K-Nearest Neighbors

K Sri Rama Murty

IIT Hyderabad

ksrm@ee.iith.ac.in

K-Nearest Neighbor Classifier

K-Nearest Neighbor Classifier

- Classification is a supervised learning problem

K-Nearest Neighbor Classifier

- Classification is a supervised learning problem
- In classification, the target variable admits a discrete set of values.
  eg. [(x_1=6 ft, x_2=65 kg), y=A], [(x_1=3.4 ft, x_2=18 kg), y=K]

K-Nearest Neighbor Classifier

- Classification is a supervised learning problem
- In classification, the target variable admits a discrete set of values. eg. [(x_1=6 ft, x_2=65 kg), y=A], [(x_1=3.4 ft, x_2=18 kg), y=K]
- Given a set of labeled data $[\mathbf{x}_n, y_n]$, $n = 1, 2, \cdots, N$ for training, classify a new instance $\mathbf{x}_0$

K-Nearest Neighbor Classifier

- Classification is a supervised learning problem
- In classification, the target variable admits a discrete set of values.
  eg. [(x_1=6 ft, x_2=65 kg), y=A], [(x_1=3.4 ft, x_2=18 kg), y=K]
- Given a set of labeled data $[\mathbf{x}_n, y_n]$, $n = 1, 2, \cdots, N$ for training, classify a new instance $\mathbf{x}_0$
- KNN classification algorithm

K-Nearest Neighbor Classifier

- Classification is a supervised learning problem
- In classification, the target variable admits a discrete set of values. eg. [(x_1=6 ft, x_2=65 kg), y=A], [(x_1=3.4 ft, x_2=18 kg), y=K]
- Given a set of labeled data $[\mathbf{x}_n, y_n]$, $n = 1, 2, \cdots, N$ for training, classify a new instance $\mathbf{x}_0$
- KNN classification algorithm
  - Evaluate distance of $\mathbf{x}_0$ to all points $\mathbf{x}_n$, $n = 1, 2, \cdots N$ in training set

$$d_n = ||\mathbf{x}_n - \mathbf{x}_0||_p \qquad\qquad d_n = 1 - \frac{\mathbf{x}_n^T \mathbf{x}_0}{||\mathbf{x}_n|| \;\; ||\mathbf{x}_0||}$$

K-Nearest Neighbor Classifier

- Classification is a supervised learning problem
- In classification, the target variable admits a discrete set of values. eg. [(x_1=6 ft, x_2=65 kg), y=A], [(x_1=3.4 ft, x_2=18 kg), y=K]
- Given a set of labeled data $[\mathbf{x}_n, y_n]$, $n = 1, 2, \cdots, N$ for training, classify a new instance $\mathbf{x}_0$
- KNN classification algorithm
  - Evaluate distance of $\mathbf{x}_0$ to all points $\mathbf{x}_n$, $n = 1, 2, \cdots N$ in training set

$$d_n = ||\mathbf{x}_n - \mathbf{x}_0||_p \qquad\qquad d_n = 1 - \frac{\mathbf{x}_n^T \mathbf{x}_0}{||\mathbf{x}_n|| \;\; ||\mathbf{x}_0||}$$

  - Sort the distances and identify K nearest neighbors - X_{NN}

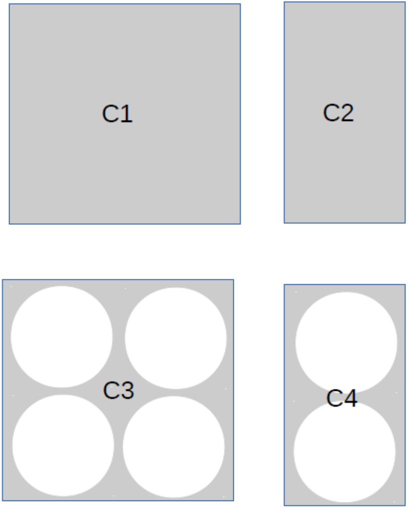
K-Nearest Neighbor Classifier

- Classification is a supervised learning problem
- In classification, the target variable admits a discrete set of values. eg. [(x_1=6 ft, x_2=65 kg), y=A], [(x_1=3.4 ft, x_2=18 kg), y=K]
- Given a set of labeled data $[\mathbf{x}_n, y_n]$, $n = 1, 2, \cdots, N$ for training, classify a new instance $\mathbf{x}_0$
- KNN classification algorithm
  - Evaluate distance of $\mathbf{x}_0$ to all points $\mathbf{x}_n$, $n = 1, 2, \cdots N$ in training set

$$d_n = ||\mathbf{x}_n - \mathbf{x}_0||_p \qquad d_n = 1 - \frac{\mathbf{x}_n^\mathsf{T}\mathbf{x}_0}{||\mathbf{x}_n|| \ \ ||\mathbf{x}_0||}$$

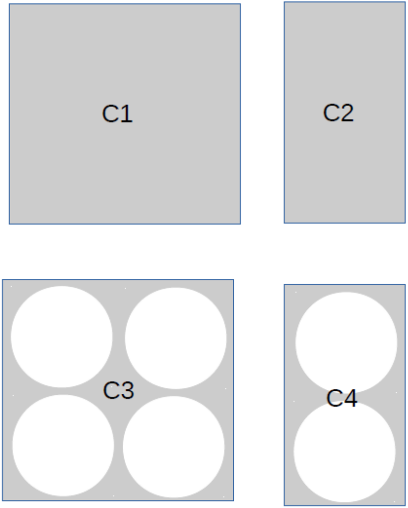
  - Sort the distances and identify K nearest neighbors - X_{NN}
  - Assign output label y_0 for test point $\mathbf{x}_0$ from labels of y_{NN} - Voting

