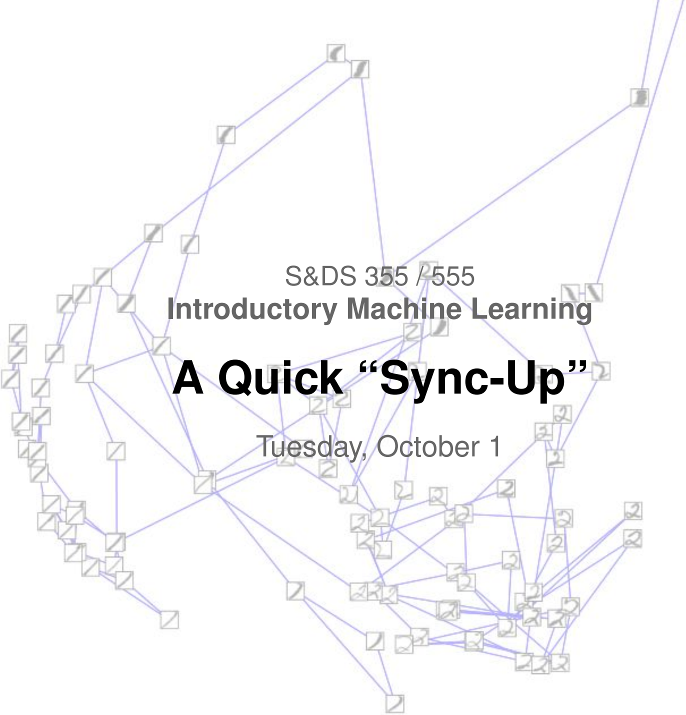


S&DS 355 / 555

Introductory Machine Learning

A Quick “Sync-Up”

Tuesday, October 1

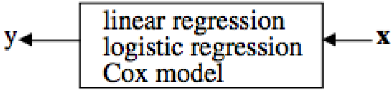
Yale

Leo Brieman – "Keep it Simple"

https://www.youtube.com/watch?v=t8ooi_tJHSE

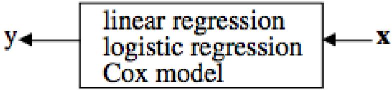
Brieman's Two Cultures

Traditional statistics:



Brieman's Two Cultures

Traditional statistics: culture



You put faith in the model & try to infer the parameters

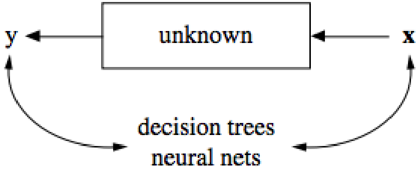
- Assume: $y = f(x)$ with a specific form of f.
  e.g. $y = \beta_0 + \beta_1 x + \epsilon$ with $\epsilon \sim N(0, \sigma^2)$
- Estimate parameters based on data.
- Do inference and prediction.
- Hypothesis testing, assumption checking...

Brieman's Two Cultures

CS culture

In the other culture, you don't try to do inference over the model, since you don't believe model is correct

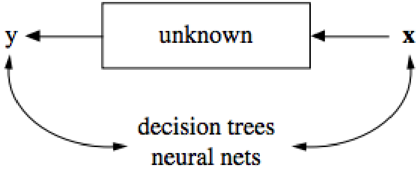
Machine learning:



At one point, the two cultures merged, but now they're drifting apart again (e.g. deep learning)

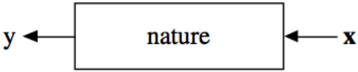
Brieman's Two Cultures

Machine learning:

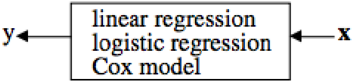


- Assume: $y = f(x)$ where f is complex and unknown.
- Use algorithm to predict y from x.
- Only care about prediction accuracy.

