PGE 383 Subsurface Machine Learning

Lecture 11: k-nearest Neighbours

Lecture outline:

- **Mapping in the Feature Space**
- **Hyperparameter Training**
- **k-Nearest Neighbour**
- **k-Nearest Neighbour Example**
- **k-Nearest Neighbour Hands-on**

k-Nearest Neighbour Regression

Motivation to Cover this k-nearest neighbours regression

- simple and interpretable
- instance-based, lazy learning
- linkage to variance-bias trade-off
- introduce our first hyperparameter
- very flexible, performs well in many situations

Vintage line-colored 1791 map of North America printed in England.

PGE 383 Subsurface Machine Learning

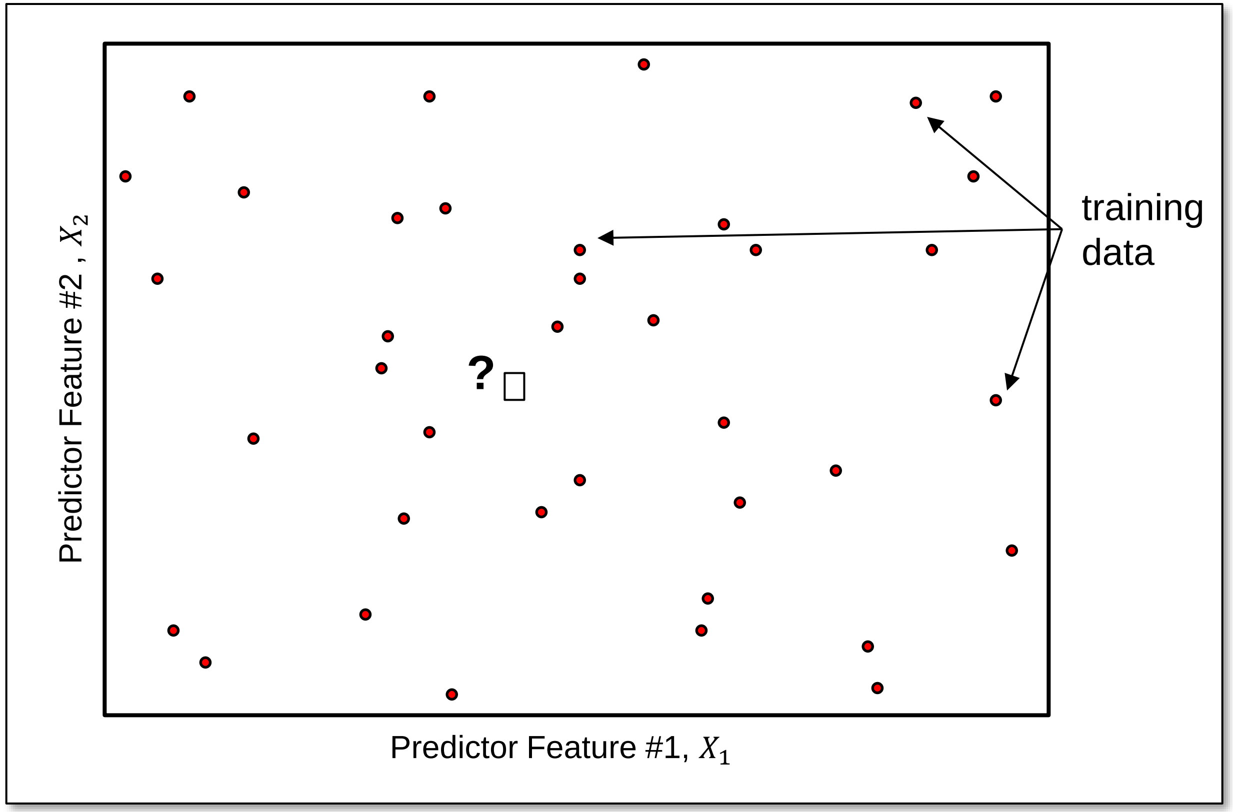
Lecture 11: k-nearest Neighbours

Lecture outline:

- **Mapping in the Feature Space**

Mapping the Response in the Predictor Feature Space

Mapping the Response Feature in the Predictor Feature Space

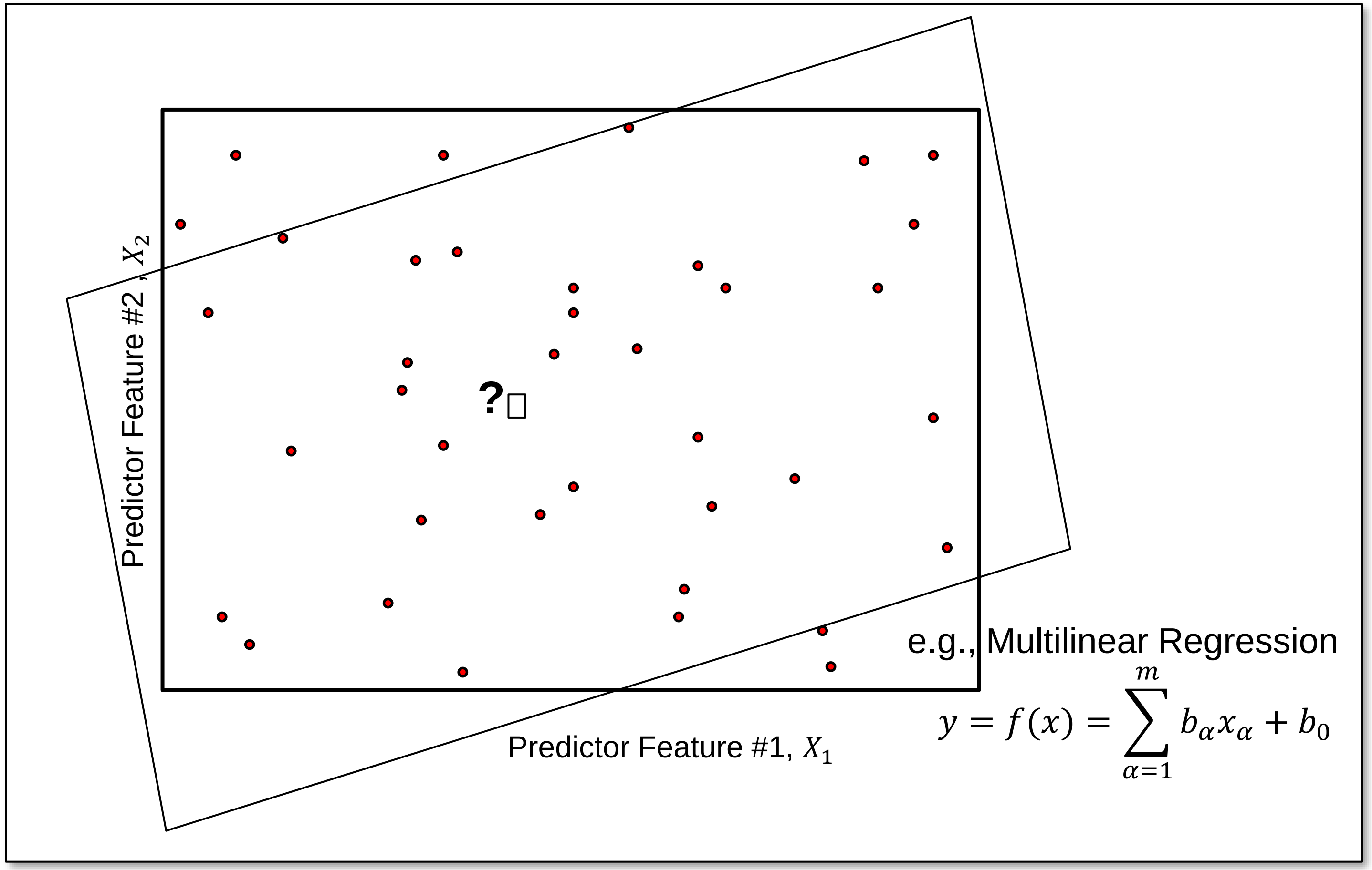


Training data in the predictor feature space.

- We want to make predictions away from training data

Mapping the Response in the Predictor Feature Space

Consider a Parametric Model for $\hat{y} = \hat{f}(x)$, Multilinear Regression



Training data in the predictor feature space.

Recall Parametric Models

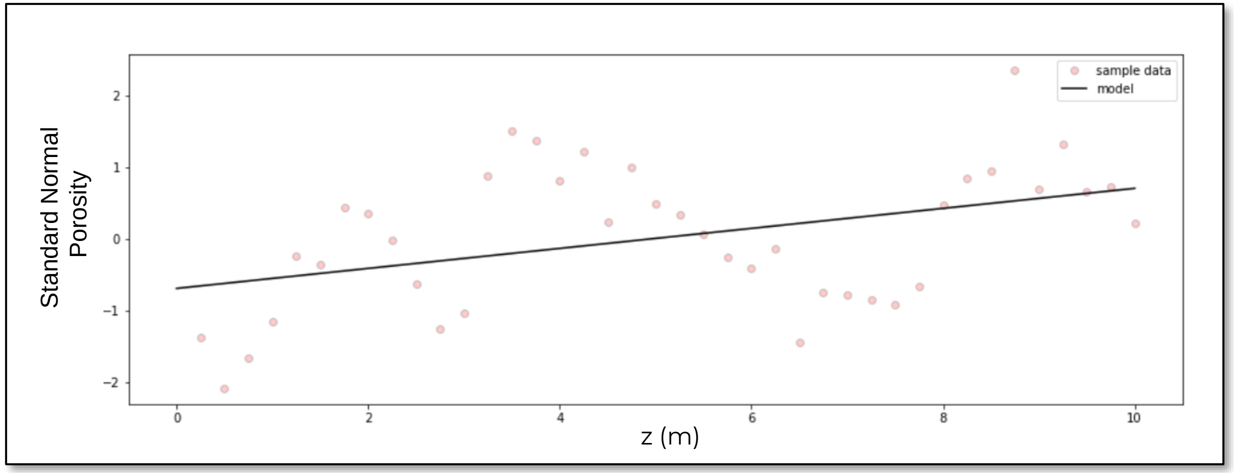
Working with Parametric Models

Makes an assumption about the functional form, shape

- We gain simplicity and advantage of only a few parameters

- For example, here is a linear model:

$$Y = f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_m X_m$$



Linear regression model to predict porosity from the z coordinate.

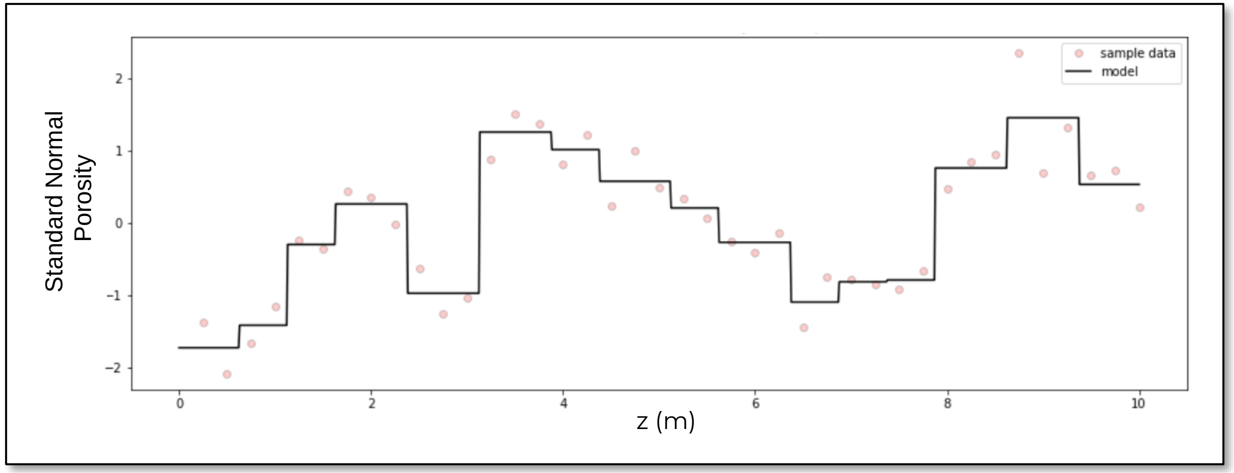
- There is a risk that $\hat{f}$ is quite different than f, then we get a poor model!