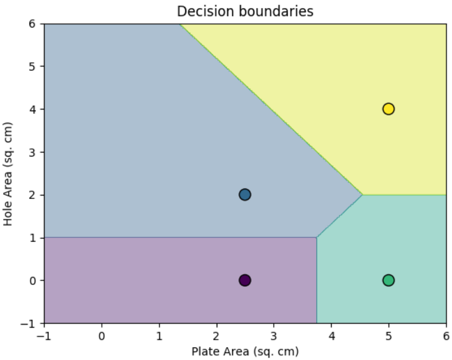
Geometry of KNN Decision Boundaries (k=1)



Steel-plate classification

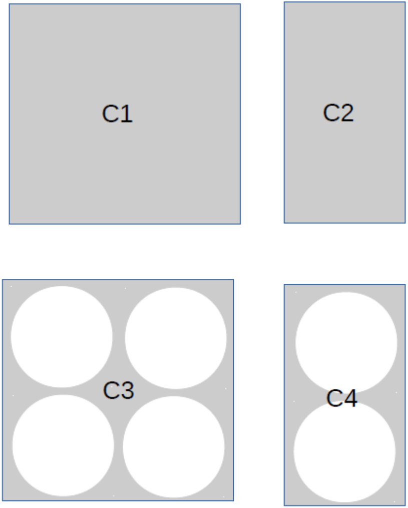
Geometry of KNN Decision Boundaries (k=1)



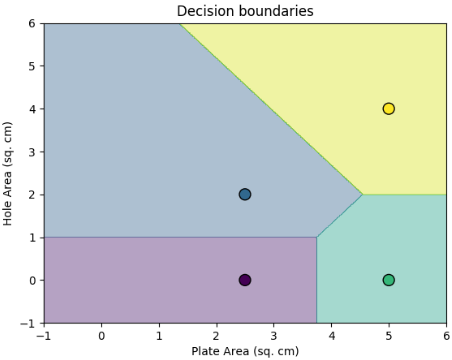


Steel-plate classification

Geometry of KNN Decision Boundaries (k=1)

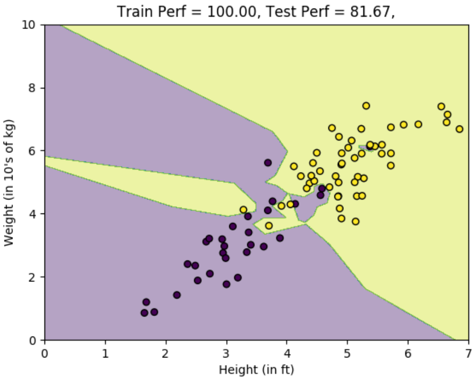


Steel-plate classification



Decision boundaries are perpendicular bisectors of the lines joining nearest competitors!

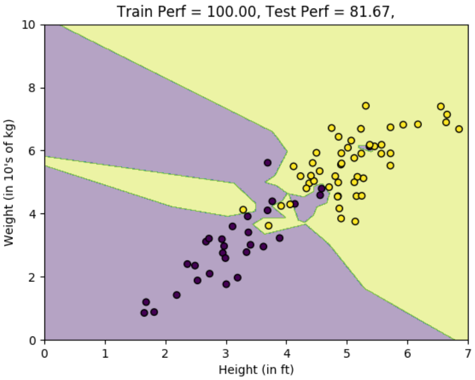
KNN Classifier (Kid vs Adult) $k = 1$



KNN Classifier (Kid vs Adult) $k = 1$

- 50 points from adult-class and 30 points from kid-class

KNN Classifier (Kid vs Adult) $k = 1$

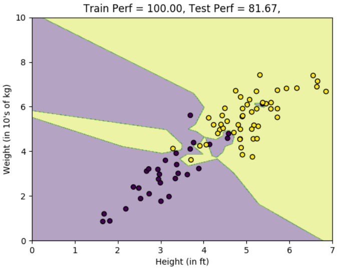


- 50 points from adult-class and 30 points from kid-class
- Big difference between train and test performances ($k = 1$ is a overfit)

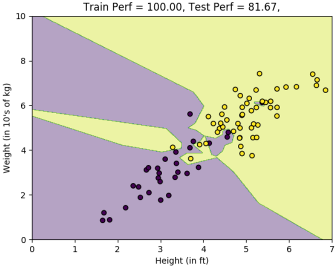
KNN Classifier (Kid vs Adult) $k = 1$

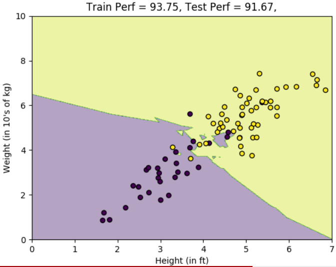
- 50 points from adult-class and 30 points from kid-class
- Big difference between train and test performances ($k = 1$ is a overfit)
- In a KNN-classifier, the decision regions are typically nonconvex