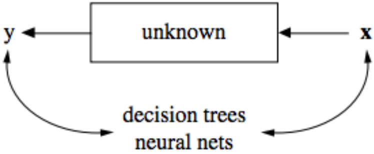
Brieman's Two Cultures



Traditional statistics:



Machine learning:



Some Terminology

- supervised vs. unsupervised
- classification vs. regression
- prediction vs. inference

Supervised Learning vs. Unsupervised Learning

Supervised learning:

- Given a set of (x, y), learn to predict y using x.
- e.g.
  - Predicting whether a loan will default based on customer characteristics

Supervised Learning vs. Unsupervised Learning

Supervised learning:

- Given a set of (x, y), learn to predict y using x.
- e.g.
  - Predicting whether a loan will default based on customer characteristics

Unsupervised learning:

- Given a set of x, learn underlying structure or relationships of x.
- e.g.
  - Identifying market segments with similar spending patterns.

Classification vs. Regression

The `Income` dataset:

Education	Seniority	Income
21.58621	113.1034	99.91717
18.27586	119.3103	92.57913
12.06897	100.6897	34.67873
17.03448	187.5862	78.70281
19.93103	20.0000	68.00992
18.27586	26.2069	71.50449

Information for 30 *simulated individuals*.

Classification vs. Regression

The `Income` dataset:

Education	Seniority	Income
21.58621	113.1034	99.91717
18.27586	119.3103	92.57913
12.06897	100.6897	34.67873
17.03448	187.5862	78.70281
19.93103	20.0000	68.00992
18.27586	26.2069	71.50449

Information for 30 *simulated individuals*.

Regression: Model income based on other characteristics.

Classification vs. Regression

The `Income` dataset:

Education	Seniority	Income
21.58621	113.1034	99.91717
18.27586	119.3103	92.57913
12.06897	100.6897	34.67873
17.03448	187.5862	78.70281
19.93103	20.0000	68.00992
18.27586	26.2069	71.50449

Information for 30 *simulated individuals*.

Regression: Model income based on other characteristics.

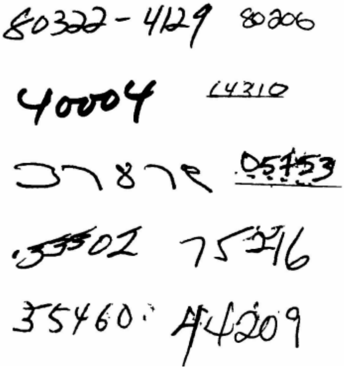
Classification: Model whether someone will earn above the median income based on other characteristics.

Example: Handwritten Digit Recognition

- Data: images of handwritten digits (grayscale pixel values)
- Classify images as digits 0 to 9.

Example: Handwritten Digit Recognition

- Data: images of handwritten digits (grayscale pixel values)
- Classify images as digits 0 to 9.



Example: Spam Mail

Classification problem

A[redacted]
to

12/27/16

Why is this message in Spam? It's similar to messages that were detected by our spam filters. Learn more

Hi Am Anita,
Nice to meet you,
i saw your email on github and i decided to communicate

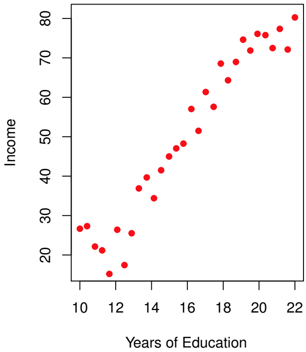
with you,in your usual time may it please you to
reply me here for my pictures, my details,

and my purpose of writing to you.
please i don't have much access on github due to some

personal reasons,
thanks from

Regression Example

The `Income` dataset:



Quantitative response Y

Predictors $X = (X_1, \ldots, X_p)$

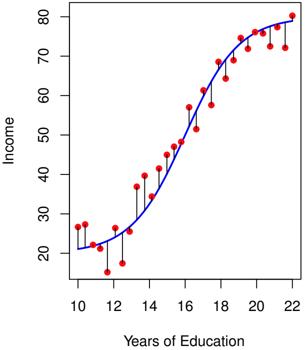
Assume the relationship can be expressed by:

$$Y = f(X) + \epsilon,$$

where f is a fixed, unknown function and ϵ is error term.

