REGRESSION PART 1: K NEAREST NEIGHBORS (KNN)

Hsin-Min Lu
盧信銘
台大資管系

K Nearest Neighbor (KNN) Regression

- training data: (y_i, x_i)
    - y_i is the outcome (label) with continuous values (i.e., real valued outcomes)
    - x_i is the feature vector of length m.
- To predict Y for a given value of X, consider k closest points to X in training data and take the average of the responses. i.e.
- $f(x) = \frac{1}{K}\sum_{x_i \in N_i} y_i$
- KNN is often referred to as the nonparametric model because it does not assume any parametric form of the prediction model (as oppose of linear regression)

Measuring Distance (or Similarity)

- L2 Norm (Euclidian distance):

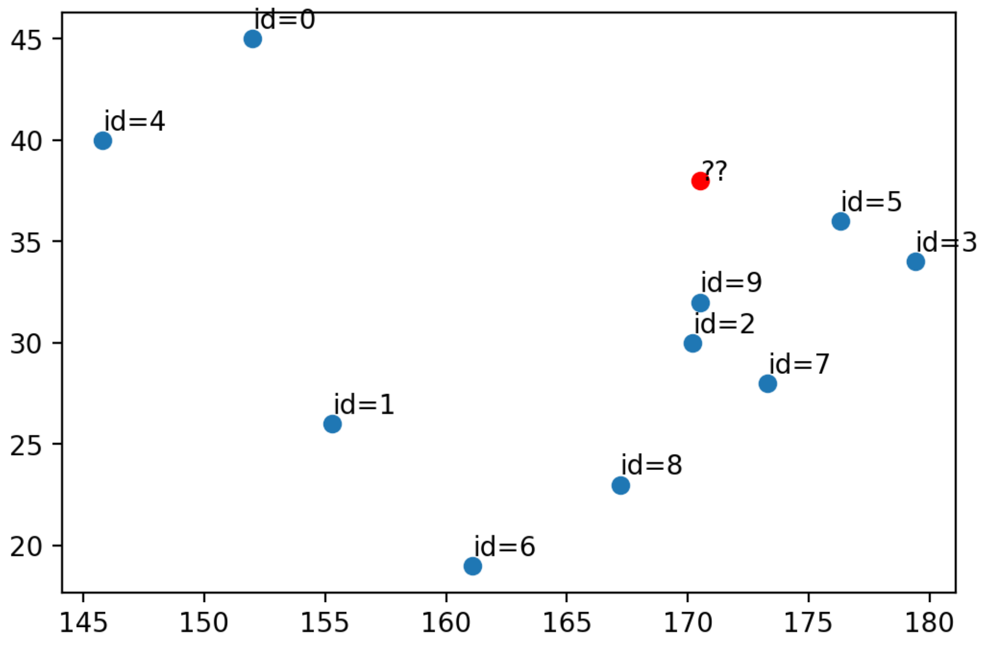
$$\left\| x_i - x_j \right\| = \sqrt{\sum_{q=1}^{m} \left(x_{i,j} - x_{j,q} \right)^2}$$

- L1 Norm:

$$\left\| x_i - x_j \right\|_1 = \sum_{q=1}^{m} \left| x_{i,j} - x_{j,q} \right|$$

