STAT 5630, Fall 2019

Statistical Overview of Machine Learning

Xiwei (Denny) Tang, Ph.D. <xt4yj@virginia.edu>

University of Virginia
September 5, 2019

Different Perspectives of ML

- Learning Target: Prediction, Description, Representation

Different Perspectives of ML

- Learning Target: Prediction, Description, Representation
- Data: Structured data v.s. Unstructured data; Big size, High-dimension

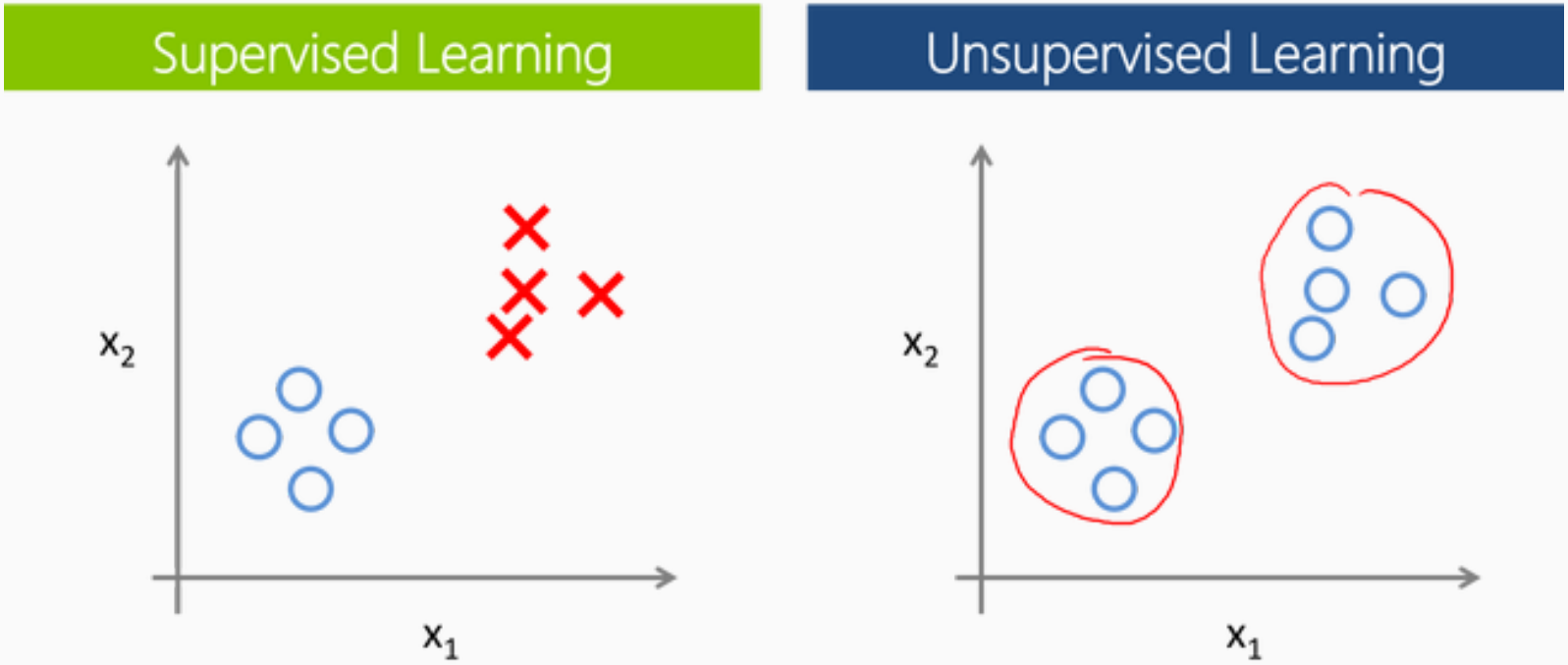
Different Perspectives of ML

- Learning Target: Prediction, Description, Representation
- Data: Structured data v.s. Unstructured data; Big size, High-dimension
- Philosophy: Statistics v.s. Computer Science

Types of Machine Learning

- **Prediction: Supervised Learning**
- **Description: Unsupervised Learning**
- Action: Reinforcement learning
- Additional: Feature learning (sparse dictionary learning); Federate learning, etc.

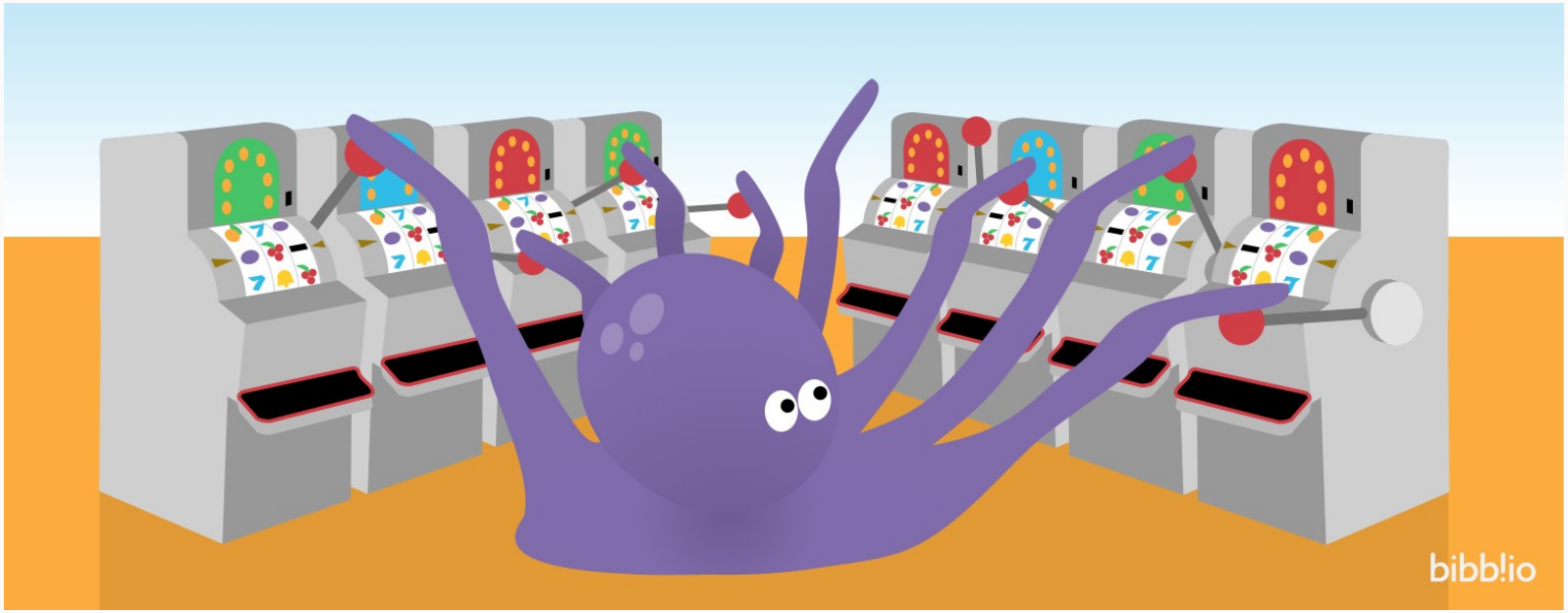
Supervised vs. Unsupervised Learning



Reinforcement learning

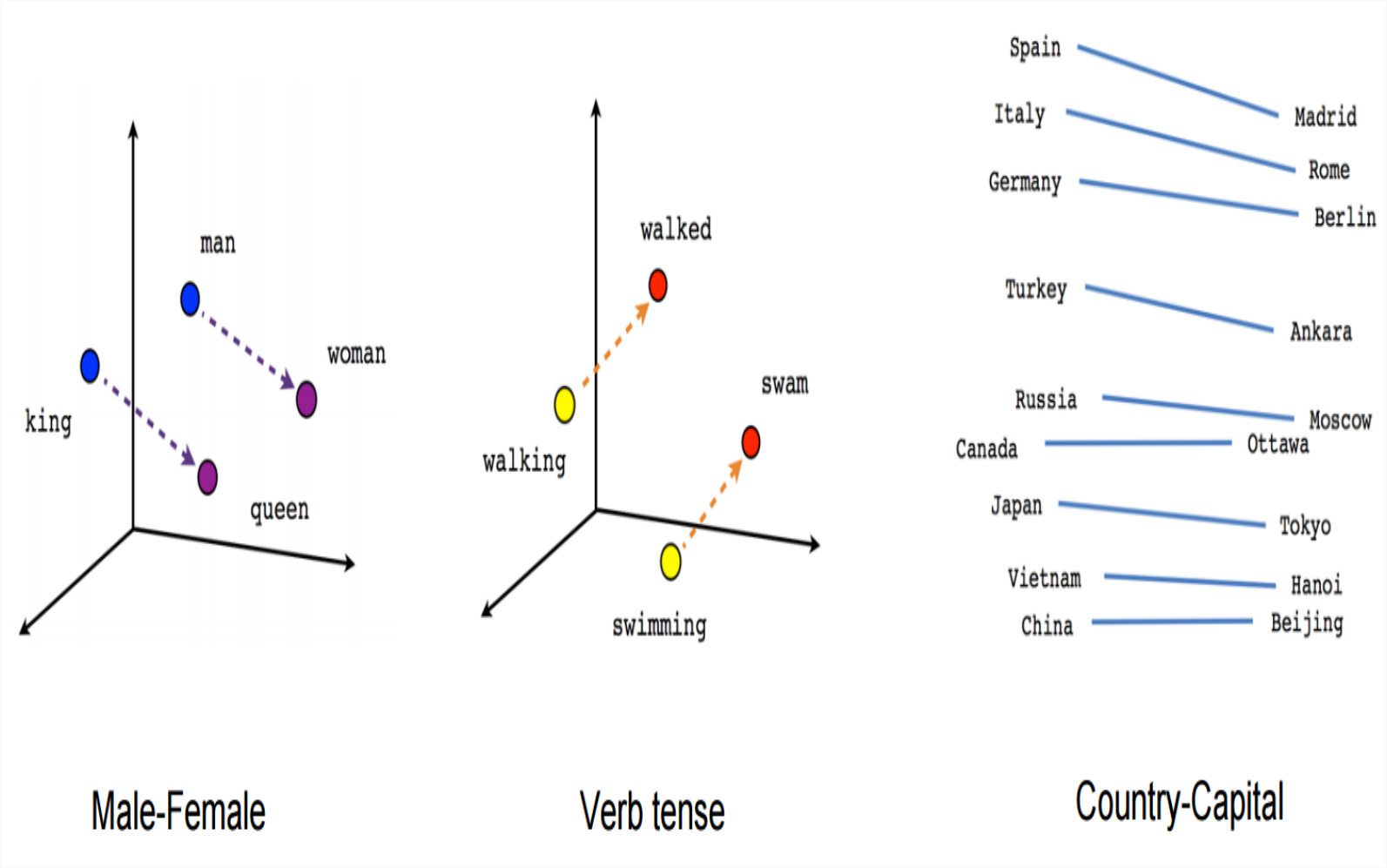
- How to take actions in an environment to maximize some notion of cumulative reward

- The environment is usually dynamic

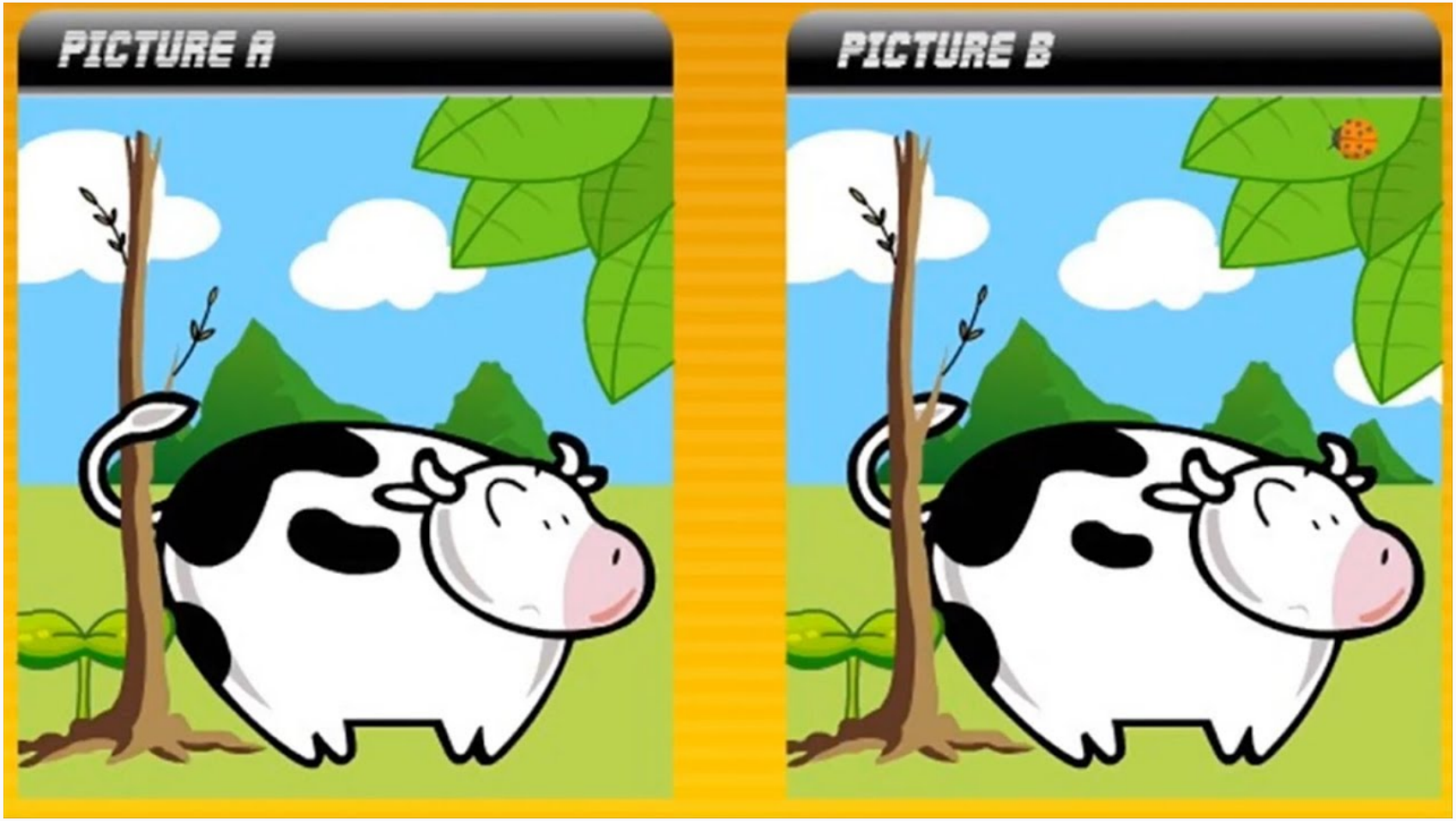


Dictionary learning: Word2Vector

- Represent unstructured text data to structured numerical data



Have a Break: Find the Difference!!!



