STAT 542: Statistical Learning

K-Nearest Neighbor and the Bias-Variance Trade-Off

Ruoqing Zhu, Ph.D. <rqzhu@illinois.edu>

Course Website: https://teazrq.github.io/stat542/

January 24, 2022

Department of Statistics
University of Illinois at Urbana-Champaign

K-Nearest Neighbour

Regression Models

- Let's consider a regression model,

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

  where $\mathrm{E}(\epsilon) = 0$ and $\mathrm{Var}(\epsilon) = \sigma^2$.

- Collect a set of i.i.d. training data $\mathcal{D}_n = \{x_i, y_i\}_{i=1}^n$
- From $\mathcal{D}_n$, estimate the regression function as $\widehat{f}$ ("f-hat").
- Predict the value of testing data Y at a target point x_0.

k-Nearest Neighbour

- k-Nearest Neighbour (kNN) is a nonparametric method that predicts a target point x_0 with the average of nearby observations in the training data
- For regression, the prediction at a given target point x_0 is

$$\widehat{y} = \frac{1}{k} \sum_{x_i \in N_k(x_0)} y_i,$$

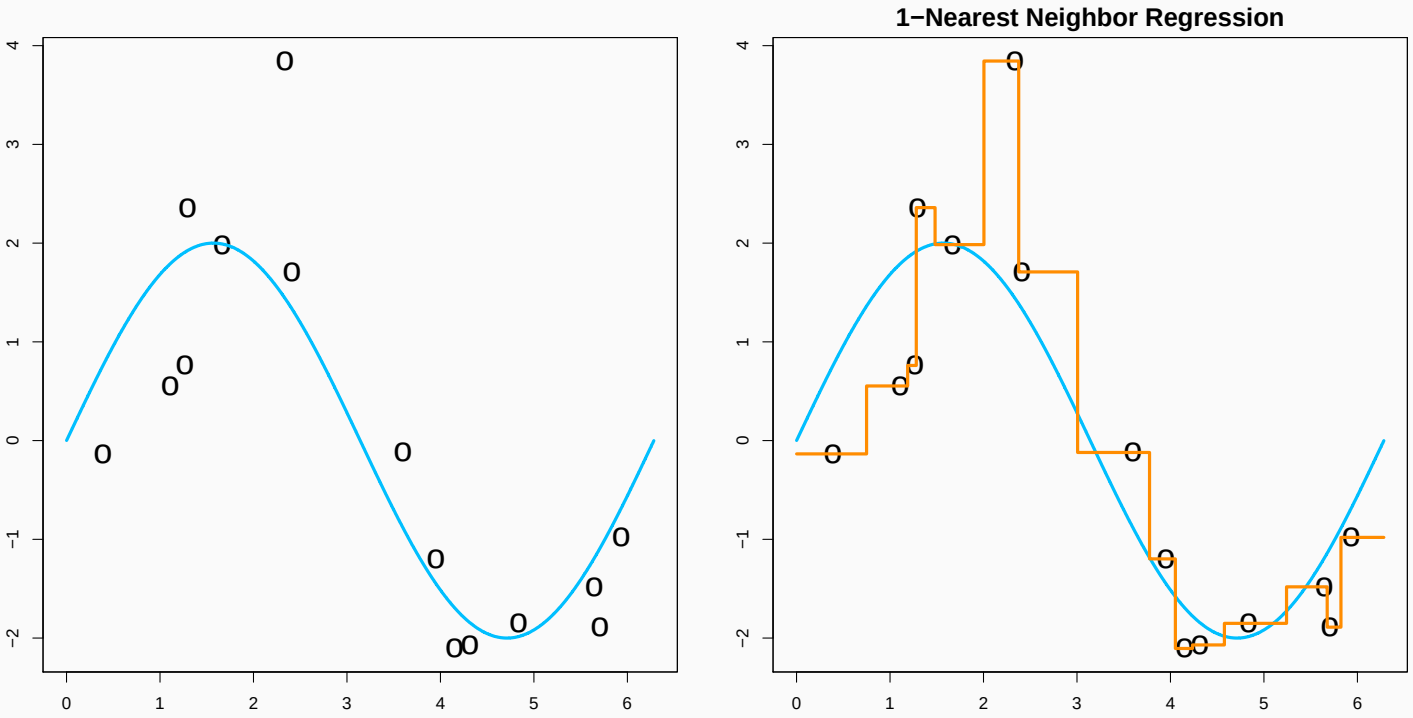
  where $N_k(x)$ defines a set of k samples from the training data (in terms of their feature values) that are closest to x_0.
- Can also be used for classification