Effect of K on Performance

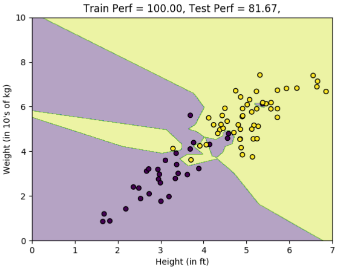


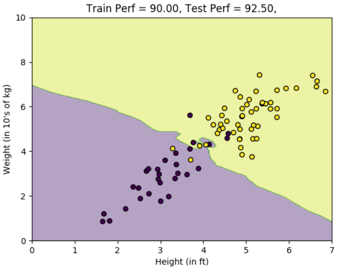
Effect of K on Performance

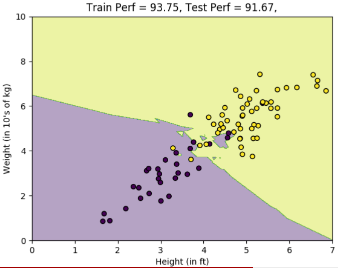




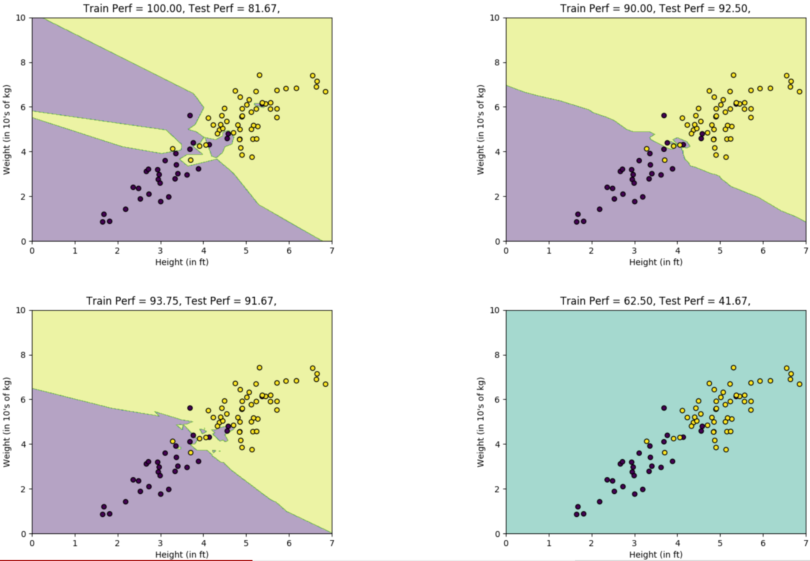
Effect of K on Performance



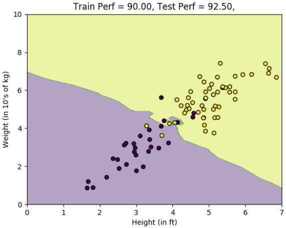




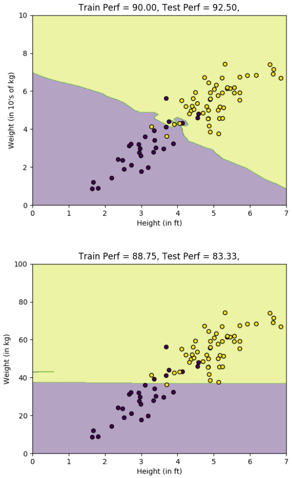
Effect of K on Performance



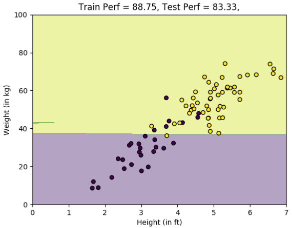
Why not Weight in kg



Why not Weight in kg



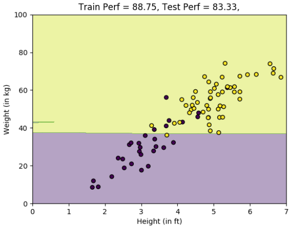
Why not Weight in kg



- Distance between two points **x** and **y** is

$$d_{xy} = \sqrt{(x_h - y_h)^2 + (x_w - y_w)^2}$$

Why not Weight in kg

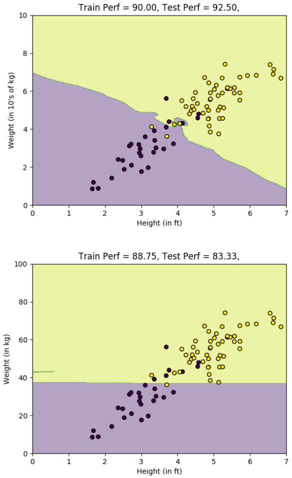


- Distance between two points **x** and **y** is

$$d_{xy} = \sqrt{(x_h - y_h)^2 + (x_w - y_w)^2}$$

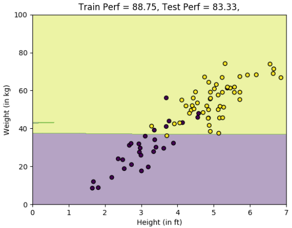
- Contribution of a dimension to distance is proportional to its variance

Why not Weight in kg



- Distance between two points **x** and **y** is
$$d_{xy} = \sqrt{(x_h - y_h)^2 + (x_w - y_w)^2}$$
- Contribution of a dimension to distance is proportional to its variance
- When weight is in kg, distance is dominated by it (horizontal boundary)

Why not Weight in kg



- Distance between two points **x** and **y** is

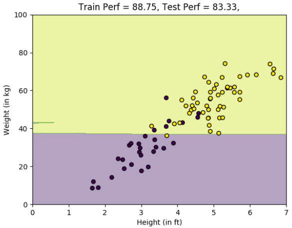
$$d_{xy} = \sqrt{(x_h - y_h)^2 + (x_w - y_w)^2}$$

