UVA CS 4774: Machine Learning

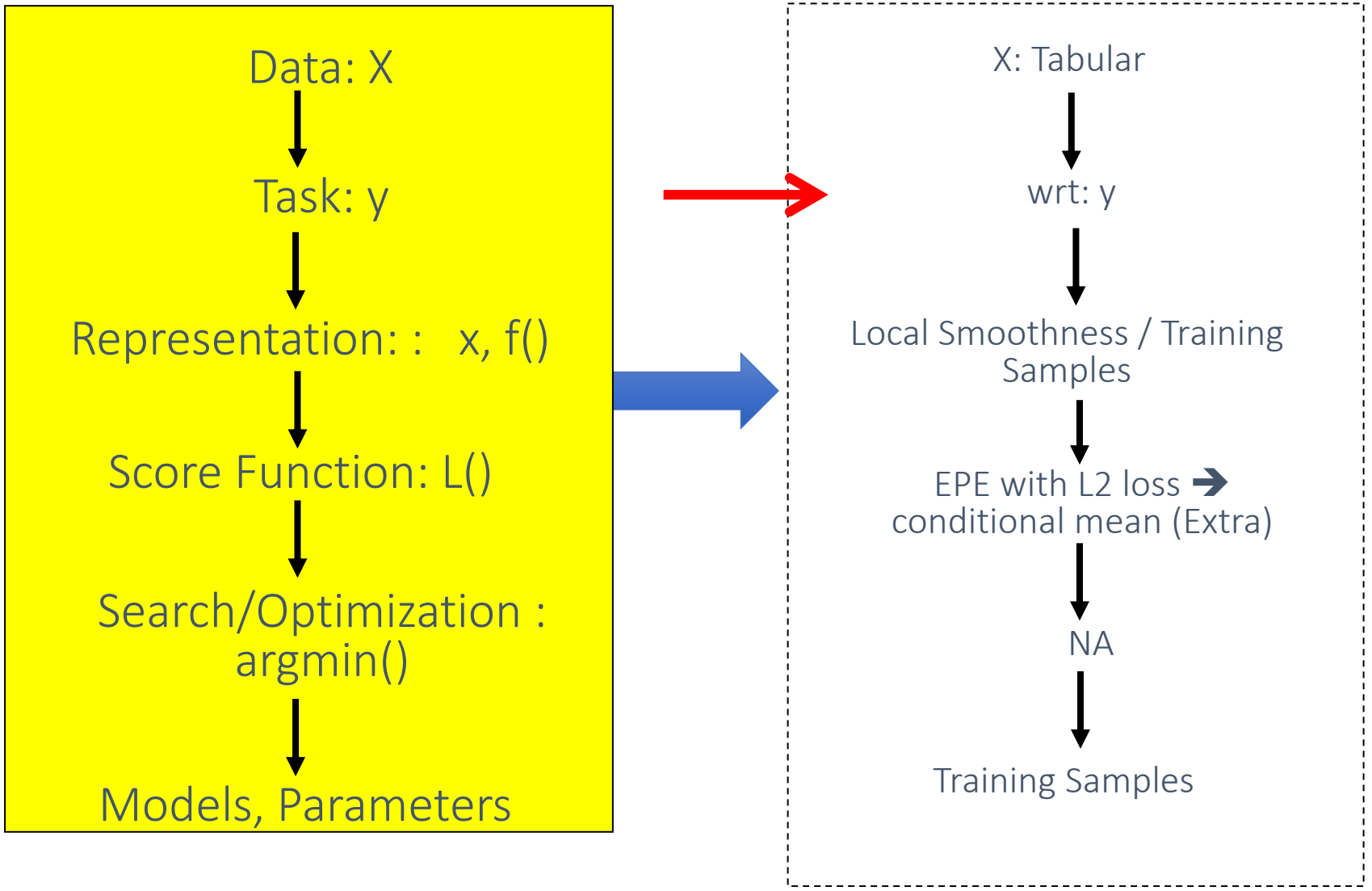
Lecture 8: K-nearest-neighbor (regressor or classifier)

