Lecture #2: Prediction, k-Nearest Neighbors

CS 109A, STAT 121A, AC 209A: Data Science

Weiwei Pan, Pavlos Protopapas, Kevin Rader

Fall 2016

Harvard University

Announcements

- Detailed instructions on how to submit in Vocareum will be posted under announcements on Canvas tonight.
- Note: Vocareum does not send you a notification, and please ignore Canvas' warnings after you submit.
- HW #0 must be submitted (deadline is tonight at midnight). Submit it complete or incomplete.
- HW #0 comments will be sent out shortly thereafter.
- HW #1 Due Tuesday at 11:59pm. Handed into Vocareum by clicking through Canvas.
- Expect to hear more details about projects when HW #2 is released.
- Always, always, always, check the announcements on Canvas.

Quiz Time

Time for Quiz.

The Data Science Process

Recall the data science process.

- Ask questions
- Data Collection
- Data Exploration
- Data Modeling
- Data Analysis
- Visualization and Presentation of Results

Today we will begin introducing the data modeling procedure (for prediction for now).

Lecture Outline

Statistical Modeling

Regression vs. Classification

Error/Loss Functions

k-Nearest Neighbors

Statistical Modeling

Predicting a Variable

Let's imagine a scenario where we'd like to predict one variable from another (or set of other) variables.

Examples:

- Predicting the amount of views a YouTube video will get next week based on video length, the date it was posted, previous number of views, etc...
- Predicting which movies a Netflix customer will rate highly based on his or her previous movie ratings, demographic data, etc...
- Predicting whether or not a customer on Amazon will purchase something based on mouse clicks, demographic data, what ads they are presented, etc...

Outcome vs. Predictor Variables

There is an asymmetry in a lot of these problems. The variable we'd like to predict may be more difficult to measure, is more important than the others, or may be directly or indirectly influenced by the levels of the other variable(s). Thus we define:

- The *outcome* or *response variable* as the variable we'd like to predict, and typically call this variable Y (and the measurements y_i).
- The *features* or *predictor variables* as the variables we can use to do the predictions, and typically call these variables $X = (X_1, ..., X_p)$ (and the measurements $x_{i,j}$).

Note: i represents which observation ($i = 1, 2, ..., n$) and j represents which predictor variable we are referring to ($j = 1, 2, ..., p$).

True vs. Statistical Model

We will assume there is a general form of how the response, Y, relates to the predictors, X, in the general, theoretical sense:

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

Here f is the unknown function that relates Y to X, and ϵ is the random error term (unrelated to the predictors, X).

We define a statistical model as any algorithm that attempts to estimate f. This estimated function is written as $\hat{f}$.

A measured set of predictors for an observation $(x_{i,1}, ..., x_{i,p})$ can then be used to predict the response variable. This resulting predicted value is called $\hat{y}_i$.

Prediction vs. Estimation

For some problems, what we care about estimating is f, the function that relates Y to X. These are often called *inference* problems.

For other problems we don't necessarily care about what the mathematical function that f takes on, but what we care about is getting $\hat{y}_i$ as close to y_i as possible. These are called *prediction* problems.

We'll see that some algorithms are better suited for inference, some are better for prediction, and others are a little bit of both.

Regression vs. Classification

Outcome Variables

There are two main types of prediction problems that we will see this term:

- *Regression* problems are ones with a quantitative response variable.
- *Classification* problems are ones with a categorical response variable.

The distinction is important as measuring how effective a method is depends on this distinction.

Error & Loss Functions

Error & Loss Functions

In order to determine how well a modeling approach works, one can define a *Loss function*, aka Error function, to quantify this.

What would be a reasonable Loss function for a quantitative outcome variable? How about for a categorical one?