Example

Data with only 1 feature from uniform$[0, 2\pi]$. The true model (blue) is

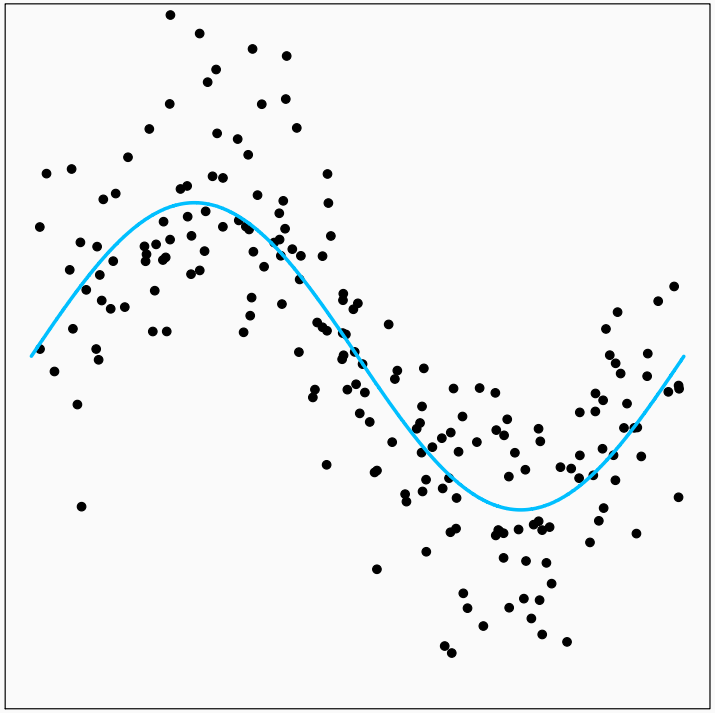
$$Y = 2\sin(X) + \epsilon,$$

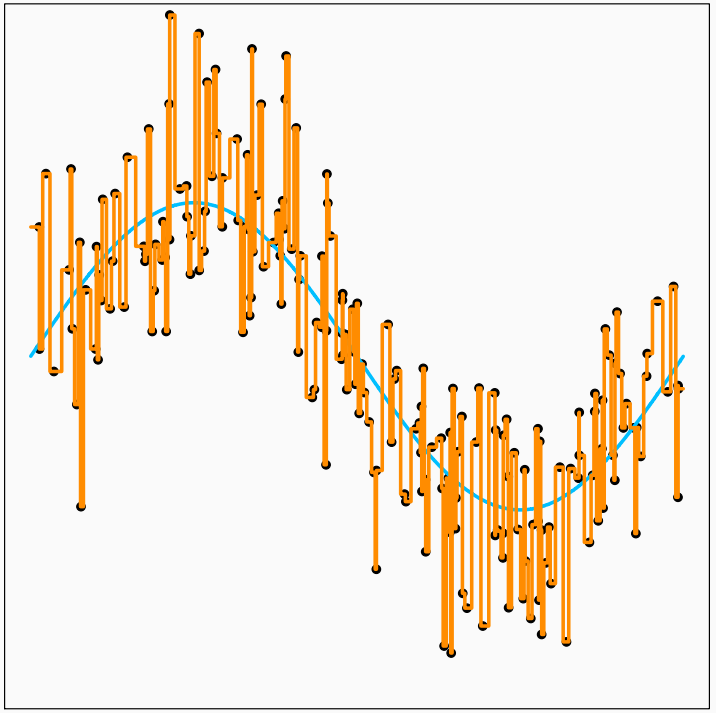
where ϵ is standard normal error. We fit the data with 1NN.



k-Nearest Neighbour in Regression

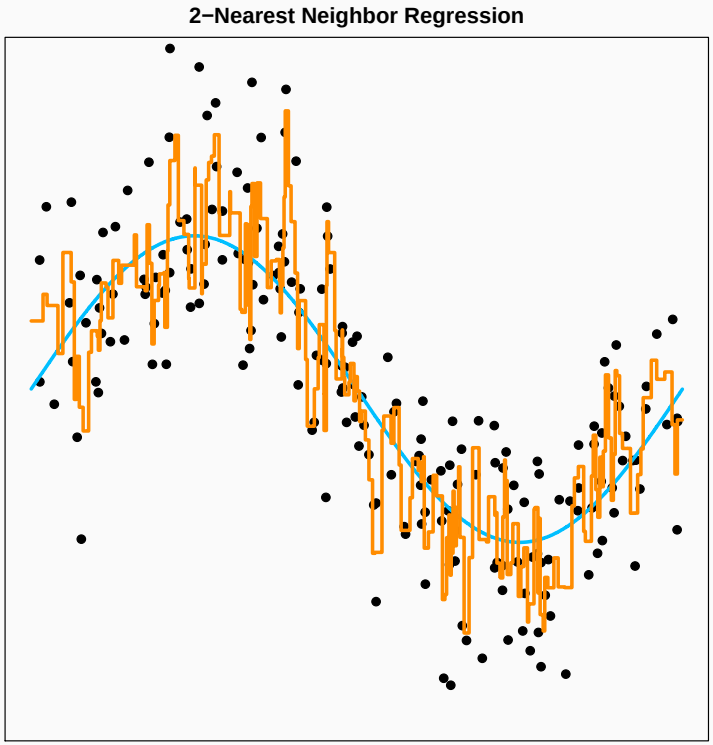
Simulate 200 observations, and see how the model changes over k.

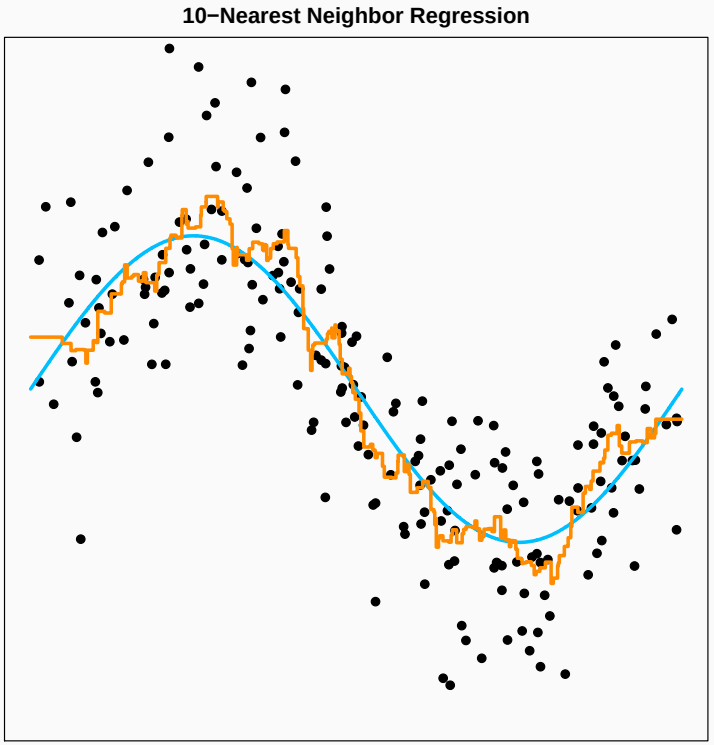




k-Nearest Neighbour in Regression

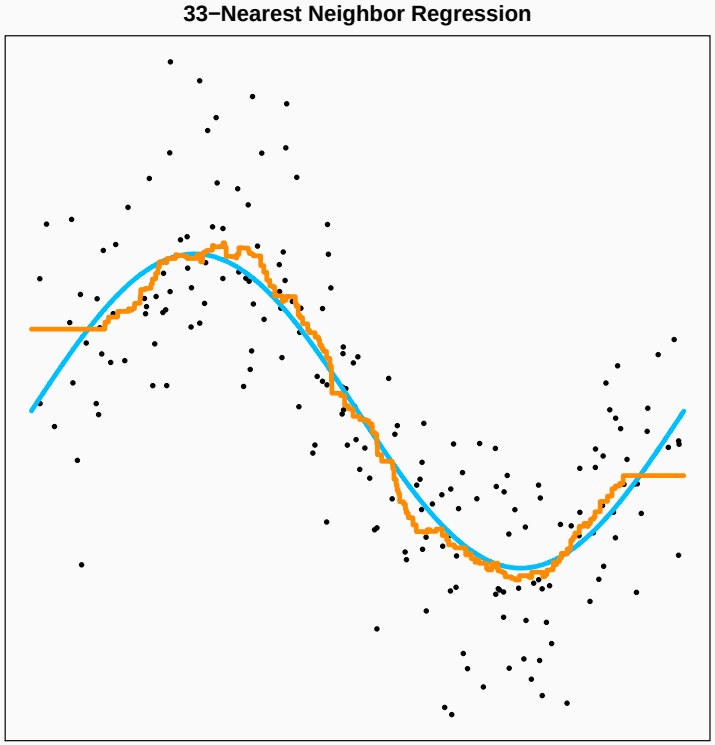
Simulate 200 observations, and see how the model changes over k.