- Contribution of a dimension to distance is proportional to its variance
- When weight is in kg, distance is dominated by it (horizontal boundary)
- Normalize each dim. to unit variance

$$\tilde{x}_w = \frac{x_w - \mu_w}{\sigma_w}$$

$$\sigma_w^2 = \frac{1}{N}\sum_{n=1}^{N}(x_{nw} - \mu_w)^2 \quad \mu_w = \frac{1}{N}\sum_{n=1}^{N} x_{nw}$$

Why not Weight in kg



- Distance between two points $\mathbf{x}$ and $\mathbf{y}$ is

$$d_{xy} = \sqrt{(x_h - y_h)^2 + (x_w - y_w)^2}$$

- Contribution of a dimension to distance is proportional to its variance
- When weight is in kg, distance is dominated by it (horizontal boundary)
- Normalize each dim. to unit variance

$$\tilde{x}_w = \frac{x_w - \mu_w}{\sigma_w}$$

$$\sigma_w^2 = \frac{1}{N}\sum_{n=1}^{N}(x_{nw} - \mu_w)^2 \quad \mu_w = \frac{1}{N}\sum_{n=1}^{N} x_{nw}$$

- This process is called *data whitening*

k-Nearest Neighbor (k-NN) Regression

k-Nearest Neighbor (k-NN) Regression

- Regression is a supervised learning problem

k-Nearest Neighbor (k-NN) Regression

- Regression is a supervised learning problem
- In regression, the target variable admits continuous values

S.No	1	2	3	4	5	6	7	8	9	10	t
x	5.2	4.8	5.8	5.7	5.4	5.1	4.9	5.3	5.9	4.5	5.6
y	58	53	63	65	60	50	52	55	61	49	?

k-Nearest Neighbor (k-NN) Regression

- Regression is a supervised learning problem
- In regression, the target variable admits continuous values

S.No	1	2	3	4	5	6	7	8	9	10	t
x	5.2	4.8	5.8	5.7	5.4	5.1	4.9	5.3	5.9	4.5	5.6
y	58	53	63	65	60	50	52	55	61	49	?

- Average of the k-nearest neighbors in the training data.

k-Nearest Neighbor (k-NN) Regression

- Regression is a supervised learning problem
- In regression, the target variable admits continuous values

S.No	1	2	3	4	5	6	7	8	9	10	t
x	5.2	4.8	5.8	5.7	5.4	5.1	4.9	5.3	5.9	4.5	5.6
y	58	53	63	65	60	50	52	55	61	49	?

- Average of the k-nearest neighbors in the training data.
- $(k = 1)$ NN - $\{(5.7, 65)\}$: $y_{11} = 65$ (overfit - high variance)

k-Nearest Neighbor (k-NN) Regression

- Regression is a supervised learning problem
- In regression, the target variable admits continuous values

S.No	1	2	3	4	5	6	7	8	9	10	t
x	5.2	4.8	5.8	5.7	5.4	5.1	4.9	5.3	5.9	4.5	5.6
y	58	53	63	65	60	50	52	55	61	49	?

- Average of the k-nearest neighbors in the training data.
- ($k = 1$) NN - $\{(5.7, 65)\}$: $y_{11} = 65$ (overfit - high variance)
- ($k = 3$) NN - $\{(5.7, 65), (5.8, 63), (5.4, 60)\}$: $y_{11} = 62.66$

k-Nearest Neighbor (k-NN) Regression

- Regression is a supervised learning problem
- In regression, the target variable admits continuous values

S.No	1	2	3	4	5	6	7	8	9	10	t
x	5.2	4.8	5.8	5.7	5.4	5.1	4.9	5.3	5.9	4.5	5.6
y	58	53	63	65	60	50	52	55	61	49	?

- Average of the k-nearest neighbors in the training data.
- ($k = 1$) NN - $\{(5.7, 65)\}$: $y_{11} = 65$ (overfit - high variance)
- ($k = 3$) NN - $\{(5.7, 65), (5.8, 63), (5.4, 60)\}$: $y_{11} = 62.66$
- ($k = 10$) assigns average weight irrespective of test height (high bias)

k-Nearest Neighbor (k-NN) Regression

- Regression is a supervised learning problem
- In regression, the target variable admits continuous values

S.No	1	2	3	4	5	6	7	8	9	10	t
x	5.2	4.8	5.8	5.7	5.4	5.1	4.9	5.3	5.9	4.5	5.6
y	58	53	63	65	60	50	52	55	61	49	?

- Average of the k-nearest neighbors in the training data.
- ($k = 1$) NN - $\{(5.7, 65)\}$: $y_{11} = 65$ (overfit - high variance)
- ($k = 3$) NN - $\{(5.7, 65), (5.8, 63), (5.4, 60)\}$: $y_{11} = 62.66$
- ($k = 10$) assigns average weight irrespective of test height (high bias)
- KNN-regression gives equal importance to all neighbors

k-Nearest Neighbor (k-NN) Regression

- Regression is a supervised learning problem
- In regression, the target variable admits continuous values

S.No	1	2	3	4	5	6	7	8	9	10	t
x	5.2	4.8	5.8	5.7	5.4	5.1	4.9	5.3	5.9	4.5	5.6
y	58	53	63	65	60	50	52	55	61	49	?