HI v.s. AI

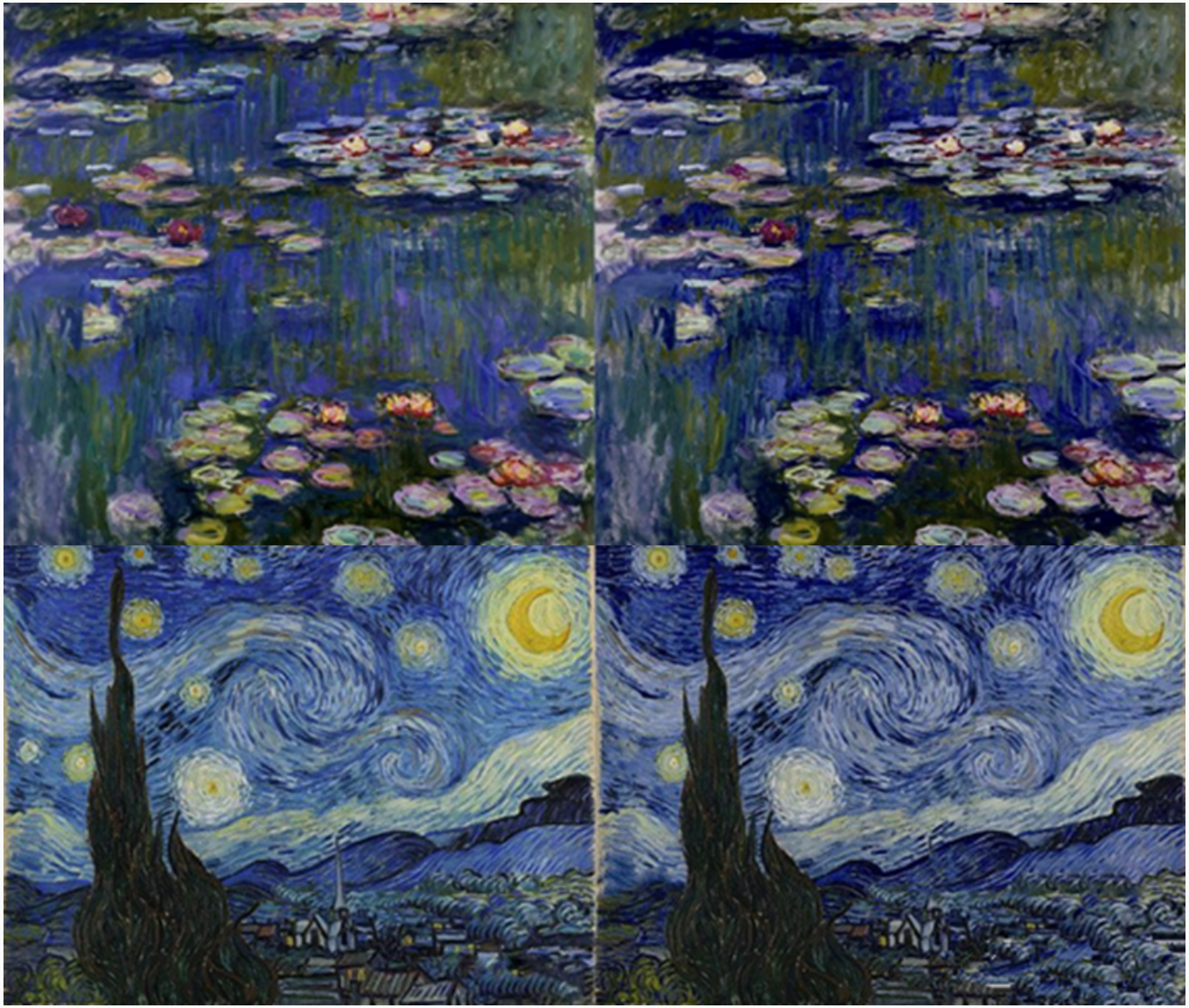
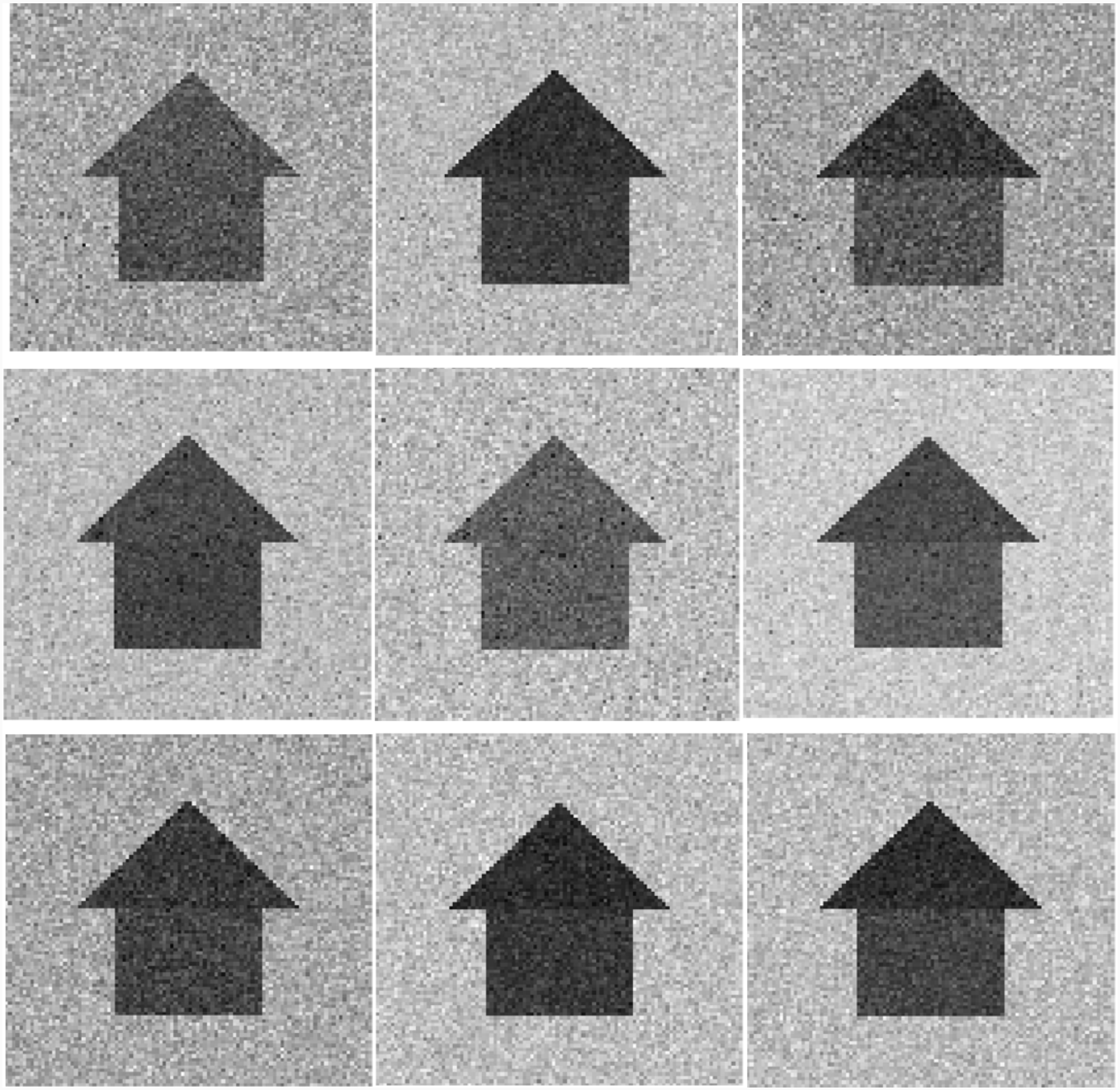


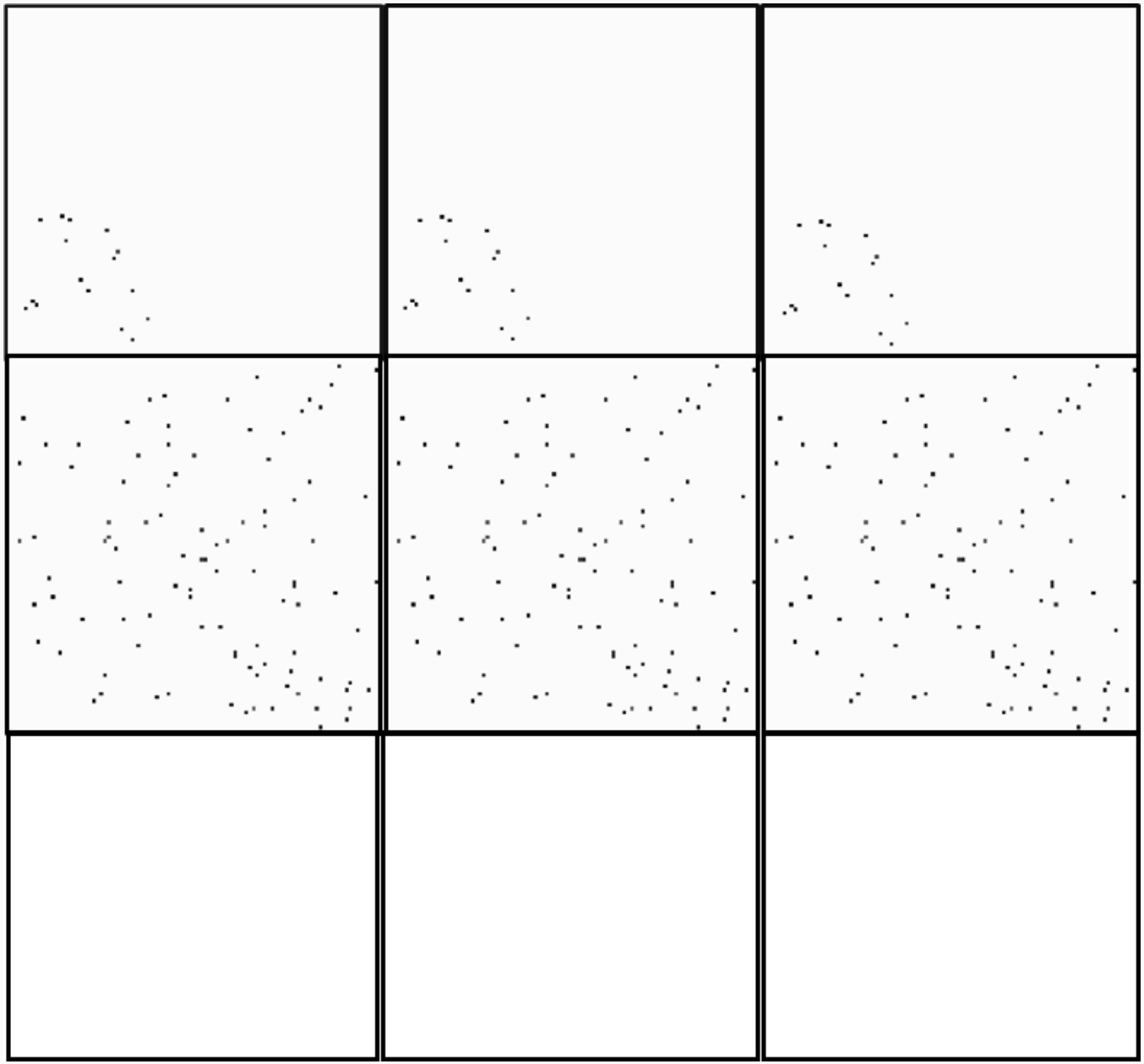
Illustration: How to Distinguish Individuals



Multi-layer Decomposition: Population+Individualized

- Raw Image $\longrightarrow$ Population Structure (Low-rank recovery) + Individualized Structure (Sparse)

Discriminant Features: Individualized Layer



Statistics v.s. ??

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant = $50,000
nice place to have a meeting: Snowbird, Utah, French Alps	nice place to have a meeting: Las Vegas in August

Statistics v.s. ??

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant = $50,000
nice place to have a meeting: Snowbird, Utah, French Alps	nice place to have a meeting: Las Vegas in August

- Why it works?

Statistics v.s. ??

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant = $50,000
nice place to have a meeting: Snowbird, Utah, French Alps	nice place to have a meeting: Las Vegas in August

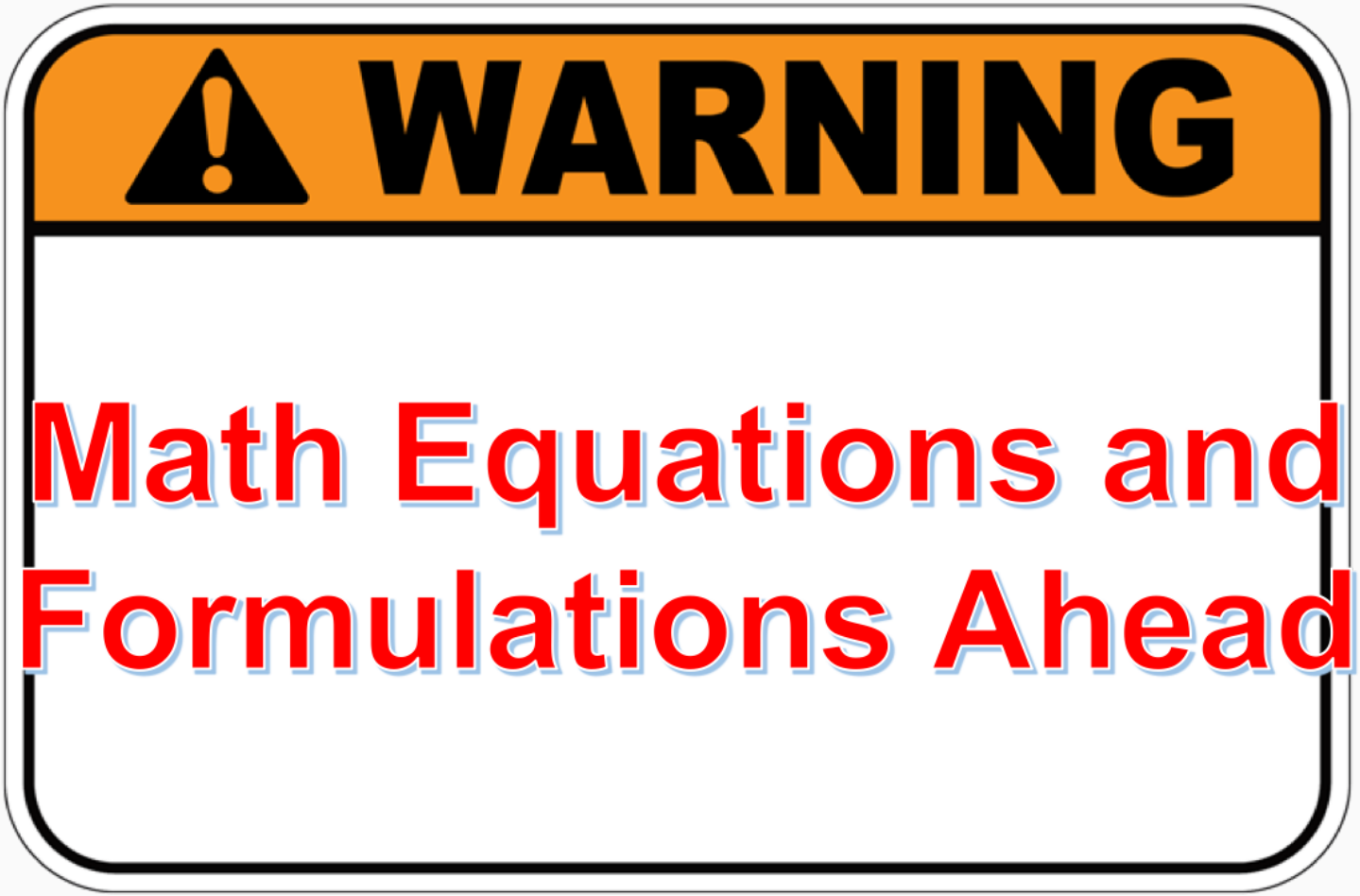
- Why it works? : Interpretability
- Sample $\longrightarrow$ Population

Statistics v.s. ??