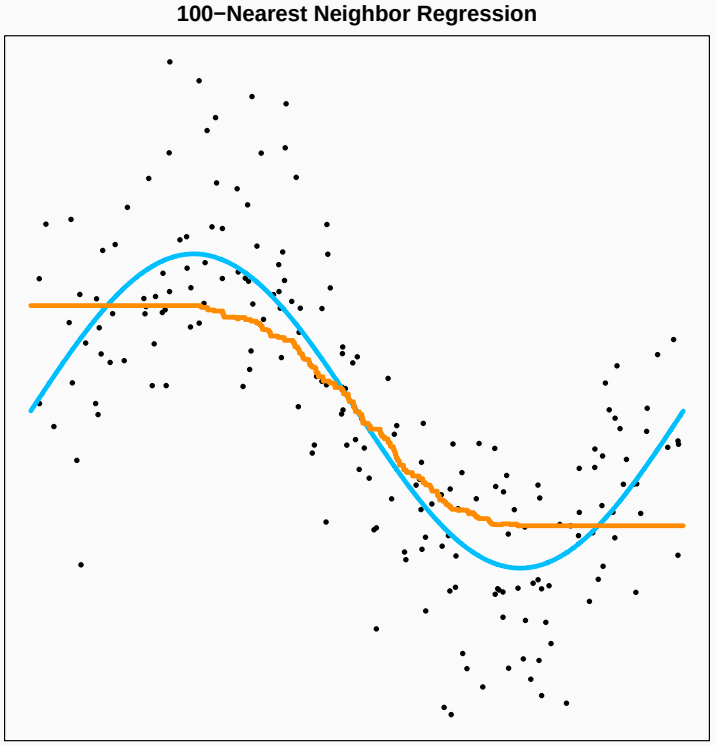




k-Nearest Neighbour in Regression

The model becomes “smoother” as k increases. However, this eventually deviates from the truth if k is too large.





The Bias-variance Trade-off

Variance

- We can see when k is small, the estimated model is unstable.
- Also, each time we observe a new training data, we may get a very different estimation. (due to the closest sample and ϵ)
- The statistical quantity to describe this property is the variance of the estimator $\widehat{f}$.
- For a target point x_0, variance of $\widehat{f}(x_0)$ is

$$\mathsf{Var}(\widehat{f}(x_0)) = \mathsf{E}\left[(\widehat{f}(x_0) - \mathsf{E}\widehat{f}(x_0))^2\right]$$

Bias

- When k is large, the estimated model eventually deviates (systematically) from the truth.
- The statistical quantity to describe this property is the bias of the estimator $\widehat{f}$.
- For a target point x_0, bias of $\widehat{f}(x_0)$ is

$$\text{Bias}\big(\widehat{f}(x_0)\big) = f(x_0) - \text{E}\widehat{f}(x_0)$$

An accurate prediction

- Using squared-error loss, the prediction error is

$$
\begin{aligned}
&\mathrm{Err}(x_0) \\
&= \mathsf{E}_{\mathcal{D}_n, Y_0}\left[\left(Y_0 - \widehat{f}(x_0)\right)^2\right] \\
&= \mathsf{E}_{\mathcal{D}_n, Y_0}\left[\left(Y_0 - f(x_0) + f(x_0) - \mathsf{E}_{\mathcal{D}_n}\widehat{f}(x_0) + \mathsf{E}_{\mathcal{D}_n}\widehat{f}(x_0) - \widehat{f}(x_0)\right)^2\right] \\
&= \ldots \\
&= \underbrace{\mathsf{E}_{Y_0}\left[\left(Y_0 - f(x_0)\right)^2\right]}_{\text{Irreducible Error}} + \underbrace{\left(f(x_0) - \mathsf{E}_{\mathcal{D}_n}\widehat{f}(x_0)\right)^2}_{\text{Bias}^2} + \underbrace{\mathsf{E}_{\mathcal{D}_n}\left[\left(\widehat{f}(x_0) - \mathsf{E}_{\mathcal{D}_n}\widehat{f}(x_0)\right)^2\right]}_{\text{Variance}}
\end{aligned}
$$

- All the cross terms are zero

An accurate prediction

- The prediction error is a sum of three terms
- $\mathrm{E}[(Y - f(x))^2] = \sigma^2$ is the irreducible error term that cannot be avoided, because we cannot predict ϵ
- Of course, we want to minimize both Bias^2 and the Variance, however, this is not always possible...

An accurate prediction

- The first instinct is that one should minimize the bias^2 term $\mathsf{E}[(f(x_0) - \mathsf{E}\widehat{f}(x))^2]$
- Why producing a model that is asymptotically incorrect, i.e., $f(x_0) \neq \mathsf{E}\widehat{f}(x)$?
- This instinct is wrong! — This is usually at the expense of high variance, which eventually damages the prediction performance...
- We already see this with 1NN

An accurate prediction

- 1NN has small Bias^2, as the closet neighbor converges to the target point x_0 as $n \to \infty$
- However, the variance is large because the estimator only uses one observation

$$\mathsf{E}\left[(\widehat{f}(x_0) - \mathsf{E}\widehat{f}(x_0))^2\right] = \mathsf{E}\epsilon^2 = \sigma^2.$$

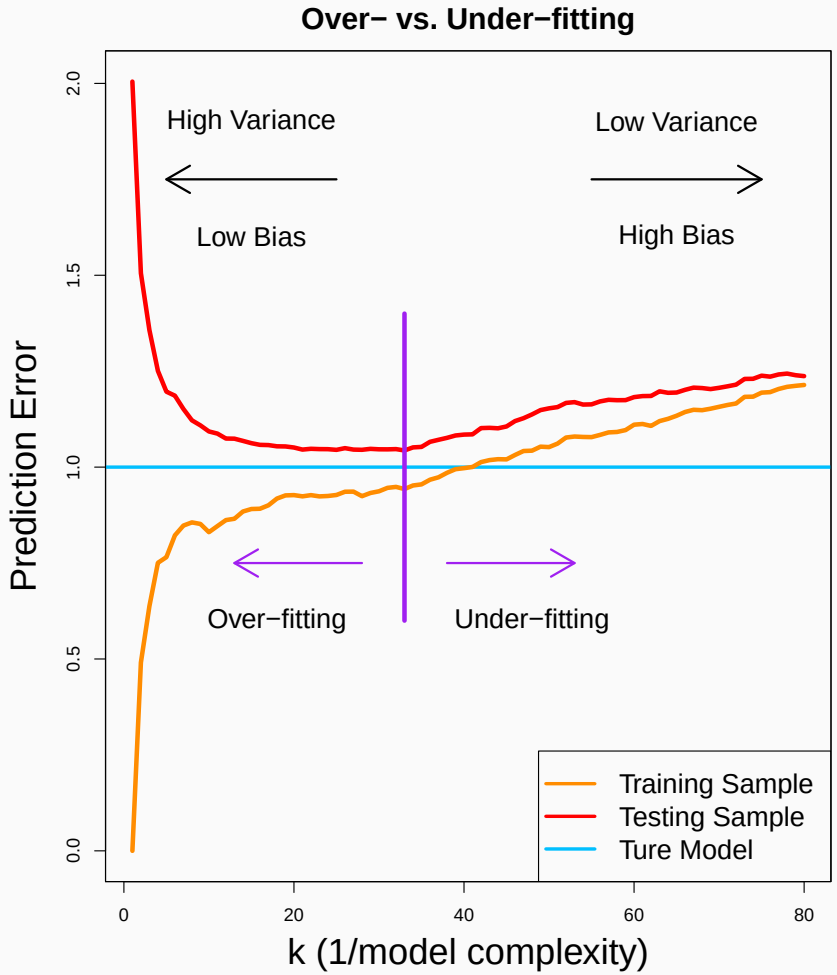
- If we use more "neighbouring" points, say k, the variance would reduce to approximately σ^2/k. But the bias^2 will increase as neighbours are far away from x_0.