Dr. Yanjun Qi

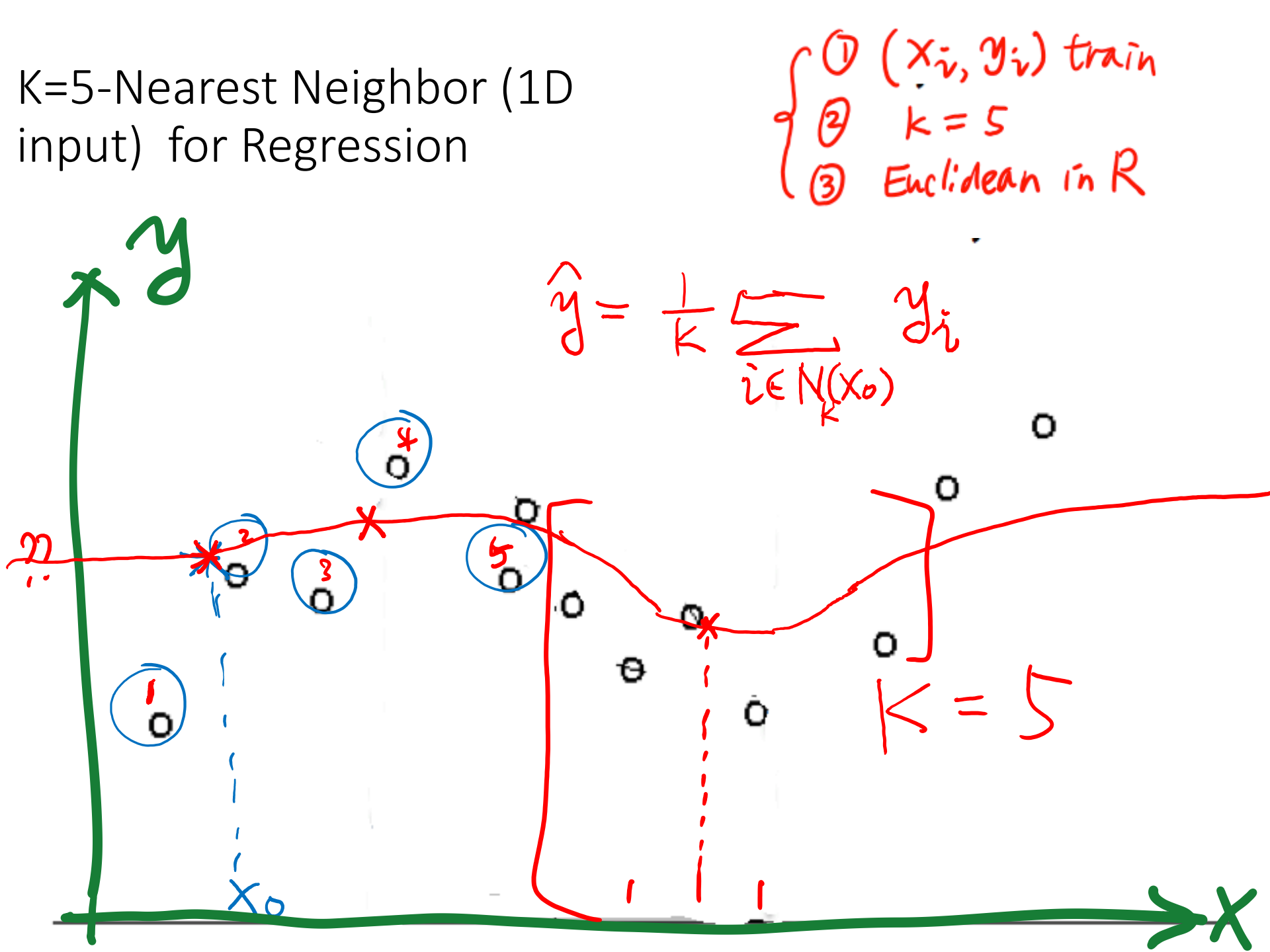
University of Virginia

Department of Computer Science

Today: **K-nearest-neighbor (regressor or classifier)**



K=5-Nearest Neighbor (1D input) for Regression

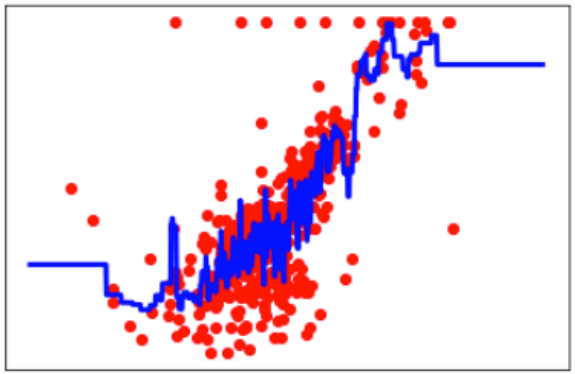


I will code run:
https://colab.research.google.com/drive/1Wzmg5ziMBMvVvLHx_0C4u0x7syF8jPT9?usp=sharing

```
[42] # import regressor
     from sklearn.neighbors import KNeighborsRegressor
     # instantiate with K=5
     knn = KNeighborsRegressor(n_neighbors=5)
     # fit with data
     knn.fit(X, y)
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=3
                    metric_params=None, n_jobs=No
                    weights='uniform')
```