One common loss function for quantitative outcomes is called Mean Squared Error (MSE) loss:

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

For categorical outcomes a common loss function is the number of misclassified outcomes:

$$L = \sum_{i=1}^{n} I\left(y_i \neq \hat{y}_i\right)$$

Using Loss Functions

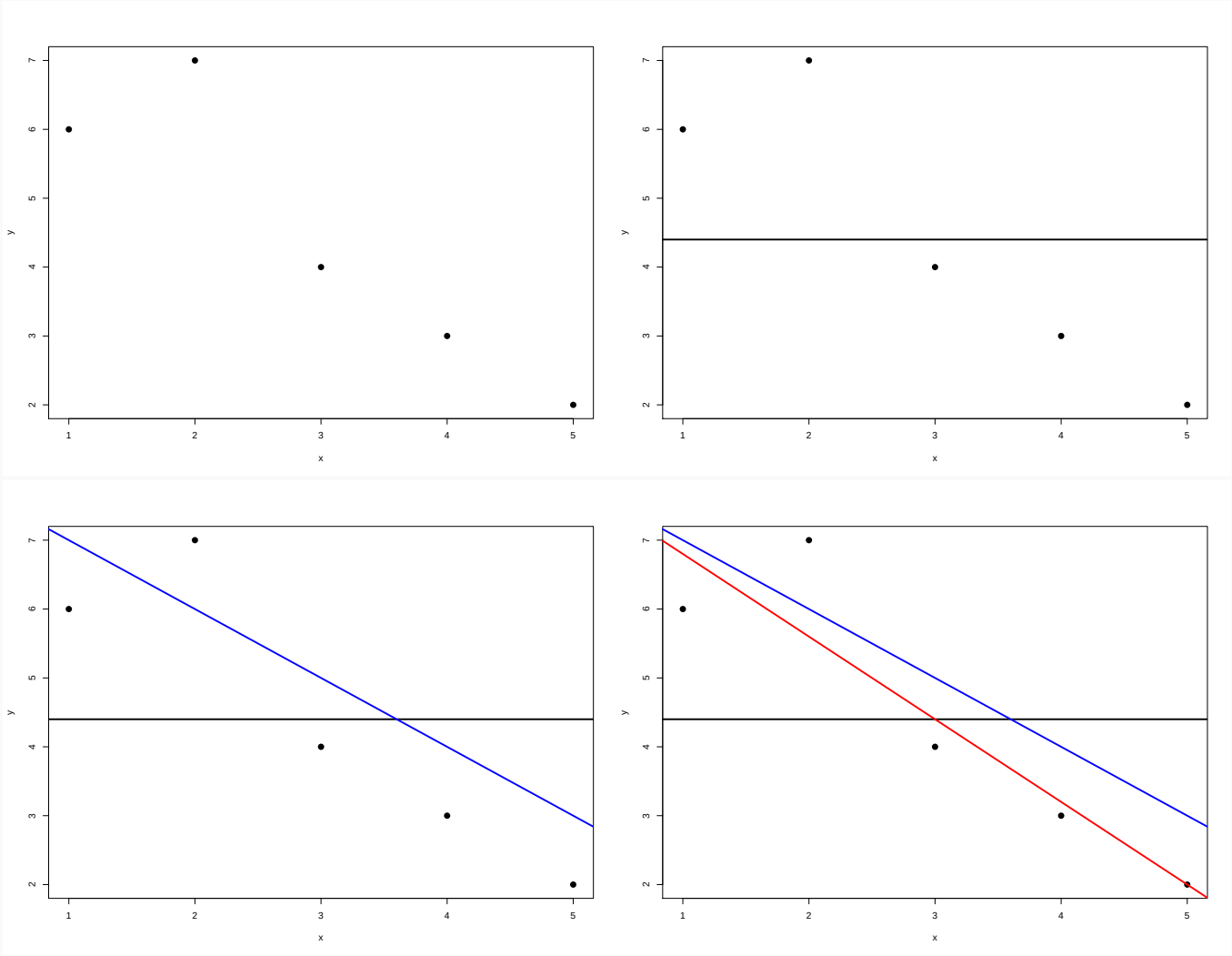
A statistical modeling approach is often an algorithm that may put some mathematical framework on f, and then

For example: linear regression says that f is a linear function. More on that next time.

