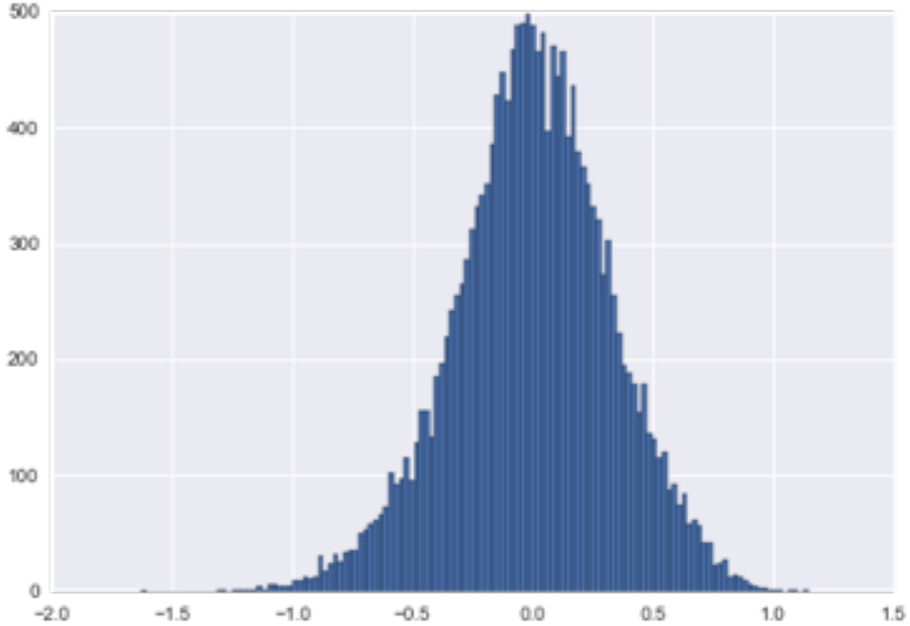
- The adjusted R-squared is a modified version of R-squared for the number of predictors in a model

- R-squared will always increase with the addition of predictor variable, but adjusted R-squared only increases when a new variable is added if the new variable is significant

# F-statistic

# RECALL:  WHAT'S RESIDUAL ERROR?

- In linear models, residual error must be normal with a mean close to zero.
- Individual residuals are useful to see the error of specific points, but it doesn't provide an overall picture for optimization.
- We need a metric to summarize the error in our model into one value.



# F-STATISTIC

## What $\beta_i$ would make our multiple linear regression model useless?

- (the multiple linear regression model again)

$$y = \beta_0 + \beta_1 \cdot x_1 + \cdots + \beta_k \cdot x_k + \varepsilon$$

- Answer: If $\beta_0 = \beta_1 = \cdots = \beta_k = 0$, we don't have a model

# F-STATISTIC

## Model's F-statistic Hypothesis Test

- The *null hypothesis* ($H_0$) represents the status quo; that all $\beta_i$ are zeros.

$$H_0: \beta_0 = \beta_1 = \cdots = \beta_k = 0$$

- The *alternate hypothesis* ($H_a$) represents the opposite of the null hypothesis (that at least one $\beta_i$ is not zero) and holds true if $H_0$ is found to be false:

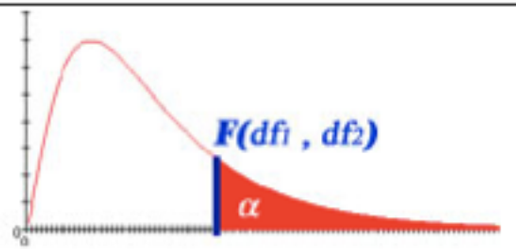
$$H_a: \exists i: \beta_i \neq 0$$

# F-STATISTIC

$$F = MSM / MSE = (\text{explained variance}) / (\text{unexplained variance})$$

# F-STATISTIC

F Table for $\alpha = 0.05$




| / | $df_1$=1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 15 | 20 | 24 | 30 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $df_2$=1 | 161.4476 | 199.5000 | 215.7073 | 224.5832 | 230.1619 | 233.9860 | 236.7684 | 238.8827 | 240.5433 | 241.8817 | 243.9060 | 245.9499 | 248.0131 | 249.0518 | 250.0951 | 251.1432 |
| 2 | 18.5128 | 19.0000 | 19.1643 | 19.2468 | 19.2964 | 19.3295 | 19.3532 | 19.3710 | 19.3848 | 19.3959 | 19.4125 | 19.4291 | 19.4458 | 19.4541 | 19.4624 | 19.4707 |
| 3 | 10.1280 | 9.5521 | 9.2766 | 9.1172 | 9.0135 | 8.9406 | 8.8867 | 8.8452 | 8.8123 | 8.7855 | 8.7446 | 8.7029 | 8.6602 | 8.6385 | 8.6166 | 8.5944 |
| 4 | 7.7086 | 6.9443 | 6.5914 | 6.3882 | 6.2561 | 6.1631 | 6.0942 | 6.0410 | 5.9988 | 5.9644 | 5.9117 | 5.8578 | 5.8025 | 5.7744 | 5.7459 | 5.7170 |
| 5 | 6.6079 | 5.7861 | 5.4095 | 5.1922 | 5.0503 | 4.9503 | 4.8759 | 4.8183 | 4.7725 | 4.7351 | 4.6777 | 4.6188 | 4.5581 | 4.5272 | 4.4957 | 4.4638 |
| | | | | | | | | | | | | | | | | |
| 6 | 5.9874 | 5.1433 | 4.7571 | 4.5337 | 4.3874 | 4.2839 | 4.2067 | 4.1468 | 4.0990 | 4.0600 | 3.9999 | 3.9381 | 3.8742 | 3.8415 | 3.8082 | 3.7743 |
| 7 | 5.5914 | 4.7374 | 4.3468 | 4.1203 | 3.9715 | 3.8660 | 3.7870 | 3.7257 | 3.6767 | 3.6365 | 3.5747 | 3.5107 | 3.4445 | 3.4105 | 3.3758 | 3.3404 |

- df1 is the number of variables, and df2 is the number of observations

# ACTIVITY: MODELS F-STATISTIC

EXERCISE

## ANSWER THE FOLLOWING QUESTIONS (10 minutes)

1. Using our Zillow dataset (`zillow-07-starter.csv` in the `datasets` folder), run a simple linear regression between *SalePrice* (the *dependent* variable) and *Size* (the *independent* variable). Does the model has any predictive power? What F-value do you get? (You can choose to use today's codealong which setup the environment and loads the dataset for you)
2. Run another simple linear regression between *SalePrice* (the *dependent* variable) and *IsAStudio* (the *independent* variable). Answer the same questions: Does the model has any predictive power? What F-value do you get?
3. Using the F-distribution table, come up with a general criteria (assuming a reasonable sized dataset) to accept or reject the null hypothesis and make; also annotate when the model is useful and when it isn't
4. When finished, share your answers with your table