- Average of the k-nearest neighbors in the training data.
- ($k = 1$) NN - $\{(5.7, 65)\}$: $y_{11} = 65$ (overfit - high variance)
- ($k = 3$) NN - $\{(5.7, 65), (5.8, 63), (5.4, 60)\}$: $y_{11} = 62.66$
- ($k = 10$) assigns average weight irrespective of test height (high bias)
- KNN-regression gives equal importance to all neighbors
- Assign higher weightage to closer neighbors

Inverse Distance Weighted k-NN

Inverse Distance Weighted k-NN

- Weighted k-NN regression: the unknown quantity y_t is estimated as

$$y_t = \frac{\sum_{n=0}^{N} k(x_n, x_t) y_n}{\sum_{n=0}^{N} k(x_n, x_t)}$$

Inverse Distance Weighted k-NN

- Weighted k-NN regression: the unknown quantity y_t is estimated as

$$y_t = \frac{\sum_{n=0}^{N} k(x_n, x_t) y_n}{\sum_{n=0}^{N} k(x_n, x_t)}$$

- $k(x_n, x_t)$ denotes similarity between x_n and x_t

Inverse Distance Weighted k-NN

- Weighted k-NN regression: the unknown quantity y_t is estimated as

$$y_t = \frac{\sum_{n=0}^{N} k(x_n, x_t) y_n}{\sum_{n=0}^{N} k(x_n, x_t)}$$

- $k(x_n, x_t)$ denotes similarity between x_n and x_t
- Similarity (or inverse distance) as weight to give preference to NNs

Inverse Distance Weighted k-NN

- Weighted k-NN regression: the unknown quantity y_t is estimated as

$$y_t = \frac{\sum_{n=0}^{N} k(x_n, x_t) y_n}{\sum_{n=0}^{N} k(x_n, x_t)}$$

- $k(x_n, x_t)$ denotes similarity between x_n and x_t
- Similarity (or inverse distance) as weight to give preference to NNs
- $(k = 3)$ NN - $\{(10, 65), (5, 63), (5, 60)\}$: $y_{11} = 63.25$

Inverse Distance Weighted k-NN

- Weighted k-NN regression: the unknown quantity y_t is estimated as

$$y_t = \frac{\sum_{n=0}^{N} k(x_n, x_t) y_n}{\sum_{n=0}^{N} k(x_n, x_t)}$$

- $k(x_n, x_t)$ denotes similarity between x_n and x_t
- Similarity (or inverse distance) as weight to give preference to NNs
- $(k = 3)$ NN - $\{(10, 65), (5, 63), (5, 60)\}$: $y_{11} = 63.25$
- Number of neighbors k is not very critical.

Inverse Distance Weighted k-NN

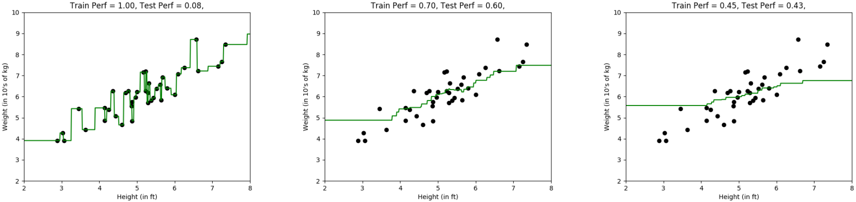
- Weighted k-NN regression: the unknown quantity y_t is estimated as

$$y_t = \frac{\sum_{n=0}^{N} k(x_n, x_t) y_n}{\sum_{n=0}^{N} k(x_n, x_t)}$$

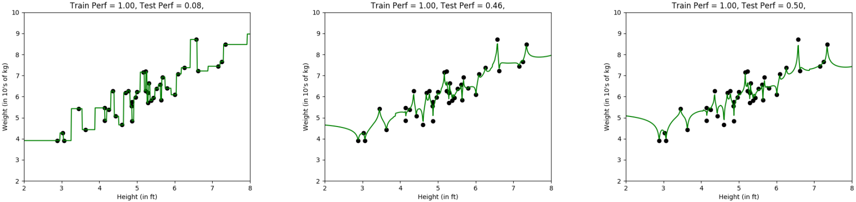
- $k(x_n, x_t)$ denotes similarity between x_n and x_t
- Similarity (or inverse distance) as weight to give preference to NNs
- $(k = 3)$ NN - $\{(10, 65), (5, 63), (5, 60)\}$: $y_{11} = 63.25$
- Number of neighbors k is not very critical.
- Similarity measure $k(x_n, x_t)$ is an important design choice

KNN-Regression Illustration

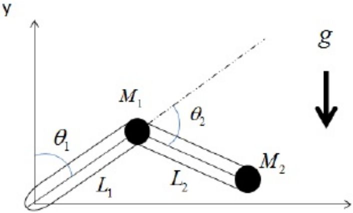
Uniform weights for the neighbors



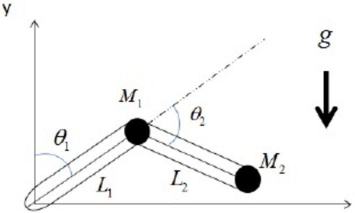
Inverse distance wights for the neighbors



Robotic Arm Control - ME Approach



Robotic Arm Control - ME Approach

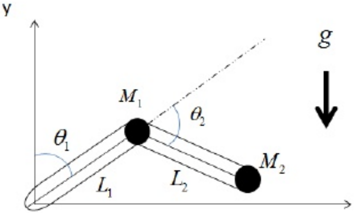


Solving for the forces:

$$F_{\theta_1} = ((M_1+M_2)L_1^2 + M_2L_2^2 + 2M_2L_1L_2cos\theta_2)\ddot{\theta}_1$$
$$+(M_2L_2^2 + M_2L_1L_2cos\theta_2)\ddot{\theta}_2 - M_2L_1L_2sin\theta_2(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2)$$
$$-(M_1+M_2)gL_1sin\theta_1 - M_2gL_2sin(\theta_1+\theta_2)$$

$$F_{\theta_2} = (M_2L_2^2 + M_2L_1L_2cos\theta_2)\ddot{\theta}_1 + M_2L_2^2\ddot{\theta}_2$$
$$-M_2L_1L_2sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 - M_2gL_2sin(\theta_1+\theta_2)$$