Nonparametric Models

Working with Nonparametric Models

Makes no assumption about the functional form, shape

- More flexibility to fit a variety of shapes for f
- Less risk that $\hat{f}$ is a poor fit for f
- Typically need a lot more data for an accurate estimate of f

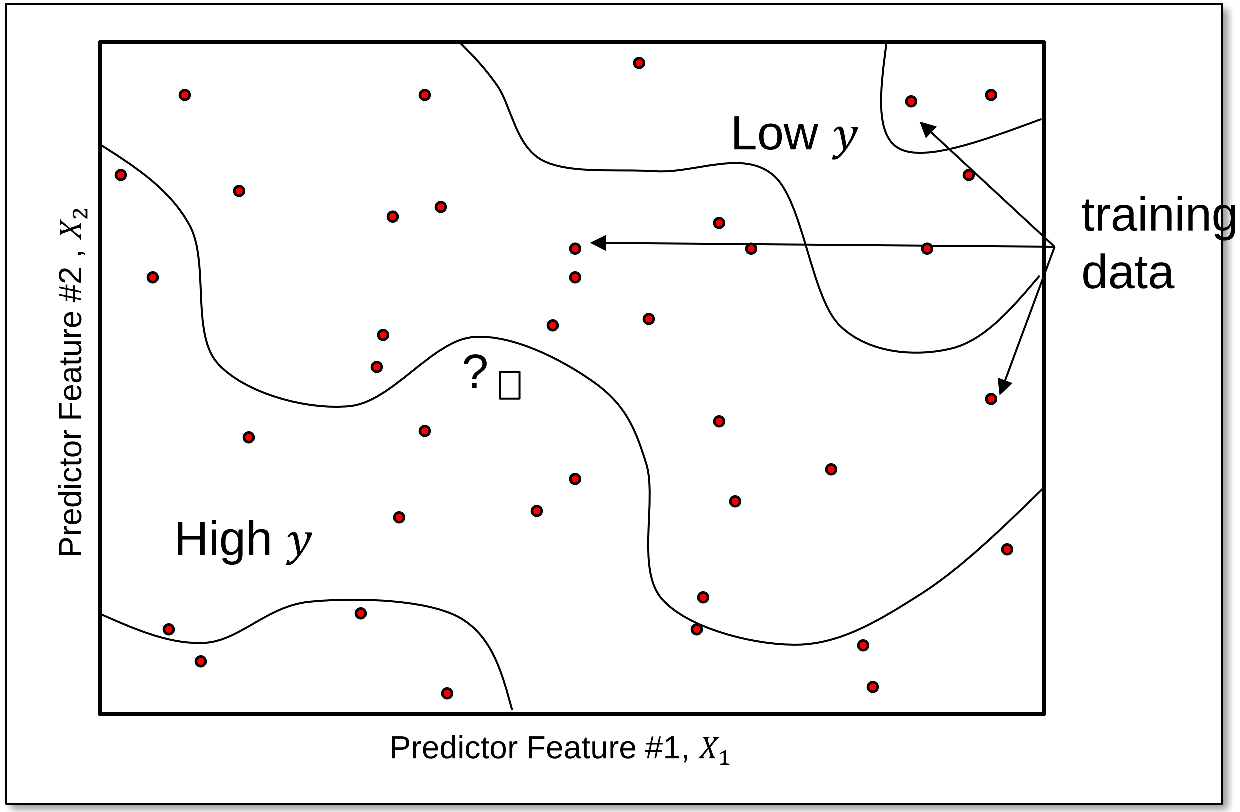


Decision tree regression model to predict porosity from the z coordinate.

- *'Nonparametric is actually parametric rich!'*

Nonparametric Model

Could We Train a More Complicated, Flexible Nonparametric Model?



Training data in the predictor feature space.

Lecture 11: k-nearest Neighbours

Lecture outline:

- **Hyperparameter Training**

Prediction

The best estimate of the response feature

$$\hat{Y} = \hat{f}(X_1, \ldots, X_m) + \epsilon$$

- Estimate the function, $\hat{f}$, for the purpose of predicting $\hat{Y}$
- We are focused on getting the most accurate estimates, $\hat{Y}$, where $\hat{Y}$ is an estimate of Y

Recall, Predictive Statistics

- given an assumption about the population, predict the outcome in the next sample
- e.g., given a fair coin what is the probability of 3 heads and 7 tails?

Assessing Model Accuracy

Method Selection is Important

- No one method performs well on all datasets.
- Based on experience, understanding the data and limitations of the methods

Measuring Quality of Fit

- for regression, the most common measure is the mean square error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left[\left(y_i - \hat{f}(x_1^i, \dots, x_m^i)\right)^2\right]$$

for $i = 1, \dots, n$ training data and for $1, \dots, m$ features.

where we have n observations. The challenge is that that real question we have is how well can we predict outside the training data – testing data.

$$\mathrm{E}\left[\left(y_0 - \hat{f}(x_1^0, \dots, x_m^0)\right)^2\right]$$

over testing data

over a variety of unsampled sets of predictors $x_1^0, \dots, x_p^0$. We want to know how our model performs when we move away from the training set of data!

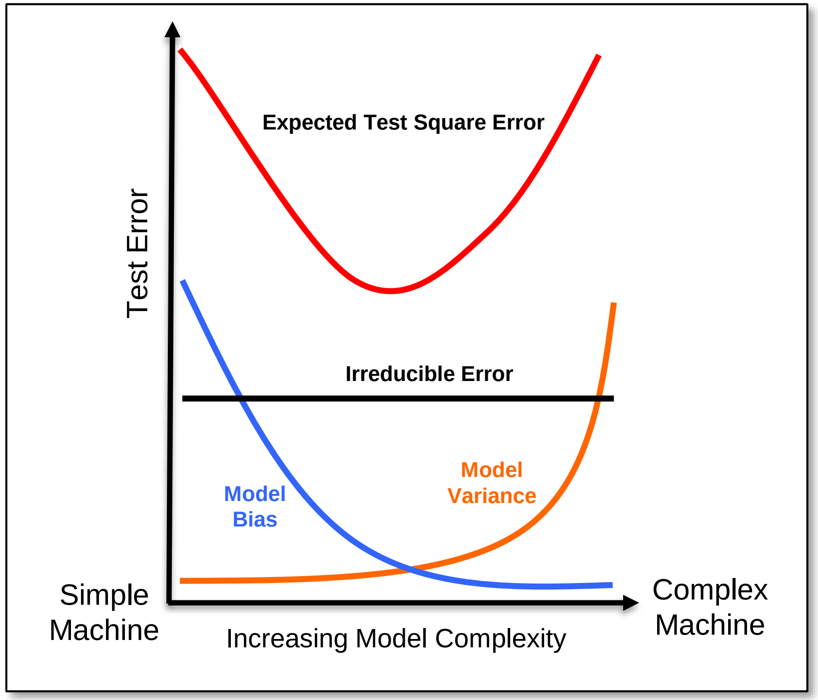
Model Bias Variance Trade-Off

The Error Components for Testing / Real-world Model Predictions

The Expected Test Square Error components:

$$\mathbf{E}\left[\left(y_0-\hat{f}(x_1^0,\ldots,x_m^0)\right)^2\right]=\underbrace{\left(E[\hat{f}(x_1^0,\ldots,x_m^0)]-f(x_1^0,\ldots,x_m^0)\right)^2}_{\text{Model Bias}^2}+\underbrace{E\left[\left(\hat{f}(x_1^0,\ldots,x_m^0)-E[\hat{f}(x_1^0,\ldots,x_m^0)]\right)^2\right]}_{\text{Model Variance}}+\underbrace{\sigma_e^2}_{\text{Irreducible Error}}$$

- **Model Variance** is error due to sensitivity to the dataset
- **Model Bias** is error due to using an approximate model
- **Irreducible Error** is due to missing variables and limited samples



Model variance and bias trade-off.

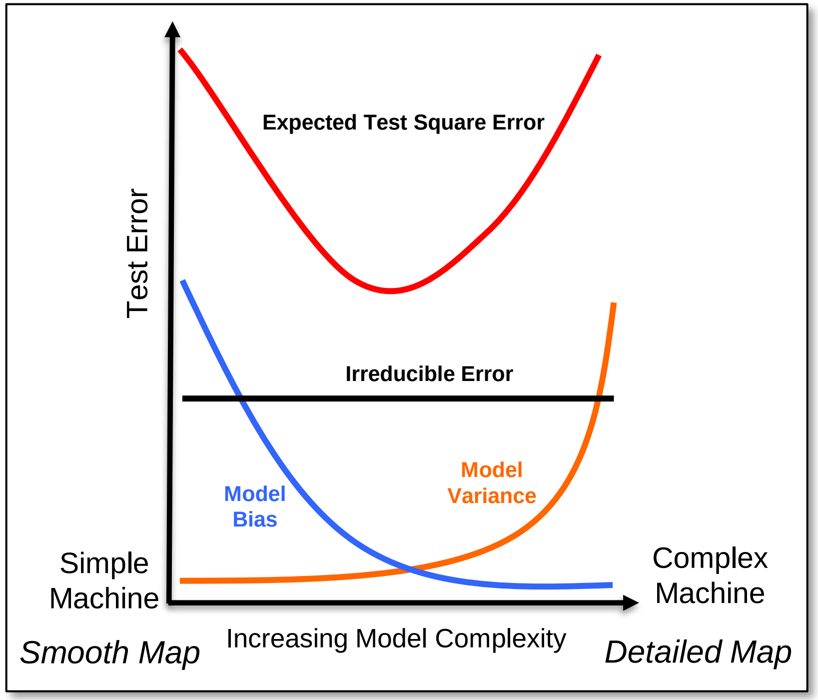
Model Bias Variance Trade-Off

The Error Components for Testing / Real-world Model Predictions

The Expected Test Square Error components:

$$\mathbf{E}\left[\left(y_0 - \hat{f}(x_1^0, \dots, x_m^0)\right)^2\right] = \underbrace{\left(E[\hat{f}(x_1^0, \dots, x_m^0)] - f(x_1^0, \dots, x_m^0)\right)^2}_{\text{Model Bias}^2} + \underbrace{E\left[\left(\hat{f}(x_1^0, \dots, x_m^0) - E[\hat{f}(x_1^0, \dots, x_m^0)]\right)^2\right]}_{\text{Model Variance}} + \underbrace{\sigma_e^2}_{\text{Irreducible Error}}$$

- **Model Variance** is error due to sensitivity to the dataset
- **Model Bias** is error due to using an approximate model
- **Irreducible Error** is due to missing variables and limited samples



Model variance and bias trade-off.