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant = $50,000
nice place to have a meeting: Snowbird, Utah, French Alps	nice place to have a meeting: Las Vegas in August

- Why it works? : Interpretability
- Sample $\longrightarrow$ Population : Inference

Warning!!!



Supervised Learning

- Based on the training data, we aim to learn/estimate a function f, which describes the dependence between the set of features X and the outcome Y:

$$Y \longleftarrow f(X)$$

  - For regression $f : \mathbf{R}^p \to \mathbf{R}$
  - For classification $f : \mathbf{R}^p \to C$ or $f : \mathbf{R}^p \to$ a probability vector over elements in C (e.g. $f : \mathbf{R}^p \to [0, 1]$ for the probability of "1" in a binary classification)
- Goal:
  1. achieve small prediction error
  2. access the effect of each features on Y

Unsupervised Learning

- No Y, just a set of features $X \in \mathbf{R}^p$
- Goal: find patterns in the data (such as clusters) understand the data generating process of X, etc.
- Different from the supervised learning, it is difficult to measure the performance of an unsupervised clustering algorithm.
- The goal is fuzzy, the objective is not well-defined, but nevertheless it is an important problem.

Supervised Learning

- Response variable or outcome Y
  - regression Y is quantitative/continuous
  - classification Y is categorical/discrete
- A set of independent variables or features $X \in \mathbf{R}^p$
  - p is the dimension
- Training data $\mathcal{D}_n = \{x_i, y_i\}_{i=1}^n$, where each $\{x_i, y_i\}$ (one observations) is a random draw of $\{X, Y\}$.
  - $y_i \in \mathbf{R}$ for regression
  - $y_i \in C$ for classification, e.g., $C = \{0, 1\}$ or $\{-1, +1\}$ for binary class.

General Framework

- Paired data (X, Y): X is the predictor and Y is the response
- Goal: find a function $f(\cdot)$

$$Y \longleftarrow f(X)$$

  - For regression $f : \mathbf{R}^p \to \mathbf{R}$
  - For classification $f : \mathbf{R}^p \to C$ or $f : \mathbf{R}^p \to$ a probability vector over elements in C (e.g. $f : \mathbf{R}^p \to [0, 1]$ for the probability of "1" in a binary classification)

- Train $\hat{f}(\cdot)$ based on fully observed data $(X_1, Y_1), \ldots, (X_n, Y_n)$
- How to find the $\hat{f}(\cdot)$?

$$\min_f \text{loss}\left(f, \{(X_i, Y_i)\}_{i=1}^n\right)$$

e.g., L_2 loss (squared loss)

$$\min_f \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

General Framework

- How to evaluate the trained $\hat{f}(\cdot)$?
  - training error: Dist($\hat{f}(X_i)$, Y_i), e.g., $\frac{1}{n}\sum_{i=1}^{n}(\hat{f}(X_i) - Y_i)^2$
  - prediction error: Dist($\hat{f}(X)$, Y), e.g., E$[(\hat{f}(X) - Y)^2]$
- Data Structure: paired data (X, Y)
  - Training set: observed data pairs $(X_1, Y_1), \ldots, (X_n, Y_n)$
  - Testing set (X, Y): independent with the trained model $\hat{f}(\cdot)$
  - *Validation set (X^v, Y^v): for model selection (pseudo testing)

k-Nearest Neighbor: an example of the bias-variance tradeoff

An accurate prediction

Let's consider a regression model,

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

where $\mathsf{E}(\epsilon) = 0$ and $\mathsf{Var}(\epsilon) = \sigma^2$. Suppose that from the training data, we are able to estimate the regression function as $\widehat{f}$ ("f-hat"). And we now want to predict the value of Y at a target point x. Using squared-error loss, the error of the prediction is

$$\begin{aligned}
&\mathsf{Err}(x) \\
&= \mathsf{E}\left[(Y - \widehat{f}(x))^2\right] \\
&= \mathsf{E}\left[\left(Y - f(x) + f(x) - \mathsf{E}\widehat{f}(x) + \mathsf{E}\widehat{f}(x) - \widehat{f}(\mathbf{x})\right)^2\right] \\
&= \underbrace{\mathsf{E}\left[\left(Y - f(x)\right)^2\right]}_{\text{Irreducible Error}} + \underbrace{\left(f(x) - \mathsf{E}\widehat{f}(x)\right)^2}_{\text{Bias}^2} + \underbrace{\mathsf{E}\left[\left(\widehat{f}(x) - \mathsf{E}\widehat{f}(x)\right)^2\right]}_{\text{Variance}}
\end{aligned}$$

An accurate prediction

- $\mathrm{E}[(Y - f(x))^2] = \sigma^2$ is the irreducible error term that cannot be avoided, because we cannot predict ϵ
- $(f(x) - \mathrm{E}\widehat{f}(x))^2$ is the squared bias term that evaluates how the average of our estimator deviates from the truth
- $\mathrm{E}[(\widehat{f}(x) - \mathrm{E}\widehat{f}(x))^2]$ is the variance term that reflects the sensitivity of the function estimate $\widehat{f}(x)$ to the training sample
- Of course we want to minimize both the Bias^2 and the Variance, however, this is not always possible...
- How about minimizing the bias first?

k-Nearest Neighbour

- k-Nearest Neighbour (kNN) is a nonparametric method that predicts the target point $\mathbf{x}$ with averages of nearby observations in the training data

- For regression, the prediction at a given target point x is

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

  where $N_k(x)$ defines the k samples from the training data (in terms of their feature values) that are closest to x.

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

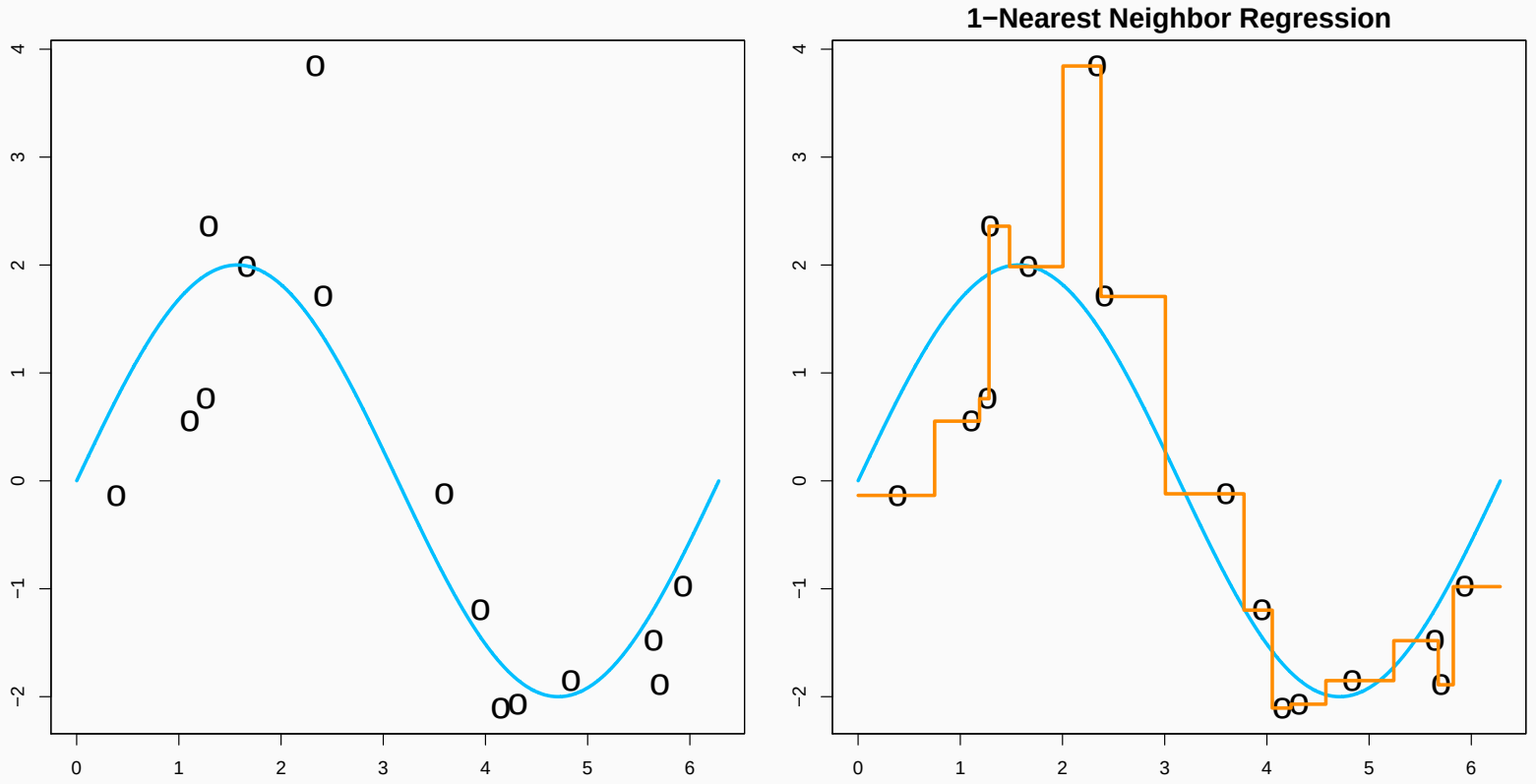
$$\widehat{y} = \arg\max_{c \in C} \sum_{x_i \in N_k(x)} \mathbf{1}\{y_i = c\},$$

k-Nearest Neighbour in Regression

The the following data is observed, with only 1 feature, uniformly from $[0, 2\pi]$. The true model (blue line) is

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

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



k-Nearest Neighbour in Regression

- Consider a new observation with x the same as one of the training data
- The bias^2 term is minimized using 1NN, why?

$$\begin{aligned}\mathsf{E}\widehat{f}(x) &= \mathsf{E}\big[\sum Y_i \mathbf{1}\{X_i \in N_1(x)\}\big] \\ &= \mathsf{E}\big[\sum f(X_i)\mathbf{1}\{X_i \in N_1(x)\}\big] = 2\sin(x)\end{aligned}$$

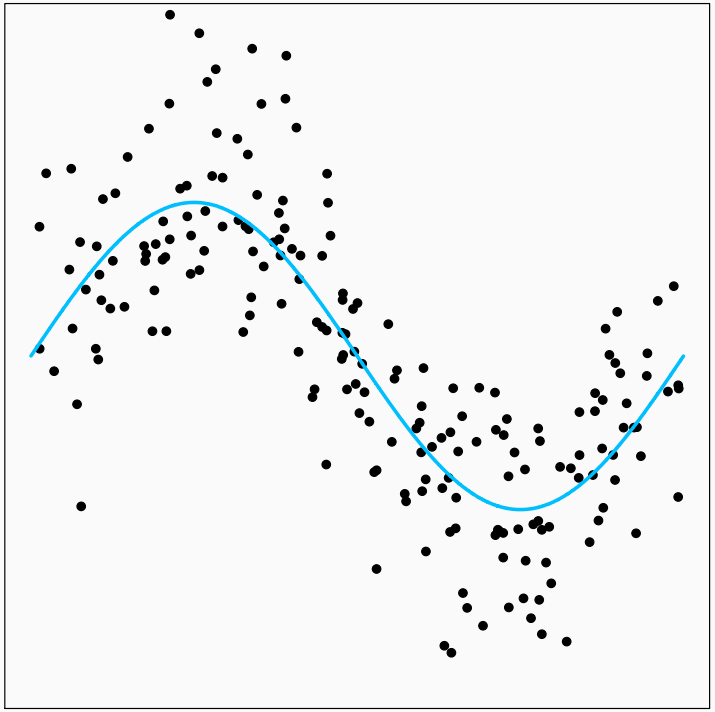
  which the closest we can get for approximating $f(x)$.
- However, the variance is large because the estimator only uses one observation

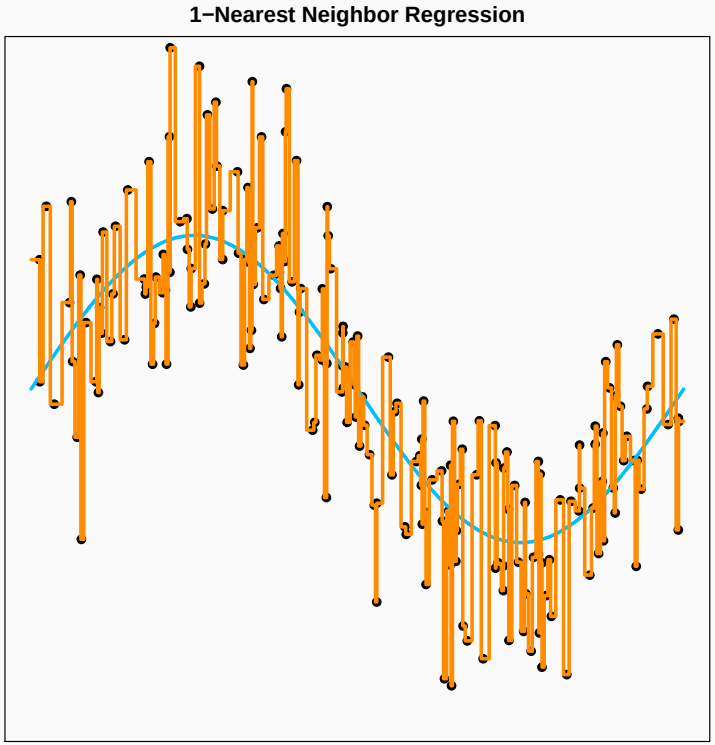
$$\mathsf{E}\big[(\widehat{f}(x) - \mathsf{E}\widehat{f}(x))^2\big] = \mathsf{E}\epsilon^2 = \sigma^2.$$

