Business Analytics II

Lecture 2

Non-Parametric Regression

Pedro Hernandez Serrano
p.hernandezserrano@maastrichtuniversity.nl

April 10, 2018

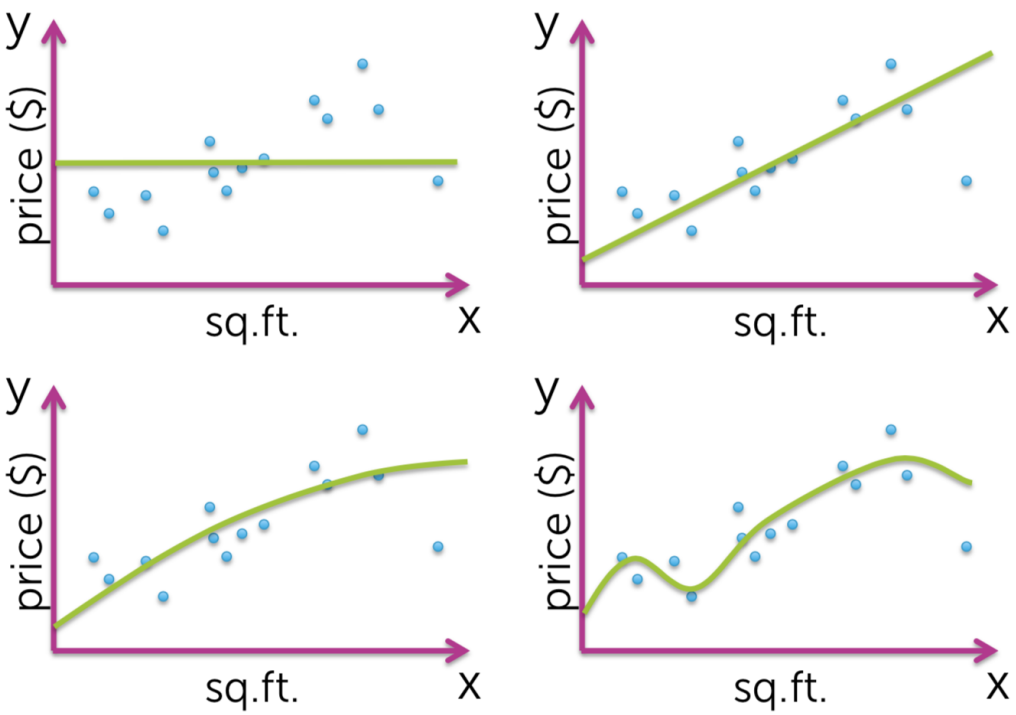
BISS Institute

Today's session

- Supervised Learning - Regression in Python
  - K-nearest Neighbour Regression
  - Decision Tree Learner
  - Random Forest Regression
- Hands-on during the session on Jupyter Notebooks
  - Construct a Non-parametric regressors
  - Regression Models using Turicreate Machine Learning Library

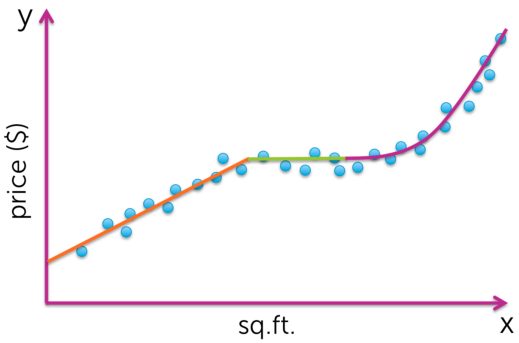
Non-parametric Regression: K-Nearest Neighbors

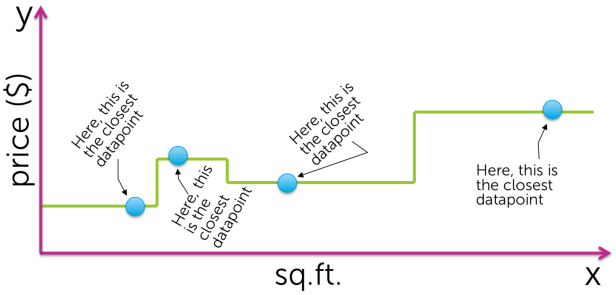
Parametric



Non-parametric Regression: K-Nearest Neighbor

Non-Parametric





Non-parametric Regression: K-Nearest Neighbor

- Nonparametric regression is a collection of techniques for fitting a curve when there is little a priori knowledge about its shape.
- The simplest nonparametric regression estimators are local versions of location estimators. For a random variable (x, y), the regression curve $\mu(x) = E(y|x)$ shows how the mean of the dependent variable y varies with the independent variable x.
- The model immediately adapts as we receive new training examples, but computational complexity grows lineary with the number of samples.
- Find a heuristically optimal number k of nearest neighbours, based on RMSE.