## DELIVERABLE

Answers to the above questions

# ACTIVITY: MODELS F-STATISTIC

EXERCISE

*SalePrice* as a function of *Size*

| | | | |
|---|---|---|---|
| **Dep. Variable:** | SalePrice | **R-squared:** | 0.236 |
| **Model:** | OLS | **Adj. R-squared:** | 0.235 |
| **Method:** | Least Squares | **F-statistic:** | 297.4 |
| **Date:** | | **Prob (F-statistic):** | 2.67e-58 |
| **Time:** | | **Log-Likelihood:** | -1687.9 |
| **No. Observations:** | 967 | **AIC:** | 3380. |
| **Df Residuals:** | 965 | **BIC:** | 3390. |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [95.0% Conf. Int.] |
|---|---|---|---|---|---|
| **Intercept** | 0.1551 | 0.084 | 1.842 | 0.066 | -0.010 0.320 |
| **Size** | 0.7497 | 0.043 | 17.246 | 0.000 | 0.664 0.835 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 1842.865 | **Durbin-Watson:** | 1.704 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 3398350.943 |
| **Skew:** | 13.502 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 292.162 | **Cond. No.** | 4.40 |

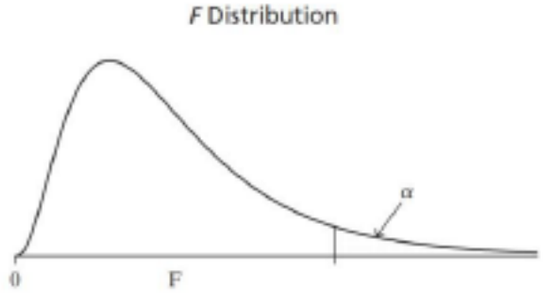
*SalePrice* as a function of *IsAStudio*

| | | | |
|---|---|---|---|
| **Dep. Variable:** | SalePrice | **R-squared:** | 0.000 |
| **Model:** | OLS | **Adj. R-squared:** | -0.001 |
| **Method:** | Least Squares | **F-statistic:** | 0.07775 |
| **Date:** | | **Prob (F-statistic):** | 0.780 |
| **Time:** | | **Log-Likelihood:** | -1847.4 |
| **No. Observations:** | 986 | **AIC:** | 3699. |
| **Df Residuals:** | 984 | **BIC:** | 3709. |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [95.0% Conf. Int.] |
|---|---|---|---|---|---|
| **Intercept** | 1.3811 | 0.051 | 27.088 | 0.000 | 1.281 1.481 |
| **IsAStudio** | 0.0829 | 0.297 | 0.279 | 0.780 | -0.501 0.666 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 1682.807 | **Durbin-Watson:** | 1.488 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 1342290.714 |
| **Skew:** | 10.942 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 182.425 | **Cond. No.** | 5.92 |

# The F-distribution table ($\alpha = .05$) (note: $df_1 \cong k, df_2 = n$) (cont.)




$\alpha = .05$

| $df_2$ | $df_1$ = 1 | 2 | 3 | 4 | 5 | 6 | 8 | 12 | 24 | ∞ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 161.4 | 199.5 | 215.7 | 224.6 | 230.2 | 234.0 | 238.9 | 243.9 | 249.0 | 254.3 |
| 2 | 18.51 | 19.00 | 19.16 | 19.25 | 19.30 | 19.33 | 19.37 | 19.41 | 19.45 | 19.50 |
| 3 | 10.13 | 9.55 | 9.28 | 9.12 | 9.01 | 8.94 | 8.84 | 8.74 | 8.64 | 8.53 |
| 4 | 7.71 | 6.94 | 6.59 | 6.39 | 6.26 | 6.16 | 6.04 | 5.91 | 5.77 | 5.63 |
| 5 | 6.61 | 5.79 | 5.41 | 5.19 | 5.05 | 4.95 | 4.82 | 4.68 | 4.53 | 4.36 |
| 6 | 5.99 | 5.14 | 4.76 | 4.53 | 4.39 | 4.28 | 4.15 | 4.00 | 3.84 | 3.67 |
| 7 | 5.59 | 4.74 | 4.35 | 4.12 | 3.97 | 3.87 | 3.73 | 3.57 | 3.41 | 3.23 |
| 8 | 5.32 | 4.46 | 4.07 | 3.84 | 3.69 | 3.58 | 3.44 | 3.28 | 3.12 | 2.93 |
| 9 | 5.12 | 4.26 | 3.86 | 3.63 | 3.48 | 3.37 | 3.23 | 3.07 | 2.90 | 2.71 |
| 10 | 4.96 | 4.10 | 3.71 | 3.48 | 3.33 | 3.22 | 3.07 | 2.91 | 2.74 | 2.54 |
| 11 | 4.84 | 3.98 | 3.59 | 3.36 | 3.20 | 3.09 | 2.95 | 2.79 | 2.61 | 2.40 |
| 12 | 4.75 | 3.88 | 3.49 | 3.26 | 3.11 | 3.00 | 2.85 | 2.69 | 2.50 | 2.30 |
| 13 | 4.67 | 3.80 | 3.41 | 3.18 | 3.02 | 2.92 | 2.77 | 2.60 | 2.42 | 2.21 |
| 14 | 4.60 | 3.74 | 3.34 | 3.11 | 2.96 | 2.85 | 2.70 | 2.53 | 2.35 | 2.13 |
| 15 | 4.54 | 3.68 | 3.29 | 3.06 | 2.90 | 2.79 | 2.64 | 2.48 | 2.29 | 2.07 |
| 16 | 4.49 | 3.63 | 3.24 | 3.01 | 2.85 | 2.74 | 2.59 | 2.42 | 2.24 | 2.01 |
| 17 | 4.45 | 3.59 | 3.20 | 2.96 | 2.81 | 2.70 | 2.55 | 2.38 | 2.19 | 1.96 |
| 18 | 4.41 | 3.55 | 3.16 | 2.93 | 2.77 | 2.66 | 2.51 | 2.34 | 2.15 | 1.92 |
| 19 | 4.38 | 3.52 | 3.13 | 2.90 | 2.74 | 2.63 | 2.48 | 2.31 | 2.11 | 1.88 |
| 20 | 4.35 | 3.49 | 3.10 | 2.87 | 2.71 | 2.60 | 2.45 | 2.28 | 2.08 | 1.84 |
| 21 | 4.32 | 3.47 | 3.07 | 2.84 | 2.68 | 2.57 | 2.42 | 2.25 | 2.05 | 1.81 |
| 22 | 4.30 | 3.44 | 3.05 | 2.82 | 2.66 | 2.55 | 2.40 | 2.23 | 2.03 | 1.78 |
| 23 | 4.28 | 3.42 | 3.03 | 2.80 | 2.64 | 2.53 | 2.38 | 2.20 | 2.00 | 1.76 |
| 24 | 4.26 | 3.40 | 3.01 | 2.78 | 2.62 | 2.51 | 2.36 | 2.18 | 1.98 | 1.73 |
| 25 | 4.24 | 3.38 | 2.99 | 2.76 | 2.60 | 2.49 | 2.34 | 2.16 | 1.96 | 1.71 |
| 26 | 4.22 | 3.37 | 2.98 | 2.74 | 2.59 | 2.47 | 2.32 | 2.15 | 1.95 | 1.69 |
| 27 | 4.21 | 3.35 | 2.96 | 2.73 | 2.57 | 2.46 | 2.30 | 2.13 | 1.93 | 1.67 |
| 28 | 4.20 | 3.34 | 2.95 | 2.71 | 2.56 | 2.44 | 2.29 | 2.12 | 1.91 | 1.65 |
| 29 | 4.18 | 3.33 | 2.93 | 2.70 | 2.54 | 2.43 | 2.28 | 2.10 | 1.90 | 1.64 |
| 30 | 4.17 | 3.32 | 2.92 | 2.69 | 2.53 | 2.42 | 2.27 | 2.09 | 1.89 | 1.62 |
| 40 | 4.08 | 3.23 | 2.84 | 2.61 | 2.45 | 2.34 | 2.18 | 2.00 | 1.79 | 1.51 |
| 60 | 4.00 | 3.15 | 2.76 | 2.52 | 2.37 | 2.25 | 2.10 | 1.92 | 1.70 | 1.39 |
| 120 | 3.92 | 3.07 | 2.68 | 2.45 | 2.29 | 2.17 | 2.02 | 1.83 | 1.61 | 1.25 |
| ∞ | 3.84 | 2.99 | 2.60 | 2.37 | 2.21 | 2.09 | 1.94 | 1.75 | 1.52 | 1.00 |