- If we use more "neighbouring" points, say k, the variance would reduce to σ^2/k. But the bias^2 will increase because as the neighbourhood expands, $f(X_i)$ is further away from $f(x)$.

k-Nearest Neighbour in Regression

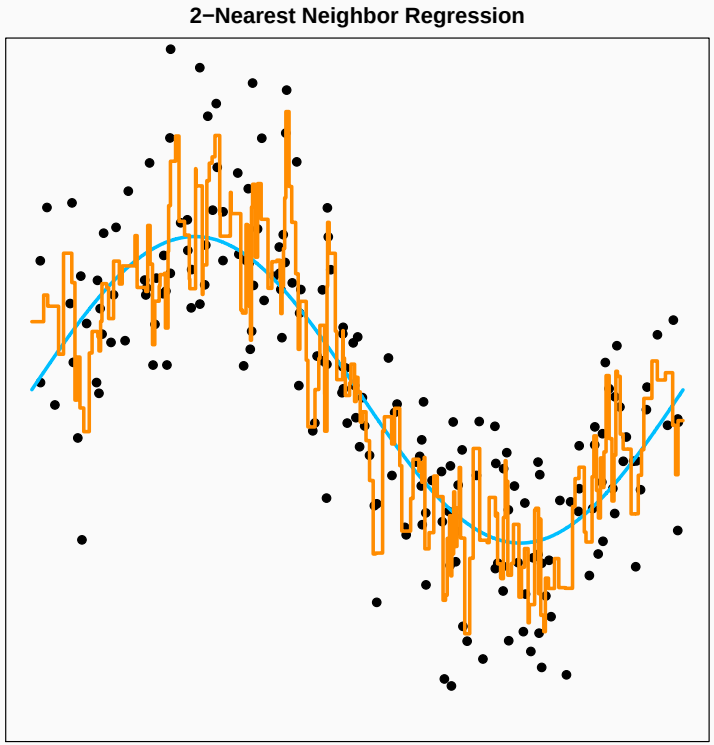
Now we simulate 200 observations, and see how the model changes over k.

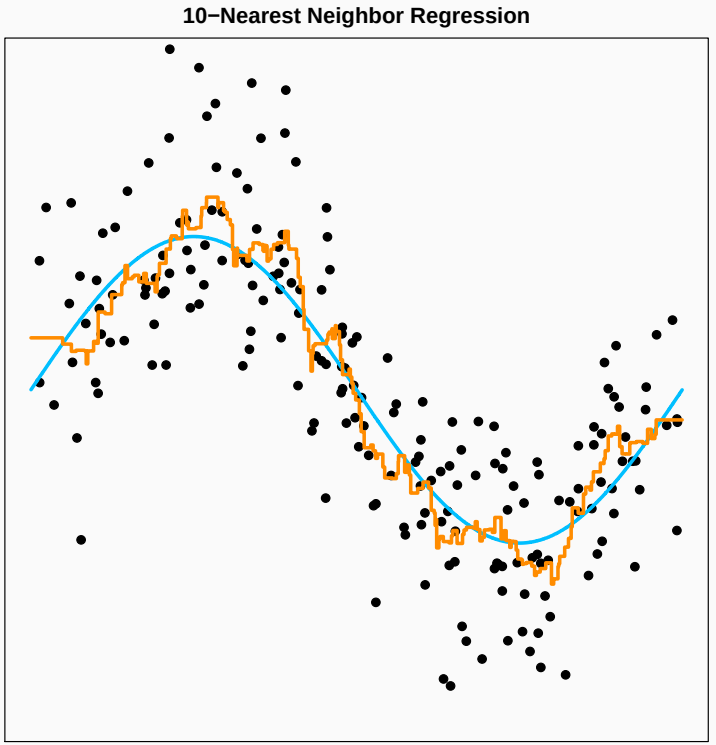




k-Nearest Neighbour in Regression

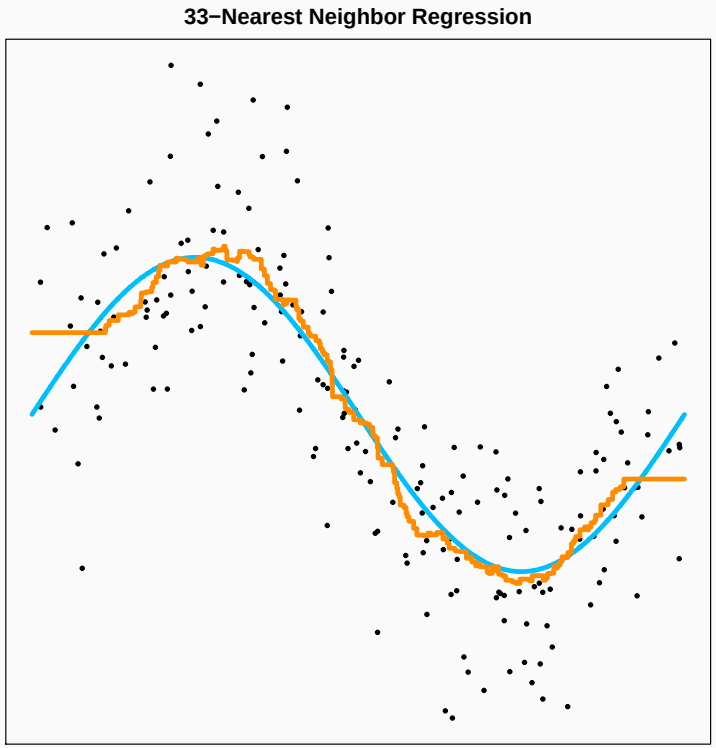
Now we simulate 200 observations, and see how the model changes over k.

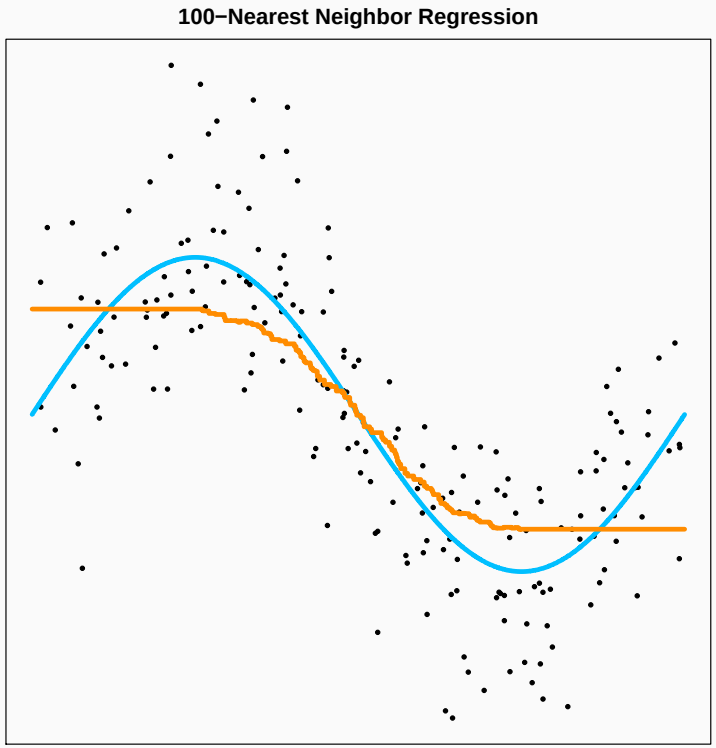




k-Nearest Neighbour in Regression

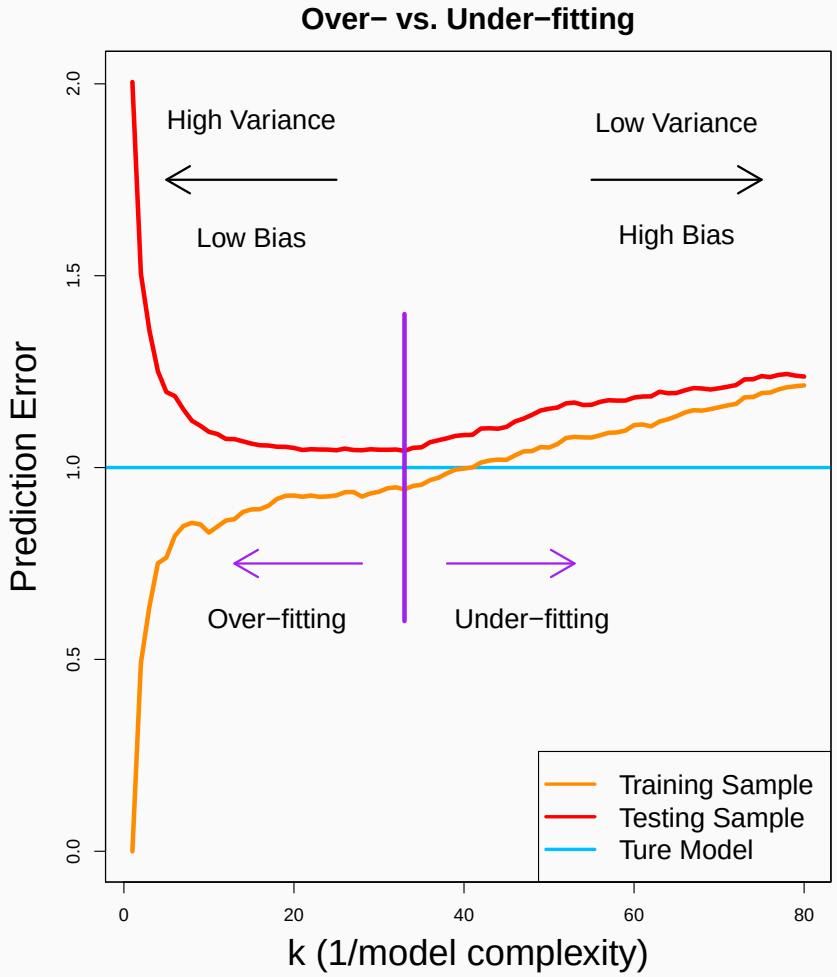
The model becomes “smoother” as k increases. However, this eventually introduces a significant bias.





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

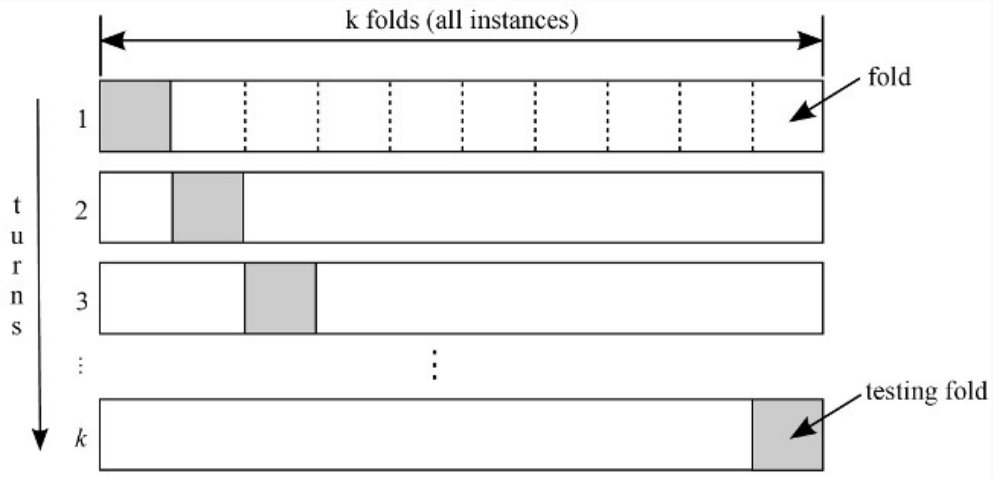
- As we can see, model complexity, bias-variance trade-off and over- and under-fitting are usually related concepts
- Over-fitting happens when the model performs well on the training sample, but not on the testing sample
- Controlling complexity can prevent overfitting. Complexity can be measured in different ways. In statistics, we often use degrees of freedom (number of parameters is a model)
- The degrees of freedom of a kNN model is roughly n/k
  - intuition: if neighborhoods don't overlap, there would be n/k neighborhoods, with one parameter for each

Prevent over-fitting

- To control complexity, one approach is to "penalize" it (we will see this a lot later on):

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

- Another common approach is to use cross-validation (CV). A 10-fold CV is carried out as follows
  - Randomly split the data into 10 equal sized subsamples
  - Fit the model using 9 out of 10 subsamples as training data and calculate the testing error using the remaining one.
  - Alternate the testing sample, and average over 10 experiments



Remark on the degrees of freedom

A rigorous definition for the degrees of freedom is

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

- The amount of covariance between outcome Y_i and its fitted values $\widehat{Y}_i$, at the same target point x_i.
- Treat $X_i = x_i$ as fixed value, but not random.
- $1/\sigma^2$ takes care of the variance of the random error term.
- In many situations, it is more convenient to define it in the matrix form, noticing that we are just adding up the diagonal elements of the covariance matrix of $\widehat{\mathbf{Y}} = (\widehat{Y}_1, \ldots, \widehat{Y}_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_n)$

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

Remark on the degrees of freedom

Based on this definition, 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 $= n/k$

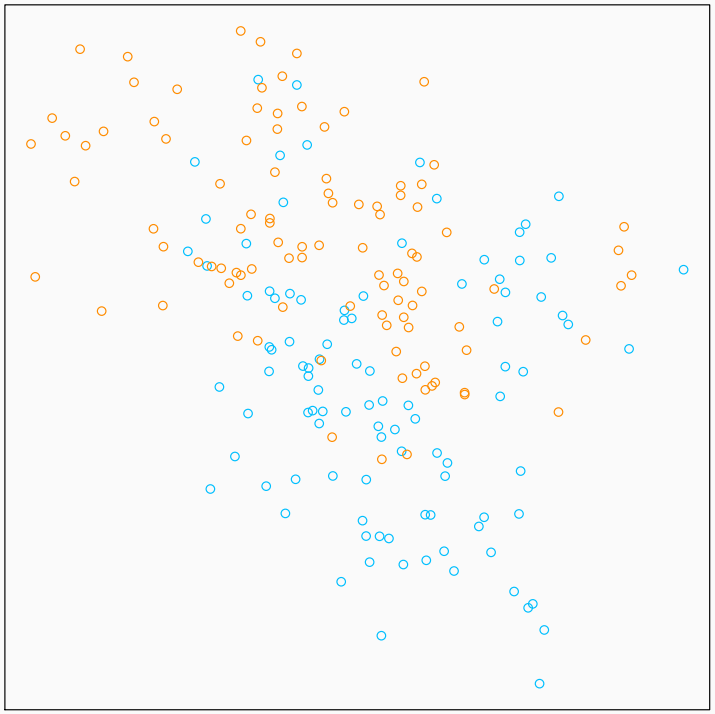
The formula works for kernel methods too, we shall see that later on.