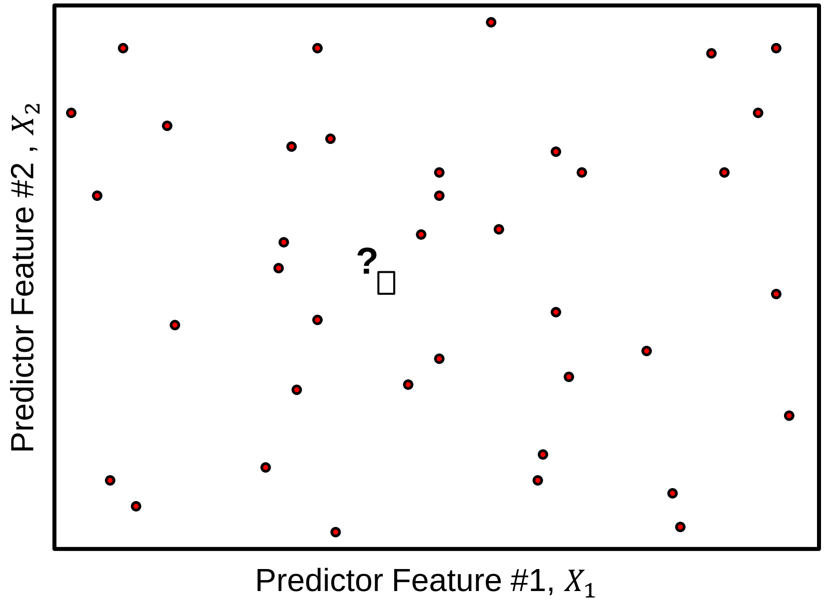
PGE 383 Subsurface Machine Learning

Lecture 11: k-nearest Neighbours

Lecture outline:

- k-Nearest Neighbour

Mapping the Response in the Predictor Feature Space



Possible methods for this interpolation,

- geostatistical, kriging
- inverse distance weighting
- **moving window average / convolution**

This is used for k-nearest neighbour

Convolution

Integral Product of Two Functions, after One is Reversed and Shifted

One interpretation is smoothing a function with weighting function,

- weighting function, $g(\tau)$, and applied to calculate the
- weighted average of function, $f(\tau)$

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\tau)\mathrm{g}(\mathrm{x}-\tau)\mathrm{d}\Delta$$

Note, $g(\tau)$ function is flipped $g(-\tau)$ and shifted, $\mathrm{g}(\mathrm{x}-\tau)$.

- this easily extends into multidimensional

$$(f * g)(x,y,z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\tau_x,\tau_y,\tau_z)\mathrm{g}\left(\mathrm{x}-\tau_x,\mathrm{y}-\tau_y,\mathrm{z}-\tau_z\right)\mathrm{d}\tau_x\mathrm{d}\tau_y\mathrm{d}\tau_z$$

Convolution

Some more details about convolution,

The choice of which function is shifted before integration does not change the result, the convolution operator has commutativity.

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\tau)g(x-\tau)d\tau$$

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-\tau)g(\tau)d\tau$$

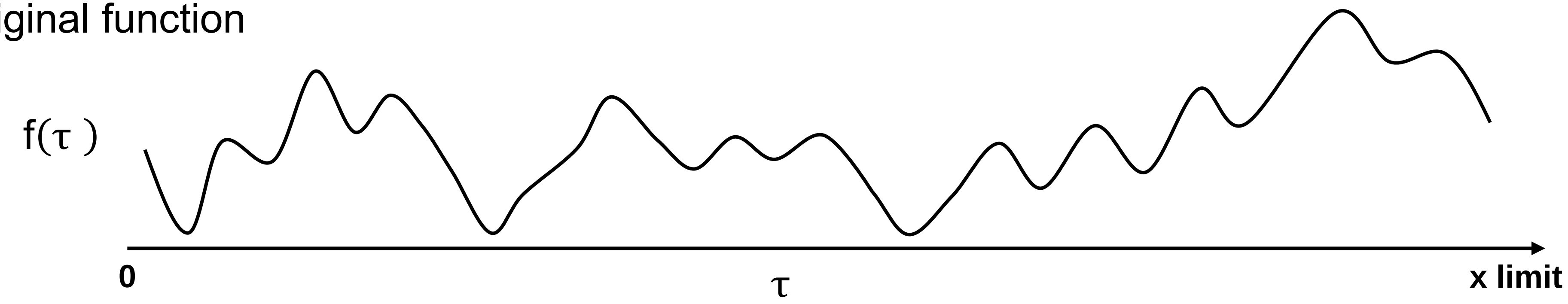
for our model, we will shift g (weighting function) over f data / signal.

- if either function is not reflected then convolution is equivalent to cross-correlation, measure of similarity between 2 signals as a function of displacement.

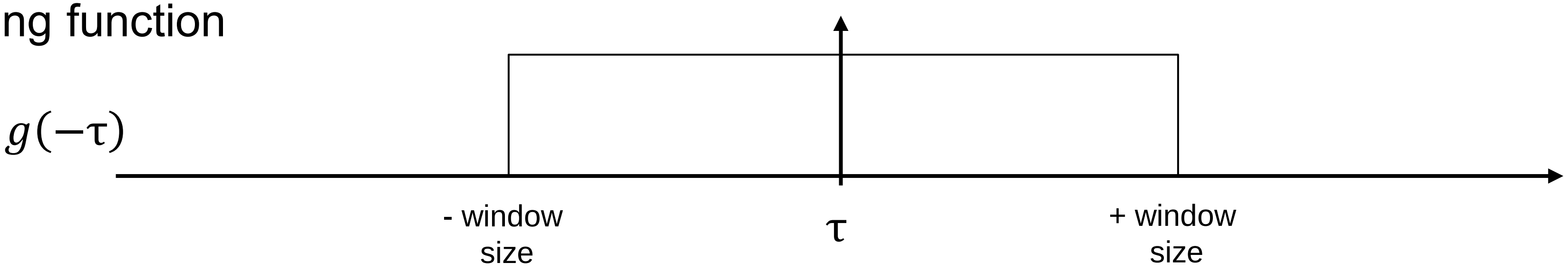
Convolution

Convolution explained graphically

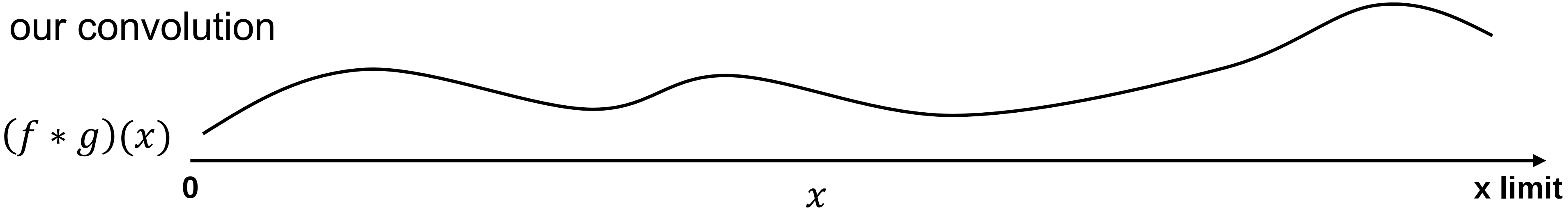
Here's our original function



- Here's our weighting function



- Here's our convolution

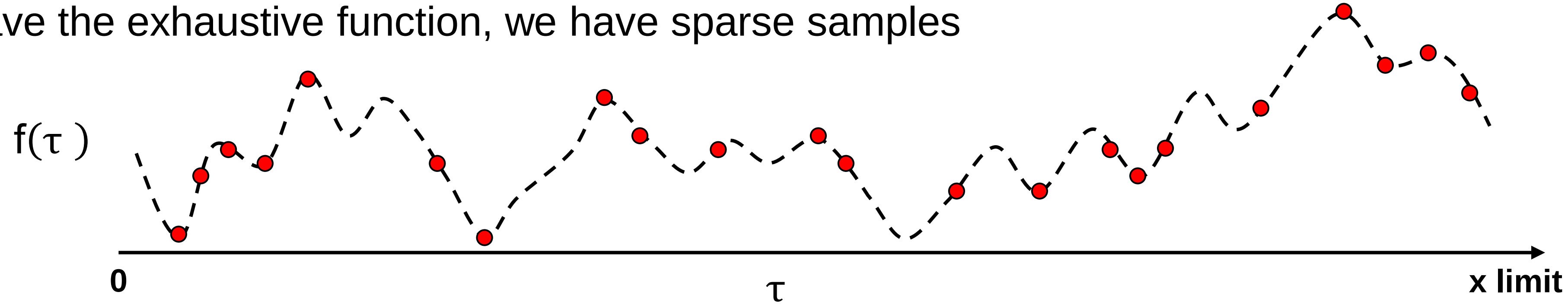


Uniform weighted moving window, window size 3m, 6m and 20m (left to right).

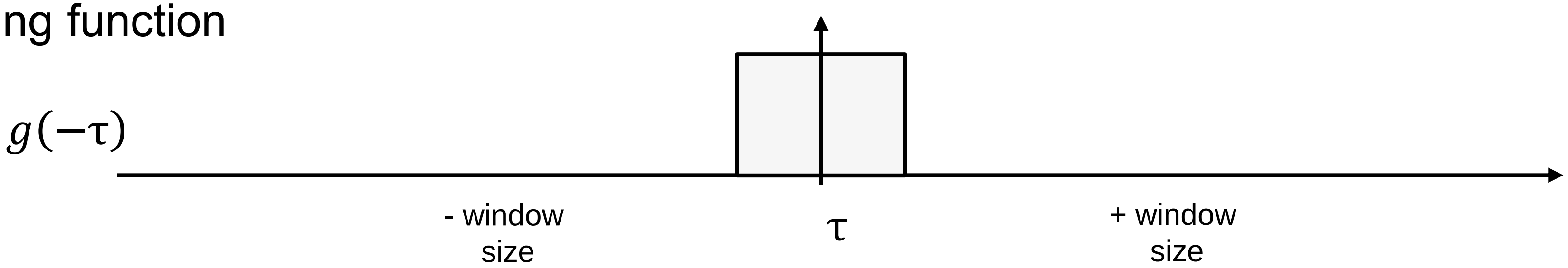
Convolution

Convolution explained graphically

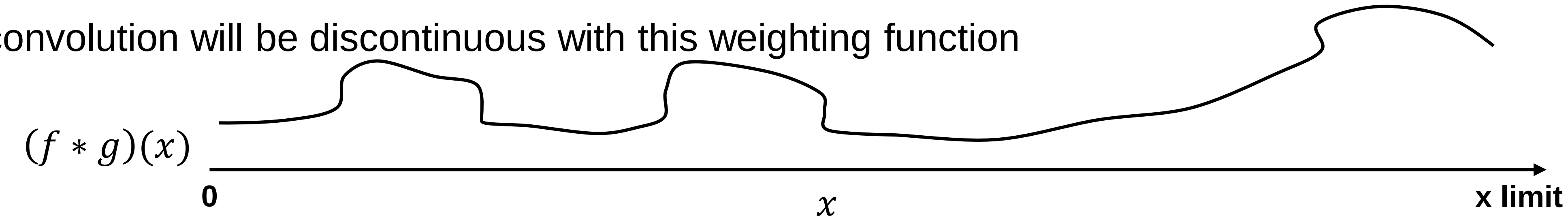
We will not have the exhaustive function, we have sparse samples



- Here's our weighting function



- Our convolution will be discontinuous with this weighting function

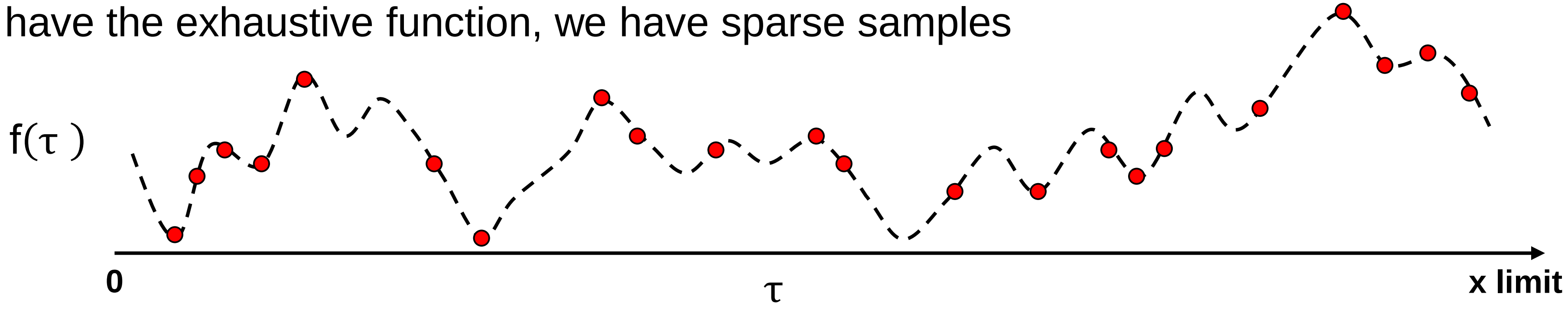


Sparse sampling convolution with uniform weighted moving window.