# Model's F-statistic ($\alpha = .05$ )

| F-value | p-value | $H_0$ / $H_a$ | Conclusion |
|---|---|---|---|
| $\geq 4$ (*)<br><br>(*) (at least one variable, at least 100 observations) | $\leq .05$ | Found evidence that $\mu \neq \mu_0$: Reject $H_0$ | At least one $\beta_i \neq 0$ ; the model is <u>useful</u> |
| $< 4$ (*) | $> .05$ | Did not find that $\mu \neq \mu_0$: Fail to reject $H_0$ | All $\beta_i = 0$ ; the model is <u>not useful</u><br><br>(assume) |

# Accessing the model’s F-statistic and its p-value

## Accessing the model's F-statistic and its p-value

### F-value (with significance level of 5%)

In [4]:
```
model.fvalue
```

Out[4]:
```
0.077751247187633807
```

### Corresponding p-value

In [5]:
```
model.f_pvalue
```

Out[5]:
```
0.78042689060390313
```

DEMO

# F-statistic, backward selection, and how to conduct a multiple regression

# Two-step guidance on how to conduct linear regression modeling

## ① Model's significance

- Always start with the F-statistics for the whole model; only then check individual variables

## ② Regressors' significance

- Prefer to work solely with significant variables: if you observe insignificant variables you *usually* need to get rid of them and rerun your regression modeling without those
- Backward selection method
  - If you have insignificant variables, start dropping the most insignificant variable. If after removing that variable you still have insignificant variables, drop them one by one, until you are left with no insignificant variables

# PUTTING IT ALL TOGETHER

- Summary statistics
- Transformation of variables
- Create dummy variables of categorical variables
- Multicollinearity testing
- Test F-statistic
- Backwards stepwise regression: remove variables one by p-values

# INTERACTIONS

# Interaction effects

- But suppose that spending money on radio advertising actually increases the effectiveness of *TV* advertising
  - the slope term for *TV* should increase as *Radio* increases
- E.g., given a fixed budget of $100,000, spending half on TV and half on radio may increase sales more than allocating the entire amount to either TV or radio
- This is known as a synergy effect in marketing; in statistics it is referred to as an interaction effect

GUIDED PRACTICE

# INTERACTION EFFECTS

# Interaction effects

- $Sales = \underset{\beta_0'}{6.76} + \underbrace{.0190}_{\beta_1'} \times TV + \underbrace{.0276}_{\beta_2'} \times Radio + \underbrace{.0011}_{\beta_3'} \times TV \times Radio$
- The interaction is important
  - $\beta_3$ is statistically significant
  - $R^2$ with this model went up to 96.8% up from 89.5% for the model without interaction. This that $1 - \frac{1-.968}{1-.895} = .70 = 70\%$ of the unexplained variability in the previous model has been explained by the interaction term

# Hierarchy Principle

- Sometimes an interaction term $x_i \cdot x_j$ is significant, but one or both of its main effects (in this case $x_i$ and/or $x_j$ ) are not

- The hierarchy principle
  - If we include an interaction in a model, we should also include the main effects, even if they aren't significant

# ACTIVITY: INTERACTION EFFECTS

EXERCISE

## ANSWER THE FOLLOWING QUESTIONS (10 minutes)

1. Our TV budget is $50,000 that we consider increasing it by $5,000. What would be the corresponding increase in sales based on different levels of radio budget?
2. When finished, share your answers with your table

## DELIVERABLE

Answers to the above questions

# ACTIVITY: INTERACTION EFFECTS

EXERCISE

| Radio budget | Model without interactions | Model with interactions |
|---|---|---|
| Formula | $\underbrace{.0457}_{\hat{\beta}_1} \times \Delta TV$ | $\left(\underbrace{.0190}_{\hat{\beta}_1} + \underbrace{.0011}_{\hat{\beta}_3} \times Radio\right) \times \Delta TV$ |
| $15,000 | .0457 × 5 = .228 = $229 | (.0190 + .0011 × 15 ) 5<br>= .178 = $178 |
| $10,000 | $229 | (.0190 + .0011 × 10 ) 5<br>= .150 = $150 |
| $5,000 | $229 | (.0190 + .0011 × 5 ) 5<br>= .123 = $123 |