Robotic Arm Control - ME Approach



Solving for the forces:

$$F_{\theta_1} = ((M_1 + M_2)L_1^2 + M_2L_2^2 + 2M_2L_1L_2cos\theta_2)\ddot{\theta}_1 + (M_2L_2^2 + M_2L_1L_2cos\theta_2)\ddot{\theta}_2 - M_2L_1L_2sin\theta_2(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) - (M_1 + M_2)gL_1sin\theta_1 - M_2gL_2sin(\theta_1 + \theta_2)$$

$$F_{\theta_2} = (M_2L_2^2 + M_2L_1L_2cos\theta_2)\ddot{\theta}_1 + M_2L_2^2\ddot{\theta}_2 - M_2L_1L_2sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 - M_2gL_2sin(\theta_1 + \theta_2)$$

- Force applied on robotic arm needs to be controlled
- Error between current and desired positions determine the forces
- F_{θ_1} and F_{θ_2} can be computing by solving Lagrangian using kinetic and potential energies
- Too complicated, and may not work
- However, we can conclude that

$$F = g(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, \ddot{\theta}_1, \ddot{\theta}_2)$$

Robotic Arm Control - KNN Approach

θ_1	θ_2	$\dot{\theta}_1$	$\dot{\theta}_2$	$\ddot{\theta}_1$	$\ddot{\theta}_2$	F_{θ_1}	F_{θ_2}

- From the current and desired positions, estimate the angles and their derivatives.
- Look for the k(=1)-nearest match in the table, and pick forces.
- If succeeded in pitching, append the current configuration to the table
- As the table size increases, the robot skill improves!

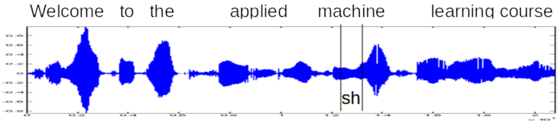
Robot Training - Testing

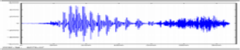
Speech Synthesis (Unit Selection)

- Regress speech waveform from the sequence of words
- Create a table of phonemes and their waveform from training data
- During synthesis, retrieve the waveforms for the sequence of phones, and concatenate them

Speech Synthesis (Unit Selection)

- Regress speech waveform from the sequence of words
- Create a table of phonemes and their waveform from training data
- During synthesis, retrieve the waveforms for the sequence of phones, and concatenate them



Phone	Prev. Phone	Next Phone	Phone position	Word Position	Duration	Waveform
/sh/	/a/	/i/	3/6	5/7	65 ms	

2h Data - 13h Data

KNN Summary

- **Pros**
  - Simple to implement
  - Flexible to feature/distance function choices (no differentiability)
  - Naturally handles multiclass data
  - Does well even on complex tasks with enough data

KNN Summary

- **Pros**
  - Simple to implement
  - Flexible to feature/distance function choices (no differentiability)
  - Naturally handles multiclass data
  - Does well even on complex tasks with enough data
- **Cons**
  - Needs to access entire training data for every inference
  - Computational complexity grows linearly with training data $O(ND)$

KNN Summary

- **Pros**
  - Simple to implement
  - Flexible to feature/distance function choices (no differentiability)
  - Naturally handles multiclass data
  - Does well even on complex tasks with enough data
- **Cons**
  - Needs to access entire training data for every inference
  - Computational complexity grows linearly with training data $O(ND)$
  - Distance function has to be chosen carefully
  - The choice of k is empirical (no theoretical guidelines)

KNN Summary

- **Pros**
  - Simple to implement
  - Flexible to feature/distance function choices (no differentiability)
  - Naturally handles multiclass data
  - Does well even on complex tasks with enough data
- **Cons**
  - Needs to access entire training data for every inference
  - Computational complexity grows linearly with training data $O(ND)$
  - Distance function has to be chosen carefully
  - The choice of k is empirical (no theoretical guidelines)
  - Nonconvex decision regions
  - Cannot naturally ignore noninformative dimensions

KNN Summary

- **Pros**
  - Simple to implement
  - Flexible to feature/distance function choices (no differentiability)
  - Naturally handles multiclass data
  - Does well even on complex tasks with enough data
- **Cons**
  - Needs to access entire training data for every inference
  - Computational complexity grows linearly with training data $O(ND)$
  - Distance function has to be chosen carefully
  - The choice of k is empirical (no theoretical guidelines)
  - Nonconvex decision regions
  - Cannot naturally ignore noninformative dimensions
  - Does not consider all the data at once, and hence, it cannot model the underlying process responsible for the observed data

Homework - Data Generation Models

- Let the distribution of human height be given by

$$X \sim \mathcal{N}(5.3, 1)$$

Let the relationship between height and weight be given by

$$Y = 10X + 5 + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 3)$. Evaluate the joint density function $p_{XY}(x, y)$, marginal density functions $p_X(x)$ & $p_Y(y)$, conditional density functions p(x/y) & p(y/x), and their corresponding expectations.