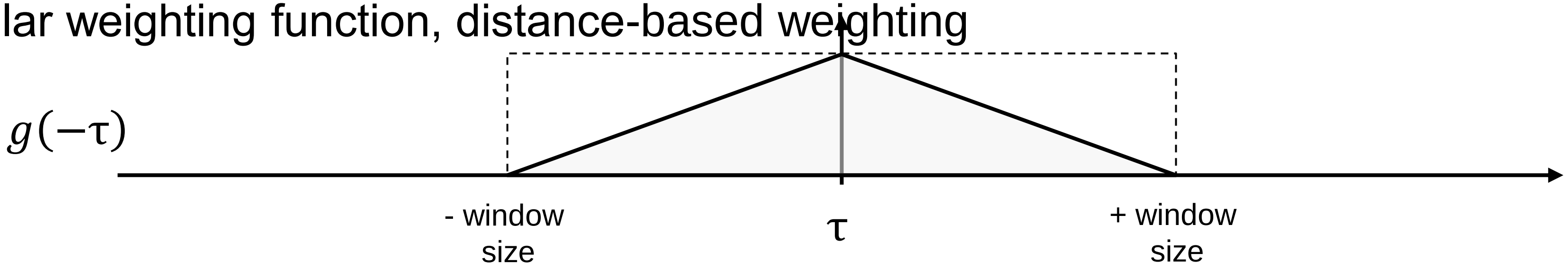
Convolution

Convolution explained graphically

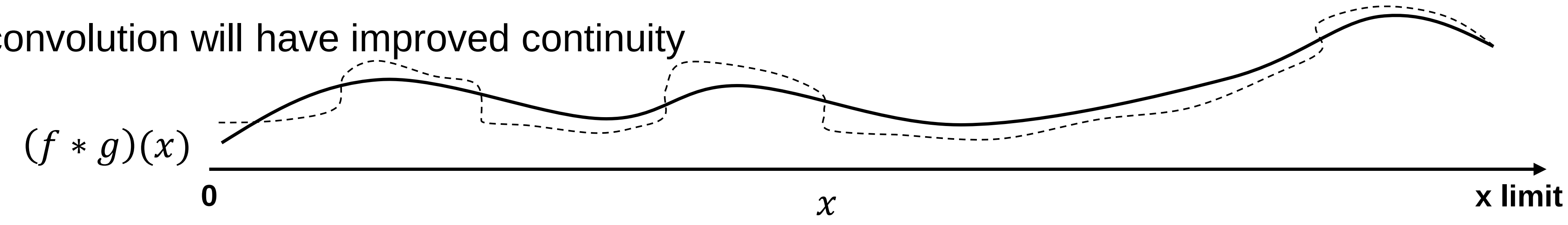
- We will not have the exhaustive function, we have sparse samples



- Here’s our triangular weighting function, distance-based weighting



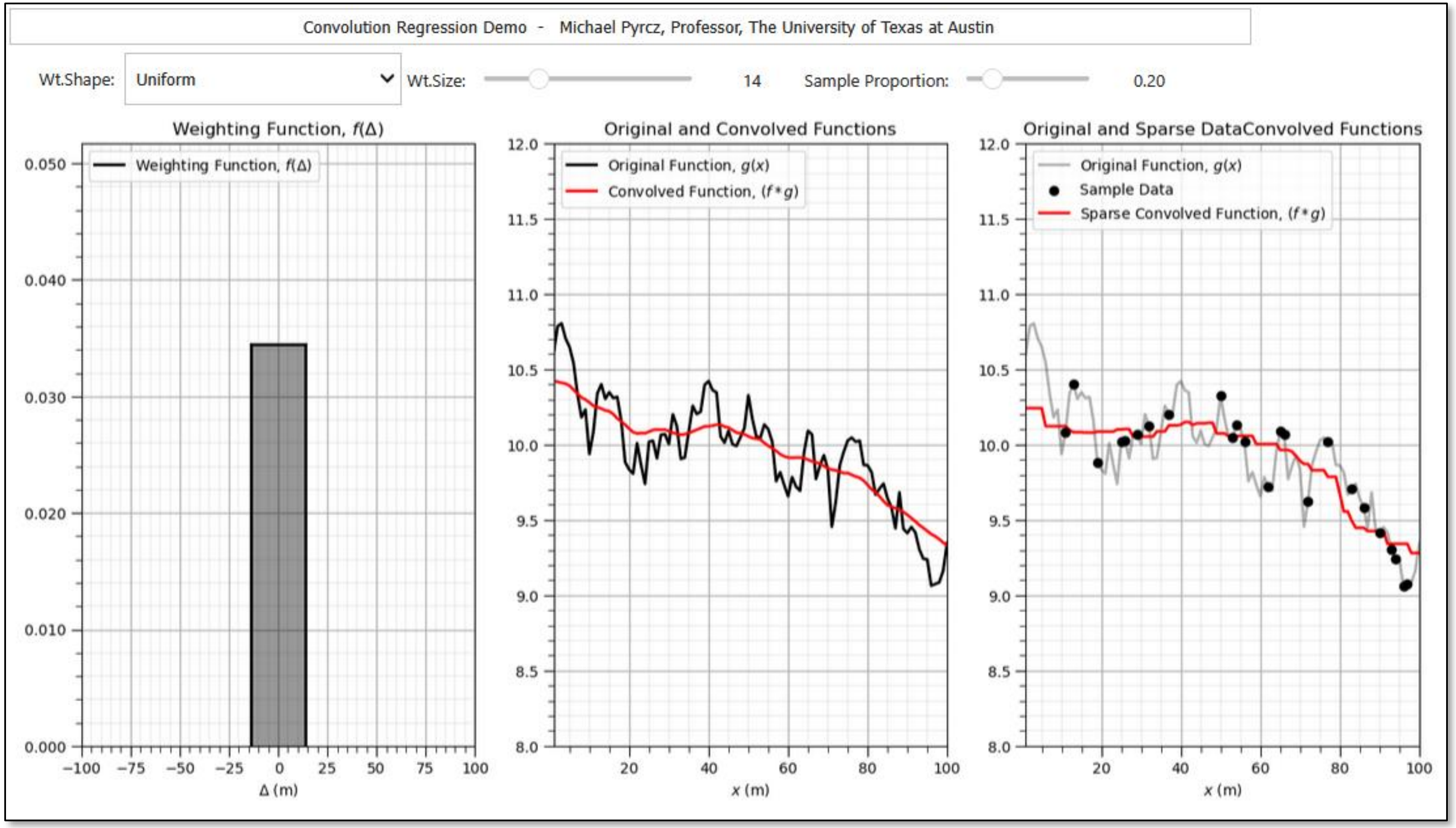
- Our convolution will have improved continuity



Sparse sample convolution with uniform and triangular weighted moving window.

Interactive Demonstration

Here's an interactive dashboard for convolution.



K-nearest neighbour weights (left), exhaustive data example (center), sparsely sampled data example (right).

K-Nearest Neighbours

What are the nearest training samples?

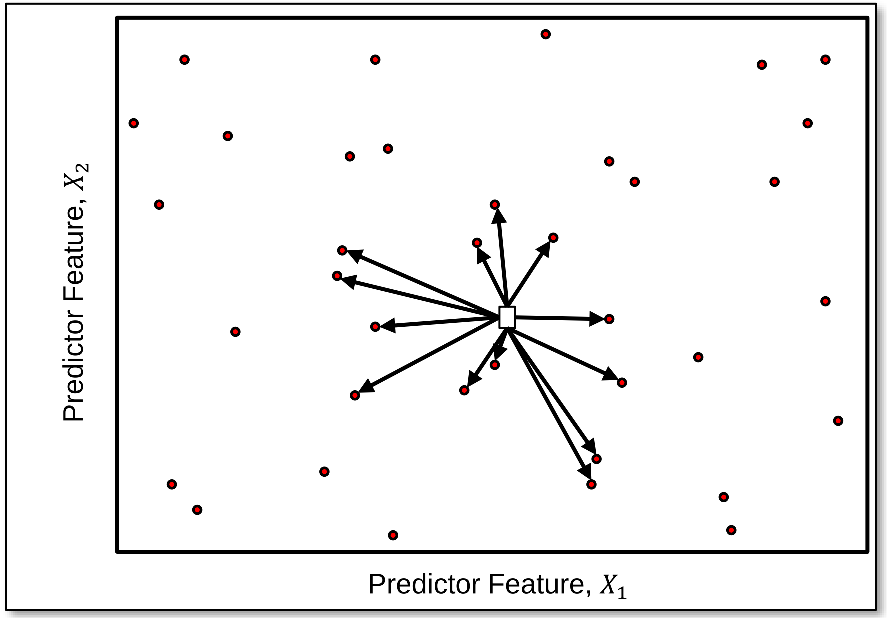
We need to rank samples by proximity in feature space.

$$\Delta_k = d_k = \sqrt{\sum_{\alpha=1}^{m} \left(\Delta X_{k,\alpha}\right)^2}$$

Given the distance separation from each nearest data $k = 1, \ldots, K$ we calculate the associated data weight.

$$w_k = f(\Delta_k)$$

where the weight for the kth nearest neighbor is a retrieved from the weighting function.



Distance metric to find the k nearest neighbours to a predication case.