# PUTTING IT ALL TOGETHER

- Summary statistics
- Transformation of variables
- Create dummy variables of categorical variables
- Multicollinearity testing
- Test F-statistic
- Backwards stepwise regression: remove variables one by p-values
- Add interaction effects

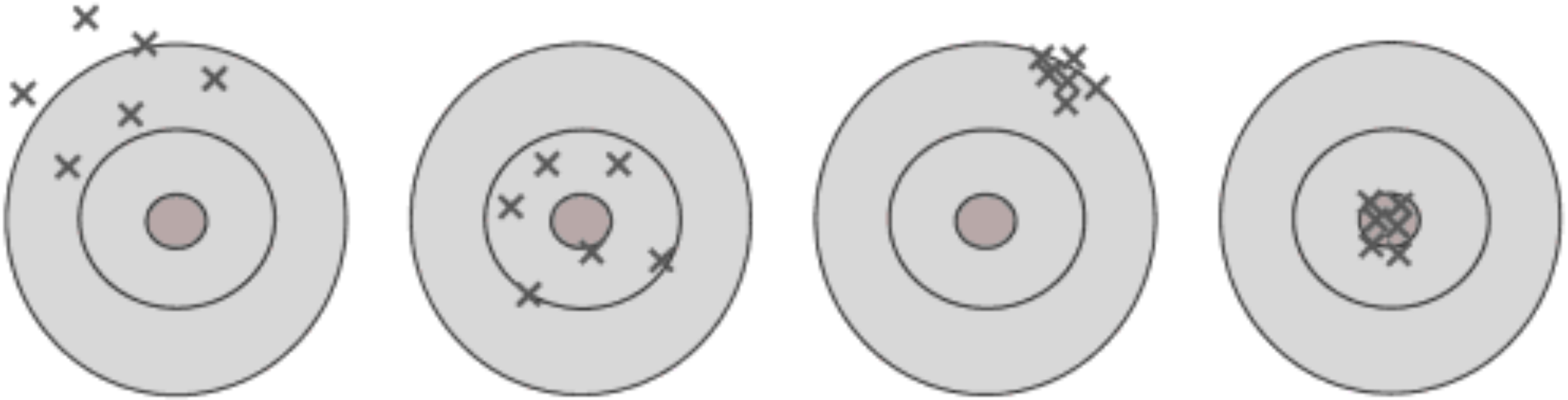
STEP BACK

# UNDERFITTING AND OVERFITTING

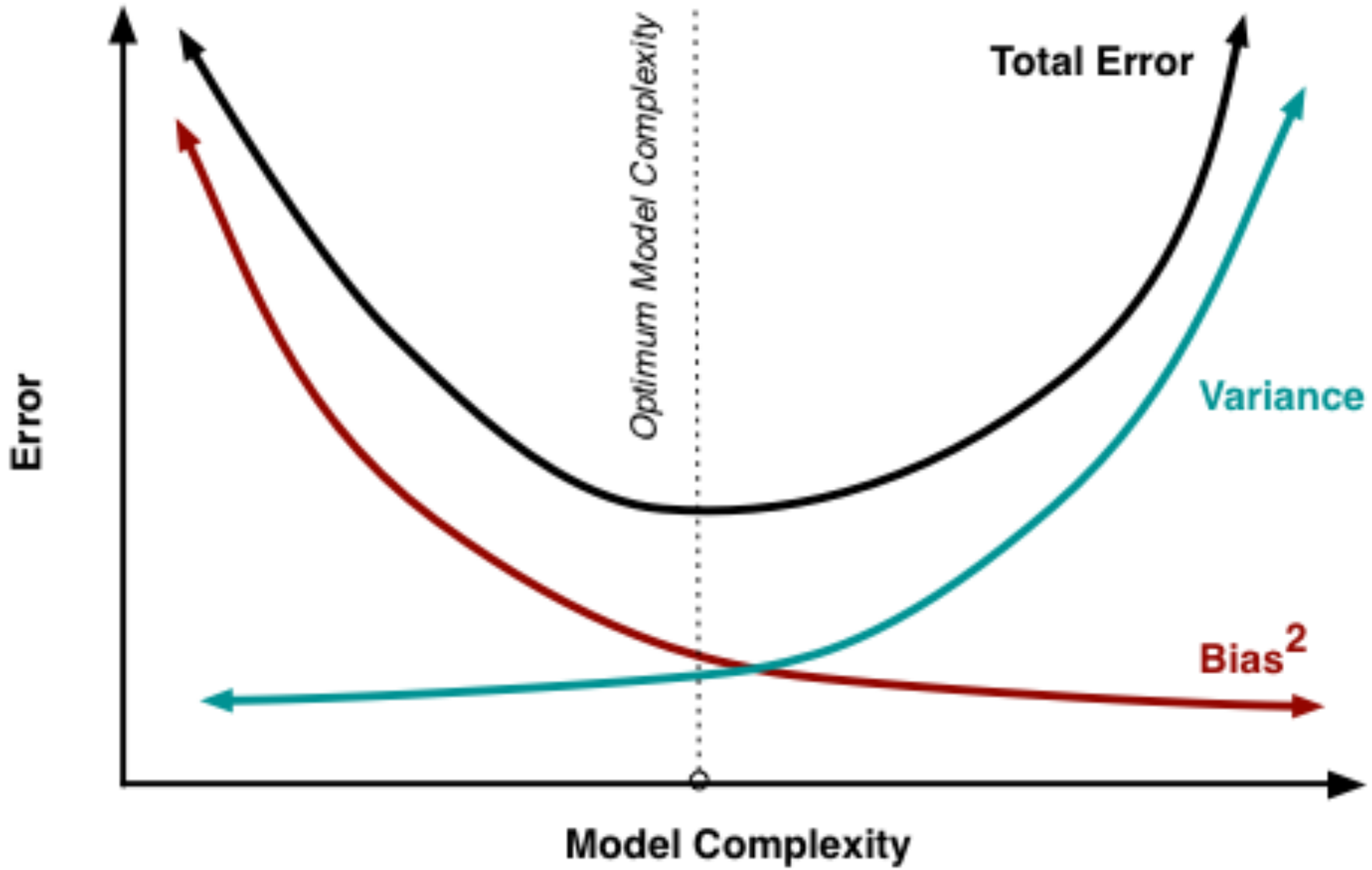
# BIAS VARIANCE TRADEOFF

- When our error is ***biased***, it means the model's prediction is consistently far away from the actual value.
- This could be a sign of poor sampling and poor data.
- One objective of a biased model is to trade bias error for generalized error. We prefer the error to be more evenly distributed across the model.
- This is called error due to ***variance***.
- We want our model to *generalize* to data it hasn't seen even if doesn't perform as well on data it has already seen.