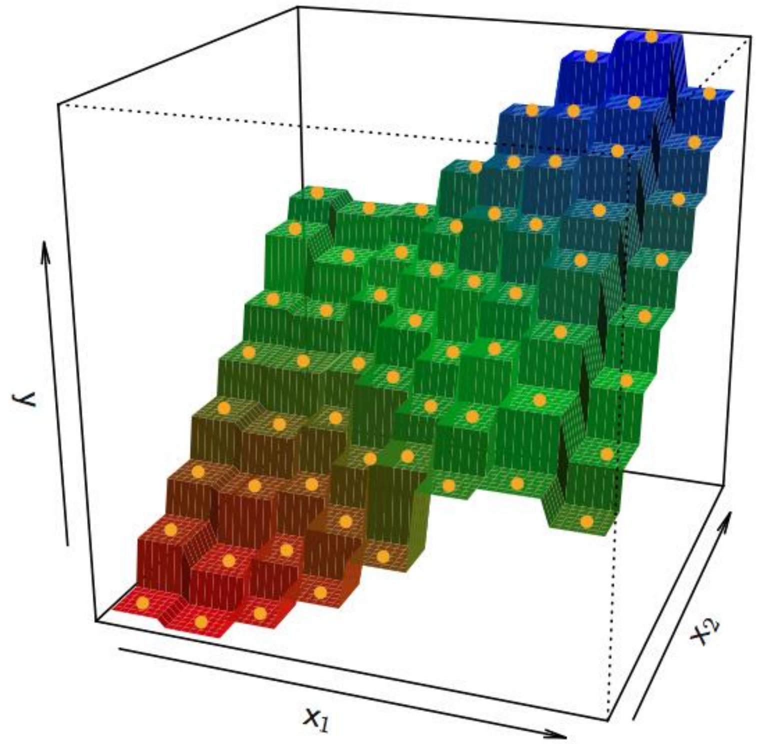
Example Dataset

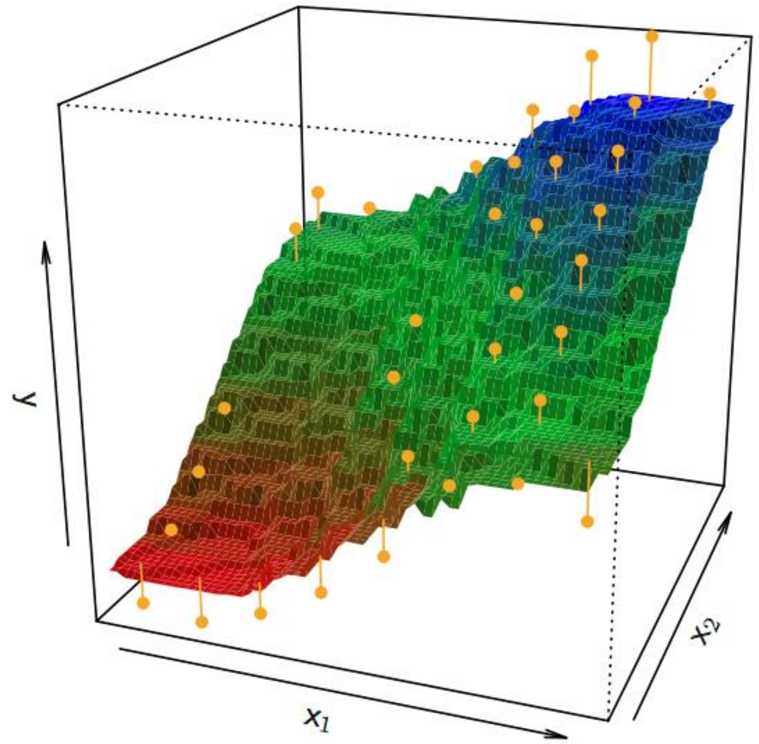
- Predicting weight using height and age
- Test data: [170.5, 38]
- K=3, ID = 9, 5, 2
- Ypred = (58 +60 +55)/3



ID	height	age	weight
0	152.0	45.0	77.0
1	155.3	26.0	47.0
2	170.2	30.0	55.0
3	179.4	34.0	59.0
4	145.8	40.0	72.0
5	176.3	36.0	60.0
6	161.1	19.0	40.0
7	173.3	28.0	60.0
8	167.2	23.0	45.0
9	170.5	32.0	58.0