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 $\propto p/n$.
  - high bias (underfit): linear assumption is very restrictive
- kNN makes no assumption on f, except some local contunuity.
  - low bias (overfit): flexible and adaptive. It can be shown that as if $k \to \infty$ and $k/n \to 0$, kNN is consistent.
  - high variance: number of parameters for kNN is roughly n/k;

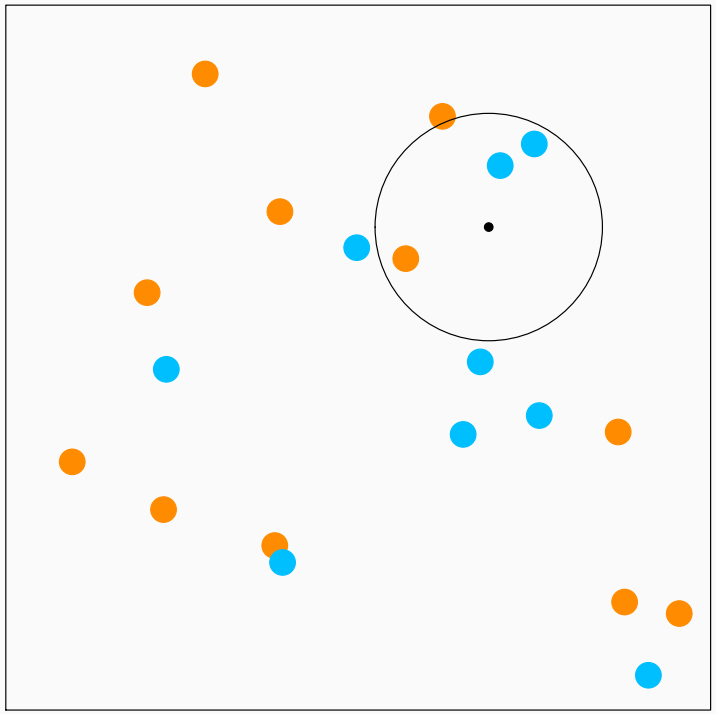
k-Nearest Neighbour in Classification

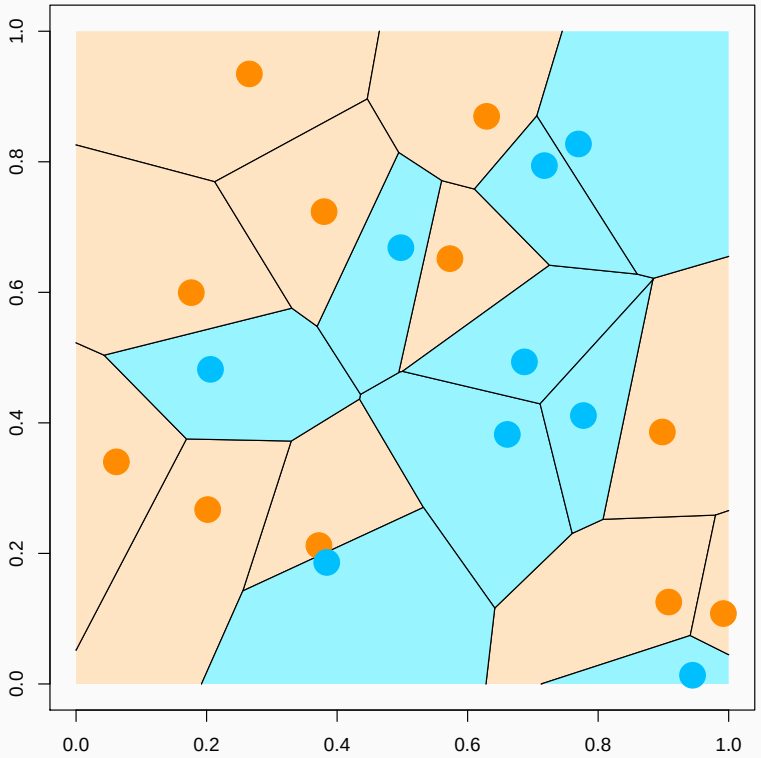
Let's look at a classification example from the HTF text book.
(BLUE = 0, ORANGE = 1)



k-Nearest Neighbour in Classification

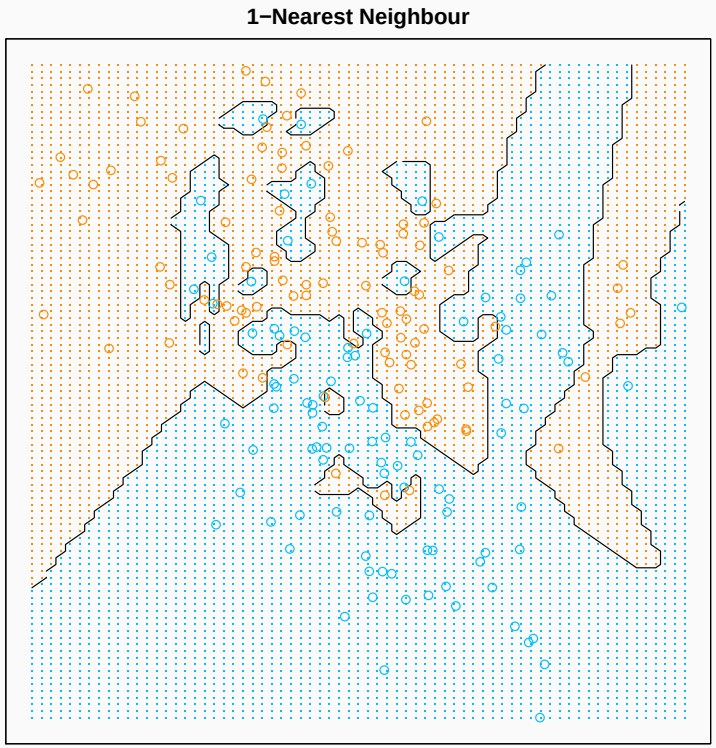
Similar to the regression case, the k-NN classification model does majority vote (the most prevalent class) within the neighborhood of a target point x. 1NN plot is a Voronoi diagram

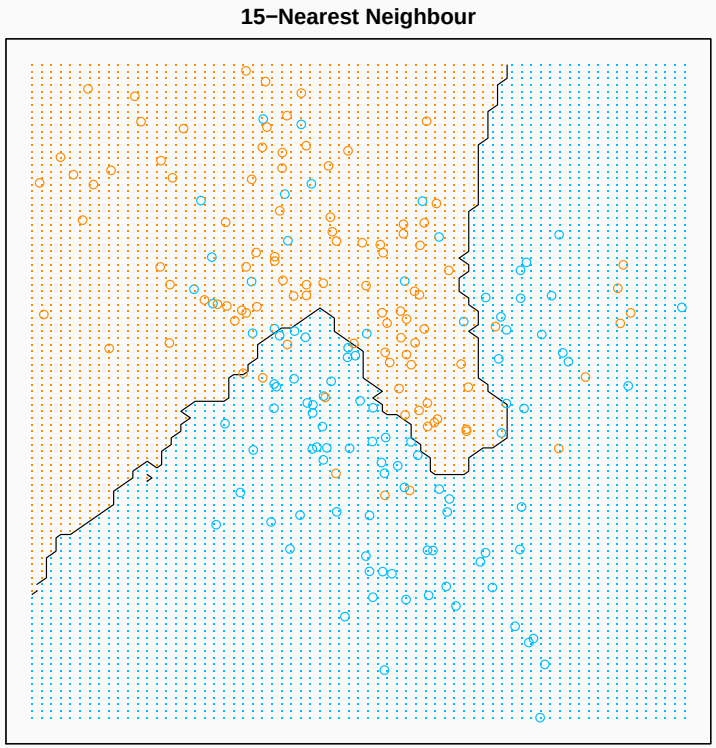




k-Nearest Neighbour in Classification

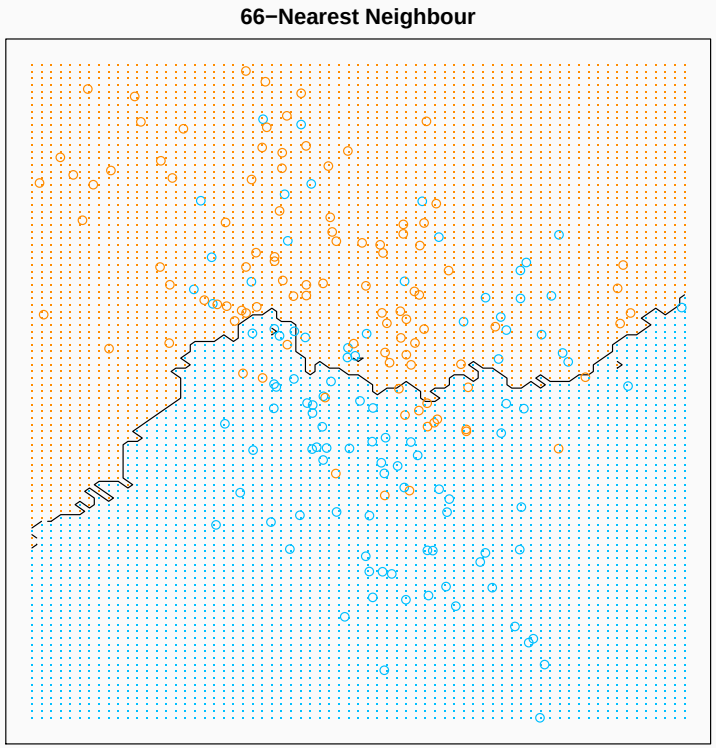
We fit k-NN classification model to the example. Of course, we would not expect 1NN to perform well...

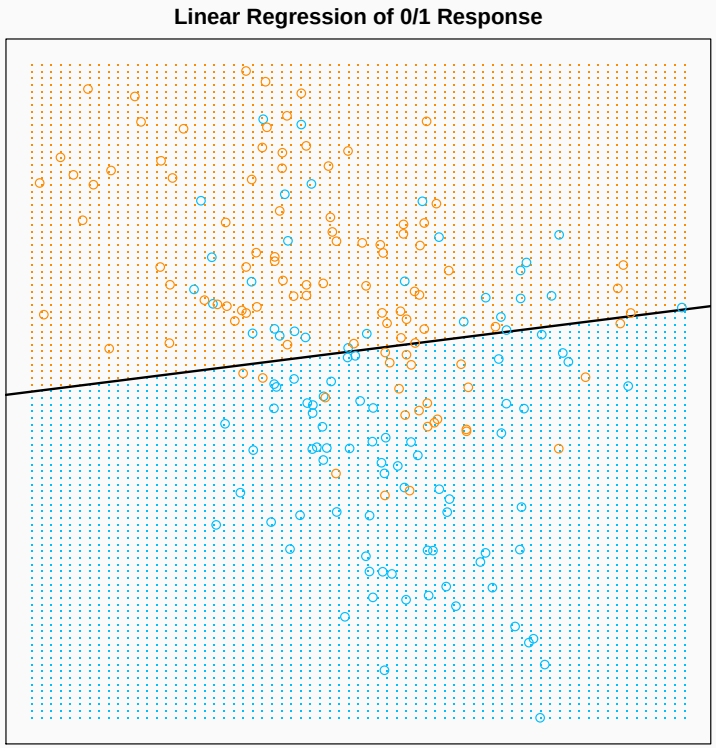




k-Nearest Neighbour in Classification

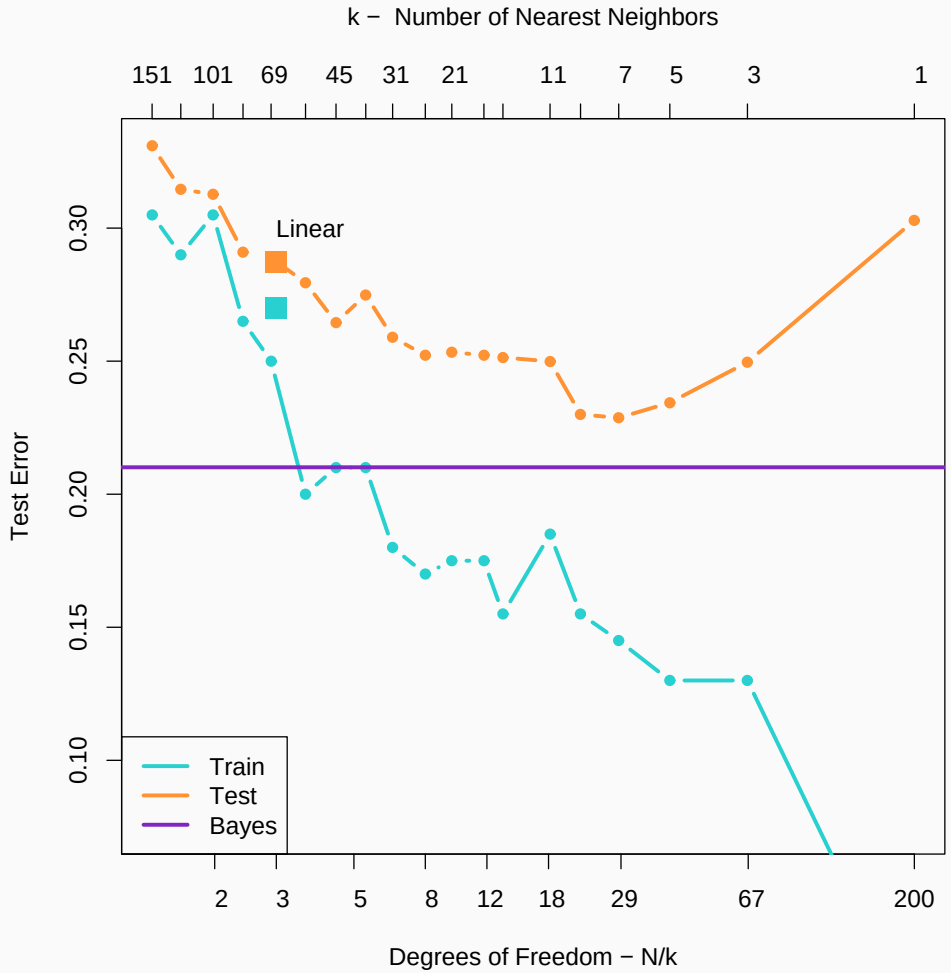
As we further increase k, the model tends to be less complex.
Compare 66NN with a linear model that uses only 3 parameters.