Model Complexity

- A related concept is the model complexity
- In linear regression, this is simply the degrees of freedom
- For kNN, k determines the model complexity
- e.g., when $k = N$, this is just the sample mean, and the model is very simple

Model Complexity, over- and under-fitting

- Model complexity $\uparrow$ (small k) $\longrightarrow$ Bias^2 $\downarrow$ and Variance $\uparrow$
- Model complexity $\downarrow$ (large k) $\longrightarrow$ Bias^2 $\uparrow$ and Variance $\downarrow$

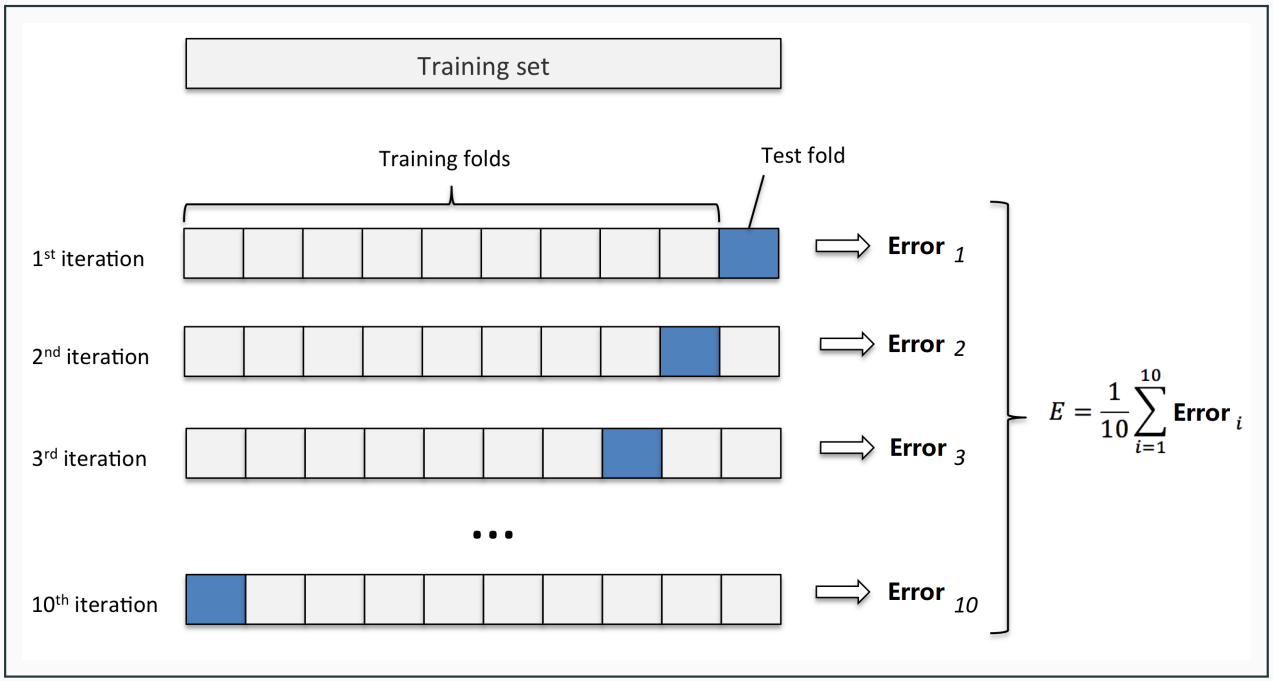


Prevent over-fitting

- As we can see, model complexity, bias-variance trade-off and over- and under-fitting are usually related concepts
- Over-fitting happens when a model performs well on the training data, but not on the testing data
- How to choose k to prevent over-fitting?

k-fold Cross-validation

- Randomly split the data into 10 portions
- Fit the model using 9 portions as training data
- Calculate the testing error using the remaining portion
- Alternate the testing set and average all testing errors



k-fold Cross-validation

- Cross-validation has many variations
- A k-fold cross-validation is random and the result can be affected by the randomness
- We could repeatedly run k-fold cross-validation several times
- Leave-one-out cross-validation is deterministic, but takes longer to run
- Repeated random sub-sampling is another choice

Model Complexity

Degrees-of-freedom and Model Complexity

- Degrees-of-freedom and model complexity are related concepts, and they can be used to prevent over-fitting
- For kNN, the tuning parameter k directly controls both
- For some other models, a penalized framework is used to control complexity

$$\arg\min_f \ \text{loss}(f) + \lambda \ \text{complexity}(f)$$

- More examples: Lasso, Ridge, tree models, etc.

Degrees-of-freedom

- Sometime the model complexity can be measured using the degrees-of-freedom, e.g., ℓ_0 penalty
- In linear regression, the degrees-of-freedom is the number of variables used in the model
- A more general definition is

$$\text{df}(\widehat{f}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \text{Cov}(\widehat{Y}_i, Y_i)$$

- Treat $X_i = x_i$'s as fixed values, not random
- $1/\sigma^2$ takes care of the variance of the random error term

Example

- If we let $\widehat{\mathbf{Y}} = (\widehat{Y}_1, \ldots, \widehat{Y}_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_n)$, we can rewrite the definition as

$$\text{df}(\widehat{f}) = \frac{1}{\sigma^2}\text{Trace}\Big(\text{Cov}(\widehat{\mathbf{Y}}, \mathbf{Y})\Big)$$

- This can be convenient for linear regression

Example

We can easily verify several cases:

- For 1NN, df $= n$
- If $\widehat{y}_i = \overline{y}$, i.e., nNN, then df = 1
- For linear regression, df $= p$
- For kNN, df $= n/k$

The formula works for kernel methods too.