Then the regression prediction model is a linear weighted average.

$$y_i = \frac{1}{\sum_{k=1}^{K} w_k} \sum_{k=1}^{K} w_k \cdot y_k$$

K-Nearest Neighbours

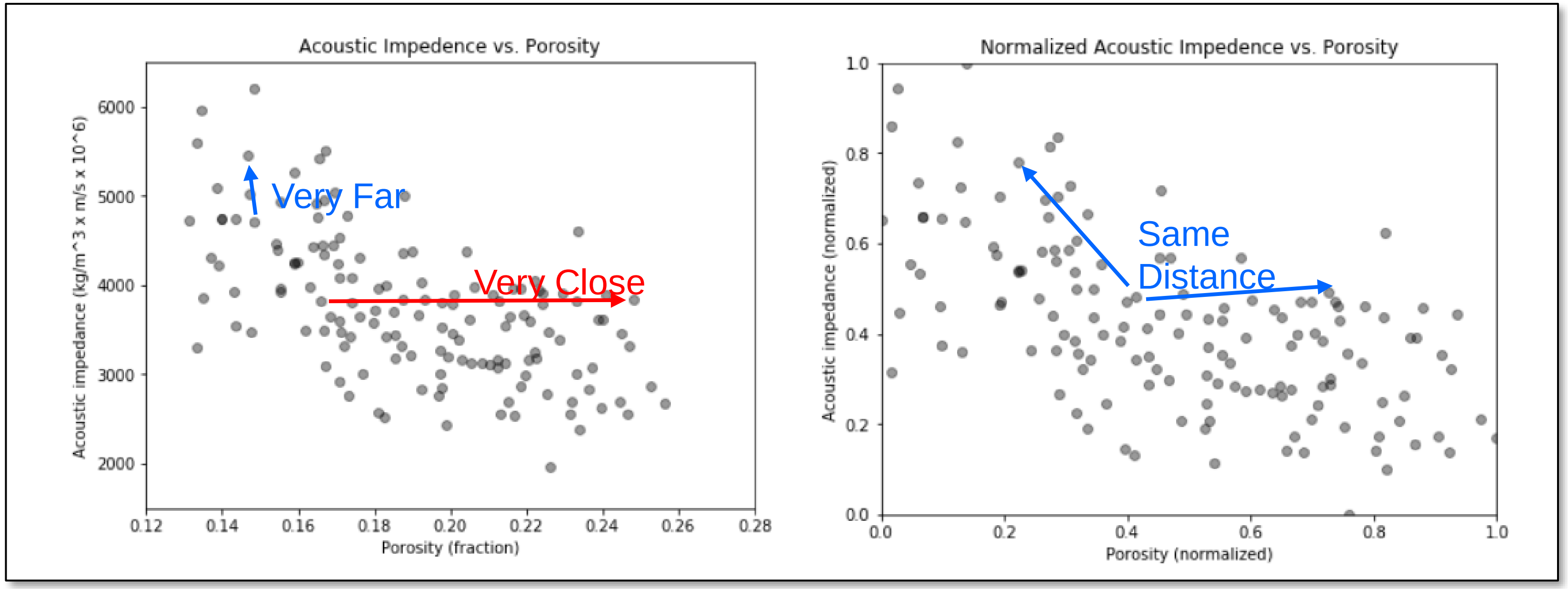
We generally require some form of

- **normalization** – constrain range [0,1]
- **standardization** – constrain the mean and variance

$$x_n = \frac{(x - x_{min})}{(x_{max} - x_{min})}$$

$$x_s = \left(\frac{\sigma_{x_s}}{\sigma_x}\right)(x - \bar{x}) + \bar{x_s}$$

To avoid artifacts, arbitrary feature weighting.

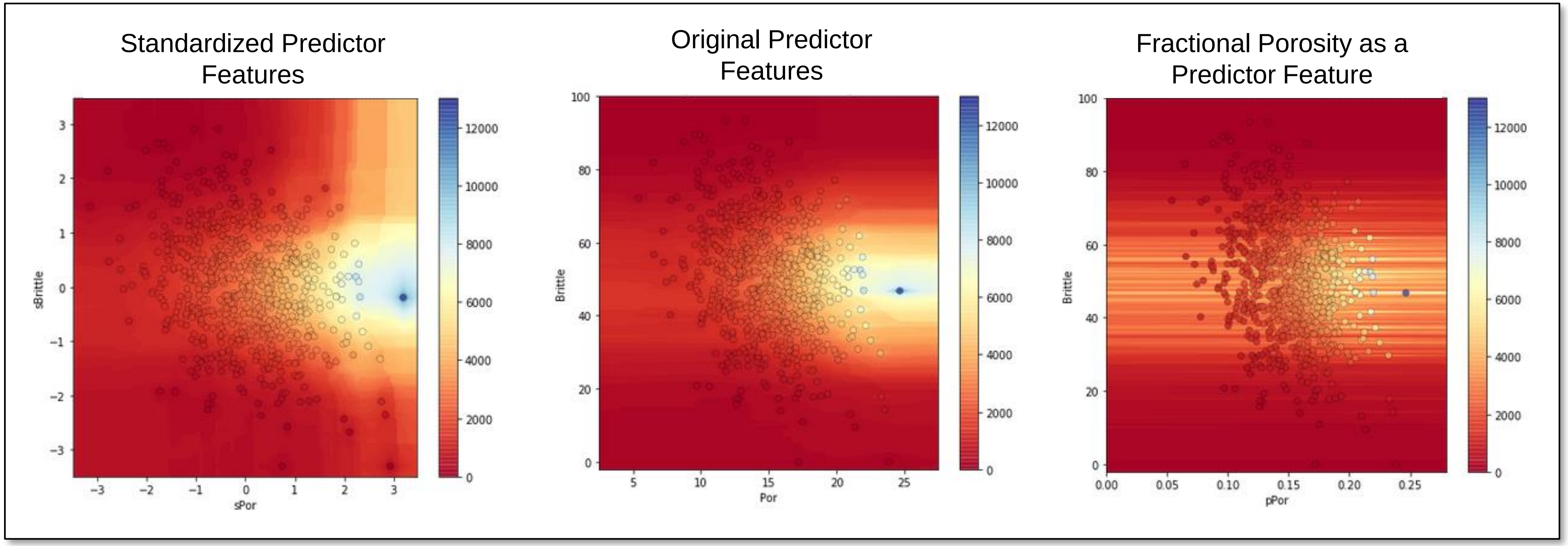


Distance calculation with original (left) and standardized (right) features, from clustering chapter of machine learning e-book.

K-Nearest Neighbours

Why must we standardize or normalize the predictor features?

Here’s three examples of k-nearest neighbour prediction models for production.



k-nearest neighbour prediction of unconventional well production from porosity and brittleness, impact of feature standardization, similar to k-nearest neighbours chapter of machine learning e-book.

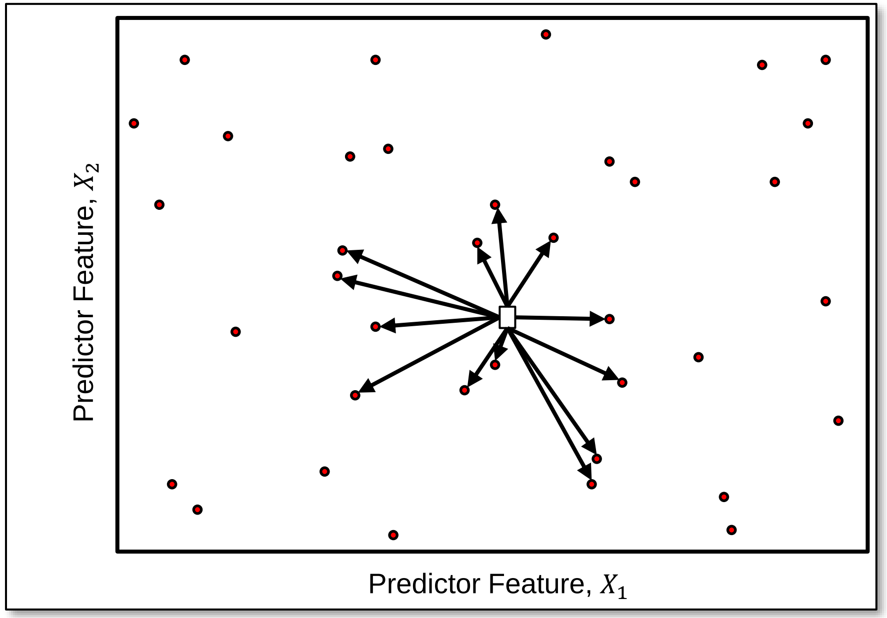
Measures of Distance

Measures of Dissimilarity / Distance, Metrics in Feature Space

Require a distance metric to find the k nearest neighbours and for distance-based weighting.

Euclidean Distance

$$d_{i,i'} = \sqrt{\sum_{j=1}^{m} \left(x_{j,i} - x_{j,i'}\right)^2}$$



Distance metric to find the k nearest neighbours to a predication case.

Manhattan / City Block Distance

$$d_{i,i'} = \sum_{j=1}^{m} \left|x_{j,i} - x_{j,i'}\right|$$

Sum of absolute differences over features.

Minkowski Distance

$$d_{i,i'} = \left(\sum_{j=1}^{m} \left(x_{j,i} - x_{j,i'}\right)^p\right)^{\frac{1}{p}}$$

Generalized form, $p = 1$ Manhattan, $P = 2$ Euclidean.

Mapping the Response in the Predictor Feature Space

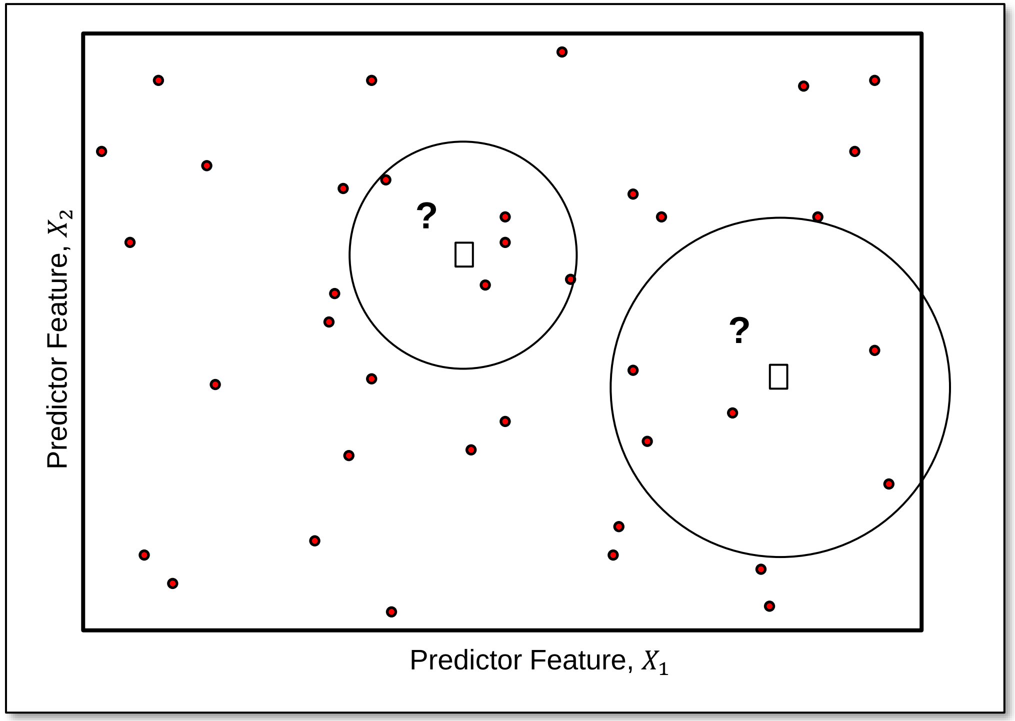
***K*-nearest neighbor regression is not exactly moving window average / convolution**

Hyperparameter, number of nearest data *k*

- size of the window is locally based on how far to go to find k data

Therefore, the window is locally adaptive,

- k-nearest neighbor is a locally adaptive search
- sparse sampled will require a larger window
- larger k results in smoother response prediction → underfit
- smaller k results in more detailed response prediction → overfit