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
- Usually use Euclidean distance (ℓ_2) for continuous variables

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

  Note: the neighbourhood is not invariant to the variable scaling.
- Mahalanobis distance is scale-invariant

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

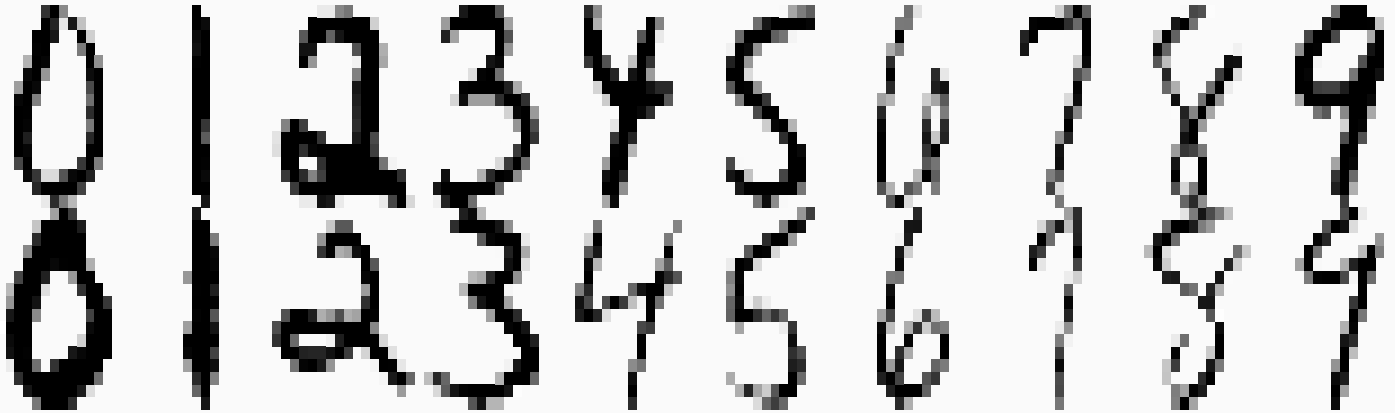
  where Σ is a covariance matrix.
- Hamming distance is usually used for categorical variables
- Weighted distance is powerful to specific target

Drawbacks of kNN

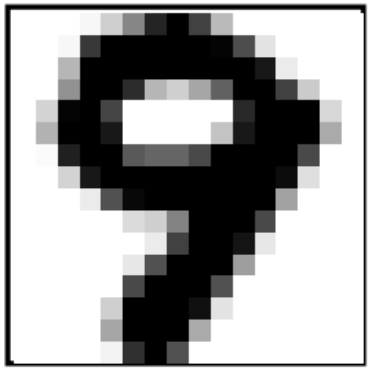
- Need to score the entire training data for future prediction
- To find the nearest neighbor of x, one needs to calculate the distance from x to all training sample and compare them. This is computationally expansive especially in high-dimensional setting (p is super large)
- A distance measure needs to be specified

Handwritten Digit Recognition Data

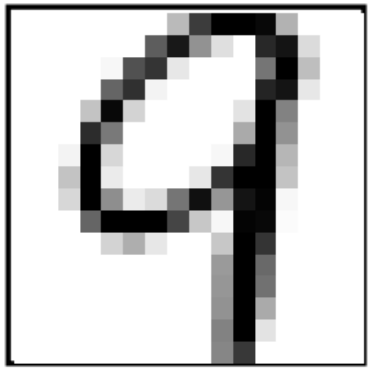
- Digits 1-9 scanned from envelopes by the U.S. Postal Service
- 7291 training samples, 2007 testing samples
- Apply kNN and calculate the errors
- 1NN with Euclidean distance gives 5.6% error rate
- 1NN with Tangent distance (Simard et al., 1993) gives 2.6% error



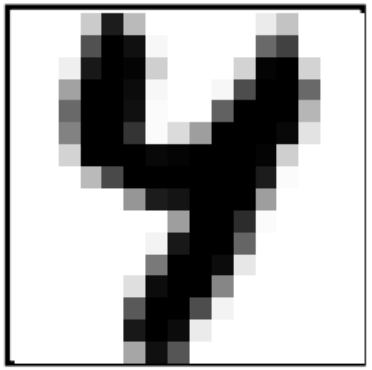
Translation Invariance



Pattern to be classified

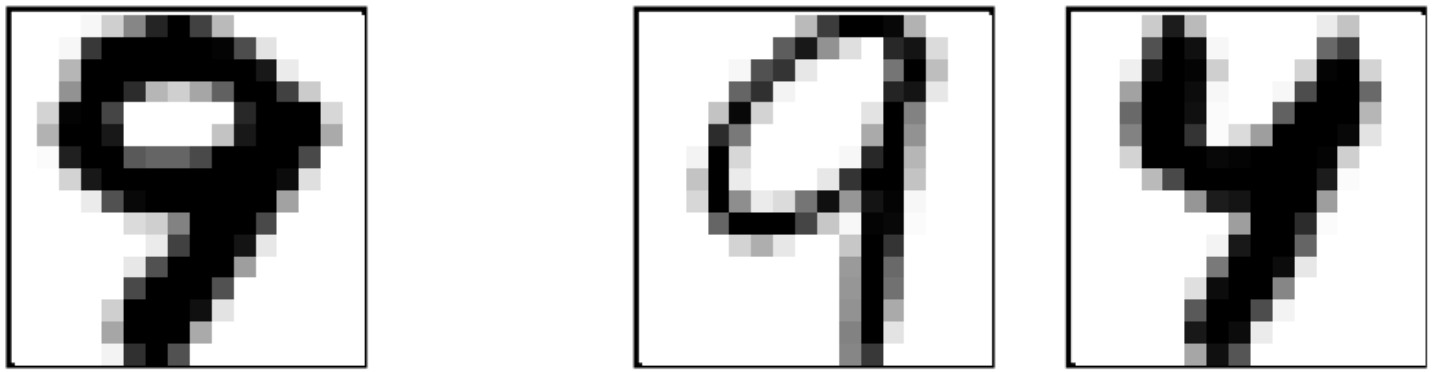


Prototype A



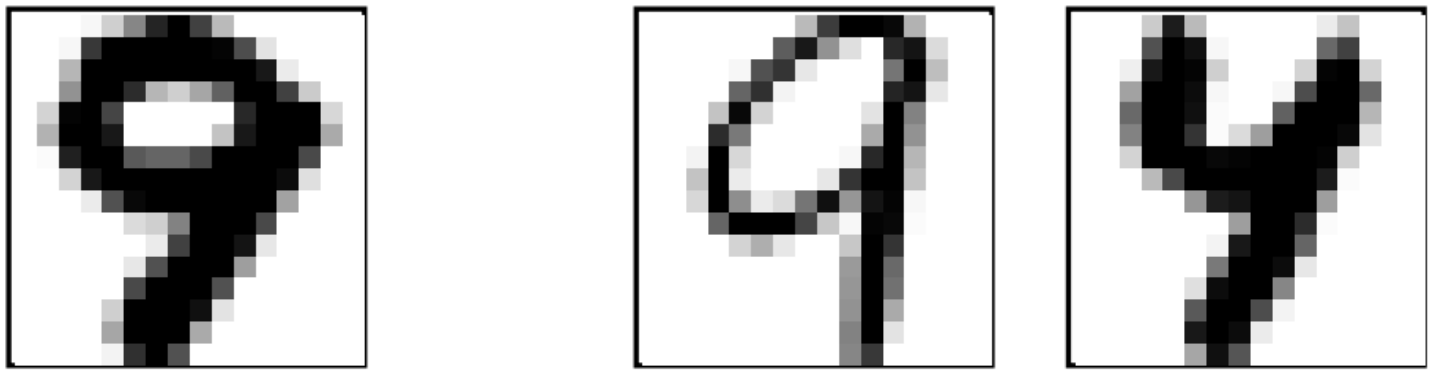
Prototype B

Translation Invariance



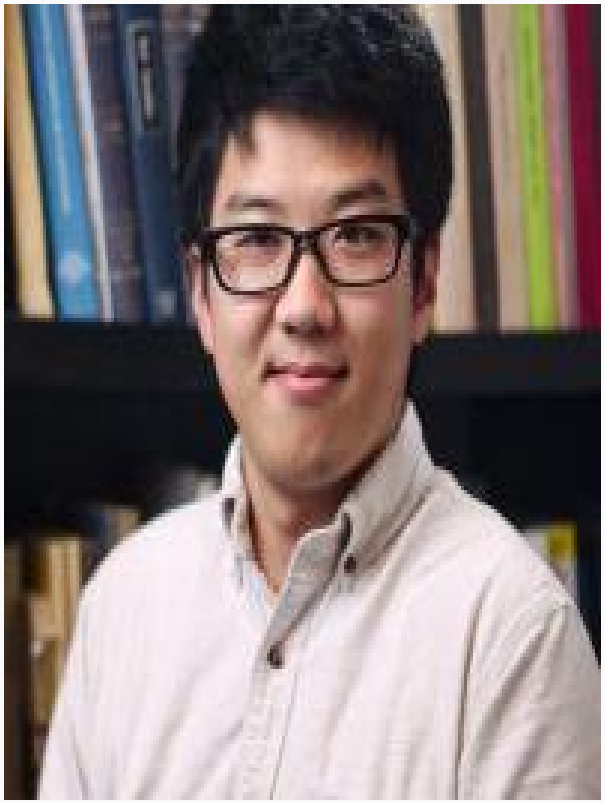
- Based on Euclidean-distance, prototype B is closer the the target

Translation Invariance

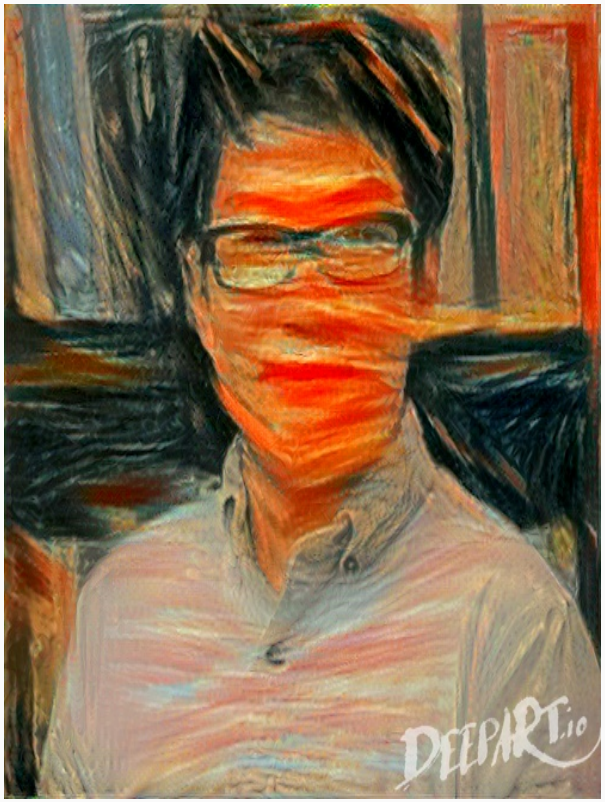


- Based on Euclidean-distance, prototype B is closer the the target
- Some natural transformations: shift, shear, rotate, scale, thinning...