Homework

- Let the joint density function of human heights and weights follow a multivariate Gaussian given by

$$[X, Y] \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

  where $\boldsymbol{\mu} \in \mathbb{R}^2$ is the mean vector and $\Sigma \in \mathbb{R}^{2\times 2}$ is the covariance matrix. Evaluate the joint density function $p_{XY}(x, y)$, marginal density functions $p_X(x)$ & $p_Y(y)$, conditional density functions p(x/y) & p(y/x), and their corresponding expectations.
- Is it related to the earlier model?

Homework

- Let V be a 2-D random vector drawn from a standard Gaussian, i.e.,

$$\mathbf{V} \sim \mathcal{N}(\boldsymbol{O}, \mathbf{I})$$

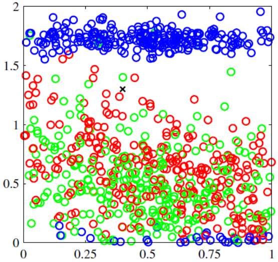
Let height and weight be obtained by an affine transformation of **V** as

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \mathbf{V} + \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$$

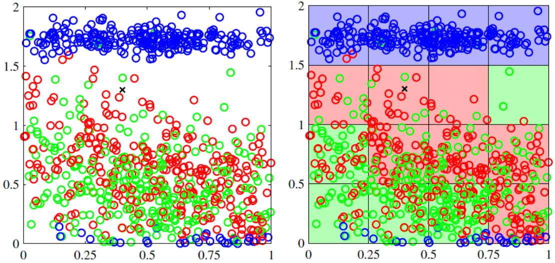
Evaluate the joint density function $p_{XY}(x, y)$, marginal density functions $p_X(x)$ & $p_Y(y)$, conditional density functions p(x/y) & p(y/x), and their corresponding expectations.

- Is it related to two earlier models?

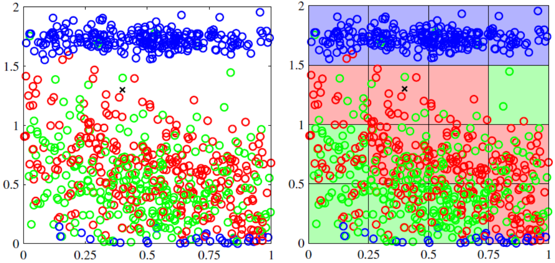
Curse of Dimensionality

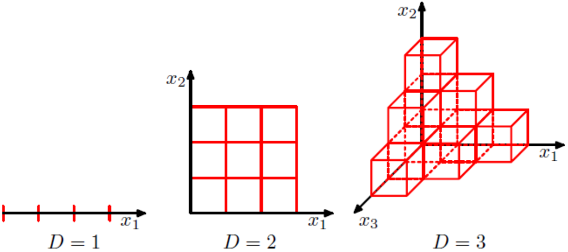


Curse of Dimensionality

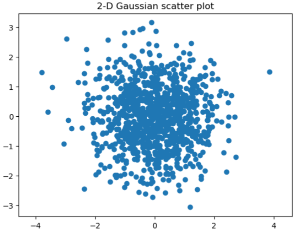


Curse of Dimensionality

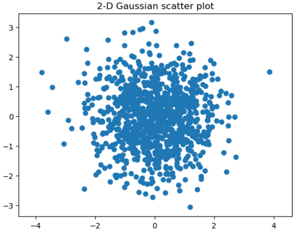


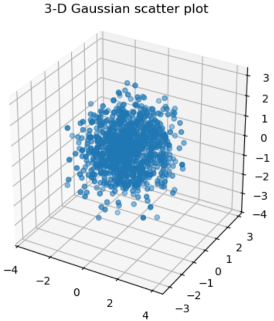


Understanding High-Dimensional Space

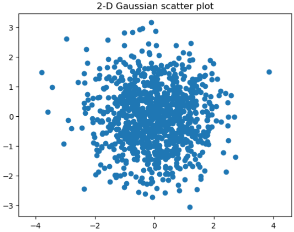


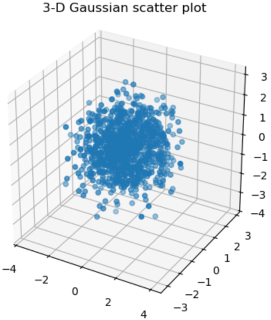
Understanding High-Dimensional Space





Understanding High-Dimensional Space

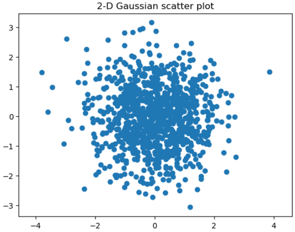


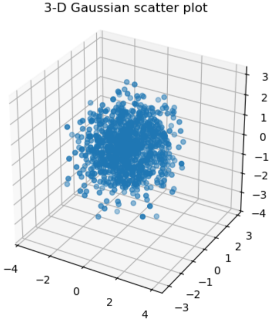


1000-D Gaussian

?

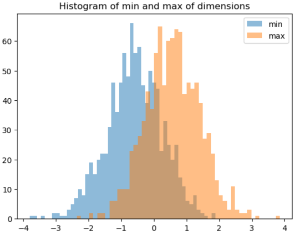
Understanding High-Dimensional Space



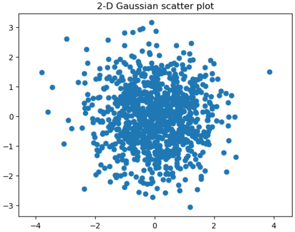


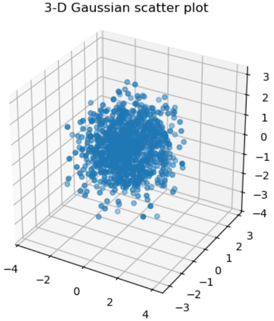
1000-D Gaussian

?