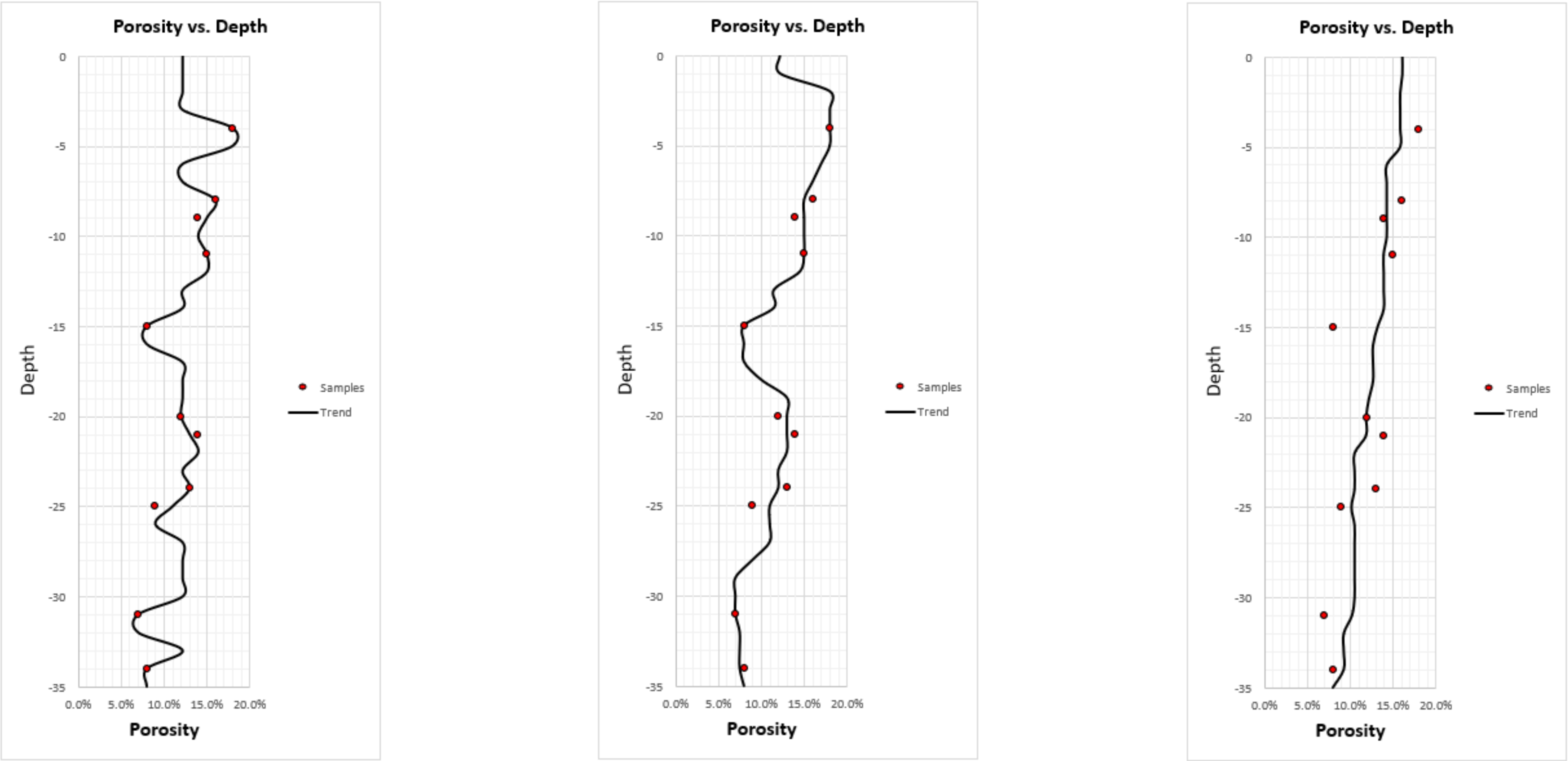
For a given k number of nearest neighbours data are collected from farther away in sparse data regions of the predictor feature space.

Convolution

Convolution for a weighted average (e.g. trend modeling).

Here’s a demonstration with uniform weighting function of variable size.

- Window size is a hyperparameter that controls the level of fit to training!



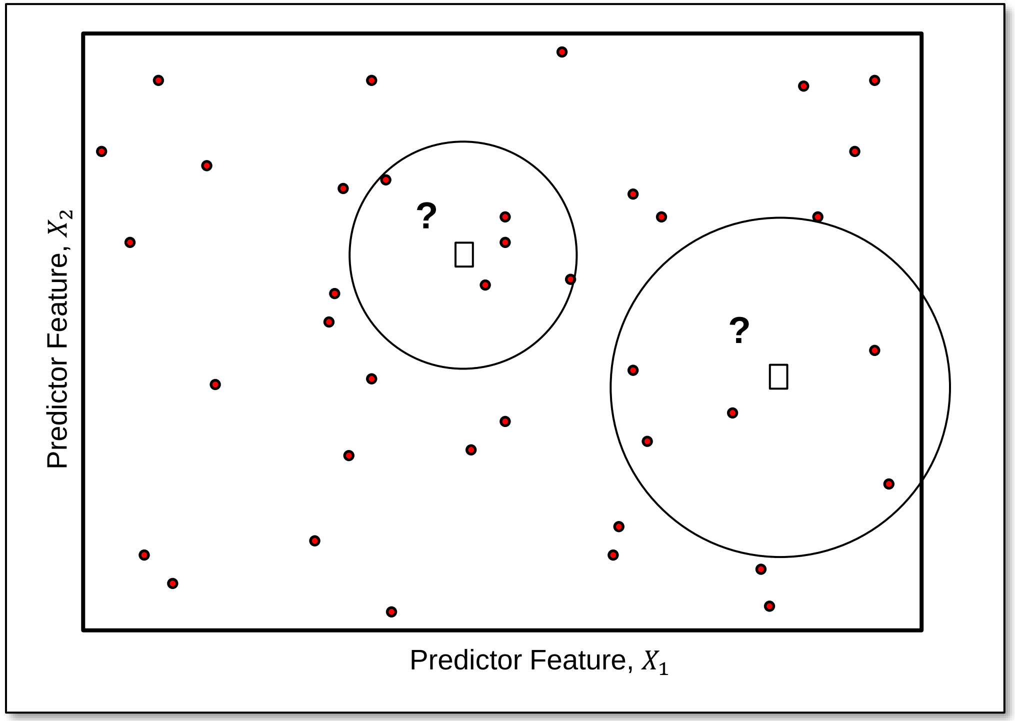
Uniform weighted moving window, window size 3m, 6m and 20m (left to right).

Mapping the Response in the Predictor Feature Space

Another K-nearest Neighbor Regression Hyperparameter

Weighting function form,

- there are generally 2 parametric forms available for the weighting function
- **uniform** – insensitive to distance of training data from estimated location
- **inverse distance weighting** with a power specified



For a given k number of nearest neighbours
data are collected from farther away in sparse data regions of
the predictor feature space.

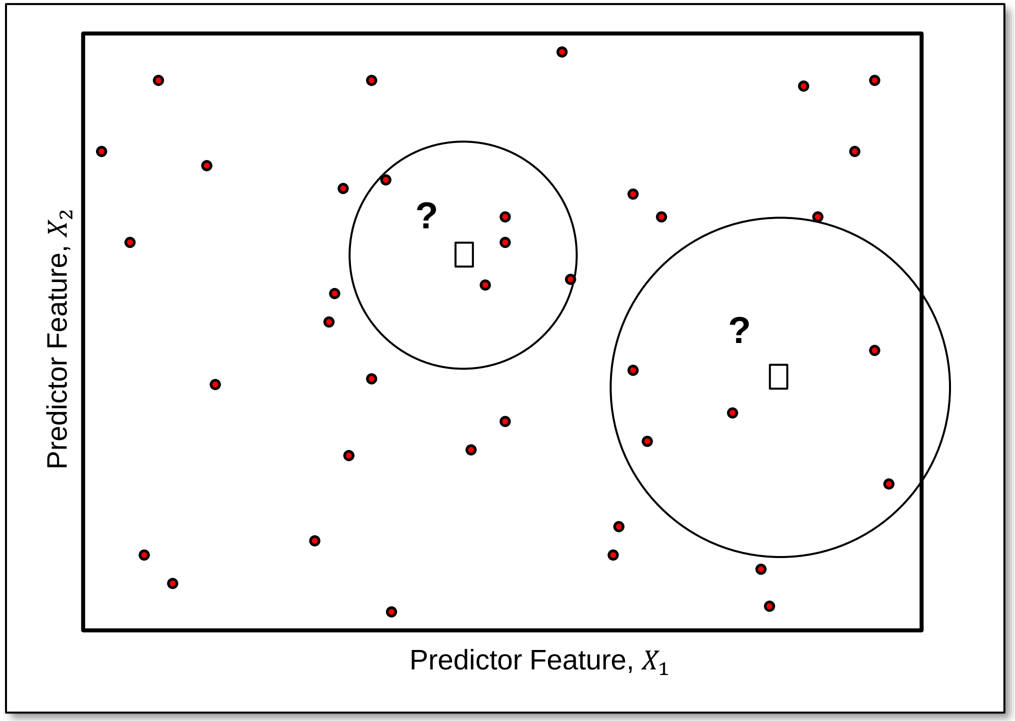
The Model
Eager and Lazy Learning

Lazy Learning

- Model is a generalization of the training data and calculation is delayed until query is made of the model
- The model is the training data and selected hyperparameters

Eager Learning

- Model is a generalization of the training data constructed prior to queries
- The model is input-independent after parameter training and hyperparameter tuning



The k-nearest neighbours model requires the data and model parameters for each prediction.

The Model
Instance-based Learning

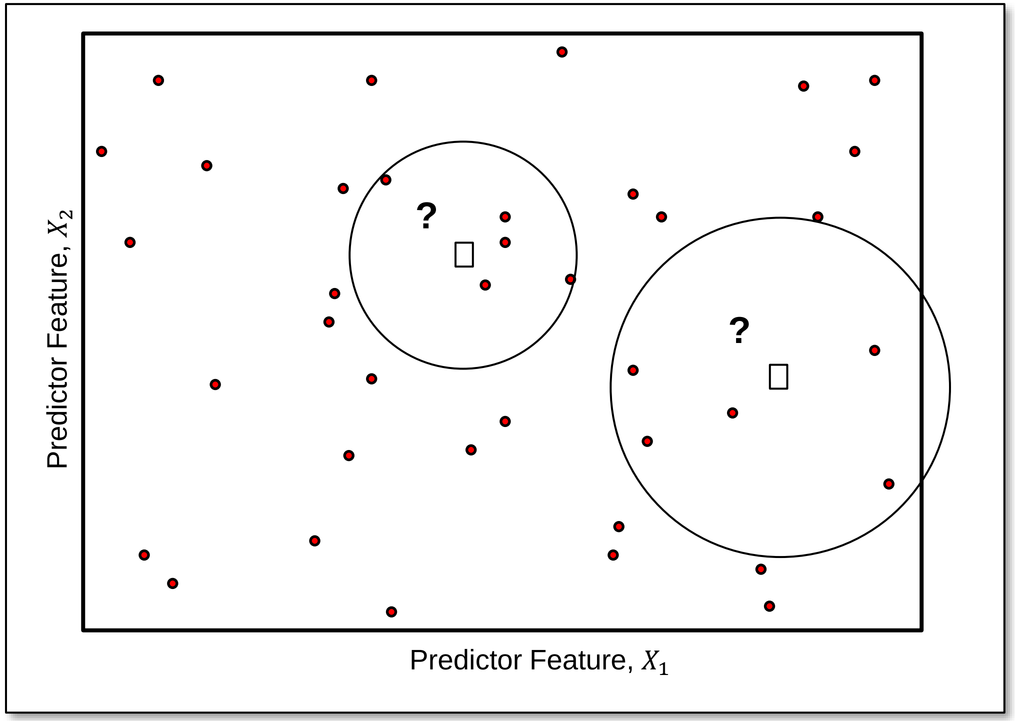
Instance-based Learning

Also known as memory-based

- Compares new prediction problems (as set of predictors, $x_1, \ldots, x_m$) with the cases observed in the training data
- Model requires access to the training data, acting as a library of observations
- Prediction directly from the training data
- Prediction complexity grows with the number of training data, n, number of neighbors, k, and number of features, m.

$$O(n \times k \times m)$$

- a specific case of lazy learning



The k-nearest neighbours model requires the data and model parameters for each prediction.