# BIAS VS. VARIANCE

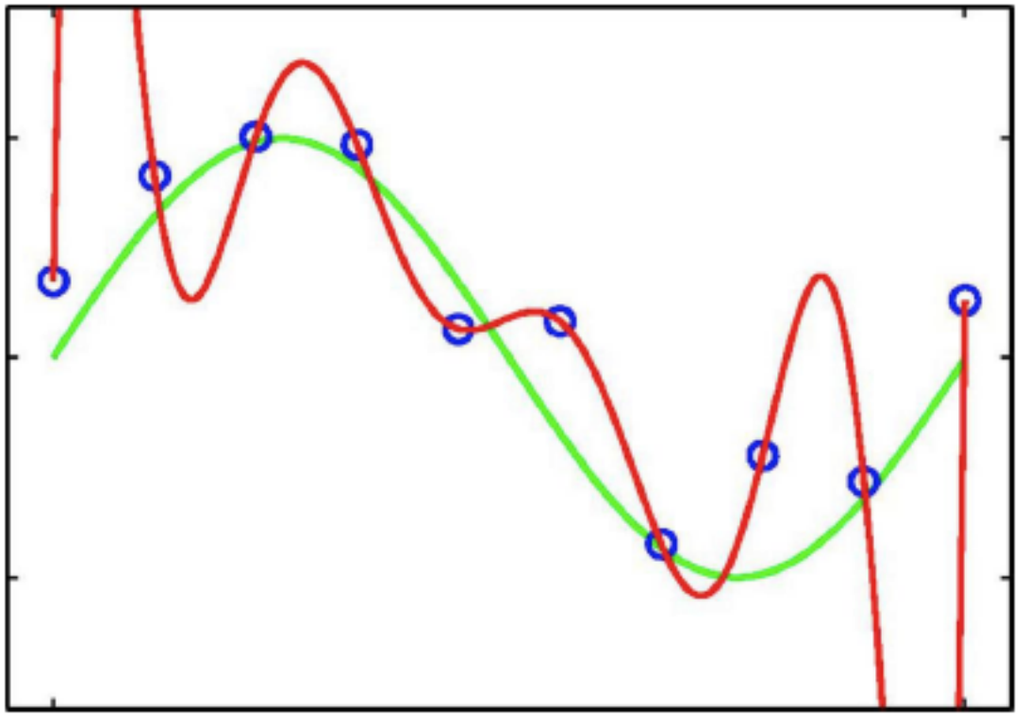


# BIAS VARIANCE TRADEOFF

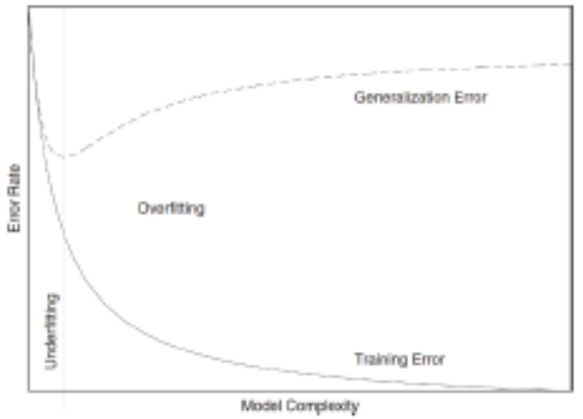


# How low can we push the training error?

- Down to zero (effectively "memorizing" the entire training set)
- However, the model is now not only too complex but it will also not generalize well to data that was not used during training
  - This is called overfitting



# Error rate, model complexity, and fit




- Underfitting
  - Model is too simple and cannot represent the desired behavior very well
  - Both its training and generalization error are poor
- Good fit
  - Model has the right level of complexity
  - It performs well on the training set (low training error) and generalize well to unknown data points (low generalization error)
- Overfitting
  - Model is too complex
  - It performs very well on the training set (low training error) but does not generalize well to unknown data points (high generalization error)

# ACTIVITY: UNDERFITTING AND OVERFITTING

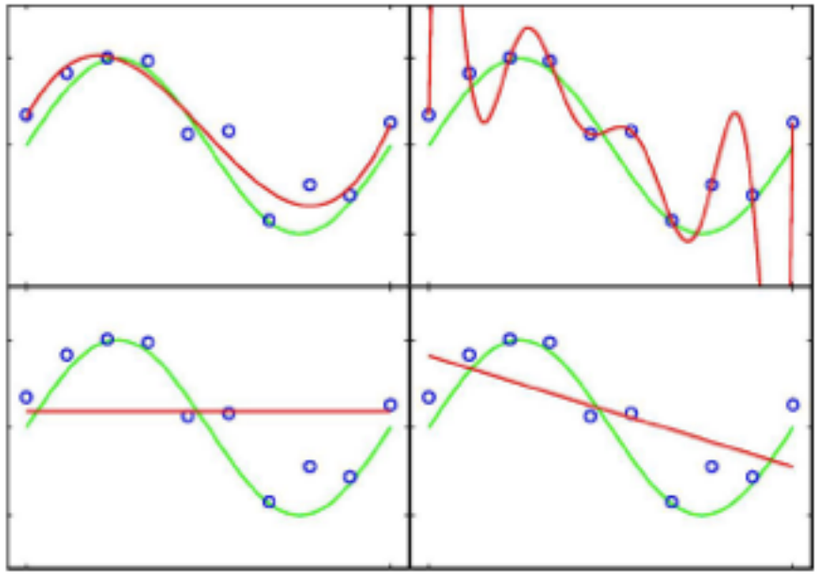
EXERCISE

ANSWER THE FOLLOWING QUESTIONS (10 minutes)

1. Classify the following polynomial regressions according to their fit:
    1. Underfitting
    2. Good fit
    3. Overfitting
2. When finished, share your answers with your table