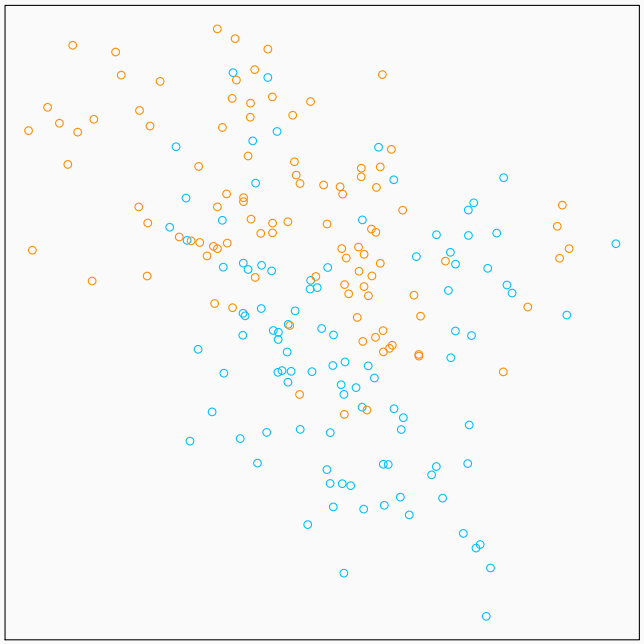
kNN for Classification

kNN for Classification

For hard classification, the most prevalent class in $N_k(x_0)$ is used

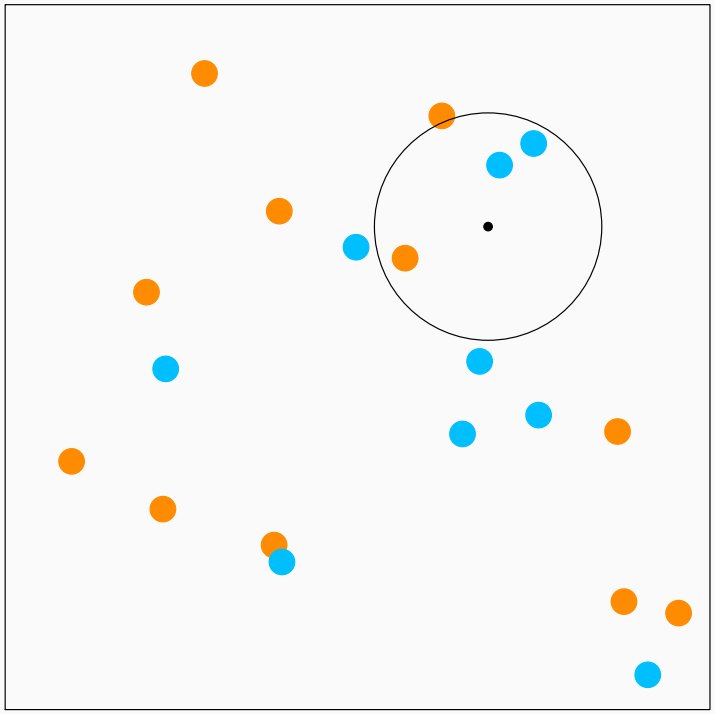
$$\widehat{y} = \arg\max_{c \in C} \sum_{x_i \in N_k(x_0)} \mathbf{1}\{y_i = c\},$$

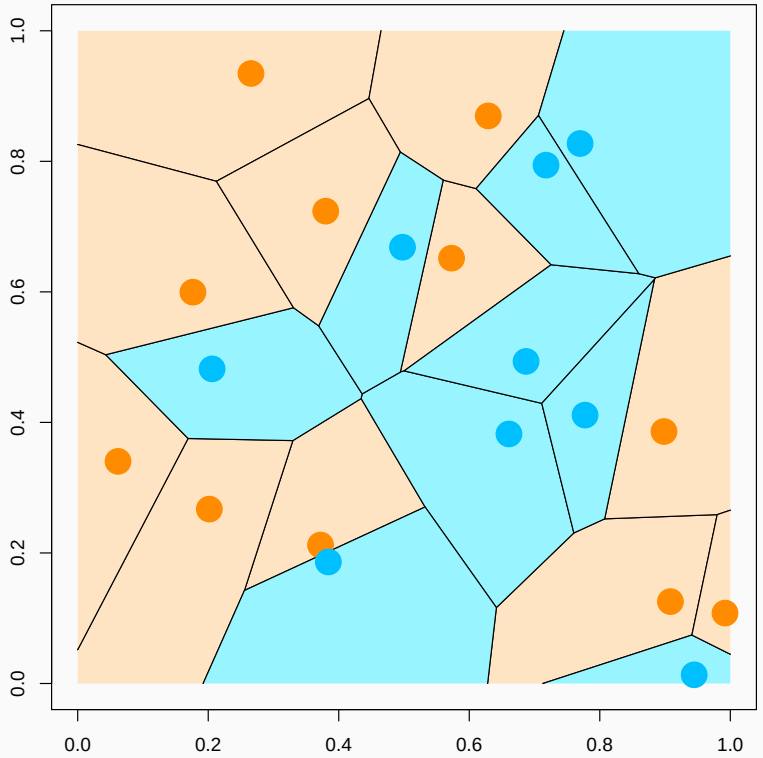
An example from the HTF textbook. (BLUE = 0, ORANGE = 1)



Voronoi Tessellation

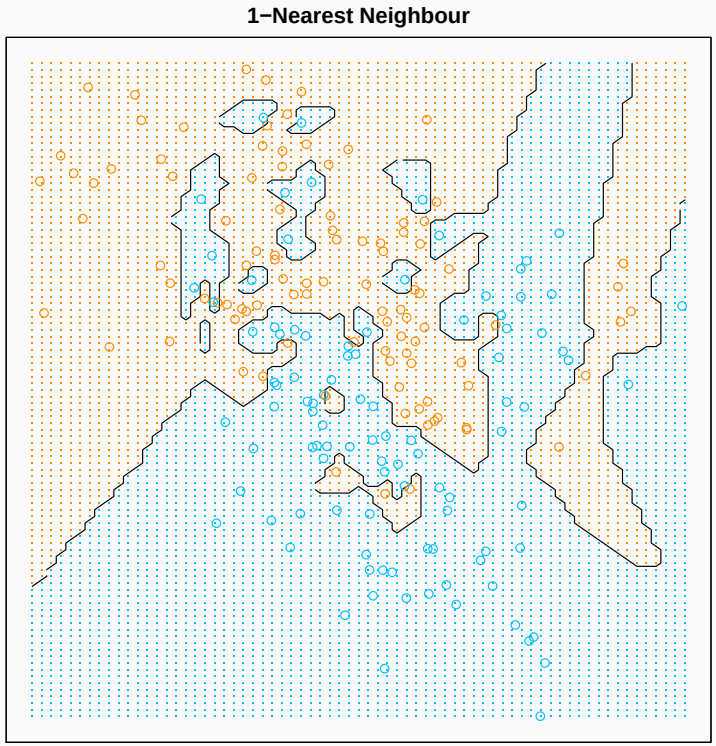
Similar to the regression case, the k-NN classification model does majority vote (the most prevalent class) within the neighborhood of a target point x. 1NN plot is a Voronoi tessellation