Regression Example

The `Income` dataset:



Quantitative response Y

Predictors $X = (X_1, \ldots, X_p)$

Assume the relationship can be expressed by:

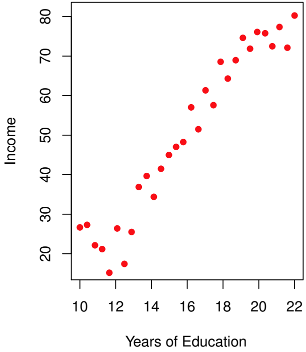
mean zero noise
e.g. Gaussian

$$Y = f(X) + \epsilon,$$

where f is a fixed, unknown function and ϵ is error term.

Regression Example

Back to regression with $p = 1$:



$$Y = f(X) + \epsilon$$

Modeling:

Use a procedure to get $\widehat{f}$. Derive estimates $\widehat{Y} = \widehat{f}(X)$.

Possible Regression Approaches

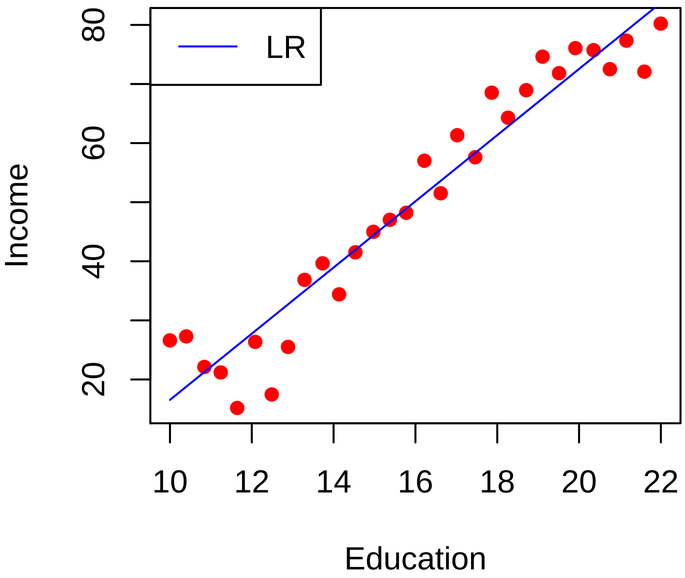
could also try minimizing
orthogonal distance

- linear regression
  - Fitting a straight line through the data.

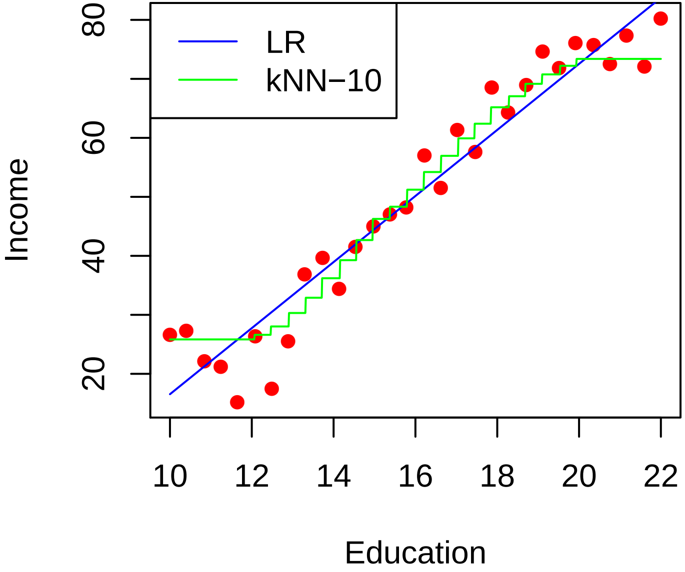
and minimize RSS
(sum of squared errors, i.e.
vertical distance)