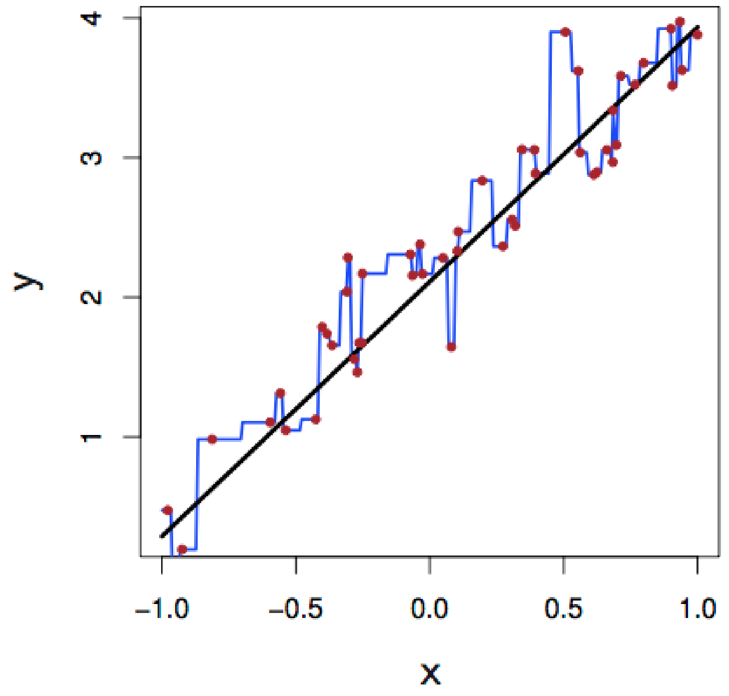
KNN Fits for k =1 and k = 9

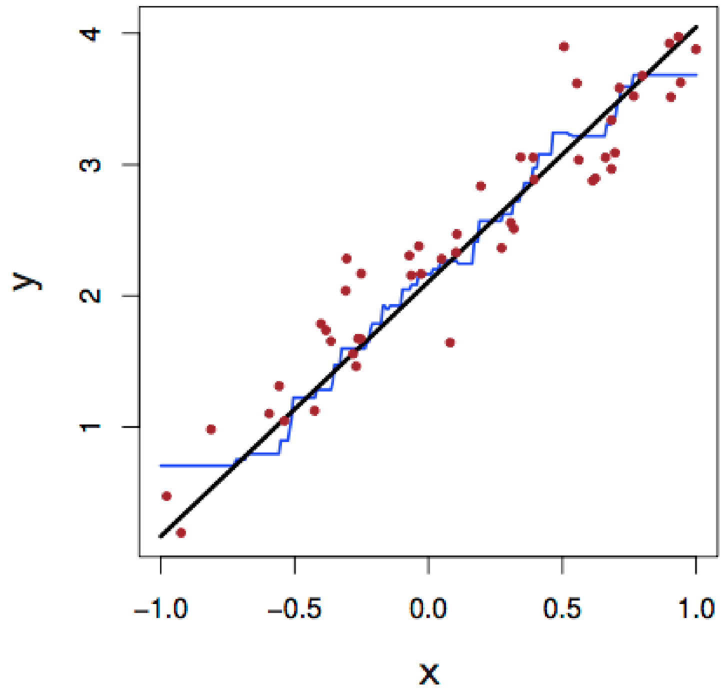




KNN Fits in One Dimension

Black line: true relation.
Left: k=1; right: k=9





Measuring Prediction Performance

- Typical arrangement:
- Random permutation, and divide the data into training (90%) and testing (10%) sets.
- Training set can be further divided into subtraining (80%) and tuning (10%) sets.
- If the model has no hyperparameters, then train on training and test on testing.
- If the model contain hyperparameters, then use subtraining and tuning to tune the hyperparameters.
- Next, fix hyperparameters, and train on training and test on testing.