Distance Metrics

1-D Euclidean norm

$$||a||_2 = \sqrt{\sum_i a_i^2} = |a_j - a_q|$$

Euclidean distance between two vectors $a, b \in R$ is

$$||a - b||_2 = \sqrt{\sum_i^P (a_i - b_i)^2}$$

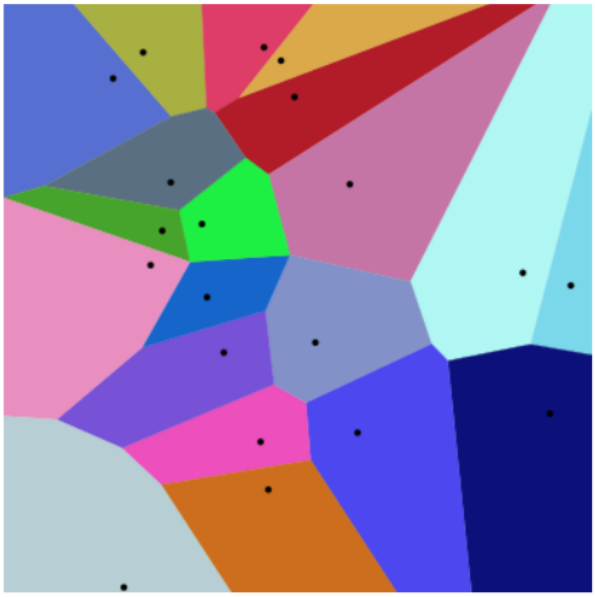
Other example distance metrics: Mahalanobis, rank-based, correlation-based, cosine similarity, Manhattan, Hamming, ...


```
import numpy as np

def dist(x,y):
        sum2 = np.sum((x-y)**2)
        distance = np.sqrt(sum2)
    return distance
```

Non-parametric Regression: K-Nearest Neighbor

Euclidean distance



Manhattan distance

Non-parametric Regression: K-Nearest Neighbor

1-NN Reasoning

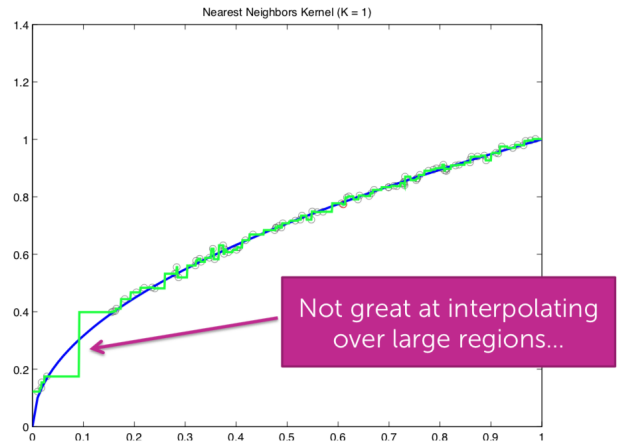
- Sort the X vector
- Compute distance a new x_i *for in* N *len*(X)
- Choose the lowest distance
- Take the Y that corresponds to that index

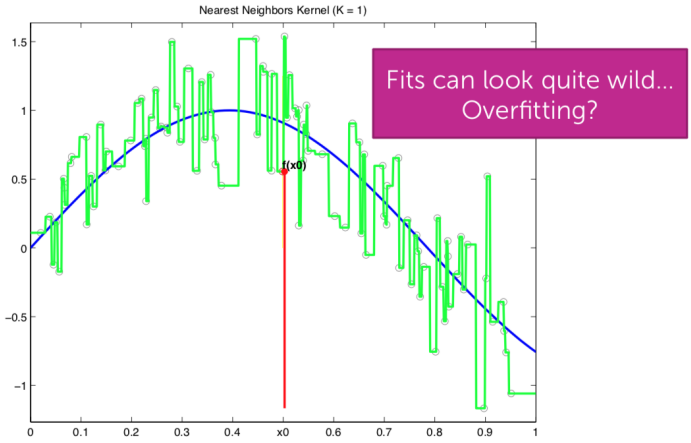
As simple as returning the price of the house with the similar characteristcs

k-NN Predict

$$\hat{Y}_q = \frac{1}{k}(Y_{NN_1}, Y_{NN_2}, \ldots, Y_{NN_k}) = \frac{1}{k}\sum_{i=i}^{k} Y_{NN_j}$$