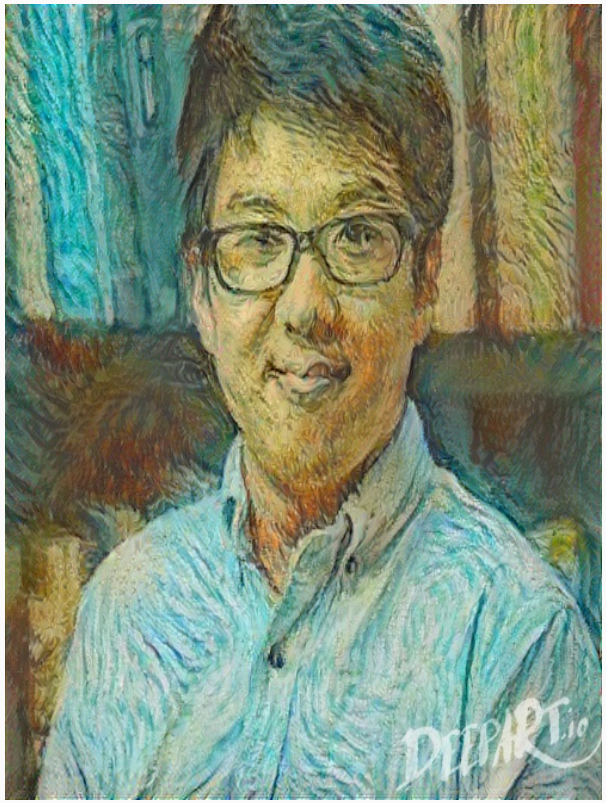
Neural Style Transfer



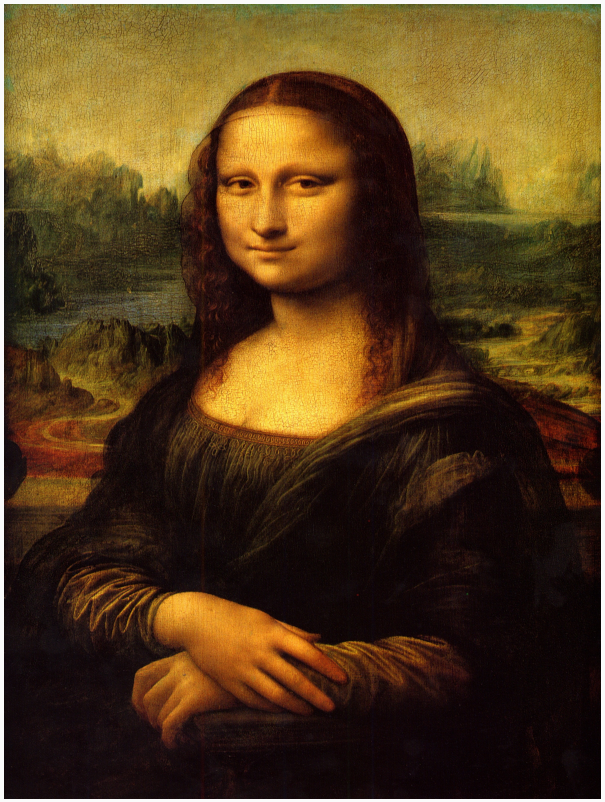
Neural Style Transfer

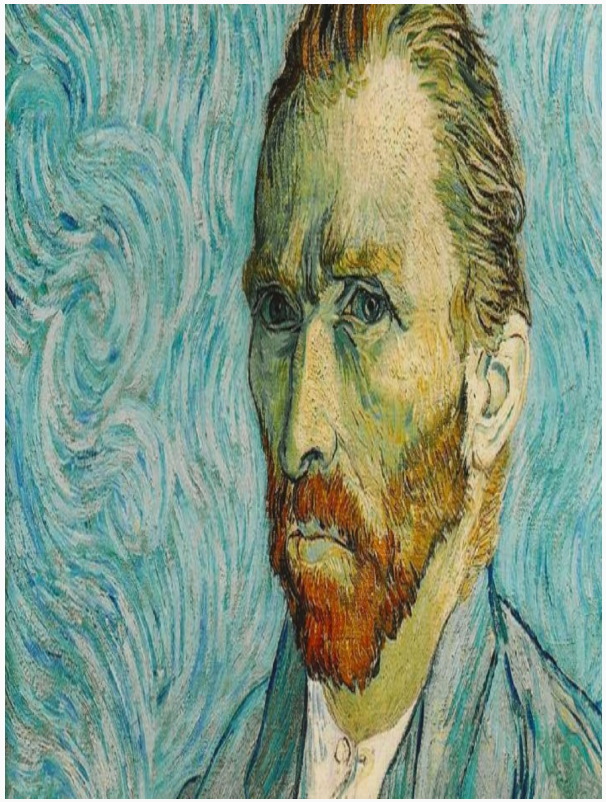


Neural Style Transfer

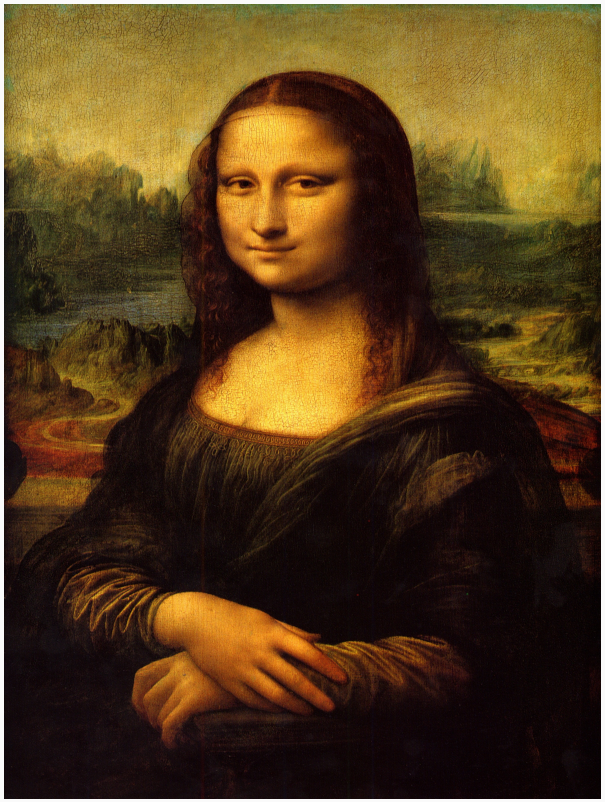


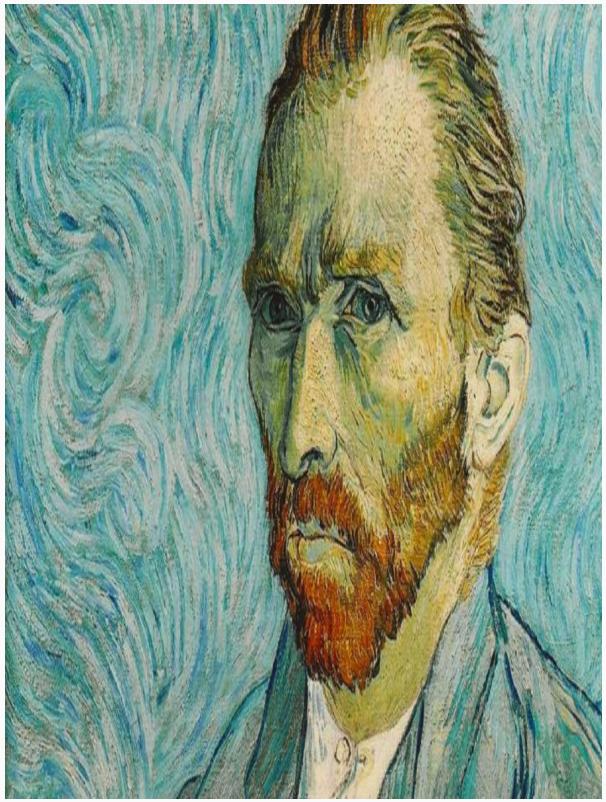
Neural Style Transfer





Neural Style Transfer

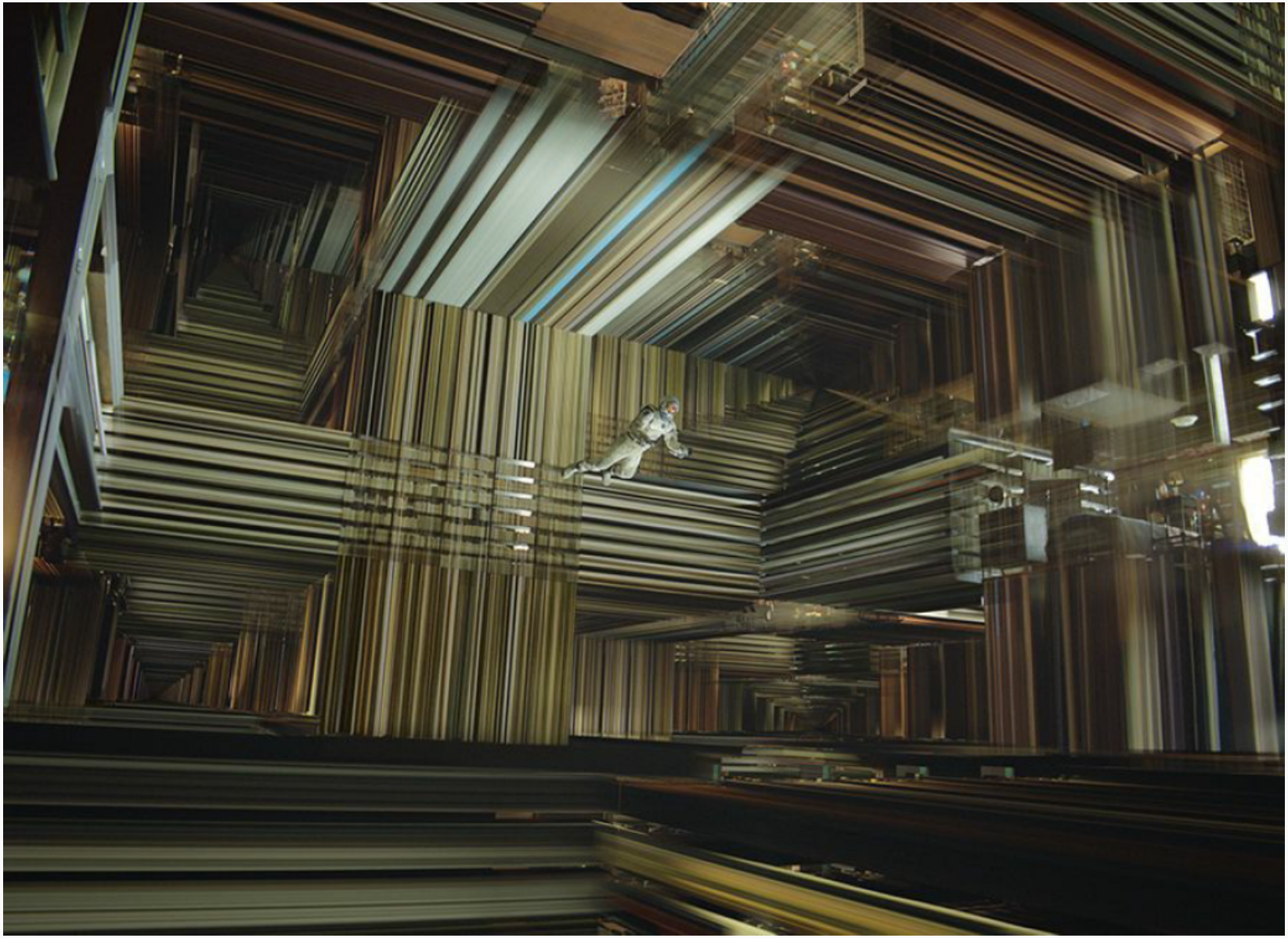




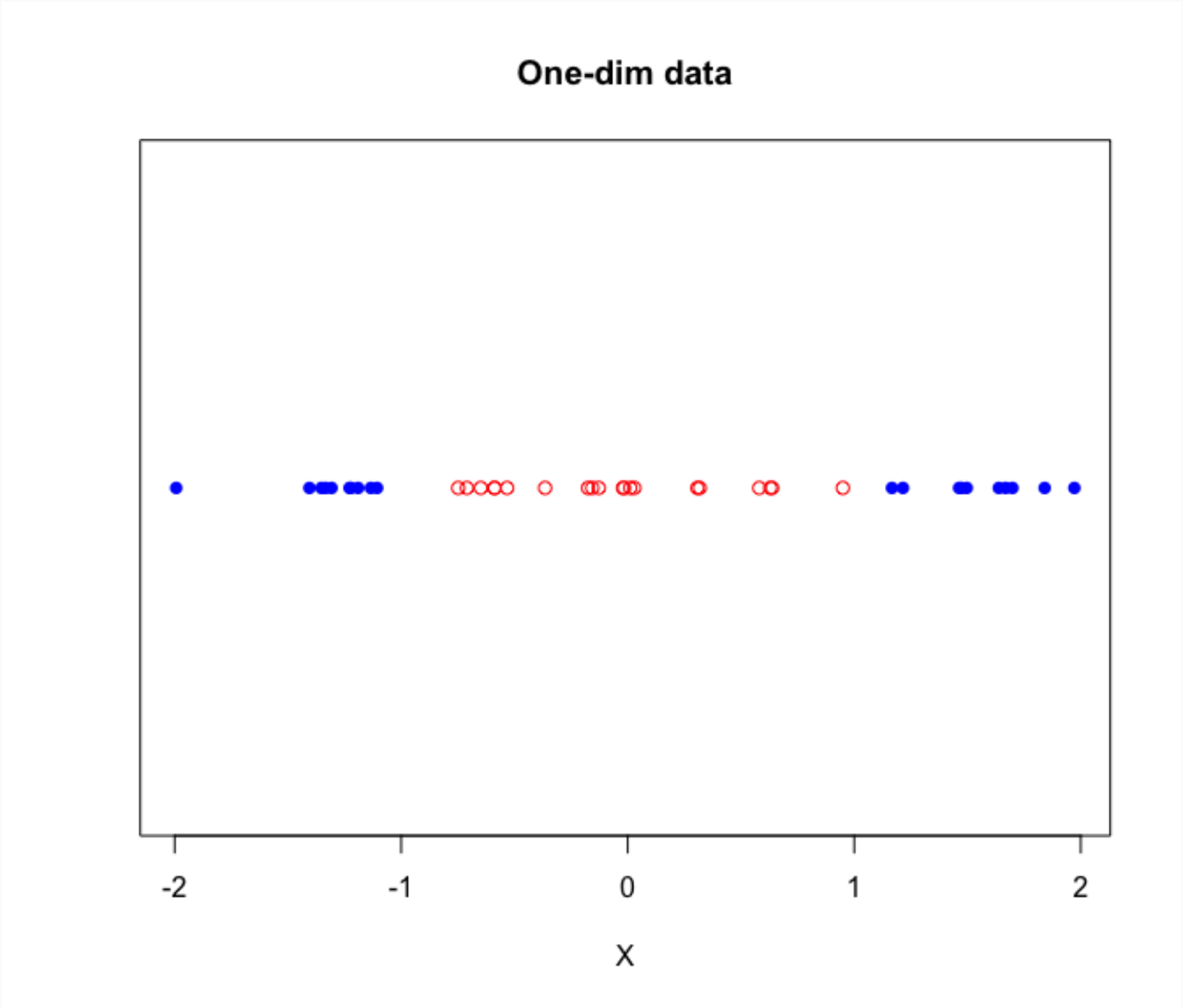
Coming Soon (Not Really)...

Challenges of High-dimension

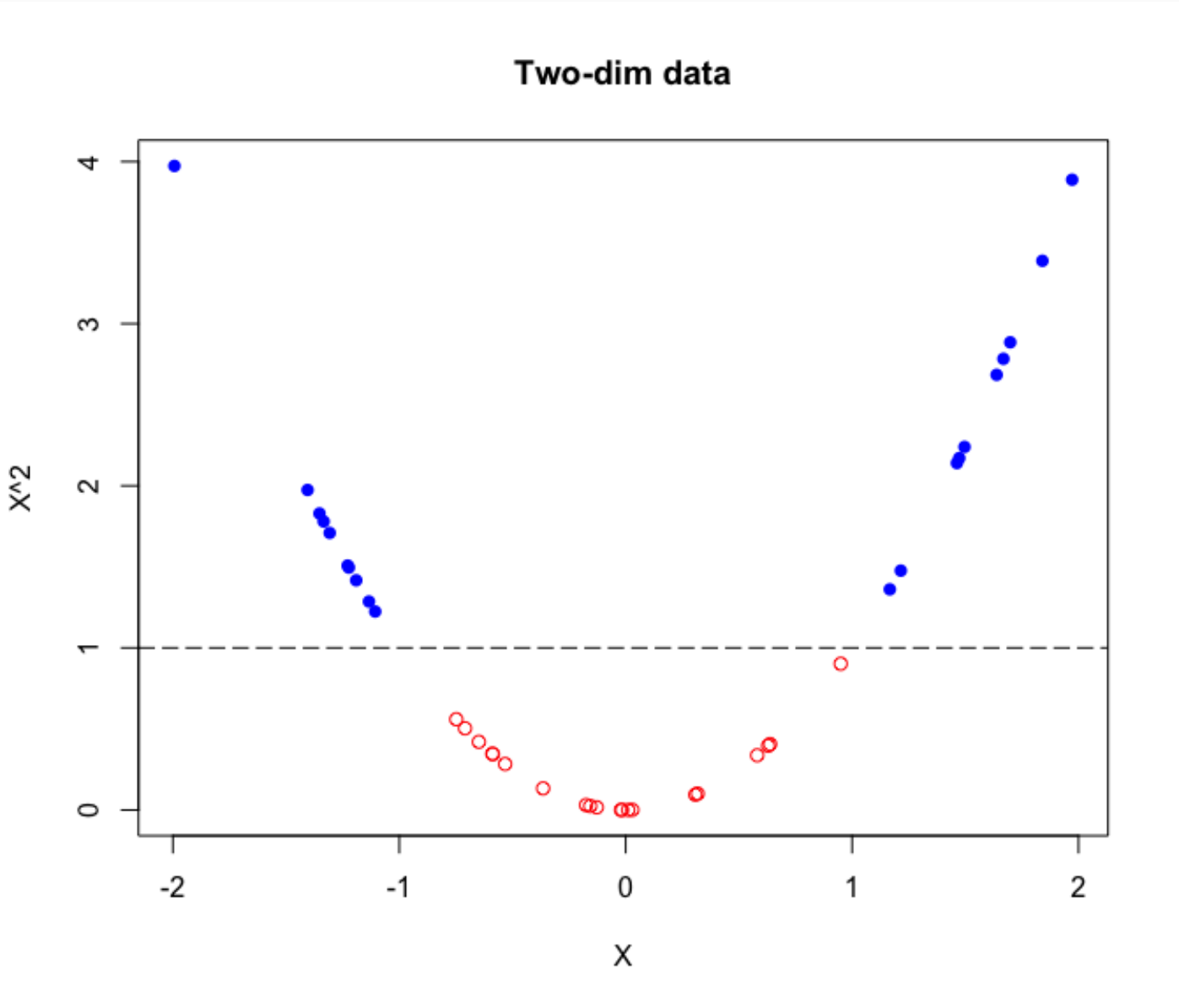
Extra Dimensions



Can you separate the one-dimensional data?