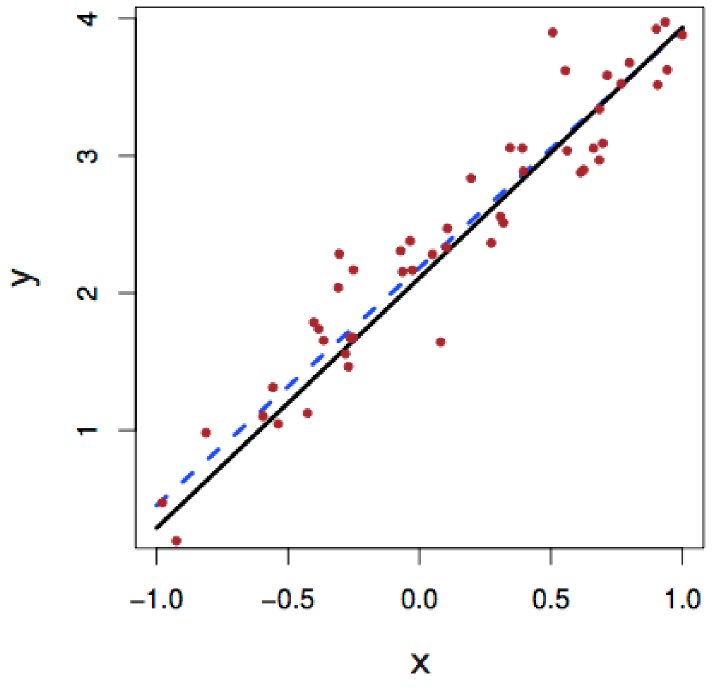
Measuring Prediction Performance

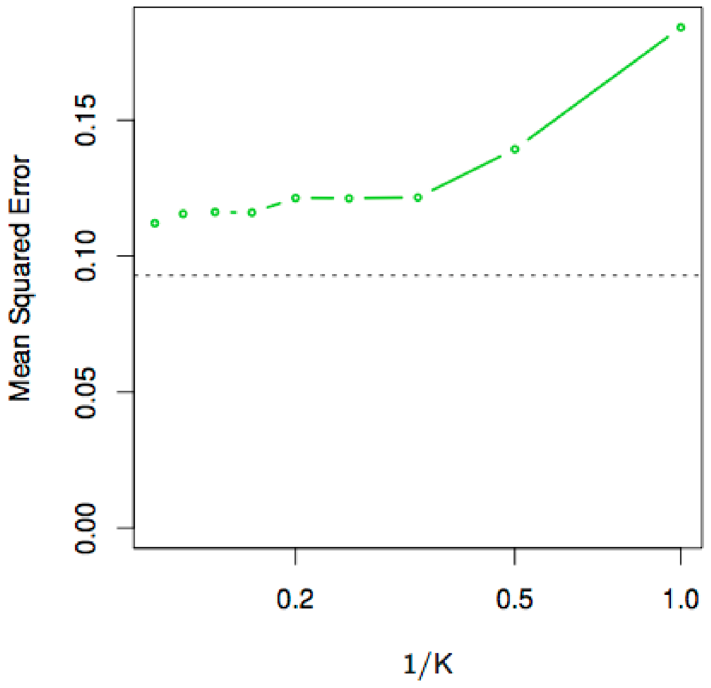
- Mean Squared Error: $MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2$
- Root Mean Squared Error: $RMSE = \sqrt{MSE}$
- Mean Absolute Error: $MAE = \frac{1}{n}\sum|y_i - \hat{y}_i|$

- In this particular example, we use simulated data, so we can generate training and testing data by ourselves.

Compare with Linear Regression Fit

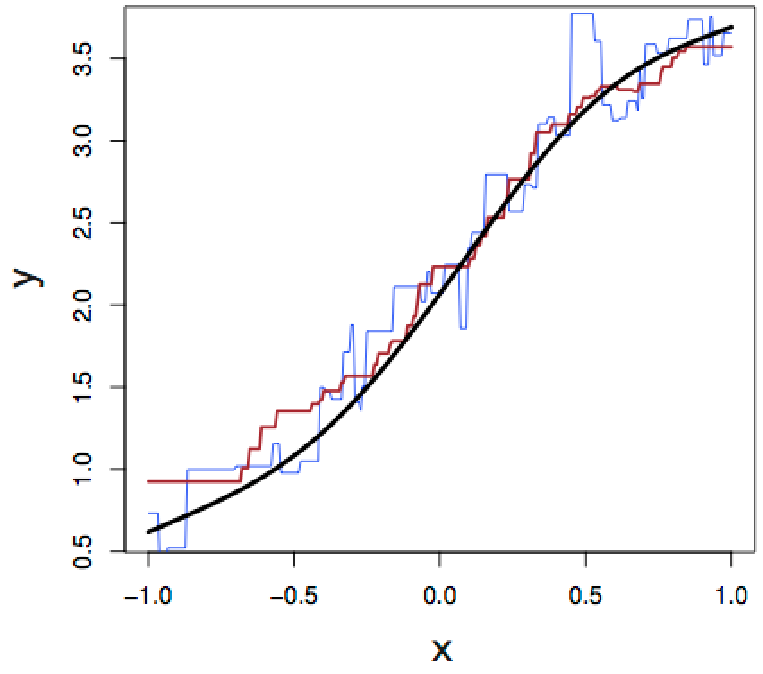
When the true relation is linear
Blue dashed line (left figure): least square fit
Dashed horizontal line (right picture): MSE of least square
Green solid line (right picture): KNN MSE.