DELIVERABLE

Answers to the above questions



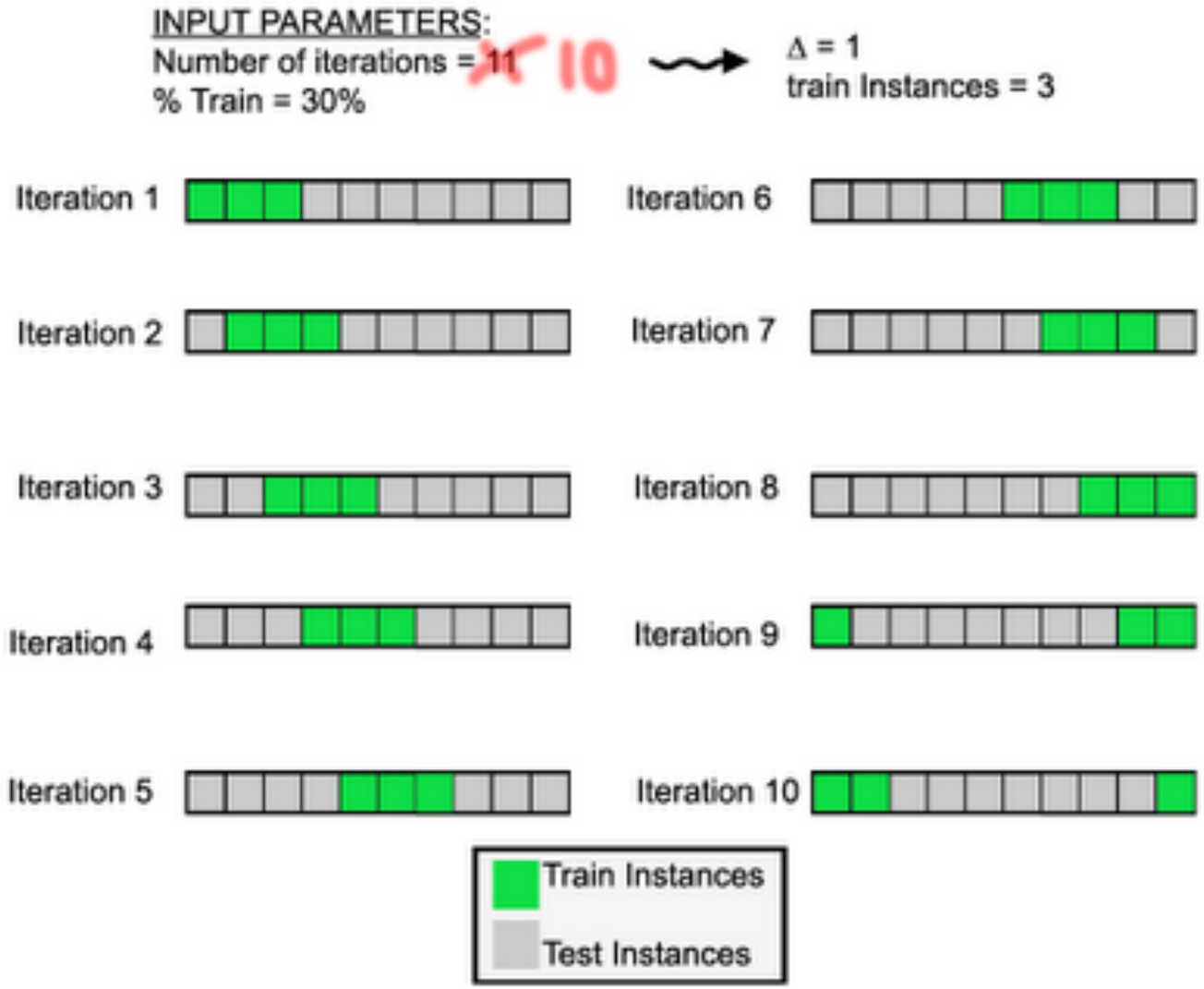
DEMO

# CROSS VALIDATION

# CROSS VALIDATION

- Cross validation can help account for bias.
- The general idea is to
  - Generate several models on different cross sections of the data
  - Measure the performance of each
  - Take the mean performance
- This technique swaps bias error for generalized error, describing previous trends accurately enough to extend to future trends.

# CROSS VALIDATION



# K-FOLD CROSS VALIDATION

- k-fold cross validation
  - Split the data into *k* group
  - Train the model on all segments except one
  - Test model performance on the remaining set
- If k = 5, split the data into five segments and generate five models.

# USING K-FOLD CROSS VALIDATION WITH MSE

- Import the appropriate packages and load data.

```
from sklearn import cross_validation
wd = '../../datasets/'
bikeshare = pd.read_csv(wd + 'bikeshare/bikeshare.csv')
weather = pd.get_dummies(bikeshare.weathersit, prefix='weather')
modeldata = bikeshare[['temp', 'hum']].join(weather[['weather_1',
'weather_2', 'weather_3']])
y = bikeshare.casual
```

# USING K-FOLD CROSS VALIDATION WITH MSE

- Build models on subsets of the data and calculate the average score.

```
kf = cross_validation.KFold(len(modeldata), n_folds=5, shuffle=True)
scores = []
for train_index, test_index in kf:
    lm = linear_model.LinearRegression().fit(modeldata.iloc[train_index],
y.iloc[train_index])
    scores.append(metrics.mean_squared_error(y.iloc[test_index],
lm.predict(modeldata.iloc[test_index])))

print np.mean(scores)
```

# USING K-FOLD CROSS VALIDATION WITH MSE

- This can be compared to the model built on all of the data.

```
- This score will be lower, but we're trading off bias error for
generalized error:
lm = linear_model.LinearRegression().fit(modeldata, y)
print metrics.mean_squared_error(y, lm.predict(modeldata))
```

- Which approach would predict new data more accurately?

GUIDED PRACTICE

# CROSS VALIDATION WITH LINEAR REGRESSION

# ACTIVITY: CROSS VALIDATION WITH LINEAR REGRESSION


## DIRECTIONS (20 minutes)

If we were to continue increasing the number of folds in cross validation, would error increase or decrease?

1. Using the previous code example, perform k-fold cross validation for all even numbers between 2 and 50.
2. Answer the following questions:
   a. What does `shuffle=True` do?
   b. At what point does cross validation no longer seem to help the model?
3. Hint: `range(2, 51, 2)` produces a list of even numbers from 2 to 50

## DELIVERABLE

Answers to questions

# REGULARIZATION AND CROSS VALIDATION

# WHAT IS REGULARIZATION? AND WHY DO WE USE IT?

- Regularization is an additive approach to protect models against overfitting (being potentially biased and overconfident, not generalizing well).
- Regularization becomes an additional weight to coefficients, shrinking them closer to zero.
- L1 (Lasso Regression) adds the extra weight to coefficients.
- L2 (Ridge Regression) adds the square of the extra weight to coefficients.
- Use Lasso when we have more features than observations ($k > n$) and Ridge otherwise.

# WHERE REGULARIZATION MAKES SENSE

- What happens to MSE if use Lasso or Ridge Regression directly?

```
lm = linear_model.LinearRegression().fit(modeldata, y)
print lm.score
lm = linear_model.Lasso().fit(modeldata, y)
print lm.score
lm = linear_model.Ridge().fit(modeldata, y)
print lm.score
```

```
1672.58110765 # OLS
1725.41581608 # L1
1672.60490113 # L2
```

# WHERE REGULARIZATION MAKES SENSE

- It doesn't seem to help.  Why is that?