- k-nearest neighbors regression
  - Average together the y_i for x_i close to x
- decision trees
  - Split input space into cells, average Y_i in each cell

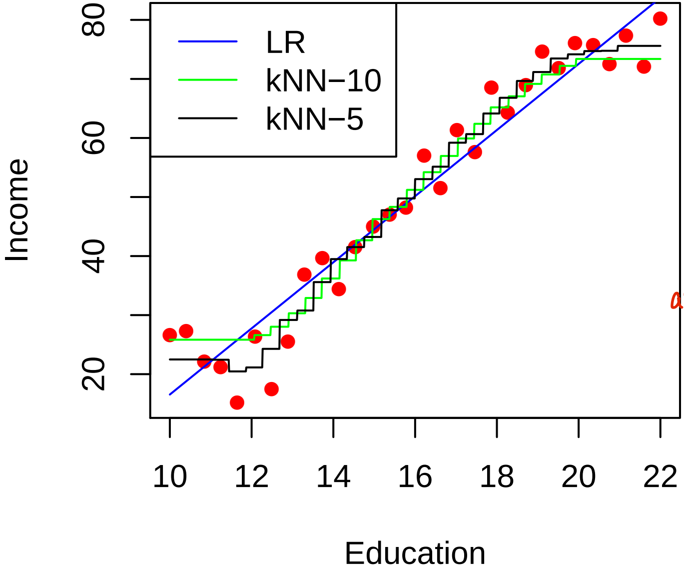
Possible Regression Approaches



Possible Regression Approaches



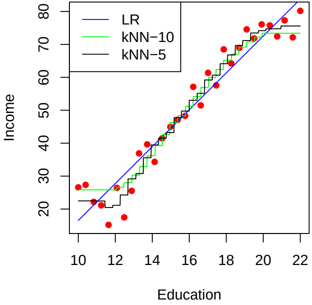
Possible Regression Approaches



as K decreases:
- bias increases
- variance decreases

curve becomes jumpier because it fits worse

Possible Regression Approaches



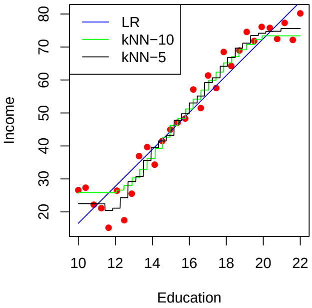
Measuring performance via **Mean Squared Error**

ground truth

predictions

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Possible Regression Approaches



Measuring performance via **Mean Squared Error**

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}(x_i))^2$$

MSEs for three methods:

Linear Regression	29.829
k-Nearest Neighbors (k=10)	23.519
k-Nearest Neighbors (k=5)	16.21

A k-nearest neighbors model with $k = 5$ achieves lowest error. Is it the best?