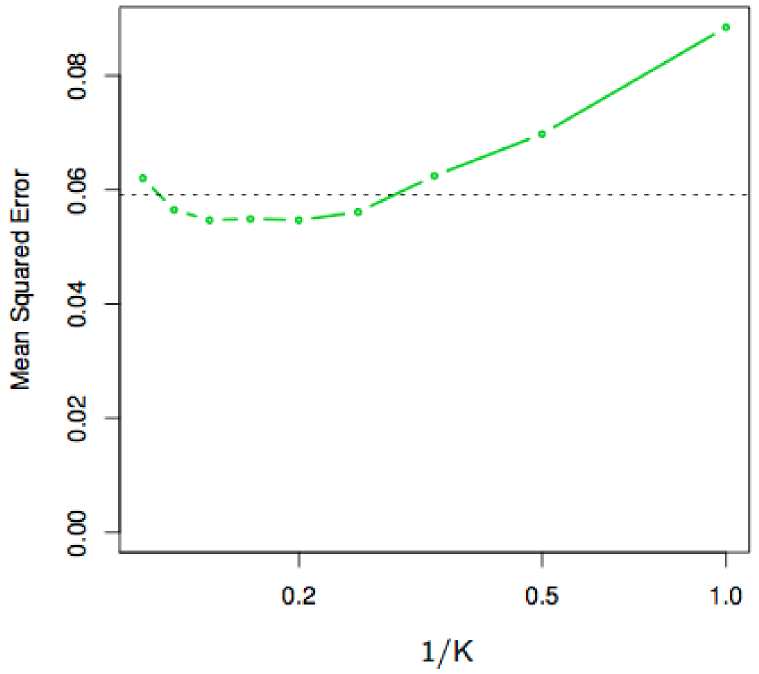




When the True Relation is Nonlinear

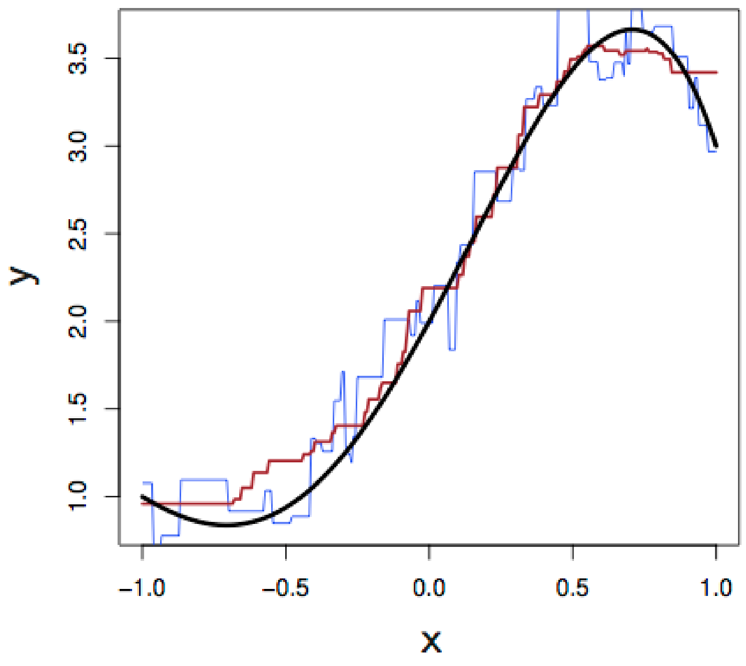
- Blue line (left): k=1; red line (left): k=9
- Black dotted line (right): linear regression MSE.
- Green solid line (right): KNN MSE