Then to estimate f, we choose the values that minimize the loss function. For linear regression, we choose the best line out of all infinitely many lines that minimizes MSE.

A picture is worth a thousand words...

Line of Best Fit



Which line performs the best? How do you know?

k-Nearest Neighbors (k-NN)

k-Nearest Neighbors

One algorithm to predict an outcome variable (if quantitative) or classify an outcome variable (if categorical) is by using the k-Nearest Neighbors approach.

The approach is simple: to predict an observation's response, use the **other** available observations that are most similar to it.

For a specified value of k, each observation's outcome is predicted to be the average of the k-closest observations as measured by some distance of the predictor(s).

With one predictor, the implementation can be easily illustrated for a small example.

k-NN: a Simple Regression Example

Suppose you have 5 observations each with an outcome and one predictor. The rectangular data set (transposed to save space) is thus:

X	$1, 2, 3, 4, 5$
Y	$6, 7, 4, 3, 2$

Calculate the predicted values for the k-NN procedure for $k = 2$.

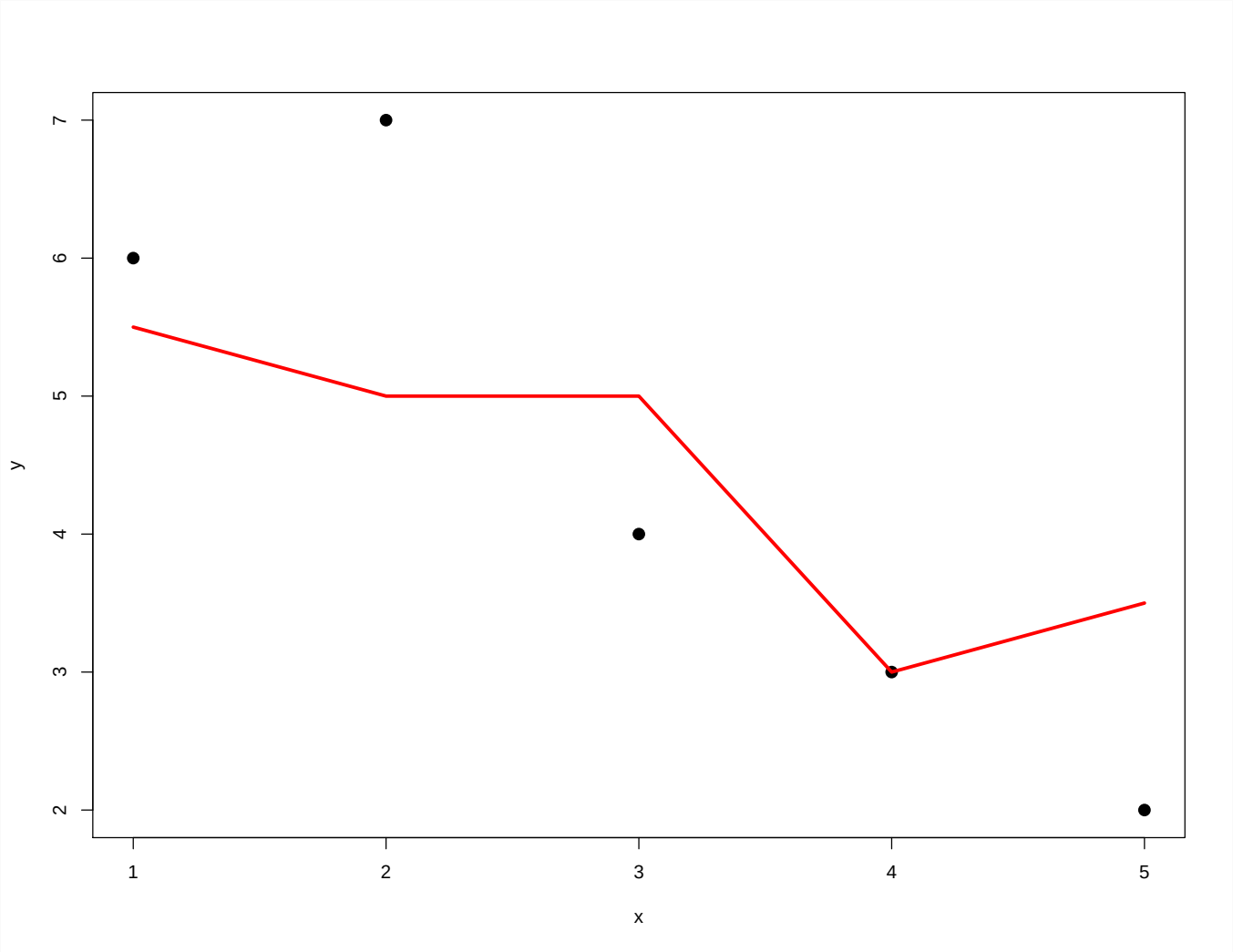
$\hat{Y} = (5.5, 5.0, 5.0, 3.0, 3.5)$

Calculate MSE for these predictions.

$$MSE = \frac{1}{5}\left((6 - 5.5)^2 + (7 - 5.0)^2 + ... + (3.5 - 2)^2\right) = 1.5$$

We could plot these predictions to see how well this procedure has performed:

Simple example plotted



Training vs. Testing set

The k-NN approach illustrates an important idea of determining how well an algorithm is performing in predicting a response.

The data are often split into two distinct subsets:

- The *training data* are the observations we used to estimate f or calculate $\hat{y}$.
- The *testing data* set are the observations we use to determine how well the model fits.

Note: often the entire set of data are used as the training set.

Training vs. Testing set (cont.)

The default in Python is to use all provided data as the training set ‘ ‘since a user fits the model on a specified training set and predicts on a separate testing set.

Tricky: for our k-NN example what were the training and testing sets? We essentially had to use the other $n - 1$ observations to train the model for the each left out observation, one at a time. Thus our training set changed for each observation.

k-NN for future observations

This leads to a slight distinction in the algorithm if we have a well specified training set (data for which the outcome is observed) and testing set (new data that we don't know the outcome and would like to predict it).

In this case we use all the observations in the training set to make the predictions on the new observations. This is sometimes called out-of-sample prediction.

Note: This illustrates a useful application of k-NN: imputation of missing values.

Choice of k

How well the predictions perform is related to the choice of k.