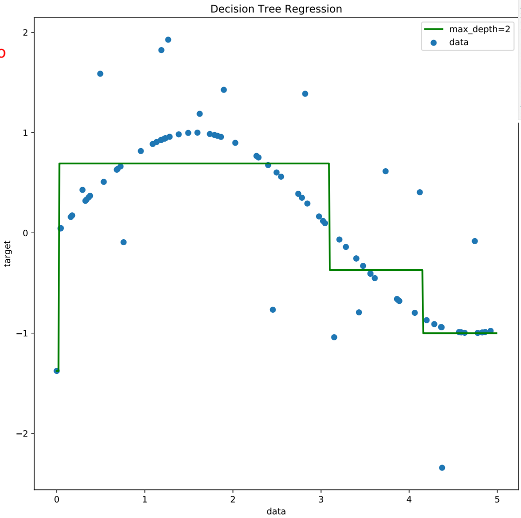
Decision tree regression

Get 80 random numbers
Apply sin
Random noise every 5th data point

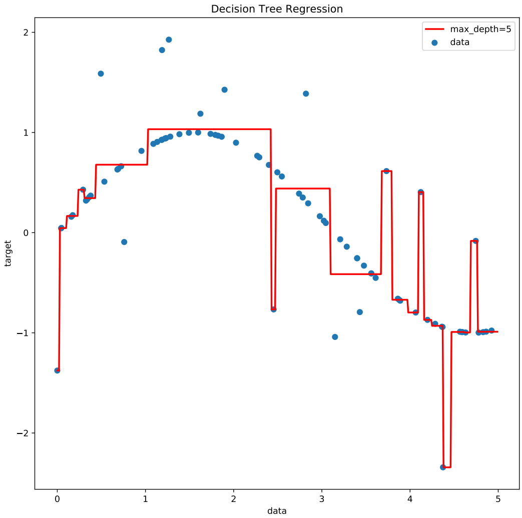
```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

# Create a random dataset
rng = np.random.RandomState(13)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))

# Fit regression model
from sklearn.tree import DecisionTreeRegressor

clf_1 = DecisionTreeRegressor(max_depth=2)
clf_2 = DecisionTreeRegressor(max_depth=5)
clf_1.fit(X, y)
clf_2.fit(X, y)

# Predict
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
y_1 = clf_1.predict(X_test)
y_2 = clf_2.predict(X_test)

plt.figure(figsize=(10,10))
plt.scatter(X, y, label="data")
plt.plot(X_test, y_1, c="g", label="max_depth=2", linewidth=2)
plt.plot(X_test, y_2, c="r", label="max_depth=10", linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
#plt.savefig('dtree_10.pdf')
plt.show()
```

We choose the splits in order to minimize MSE

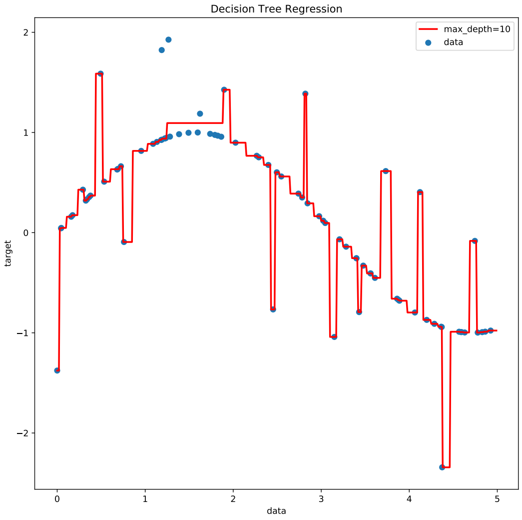
It's a greedy algorithm and recursively splits in order to most minimize the error



Decision tree regression



Decision tree regression



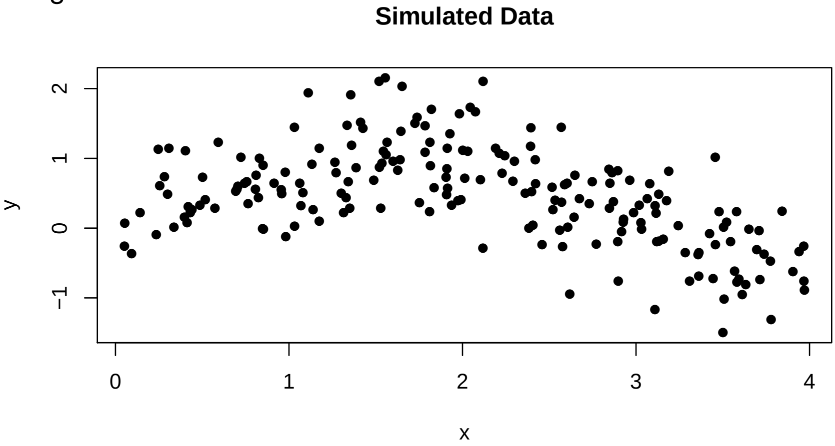
Overfitting

The training set has bias 0 and very high variance because it is the true regression

A method is **overfitting** the data when it has a small training MSE but a large test MSE.