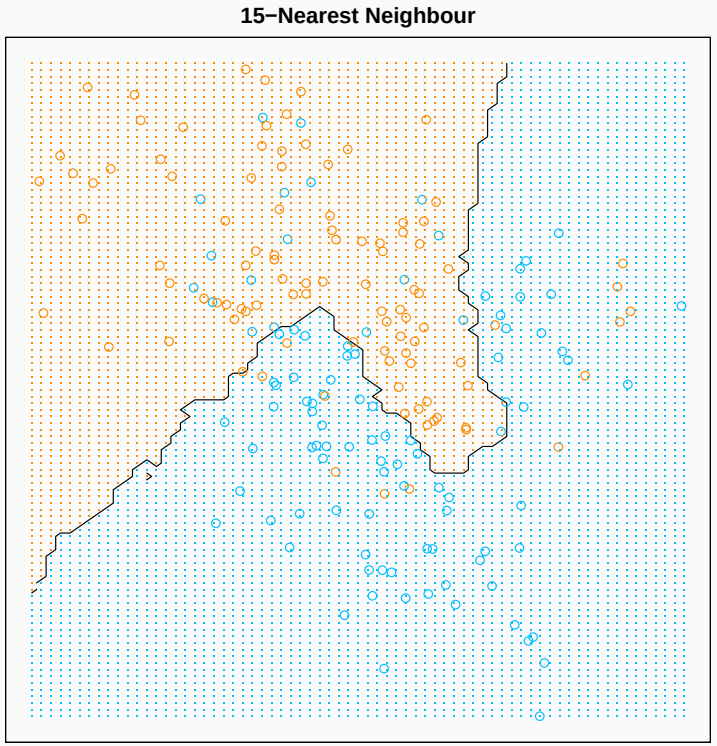




Example

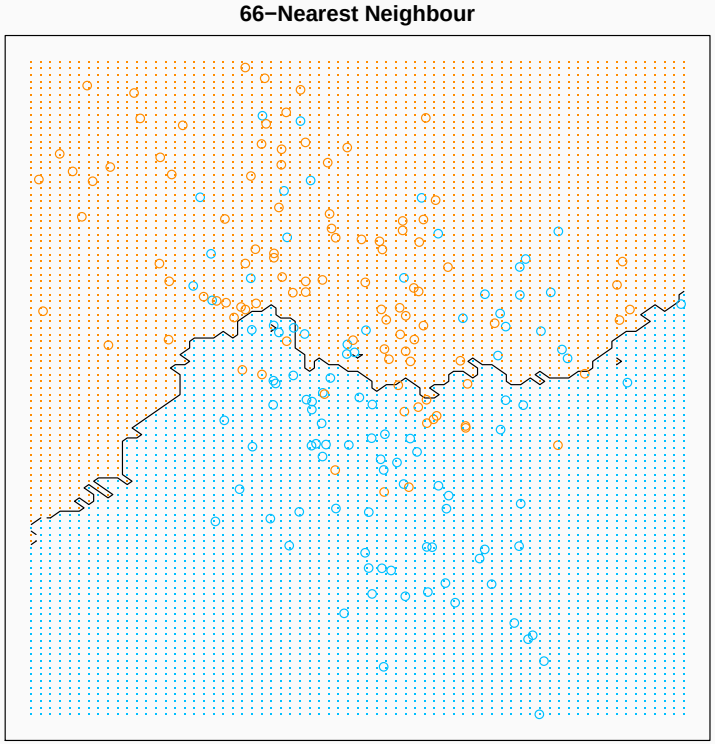
We fit k-NN classification model to the example. Of course, we would not expect 1NN to perform well...

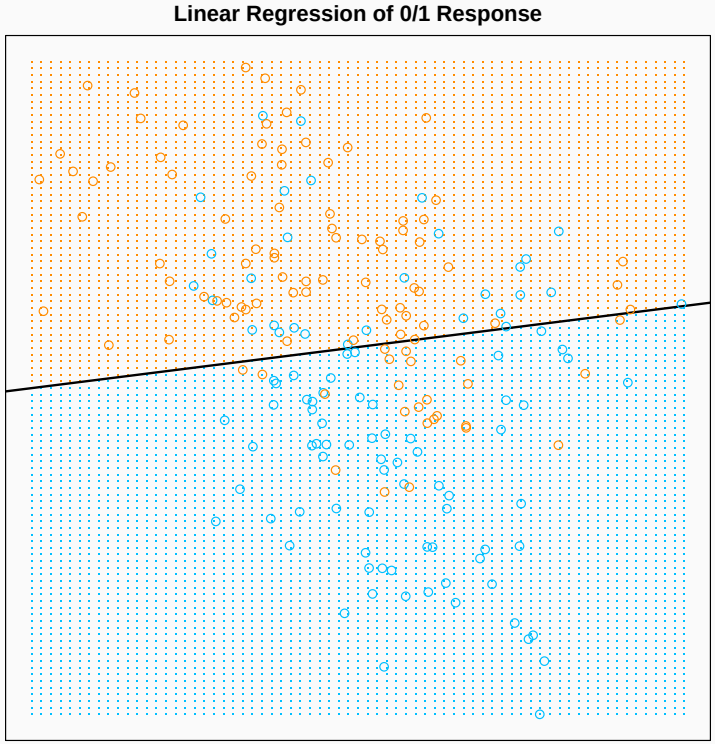




k-Nearest Neighbour in Classification

As we further increase k, the model tends to be less complex.
Compare 66NN with a linear model that uses only 3 parameters.





1NN Error Bound

- 1NN error is no more than twice of the Bayes error, as $n \to \infty$
- As $n \to \infty$, we have $d(x_0, x_{1nn}) \to 0$, where x_{1nn} is the closest neighbor of x_0. and we may assume that $\mathsf{P}(Y|x_{1nn}) \to \mathsf{P}(Y|x_0)$
- The error of 1NN is

$$\begin{aligned}
&\mathsf{P}(Y=1|x_0)[1-\mathsf{P}(Y=1|x_{1nn})] + [1-\mathsf{P}(Y=1|x_0)]\mathsf{P}(Y=1|x_{1nn}) \\
&\leq [1-\mathsf{P}(Y=1|x_{1nn})] + [1-\mathsf{P}(Y=1|x_0)] \\
&\approx 2[1-\mathsf{P}(Y=1|x_0)] \\
&= 2 \times \text{Bayes Error}
\end{aligned}$$

- This is a very crude bound, but it shows that if the noise is small, 1NN may be reasonable.