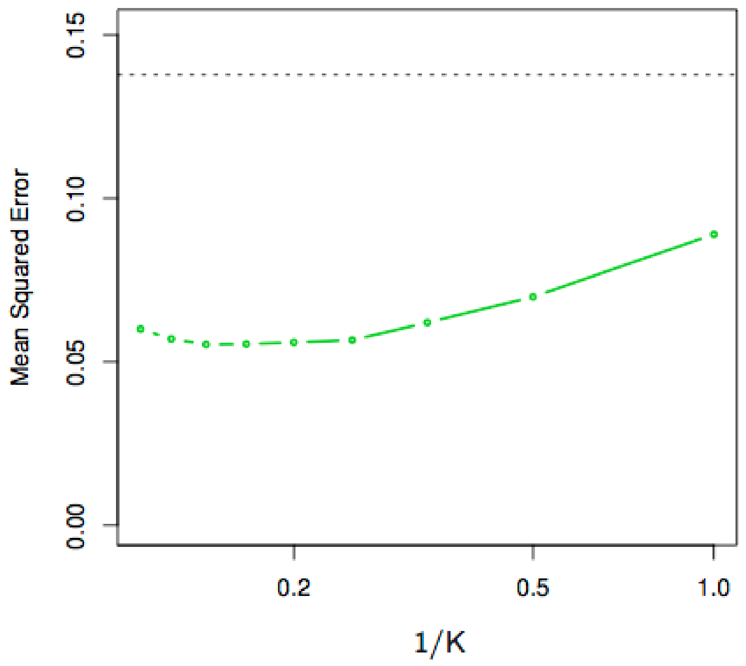




When the True Relation is Extremely Nonlinear

- Blue line (left): k=1; red line (left): k=9
- Black dotted line (right): linear regression MSE.
- Green solid line (right): KNN MSE





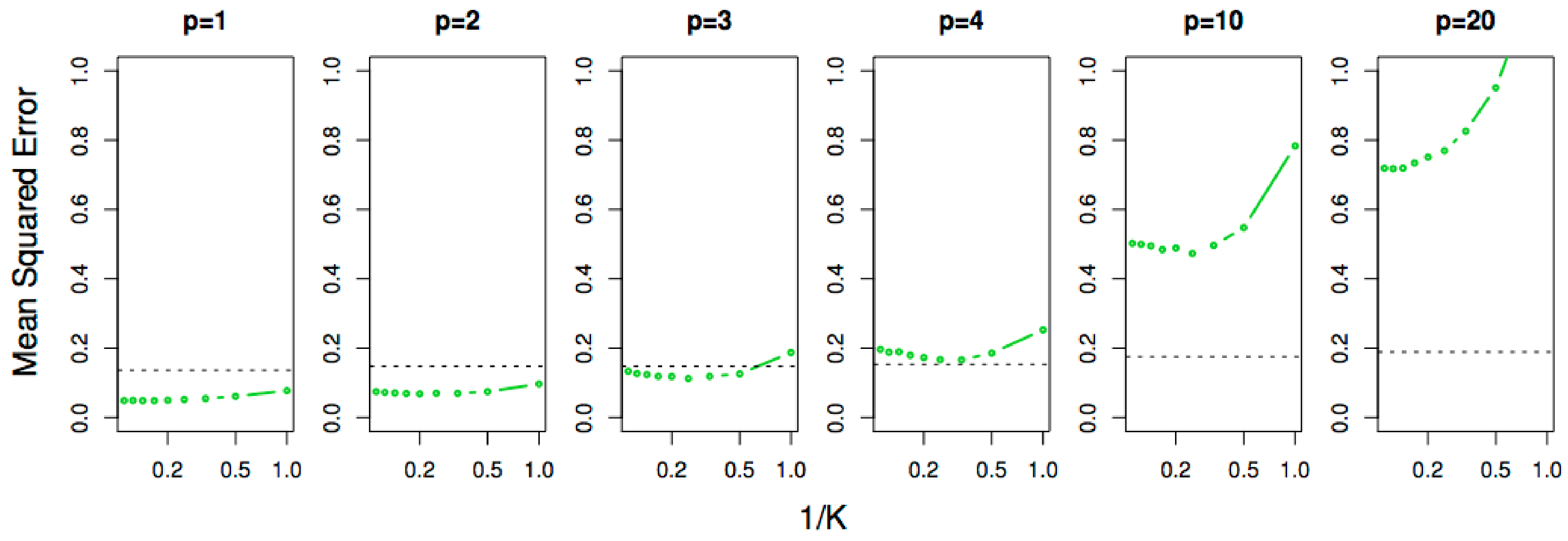
Several Observations

- KNN can be worse than linear regression if the underlying assumption for linear model is correct.
- If the true model is nonlinear, then a good choice for k can easily outperform the linear model.