Recall: Computational Complexity

Computational Resources

- **Time complexity** refers to computational time and the scaling of this time to the size of the problem for a given algorithm
- **Space complexity** refers to computer memory required and the scaling of storage to the size of the problem for a given algorithm.
- We will default to **worst-case complexity**, the worst case for complexity given a specific problem size.
  - Assumes all steps are required, e.g., data is not presorted etc.

Computational complexity for k-nearest neighbours

- calculate distances $O(n)$, sort distances $O(n\log(k))$, since $k \ll n$

k-nearest neighbours - $O(n)$

Nearest Neighbor Classification

Classification:

We sum the weight assigned to each labeled categorical nearest neighbor.

Category 1

$$w_{i,c=1} = \sum_{k=1,c=1}^{K} w_k$$

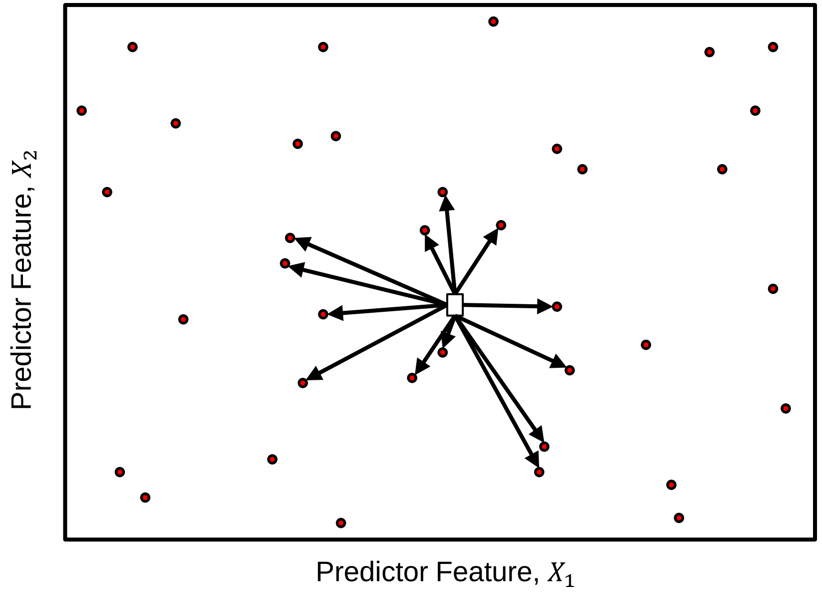
Category 2

$$w_{i,c=1} = \sum_{k=1,c=1}^{K} w_k$$

⋮

Category C

$$w_{i,c=C} = \sum_{k=1,c=C}^{K} w_k$$



Distance metric to find the k nearest neighbours to a predication case.

Then assign category with greatest weight:

$$y_i = C[arg\,max(w_{i,c=1}, \ldots, w_{i,c=C})]$$

'majority rule' is actually plurality vote.

Including Dimensionality Reduction

Dimensionality Reduction

- for high dimensional problems $m \geq 10$, it is common to apply dimensionality reduction
- this includes feature selection and projection through PCA.
- random projection can be used for real-time, big data with k-nearest neighbors since it is a lazy learner!

Recall the Curse of Dimensionality

- time and storage complexity increases
- visualization, multicollinearity, sampling and coverage
- and most importantly for k-nearest neighbors, the distances become imprecise

Recall, Curse of Dimensionality Distorted Space

Distances in High Dimensional Space

Hyperdimensional space is distorted,

- Take the ratio of the volume of an inscribed hypersphere in a hypercube.

$$\frac{\pi^{m/2}}{m 2^{m-1}\Gamma(m/2)} \rightarrow 0 \text{ as } m \rightarrow \infty$$

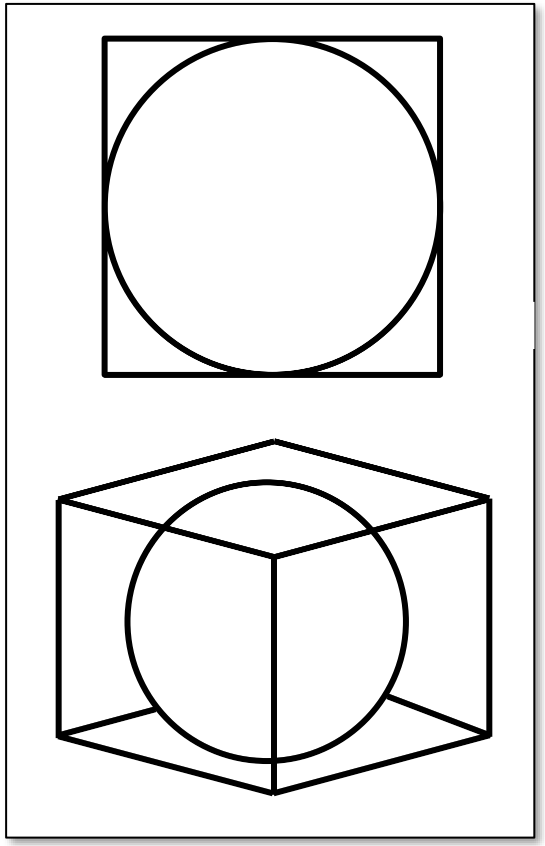
Recall, $\Gamma(n) = (n-1)!$

- High dimensional space is all corners and no 'middle' and most of high dimensional space is far from the middle (all corners!).

Distance in hyperdimensional space loses variance,

$$\lim_{m\rightarrow\infty} E\{dist_{max}(m) - dist_{min}(m)\} \rightarrow 0$$

- The limit of the expectation of the range of pairwise distances over random points in hyperdimensional space tends to zero.
  - Distances are almost all the same, Euclidian distance is no longer meaningful



Ratio of (hyper)sphere inscribed in (hyper)cube.

Lecture 11: k-nearest Neighbours

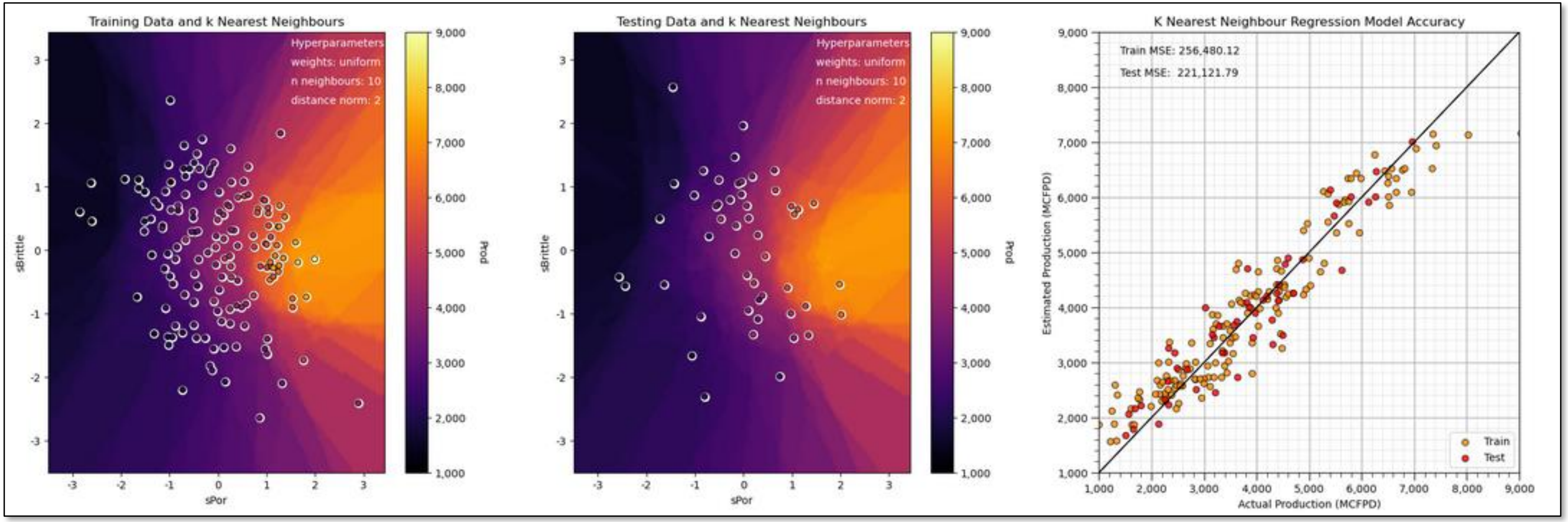
Lecture outline:

- Mapping in the Feature Space
- Hyperparameter Training
- k-Nearest Neighbour
- k-Nearest Neighbour Example
- k-Nearest Neighbour Hands-on

k-Nearest Neighbour Example

Prediction of unconventional production rates (MCFPD) from, $prod = f(porosity, bittleness)$

k = 10, Euclidian distance and uniform weighting

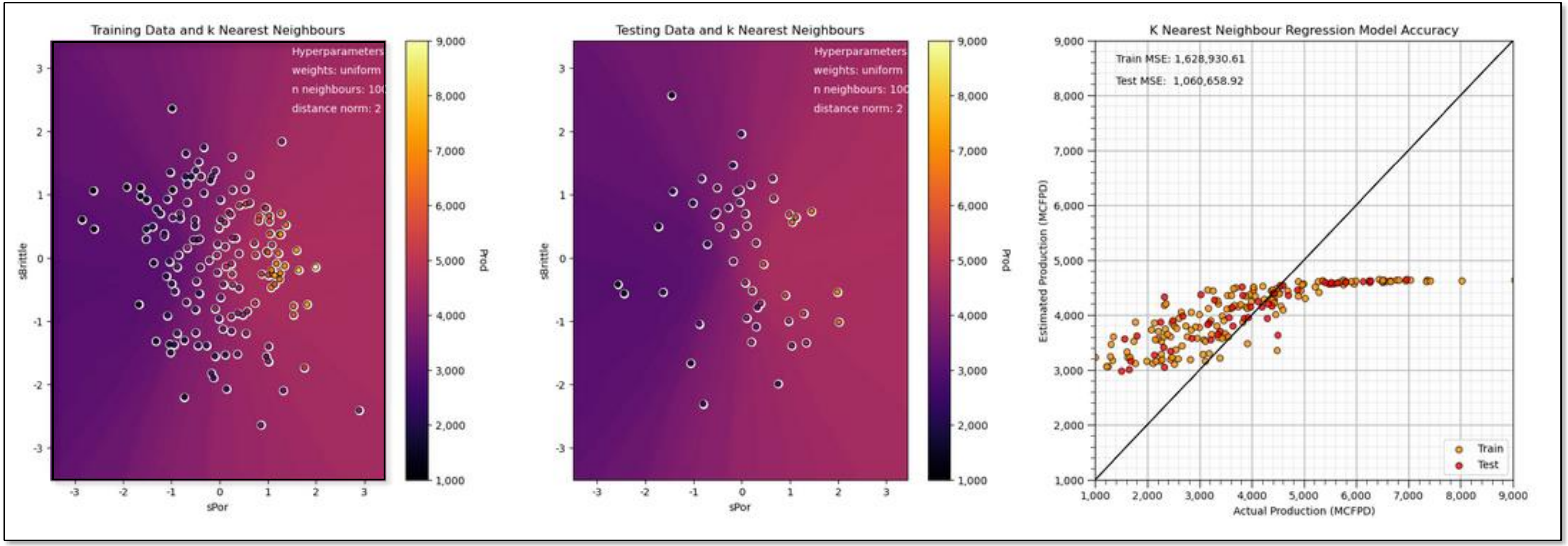


Prediction problem, production rate (MCFPD) from porosity (standardized) and brittleness (standardized), from to k-nearest neighbours chapter of machine learning e-book.