Can you separate the two-dimensional data?



"Big" on p

- 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

- In linear regression, the curse of dimensionality is easy to see from the normal equation:

$$\widehat{\beta} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$$

  where $\mathbf{X}_{n\times p}$ is the design matrix, $\mathbf{y}_{n\times 1}$ is the response vector.

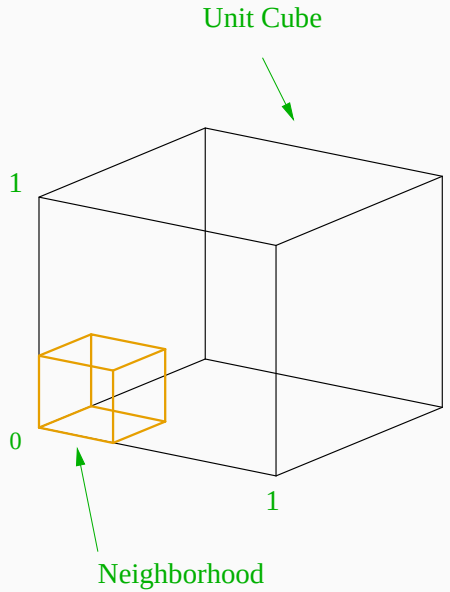
- $\mathbf{X}^\mathsf{T}\mathbf{X}$ is not invertible when $p > n$, hence $\widehat{\beta}$ is not unique

- In fact, since $\mathbf{X}$ is of rank n, we can always find some $\widehat{\beta}$ such that $\mathbf{y} = \mathbf{X}\widehat{\beta}$, this means we are modeling the error term ϵ!

- Some penalized methods can deal with this problem

Curse of Dimensionality

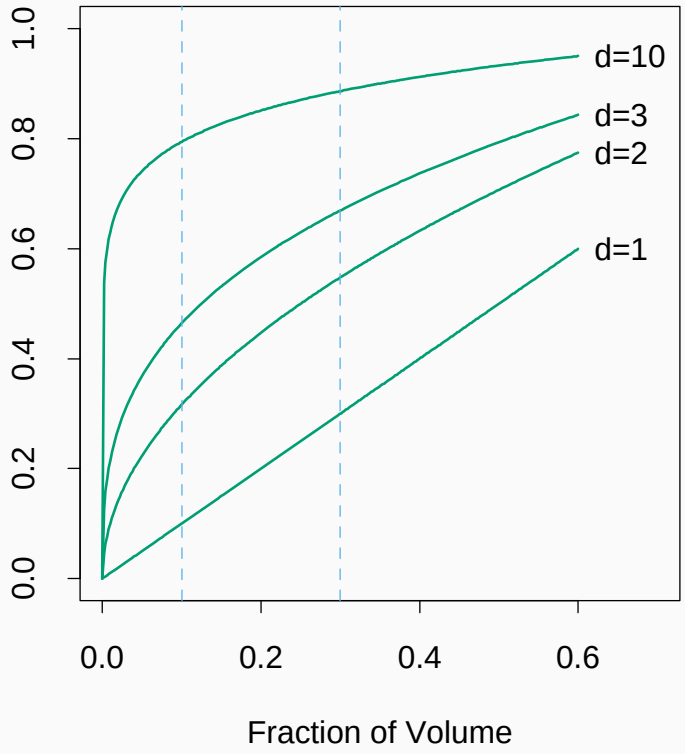
- The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube.
- Suppose the sample points are evenly spread out on $[0, 1]^p$, and we want to capture 10% of the data by constructing a hypercube neighborhood of x. What is the edge length l of this cube? Since the volume of the cube is $l^p = 10\%$, we need $l = 0.1^{1/p}$,



- When $p = 1$, l = 0.1
- When $p = 2$, l = 0.32
- When $p = 10$, l = 0.79

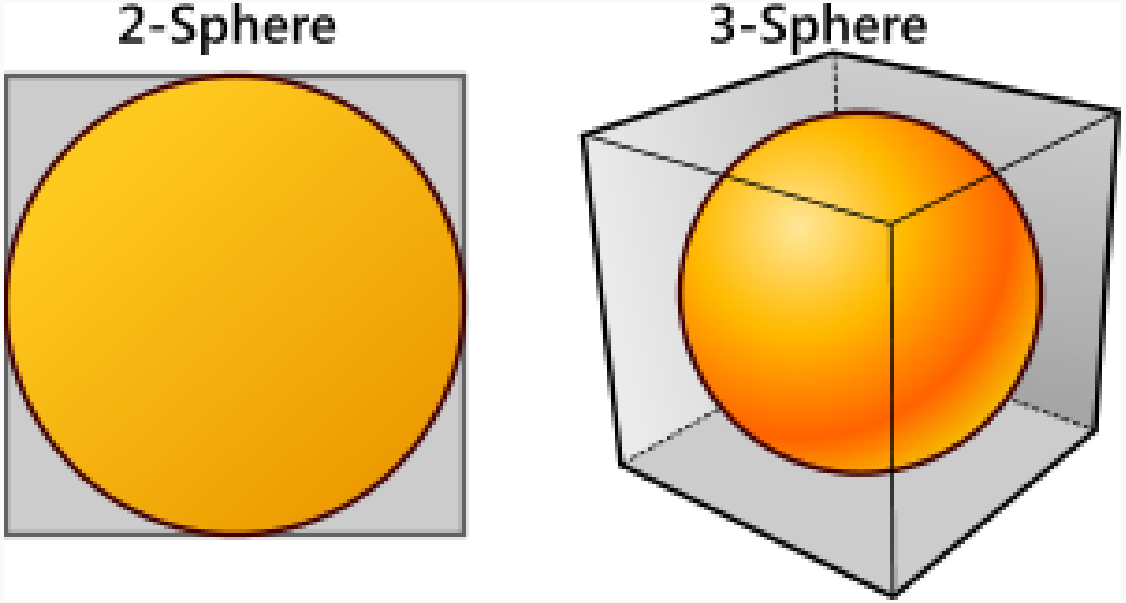
Curse of Dimensionality

- 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.



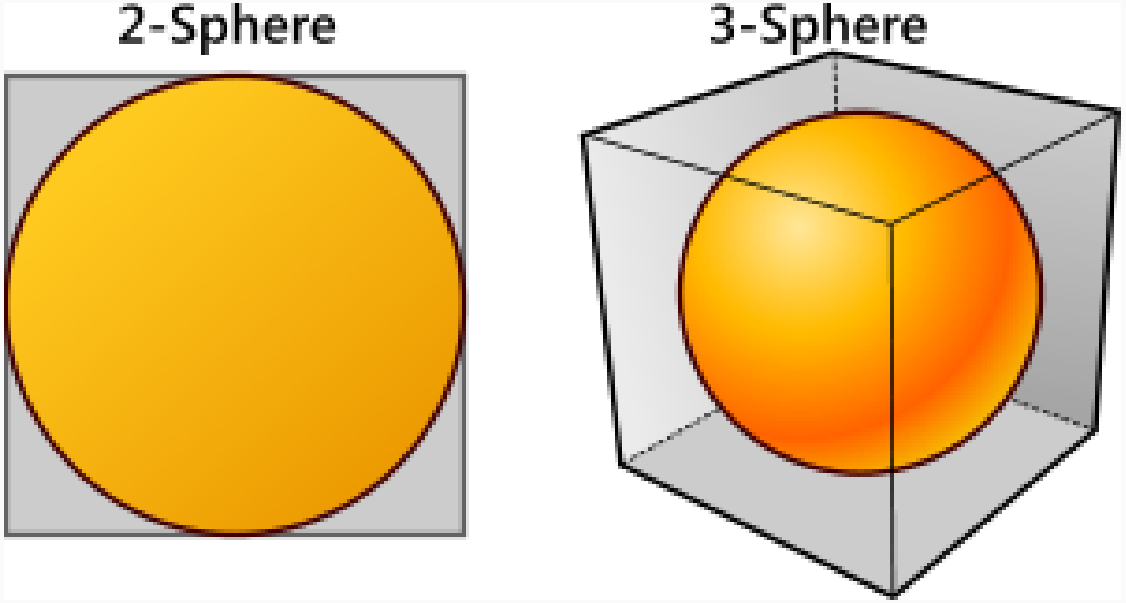
Curse of Dimensionality

- Hypersphere and Hypercubic in a d-dimensional space



Curse of Dimensionality

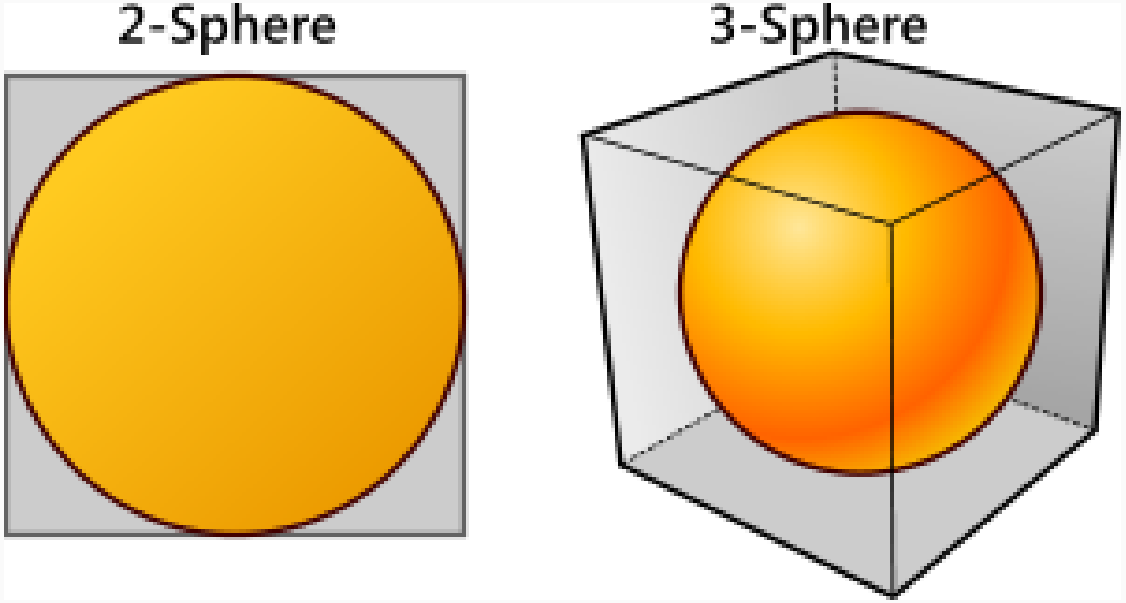
- Hypersphere and Hypercubic in a d-dimensional space



- Vol(Hypersphere) with a radius of r is $V_s = \frac{2r^d \pi^{d/2}}{d\Gamma(d/2)}$

Curse of Dimensionality

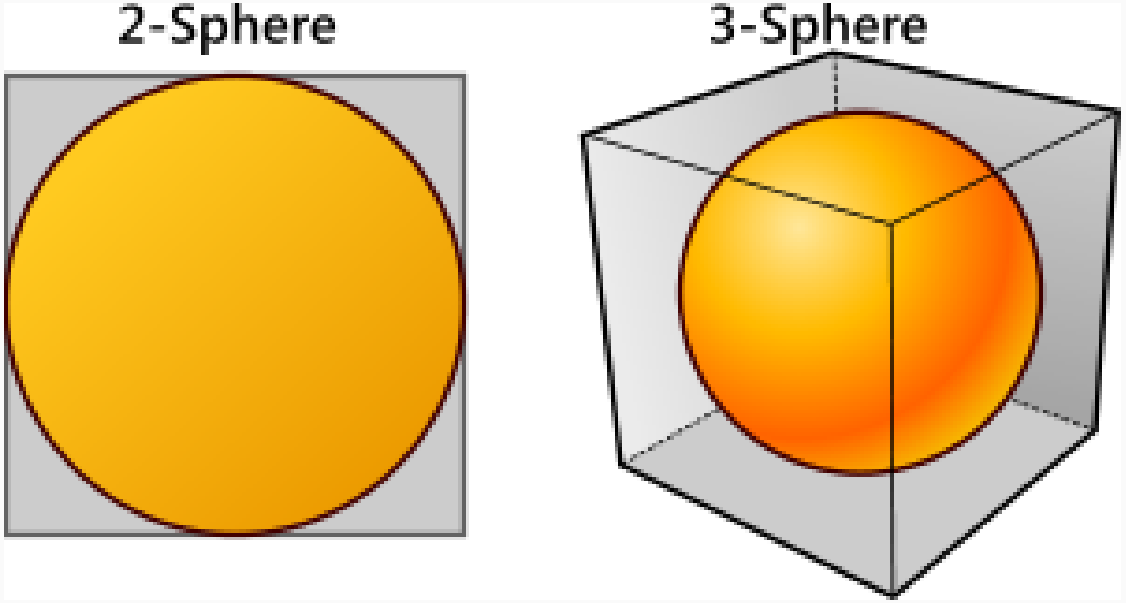
- Hypersphere and Hypercubic in a d-dimensional space



- Vol(Hypersphere) with a radius of r is $V_s = \frac{2r^d \pi^{d/2}}{d\Gamma(d/2)}$
- Vol(Hypercubic) with an edge of $2r$ is $V_c = (2r)^d$

Curse of Dimensionality

- Hypersphere and Hypercubic in a d-dimensional space



- Vol(Hypersphere) with a radius of r is $V_s = \frac{2r^d \pi^{d/2}}{d\Gamma(d/2)}$
- Vol(Hypercubic) with an edge of $2r$ is $V_c = (2r)^d$
- $\frac{V_s}{V_c} = \frac{\pi^{d/2}}{2^{d-1} d\Gamma(d/2)} \to 0$ as $d \to \infty$: Vastness
- Almost every data point is in the ”Corner”

Curse of Dimensionality