- We need to optimize the regularization weight parameter (called alpha) through cross validation.

DEMO

# UNDERSTANDING REGULARIZATION EFFECTS

# QUICK CHECK

- We are working with the bikeshare data to predict riders over hours/days with a few features.
- Does it make sense to use a ridge regression or a lasso regression?
- Why?

# UNDERSTANDING REGULARIZATION EFFECTS

- Let's test a variety of alpha weights for Ridge Regression on the bikeshare data.

```
alphas = np.logspace(-10, 10, 21)
for a in alphas:
    print 'Alpha:', a
    lm = linear_model.Ridge(alpha=a)
    lm.fit(modeldata, y)
    print lm.coef_
    print lm.score
```

- What happens to the weights of the coefficients as alpha increases? What happens to the error as alpha increases?

# WE CAN MAKE THIS EASIER WITH GRID SEARCH!

- Grid search exhaustively searches through all given options to find the best solution. Grid search will try all combos given in `param_grid`.

```
param_ grid = {
    'intercept': [True, False],
    'alpha': [1, 2, 3],
}
```

# WE CAN MAKE THIS EASIER WITH GRID SEARCH!

- This param grid has six different options:
  - intercept True, alpha 1
  - intercept True, alpha 2
  - intercept True, alpha 3
  - intercept False, alpha 1
  - intercept False, alpha 2
  - intercept False, alpha 3

```
param_ grid = {
    'intercept': [True,
False],
    'alpha': [1, 2, 3],
}
```

# WE CAN MAKE THIS EASIER WITH GRID SEARCH!

- This is an incredibly powerful, automated machine learning tool!

```
from sklearn import grid_search

alphas = np.logspace(-10, 10, 21)
gs = grid_search.GridSearchCV(
    estimator=linear_model.Ridge(),
    param_grid={'alpha': alphas},
    scoring='mean_squared_error')
```

# WE CAN MAKE THIS EASIER WITH GRID SEARCH!

```
gs.fit(modeldata, y)

print -gs.best_score_ # mean squared error here comes in
negative, so let's make it positive.
print gs.best_estimator_ # explains which grid_search setup
worked best
print gs.grid_scores_ # shows all the grid pairings and their
performances.
```

GUIDED PRACTICE

# GRID SEARCH CV, SOLVING FOR ALPHA

# ACTIVITY: GRID SEARCH CV, SOLVING FOR ALPHA


## DIRECTIONS (25 minutes)

1. Modify the previous code to do the following:
   a. Introduce cross validation into the grid search. This is accessible from the cv argument.
   b. Add fit_intercept = True and False to the param_grid dictionary.
   c. Re-investigate the best score, best estimator, and grid score attributes as a result of the grid search.

## DELIVERABLE

New code and output that meets above requirements

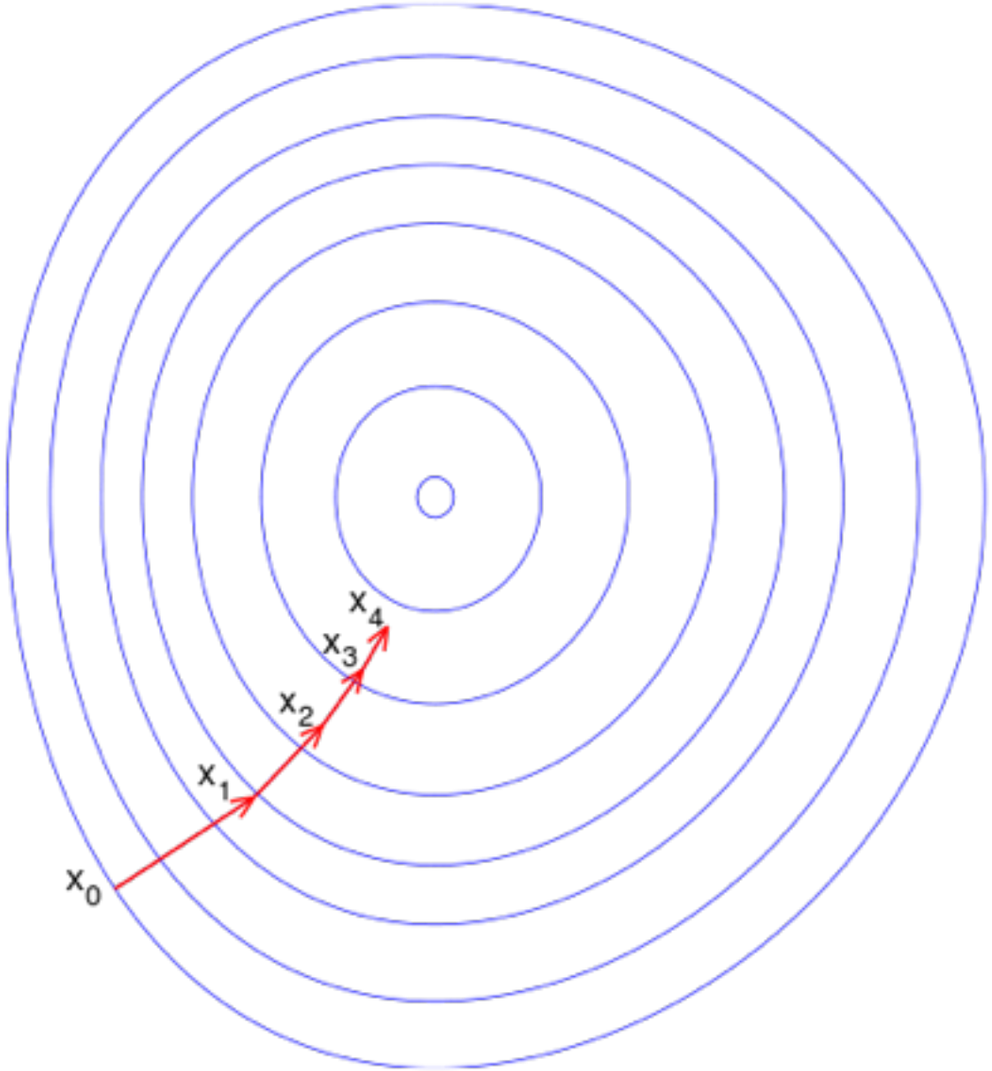
# MINIMIZING LOSS THROUGH GRADIENT DESCENT

# GRADIENT DESCENT

- Gradient Descent can also help us minimize error.