- Another point to make: If the dataset is small, then parametric models (e.g. linear regression) typically perform better.

- There is another weakness for KNN.

Not So Good in High Dimensional Situations



- The true function is non-linear in the first variable, as in the previous examples.
- Other features have no effect on the outcome variable.
- The linear regression deteriorates slowly in the presence of additional noisy variables.
- KNN degraded much quickly as the including of additional noisy variables.

Computational Complexity

- A naïve implementation does not require any computation in training. You just need to store the training data.
- To predict, the model compare an input data point to all training data points and select the k closest training data point.
- ➔ Time complexity for predicting one data point is $O(N)$
- When the training dataset is large, this approach is quite slow.
- There are good data structures such as **Ball Tree** and **K-D Tree** to reduce the time complexity to $O(\log N)$.
- However, the prediction speed is still quite slow compared to linear regression.

Hyper-parameter Tuning

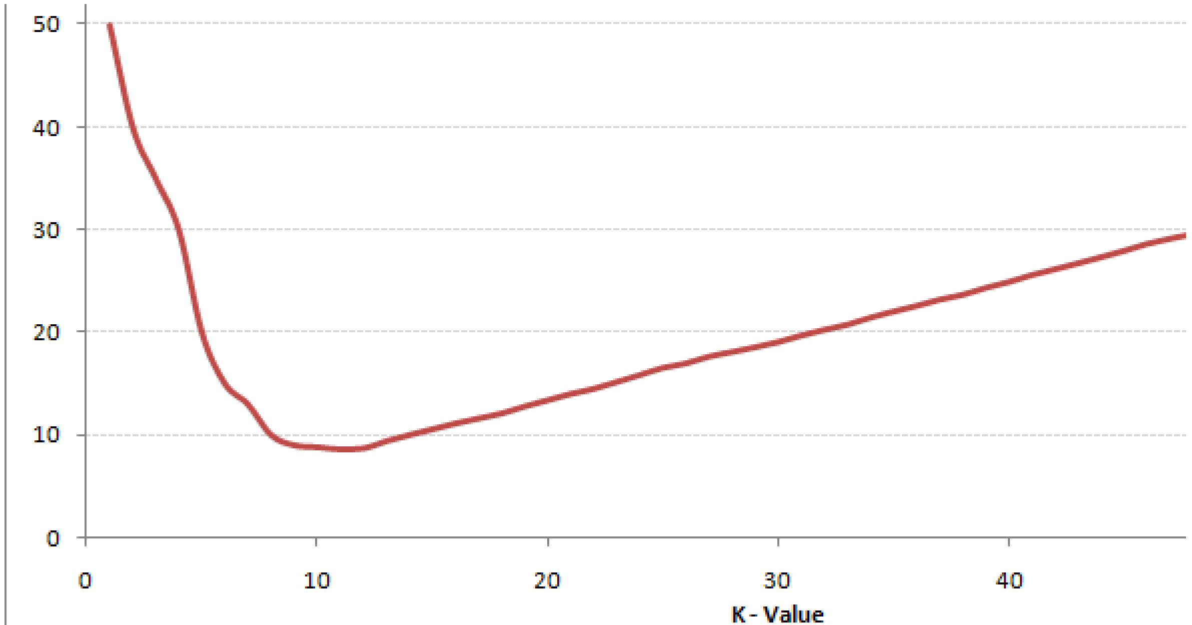
- The “k” is the hyper-parameter that needs to be tuned before we can reliably evaluate the prediction performance.
- In order to do so, randomly split the data into: subtraining (80%), tuning (10%), and testing (10%).
- Select candidate k. Usually should cover “extreme” and “good guess”
- Extreme: k=1, k=N/2
- Good guess: 5, 10, 20, 50,
- Need not be equally spaced,
- E.g. 1, 3, 5, 10, 20, 50, 100, 200, 500

Hyper-parameter Tuning (Cont'd.)

- For each candidate k,
- Train on subtraining, test on tuning, record RMSE

- Plot RMSE w.r.t. k, you should see a U shape.
- Pick one the k with the lowest RMSE.
- Fix the k, train on subtraining+tuning, test on testing.

- Issues:
- What if it is not U-shaped?
- What if several k have very close RMSE?


K - Value