Remarks

Remarks

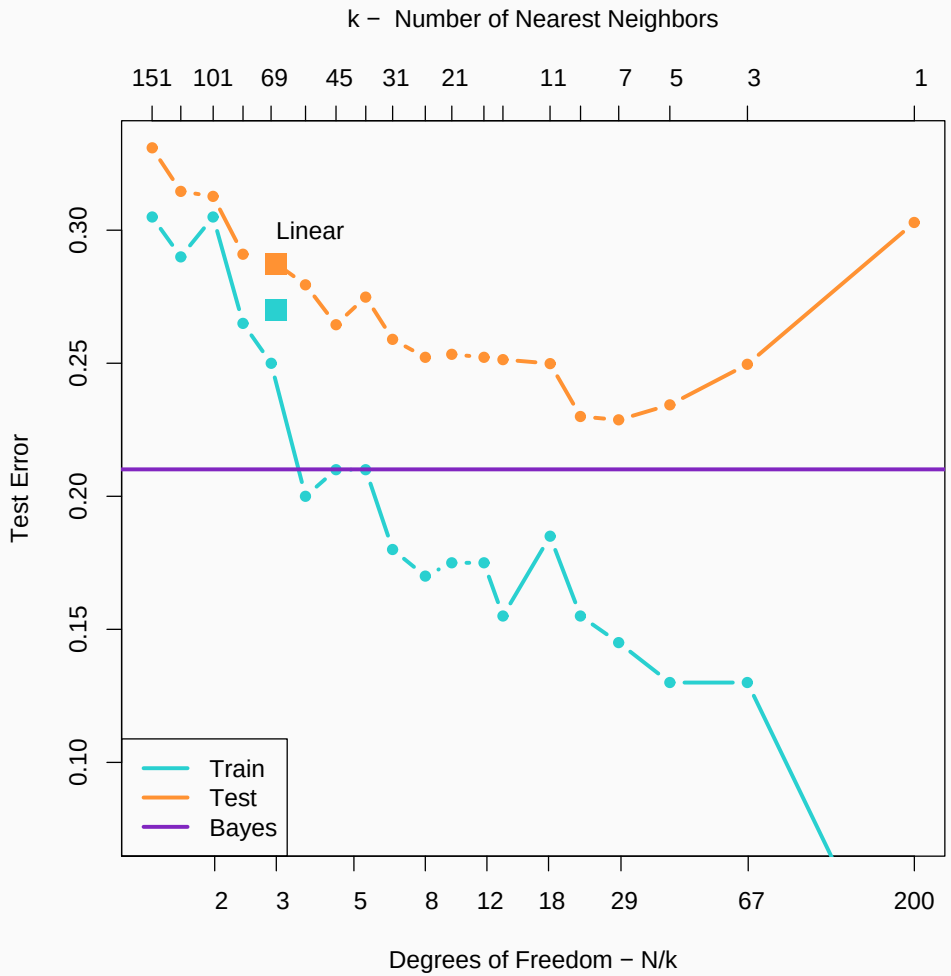
- kNN vs. linear model
- Distance measure
- Computational issue
- Curse of dimensionality
- Double descent

k-Nearest Neighbour vs. Linear Regression

- The goal is to approximate $f(x) = \mathsf{E}(Y|X = x)$
- Linear regression makes a structural assumption: f is linear.
  - low variance: Number of parameters is p (fixed); we know that when sample size n grows, the variance of $\widehat{\boldsymbol{\beta}}$ is $\propto 1/n$.
  - high bias (underfit): linear assumption is very restrictive
- kNN makes on assumption on f, except some smoothness.
  - low bias (overfit): flexible and adaptive. It can be shown that as if $k \to \infty$ and $n/k \to 0$, kNN is consistent.
  - high variance: number of parameters for kNN is roughly n/k;

k-Nearest Neighbour in Classification

An “U” shaped prediction error curve is again observed for the testing sample (Figure from HTF):



Distance measures

- Closeness between two points needs to be defined based on some distance measures
- By default, we use Euclidean distance (ℓ_2 norm) for continuous variables

$$d^2(\boldsymbol{u}, \boldsymbol{v}) = \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 = \sum_{i=1}^{p} (u_i - v_i)^2$$

Hence the neighbourhood is not invariant to the scaling of the variables.

Scaling issues

- We often scale the variables marginally when using kNN, so that the distance is

$$d^2(\boldsymbol{u}, \boldsymbol{v}) = \sum_{j=1}^{p} \frac{(u_i - v_i)^2}{\sigma_j^2}$$

  where σ_j^2 is the variance of variable j.

- Mahalanobis distance is also scale-invariant and takes care of correlation

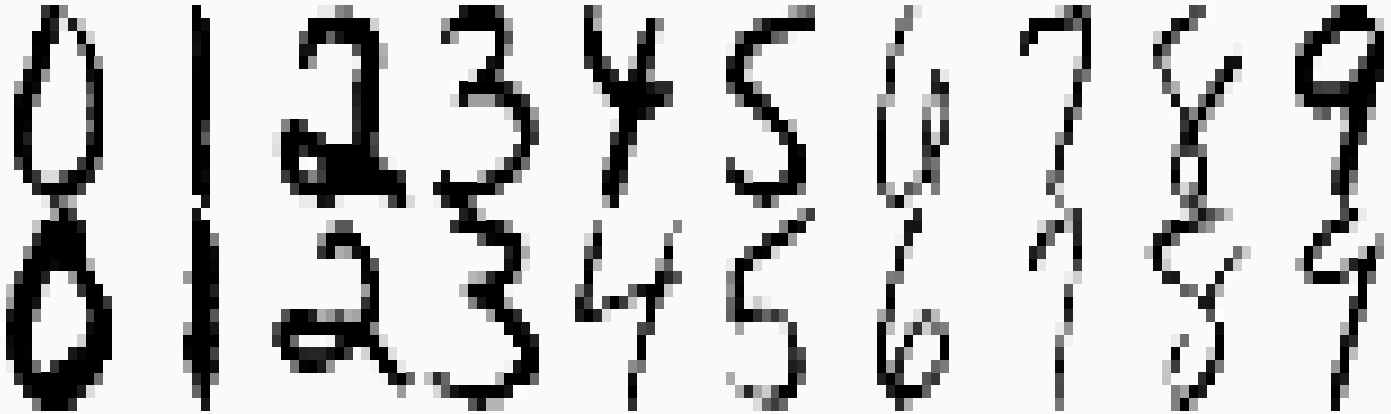
$$d^2(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u} - \boldsymbol{v})^\mathsf{T} \Sigma^{-1} (\boldsymbol{u} - \boldsymbol{v}),$$

  where Σ is a covariance matrix.