Non-parametric Regression: K-Nearest Neighbor



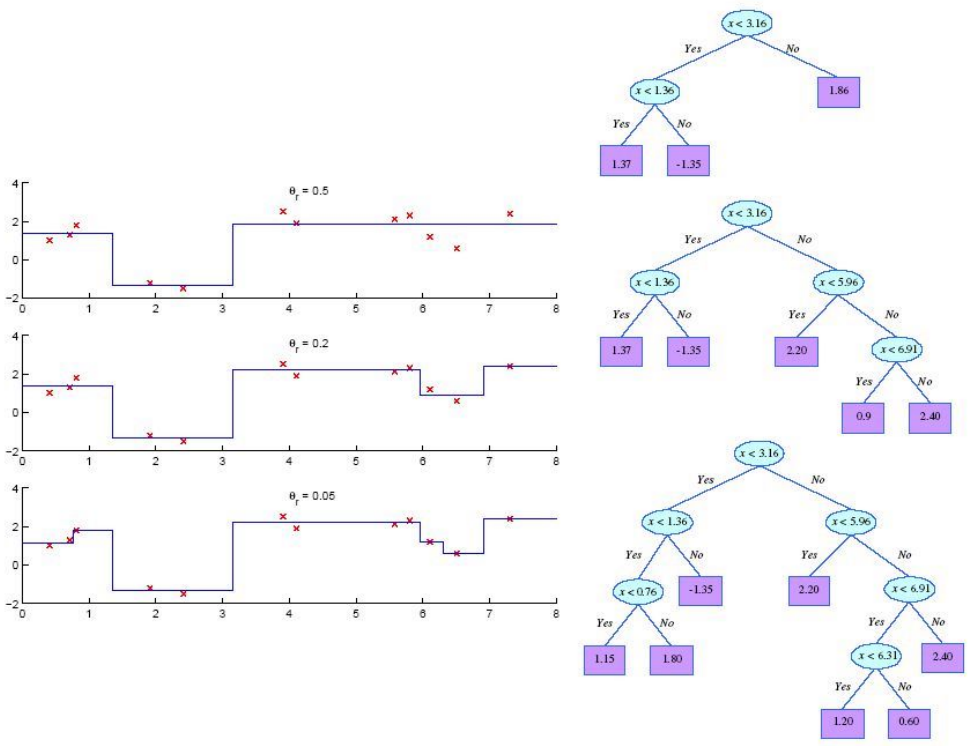


Decision Tree Learner and Random Forest Regression

Decision Tree Learner

- One of the most widely used and practical methods for inductive inference
- Approximates discrete-valued functions (including disjunctions)
- Can be used for classification or regression since accept any variable.
- Are attractive models if we care about interpretability. Like the name decision tree suggests, we can think of this model as breaking down our data by making decisions based on asking a series of questions
- Unlike linear models, decision trees are able to capture non-linear interaction between the features and the target.
- In classification is used **Information Gain (IG)** function, in regression we use **MSE**

Decision Tree Learner



Decision Tree Learner

In the context of decision tree regression, the MSE is often also referred to as within-node variance, which is why the splitting criterion is also better known as variance reduction.

$$\hat{y}_t = \frac{1}{N} \sum_{i \in D_t} y^{(i)}$$

- Each internal node represents a test on an feature resulting on the split of the current sample.
- At each step the algorithm selects the feature and a cutoff value that maximises a given metric.
- Different metrics exist for regression tree (target is continuous) or classification tree (the target is qualitative).

Decision Tree Learner

Note: is that tree based models are not designed to work with very sparse features. When dealing with sparse input data (e.g. categorical features with large dimension), we can either pre-process the sparse features to generate numerical statistics, or switch to a linear model, which is better suited for such scenarios.

Random Forest Regression

Random Forest

- An ensemble technique that combines multiple decision trees.
- A random forest usually has a better generalization performance than an individual decision tree due to randomness that helps to decrease the model variance.
- Other advantages of random forests are that they are less sensitive to outliers in the dataset and don't require much parameter tuning.
- The only parameter in random forests that we typically need to experiment with is the number of trees in the ensemble.

Random Forest Regression

Tuning hyperparameters

num trees: Controls the number of trees in the final model. Usually the more trees, the higher accuracy. However, both the training and prediction time also grows linearly in the number of trees.
step size: Also called shrinkage. It works similar to the learning rate of the gradient descent procedure: smaller value will take more iterations to reach the same level of training error of a larger step size. So there is a trade off between step size and number of iterations.
min child weight One of the pruning criteria for decision tree construction. In classification problem, this corresponds to the minimum observations required at a leaf node. Larger value produces simpler trees.
min loss reduction Another pruning criteria for decision tree construction. This restricts the reduction of loss function for a node split. Larger value produces simpler trees.

Random Forest Regression

Case: Price Rooms

- Go to **decision tree** notebook
- Follow the instructions
- Fit a Decision Tree and a Random Forest model