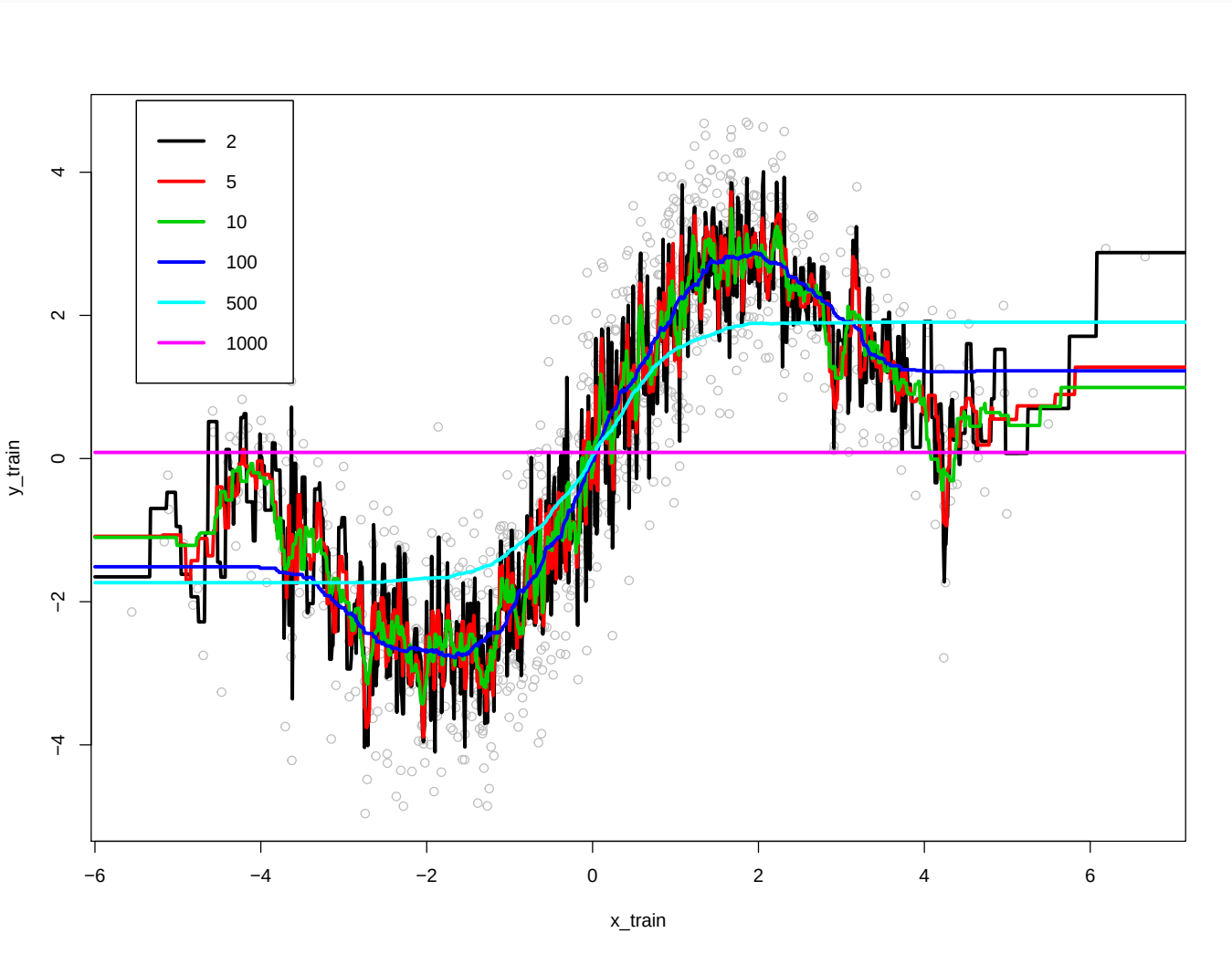
What will the predictions look like if k is very small? What if it is very large?

More specifically, what will the predictions be for new observations if $k = n$?

$$\bar{y}$$

A picture is worth a thousand words...

Choice of k Matters



k-NN with Multiple Predictors

How could we extend k-NN when there are multiple predictors?

We would need to define a measure of distance for observations in order to which are the most similar to the observation we are trying to predict.

Euclidean distance is an option. But what must we be careful about?

Differences in variability in our predictors!

k-NN for Classification

The other type of prediction problem we will run into is classification of a categorical response variable.

The approach here is the same as for k-NN for Regression: use the other available observations that are most similar to the observation we are trying to predict (classify into a group) based on the predictors at hand.

How do we classify which category a specific observation should be in based on its nearest neighbors?

The category that shows up the most among the nearest neighbors.

k-NN for Classification (cont.)

There are some issues that may arise:

- How can we handle a tie?
  With a coin flip!
- What could be a major problem with always classifying to the most common group amongst the neighbors?
  If one category is much more common than the others then all the predictions may be the same!

- How can we handle a this?
  Rather than classifying with the most likely group, used a biased coin flip to decide which group to classify to!

Note: we will revisit k-NN later in the semester and get much further into the nitty gritty.