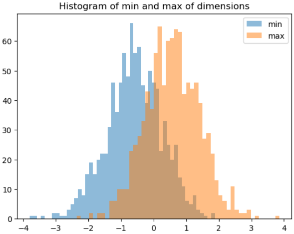
Understanding High-Dimensional Space

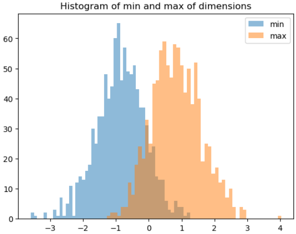




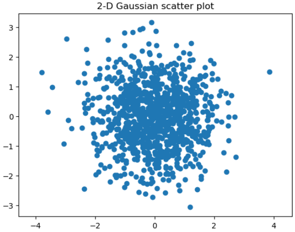
1000-D Gaussian

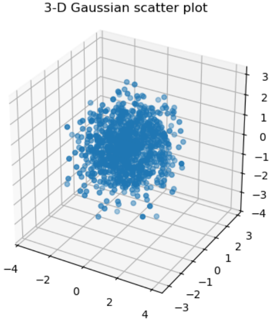
?





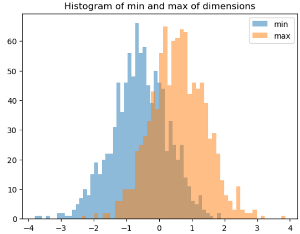
Understanding High-Dimensional Space

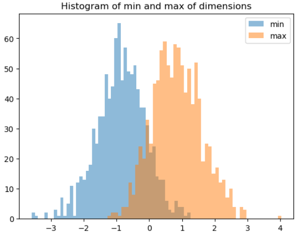


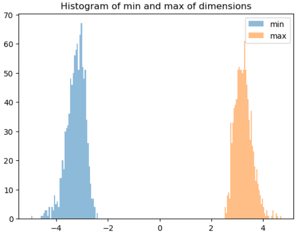


1000-D Gaussian

?







Gaussian in High Dimensional Space

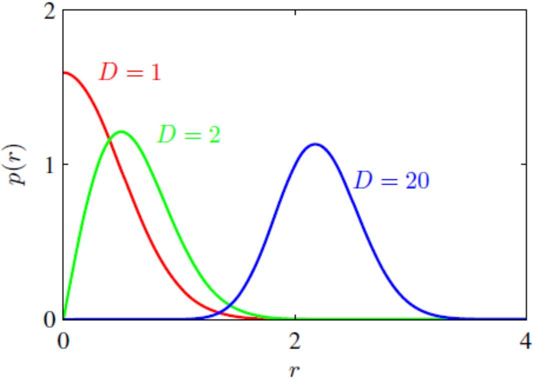
- The probability mass of a Gaussian distribution is concentrated in a thin shell

$$p(r)\Delta r = P\left[r - \frac{\Delta r}{2} < R < r + \frac{\Delta r}{2}\right]$$

Gaussian in High Dimensional Space

- The probability mass of a Gaussian distribution is concentrated in a thin shell

$$p(r)\Delta r = P\left[r - \frac{\Delta r}{2} < R < r + \frac{\Delta r}{2}\right]$$



Thank You !

Understanding High-Dimensional Space

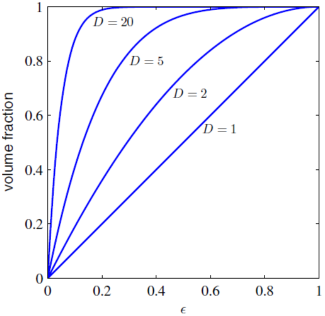
- Most of the volume of a sphere is concentrated in thin shell near the surface.

$$\frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D$$

Understanding High-Dimensional Space

- Most of the volume of a sphere is concentrated in thin shell near the surface.

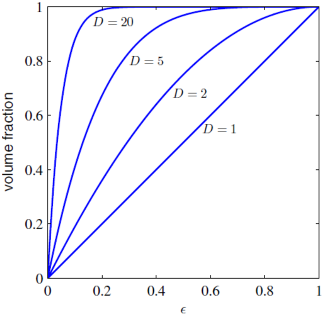
$$\frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D$$



Understanding High-Dimensional Space

- Most of the volume of a sphere is concentrated in thin shell near the surface.

$$\frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D$$



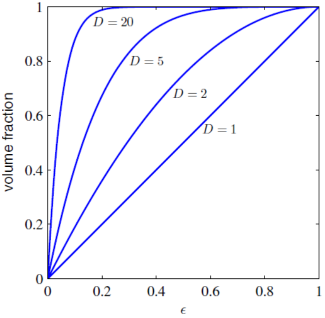
- The probability mass of a Gaussian distribution is concentrated in a thin shell

$$p(r)\Delta r = P\left[r - \frac{\Delta r}{2} < R < r + \frac{\Delta r}{2}\right]$$

Understanding High-Dimensional Space

- Most of the volume of a sphere is concentrated in thin shell near the surface.

$$\frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D$$



- The probability mass of a Gaussian distribution is concentrated in a thin shell

$$p(r)\Delta r = P\left[r - \frac{\Delta r}{2} < R < r + \frac{\Delta r}{2}\right]$$