- Hamming distance is usually used for categorical variables

Example: Handwritten Digit Recognition Data

- Digits 0-9 scanned from envelopes by the U.S. Postal Service
- 7291 training samples, 2007 testing samples
- Apply kNN and calculate the errors
- 1NN with Euclidean distance gives 5.6% error rate
- 1NN with tangent distance (Simard et al., 1993) gives 2.6% error



Computational Issue

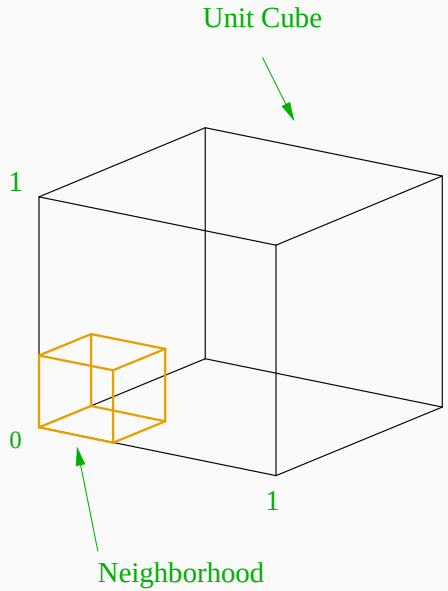
- Need to store the entire training data for future prediction
- Prediction can be slow. Needs to calculate the distance from x_0 to all training sample and sort them.
- Some fast nearest neighbor search algorithms such as kd-tree
- A distance measure may affect accuracy

Curse of dimensionality

- High-dimension low sample size ($p \gg n$)
  - The resolution of the handwritten digit example is $16 \times 16 = 256$
  - Some common imaging data in medical are 1024×1024 while only a few hundred samples are available
  - Strategy games (Go, StarCraft, etc.) may have a huge number of variables
- Curse of Dimensionality
  - For fixed n, as p increases, the data become sparse
  - As p increases, the number of possible models explodes (computation burden, variable selection necessary)

Curse of Dimensionality

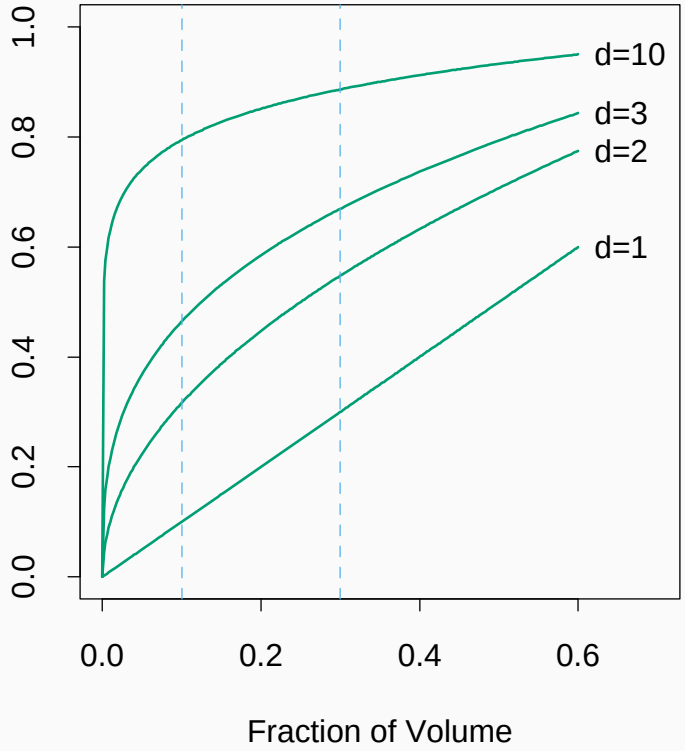
- The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube.
- Suppose the sample points are evenly spread out on $[0, 1]^p$, and we want to fit $k = 10$ nearest neighbors with $n = 1000$. Let l be the edge length of the hyper-cube that contains all k-nearest neighbor of a test point. How big is l?



- $l^p \approx \frac{k}{n}$
- When $p = 2$, l = 0.1
- When $p = 10$, l = 0.63
- When $p = 100$, l = 0.955

Curse of Dimensionality

- Suppose we have sample points evenly spread out on $[0, 1]$
- In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.

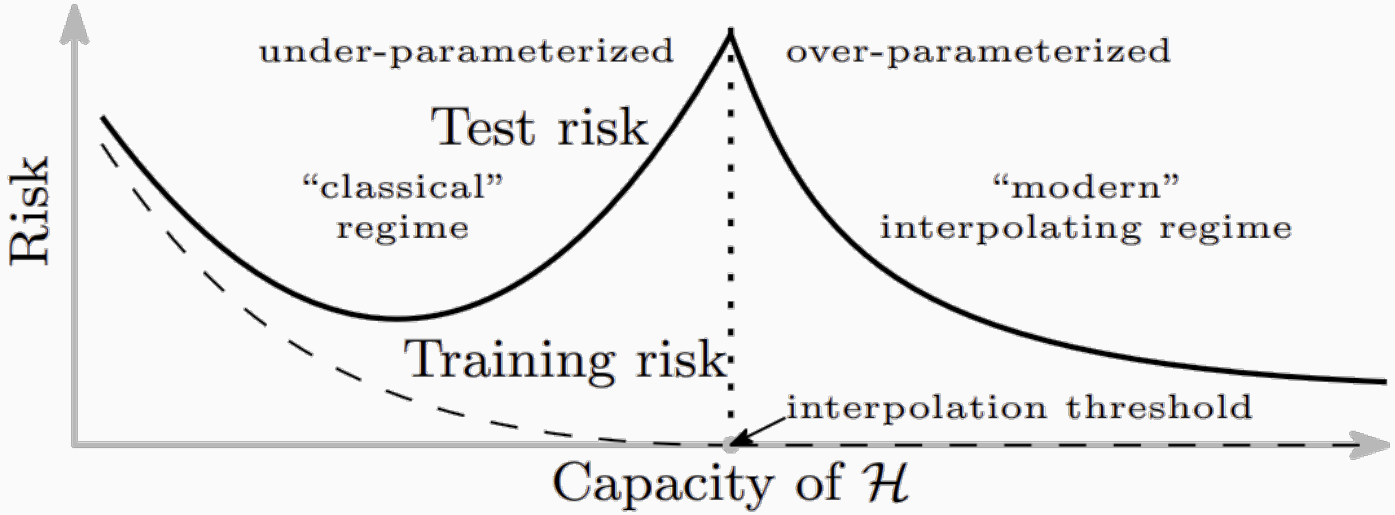


Low Dimensional Structure

- However, in the previous handwritten digit problem, KNN seems to work pretty well. Why?
- There is potential lower dimensional subspace (manifold).
- Total volume of the data is much reduced — there are more samples within the neighborhood of an existing sample

Double Descent

- Recent research shows that the bias-variance trade-off may not be everything
- E.g., deep learning models are always over-parameterized. However, they still have good performance.



Belkin, et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off." PNAS (2019)