k-Nearest Neighbour Example

Prediction of unconventional production rates (MCFPD) from, $prod = f(porosity, bittleness)$

k = 100, Euclidian distance and uniform weighting

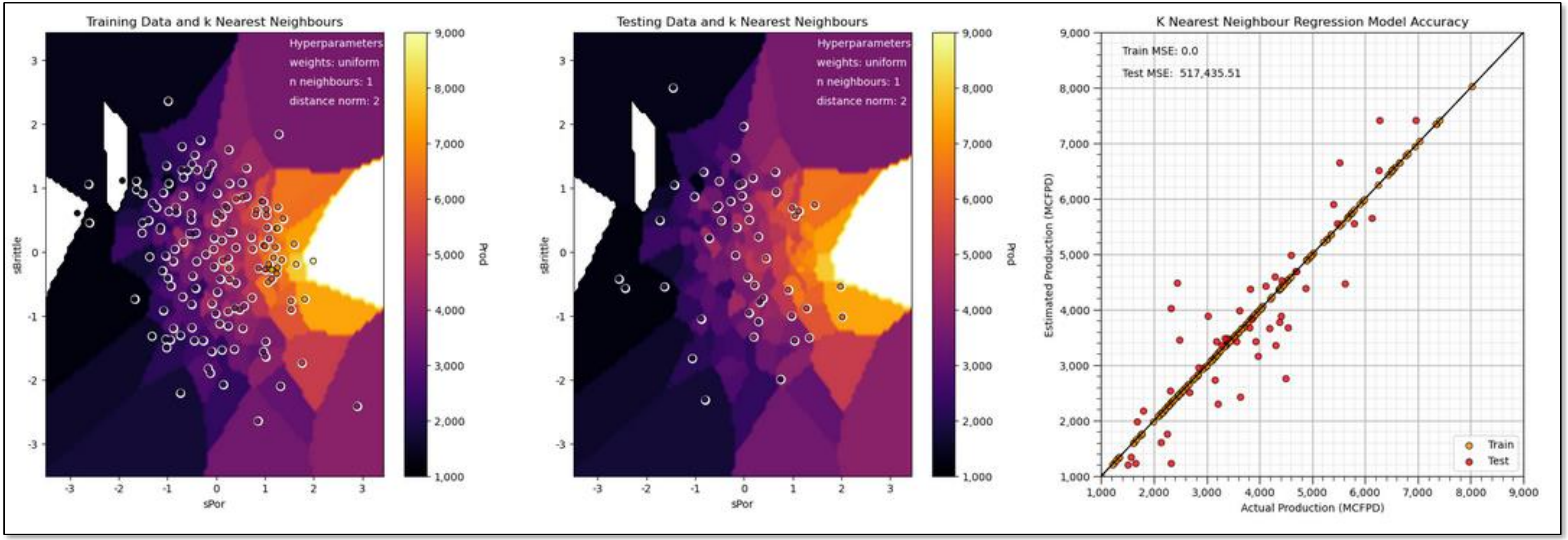


Prediction problem, production rate (MCFPD) from porosity (standardized) and brittleness (standardized), from to k-nearest neighbours chapter of machine learning e-book.

k-Nearest Neighbour Example

Prediction of unconventional production rates (MCFPD) from, $prod = f(porosity, bittleness)$

k = 1, Euclidian distance and uniform weighting

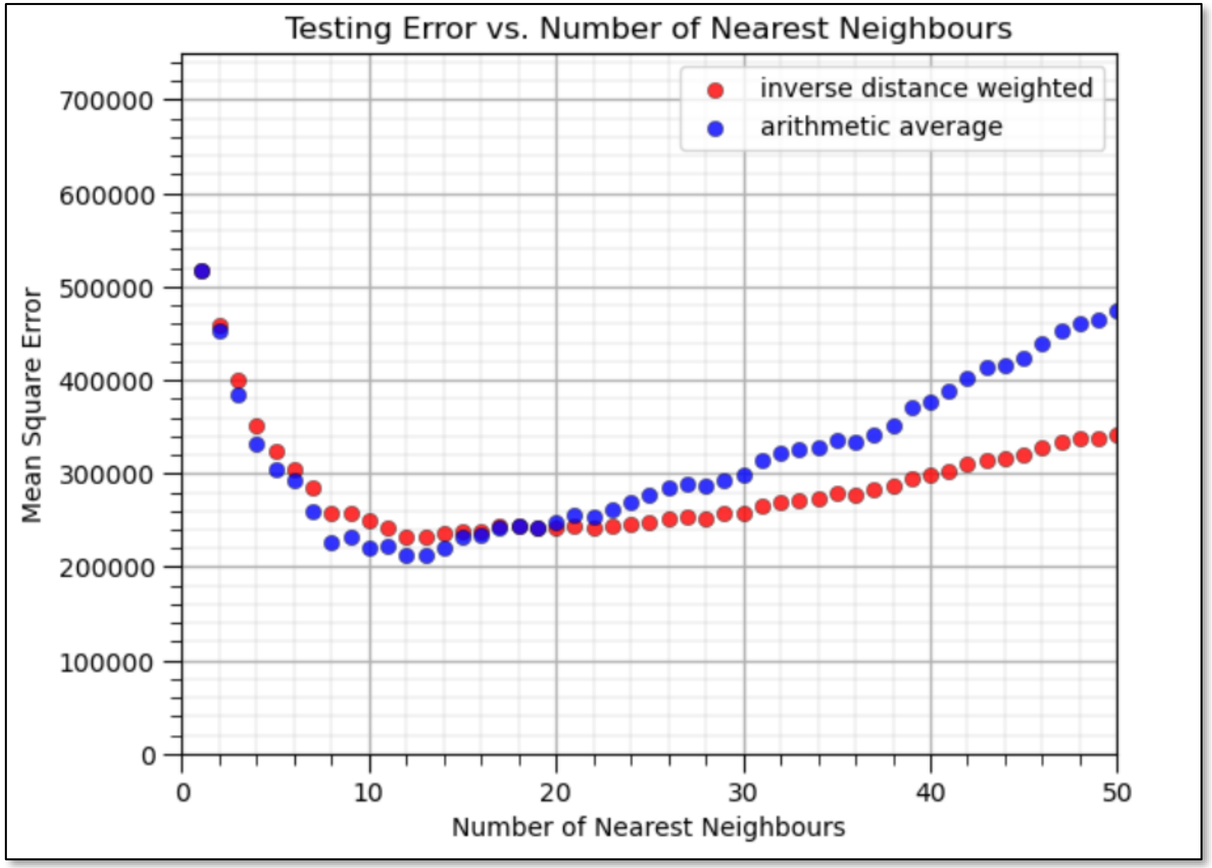


Prediction problem, production rate (MCFPD) from porosity (standardized) and brittleness (standardized), from to k-nearest neighbours chapter of machine learning e-book.

k Nearest Neighbour Example

Prediction of unconventional production rates (MCFPD) from,

- hyperparameter tuning with train and test split cross validation.

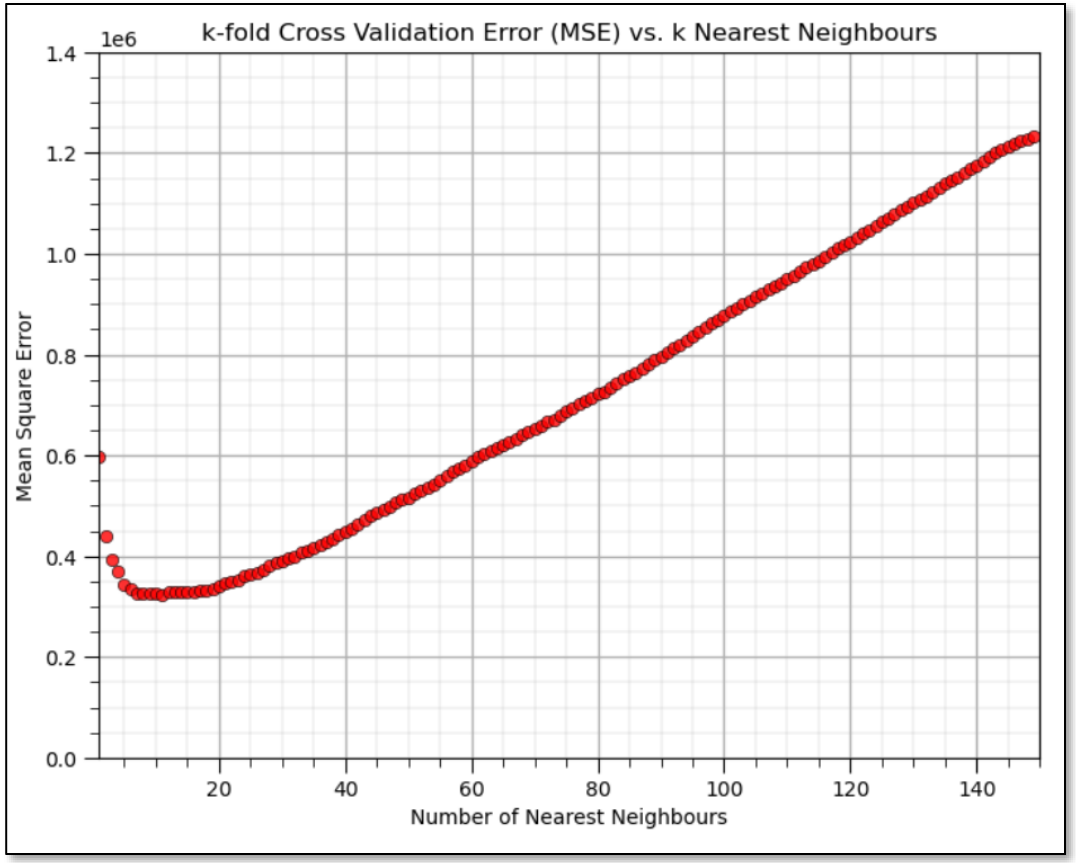


Cross validation hyperparameter tuning.

k Nearest Neighbour Example

Prediction of unconventional production rates (MCFPD) from,

- hyperparameter tuning with k-fold cross validation



k-fold cross validation hyperparameter tuning.

PGE 383 Subsurface Machine Learning

Lecture 11: k-nearest Neighbours

Lecture outline:

- k-Nearest Neighbour Hands-on

k-Nearest Neighbours Demonstration in Python

Demonstration of k-nearest neighbour with a well-documented workflow.

Applied Machine Learning in Python: a Hands-on Guide with Code

k-nearest Neighbours Regression

Michael J. Pyrcz, Professor, The University of Texas at Austin

Twitter | GitHub | Website | GoogleScholar | Book | YouTube | Applied Geostats in Python e-book | LinkedIn

Chapter of e-book “Applied Machine Learning in Python: a Hands-on Guide with Code”.

Cite this e-Book as:

Pyrcz, M.J., 2024, Applied Machine Learning in Python: a Hands-on Guide with Code, https://geostatsguy.github.io/MachineLearningDemos_Book.

The workflows in this book and more are available here:

Cite the MachineLearningDemos GitHub Repository as:

Pyrcz, M.J., 2024, MachineLearningDemos: Python Machine Learning Demonstration Workflows Repository (0.0.1). Zenodo. DOI 10.5281/zenodo.13835318

By Michael J. Pyrcz
© Copyright 2024.

This chapter is a tutorial for / demonstration of **k-nearest Neighbours Regression**.

YouTube Lecture: check out my lectures on:

- Introduction to Machine Learning
- k-nearest Neighbours Regression

k-nearest neighbour regression chapter of Applied Machine Learning in Python e-book.

PGE 383 Subsurface Machine Learning

Lecture 11: k-nearest Neighbours

Lecture outline:

- Mapping in the Feature Space
- Hyperparameter Training
- k-Nearest Neighbour
- k-Nearest Neighbour Example
- k-Nearest Neighbour Hands-on