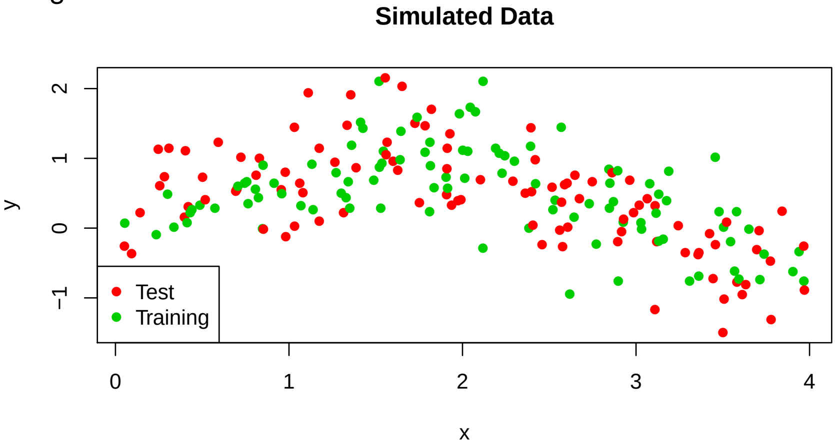
Overfitting

A method is **overfitting** the data when it has a small training MSE but a large test MSE.



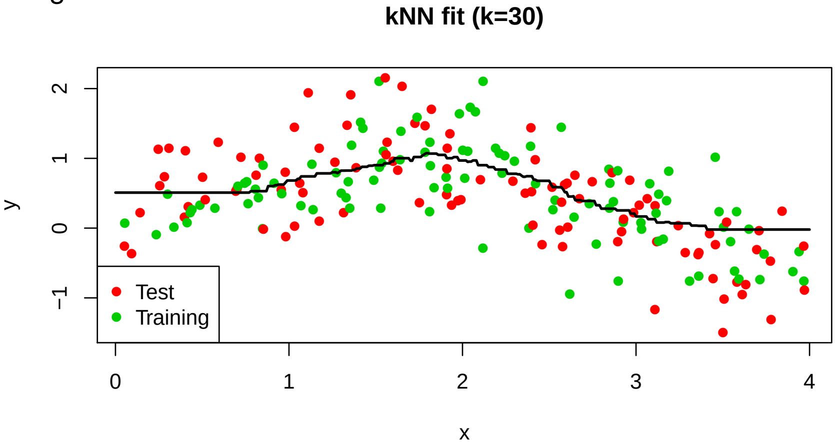
Overfitting

A method is **overfitting** the data when it has a small training MSE but a large test MSE.



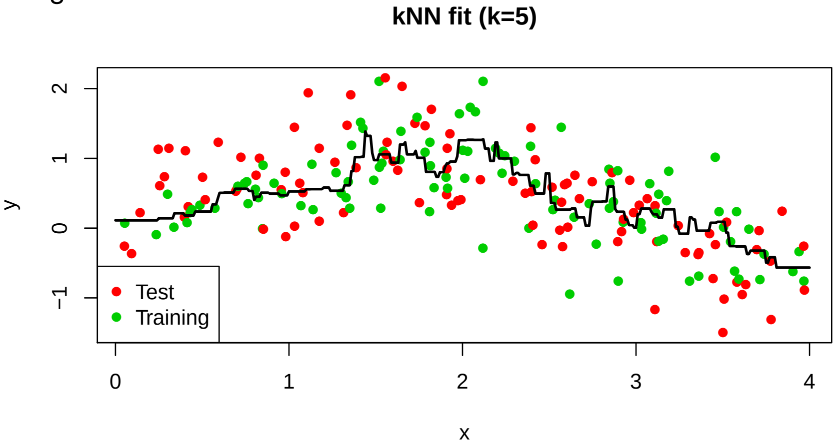
Overfitting

A method is **overfitting** the data when it has a small training MSE but a large test MSE.