- How Gradient Descent works:
  - A random linear solution is provided as a starting point
  - The solver attempts to find a next “step”:  take a step in any direction and measure the performance.
  - If the solver finds a better solution (i.e. lower MSE), this is the new starting point.
  - Repeat these steps until the performance is optimized and no “next steps” perform better.  The size of steps will shrink over time.

# GRADIENT DESCENT



# A CODE EXAMPLE OF GRADIENT DESCENT

```
num_to_approach, start, steps, optimized = 6.2, 0., [-1, 1], False
while not optimized:
    current_distance = num_to_approach - start
    got_better = False
    next_steps = [start + i for i in steps]
    for n in next_steps:
        distance = np.abs(num_to_approach - n)
        if distance < current_distance:
            got_better = True
            print distance, 'is better than', current_distance
            current_distance = distance
            start = n
```

# A CODE EXAMPLE OF GRADIENT DESCENT

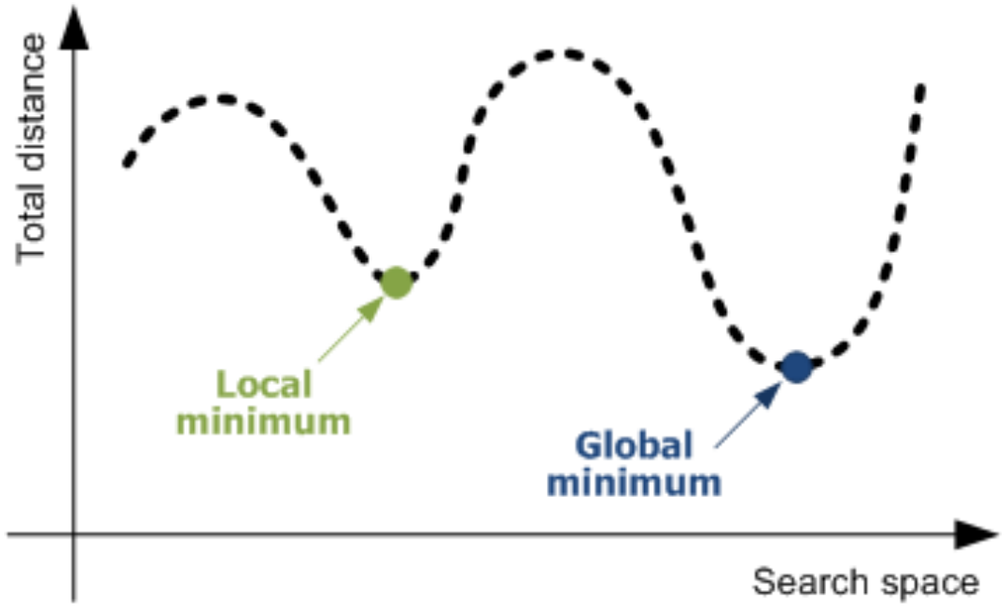
```
    if got_better:
        print 'found better solution! using', current_distance
        a += 1
    else:
        optimized = True
        print start, 'is closest to', num_to_approach
```

- What is the code doing?  What could go wrong?

# GLOBAL VS LOCAL MINIMUMS

- Gradient Descent could solve for a *local* minimum instead of a *global* minimum.

- A *local* minimum is confined to a very specific subset of solutions. The *global* minimum considers all solutions. These could be equal, but that's not always true.



DEMO

# APPLICATION OF GRADIENT DESCENT

# APPLICATION OF GRADIENT DESCENT

- Gradient Descent works best when:
  - We are working with a large dataset.  Smaller datasets are more prone to error.
  - Data is cleaned up and normalized.
- Gradient Descent is significantly faster than OLS.  This becomes important as data gets bigger.

# APPLICATION OF GRADIENT DESCENT

- We can easily run a Gradient Descent regression.
- Note:  The verbose argument can be set to 1 to see the optimization steps.

```
lm = linear_model.SGDRegressor()
lm.fit(modeldata, y)
print lm.score(modeldata, y)
print metrics.mean_squared_error(y, lm.predict(modeldata))
```

- Untuned, how well did gradient descent perform compared to OLS?

# APPLICATION OF GRADIENT DESCENT

- Gradient Descent can be tuned with
  - the learning rate:  how aggressively we solve the problem
  - epsilon:  at what point do we say the error margin is acceptable
  - iterations:  when should be we stop no matter what

## INDEPENDENT PRACTICE

# ON YOUR OWN

# ACTIVITY: ON YOUR OWN

EXERCISE

## DIRECTIONS (30 minutes)

There are tons of ways to approach a regression problem.

1. Implement the Gradient Descent approach to our bikeshare modeling problem.
2. Show how Gradient Descent solves and optimizes the solution.
3. Demonstrate the grid_search module.
4. Use a model you evaluated last class or the simpler one from today. Implement param_grid in grid search to answer the following questions:
    a. With a set of values between 10^-10 and 10^-1, how does MSE change?
    b. Our data suggests we use L1 regularization. Using a grid search with l1_ratios between 0 and 1, increasing every 0.05, does this statement hold true? If not, did gradient descent have enough iterations to work properly?
    c. How do these results change when you alter the learning rate?

## DELIVERABLE

Gradient Descent approach and answered questions

# ACTIVITY: ON YOUR OWN


Starter Code

```
params = {} # put your gradient descent parameters here
gs = grid_search.GridSearchCV(
    estimator=linear_model.SGDRegressor(),
    cv=cross_validation.KFold(len(modeldata), n_folds=5,
shuffle=True),
    param_grid=params,
    scoring='mean_squared_error',
    )

gs.fit(modeldata, y)

print 'BEST ESTIMATOR'
print -gs.best_score_
print gs.best_estimator_
print 'ALL ESTIMATORS'
print gs.grid_scores_
```

# TOPIC REVIEW

# LESSON REVIEW

- What's the (typical) range of r-squared?
- What's the range of mean squared error?
- How would changing the scale or interpretation of y (your target variable) effect mean squared error?
- What's cross validation, and why do we use it in machine learning?
- What is error due to bias? What is error due to variance? Which is better for a model to have, if it had to have one?
- How does gradient descent try a different approach to minimizing error?

# BEFORE NEXT CLASS

## BEFORE NEXT CLASS

# DUE DATE

- Homework:
- Project: Final Project, Deliverable 1

# CREDITS

## THANKS FOR THE FOLLOWING

# CITATIONS

- Title, Author: link
- Title, Author: link
- Title, Author: link

LESSON

# Q & A

LESSON

# EXIT TICKET

**DON’T FORGET TO FILL OUT YOUR EXIT TICKET**

## THANKS!

# NAME

- Optional Information:
- Email?
- Website?
- Twitter?