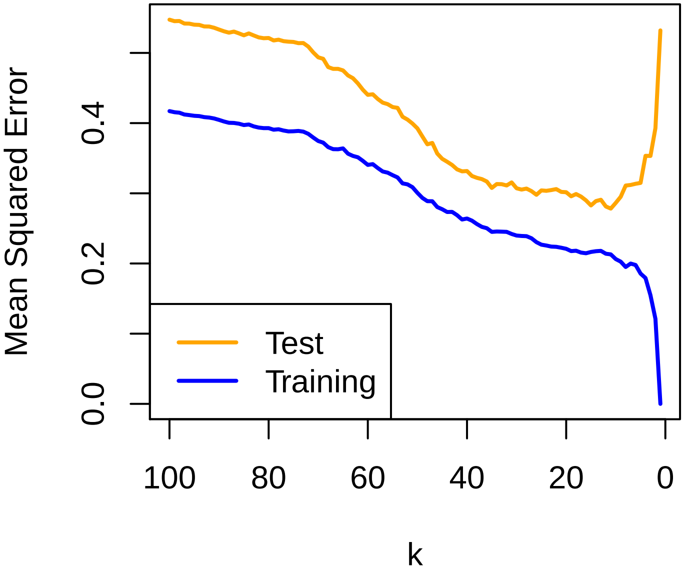


Overfitting

A method is **overfitting** the data when it has a small training MSE but a large test MSE.



Overfitting via k-Nearest Neighbors



Bias-Variance: *k*-NN

- As *k* increases, bias increases
- As *k* increases, variance decreases
- How to choose the best *k*?

variance is zero at start

"complexity" of model.

- with linear model, complexity corresponds to the number of features / parameters
- complexity also comes from the size of the parameters
  - the greater the size, the greater the complexity

- with decision trees, complexity comes from the number of leaves / depth and the minimum no. of observations in each leaf

Bias-variance: Decision trees

- As tree is grown deeper, bias decreases
- But the variance increases
- How to choose the right size of tree?

Bias-variance: Decision trees

http://www.r2d3.us/visual-intro-to-machine-learning-part-1

http://www.r2d3.us/visual-intro-to-machine-learning-part-2

Penalization

- When we “penalize” a classification or regression method, we minimize the sum of the squared error (or negative log-likelihood), plus a term that discourages big coefficients.
- This “shrinks” the coefficients compared to what they would be without the penalization.
- The result is decreased variance, at the expense of increased bias

Random forests

e.g. if you average n independent gaussians, the variance goes down as $\frac{1}{n}$

Here's the idea:

- A deep tree has small bias but big variance
- If we grow a bunch of deep trees, they will all tend to have low bias
- Averaging them will tend to reduce variance *if* they are strongly uncorrelated
- To reduce the correlation between the trees, we force them to be different by selecting random subsets of features at each split. Bootstrap sampling also is used.

Elisa's lecture

the most common technique

We inject randomness into the tree growing procedure to decorrelate the trees.

Why ever use a decision tree over a random forest ?

- The main reason is interpretability of a single tree
- In practice, you almost never use a single tree

Logistic regression

For Binary classification

- Logistic regression is the analogue of least squares linear regression for (binary) classification
- It's a linear model of the log-odds $\log(p/(1-p)) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$ where $p = P(Y = 1 \mid x)$
- To fit the parameters β_j, can use stochastic gradient descent
- This is the baseline classification method

Summary

- Two cultures: model based and prediction based
- Prediction based approaches are sometimes not interpretable
- Overfitting is easy with very flexible models and algorithms
- Finding the right complexity is a matter of balancing squared bias and variance