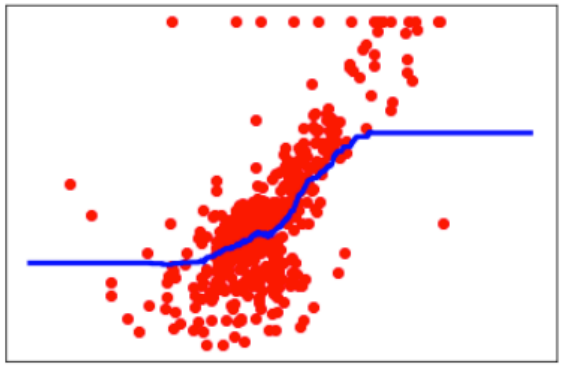
```
###
Xfit = np.linspace(3, 10, 1000).reshape(-1, 1)
yfit =knn.predict(Xfit)

# Plot outputs
plt.scatter(X, y,  color='red')
plt.plot(Xfit, yfit, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```



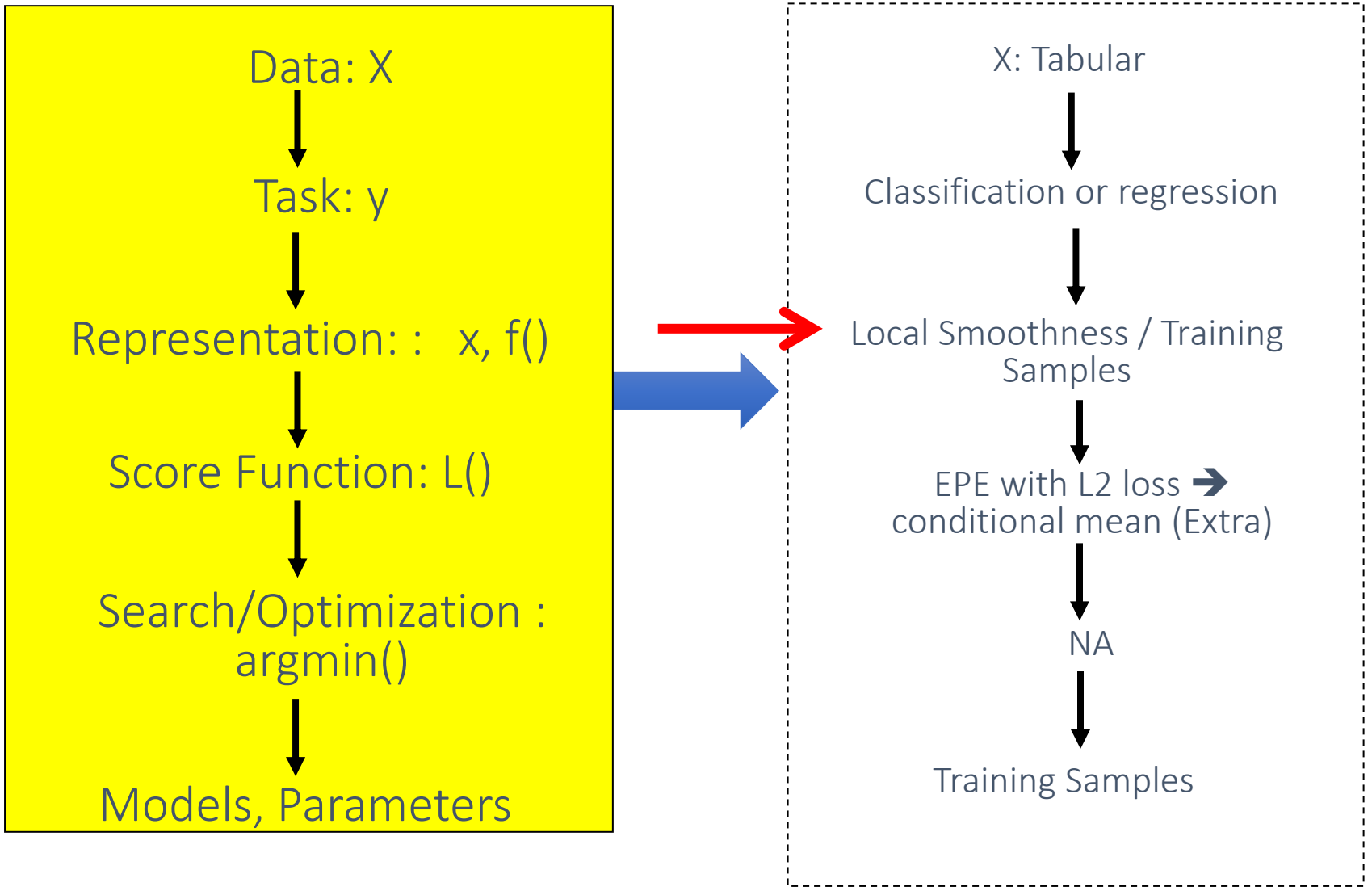
```
[44] # import regressor
     from sklearn.neighbors import KNeighborsRegressor
     # instantiate with K=5
     knn = KNeighborsRegressor(n_neighbors=100)
     # fit with data
     knn.fit(X, y)
```

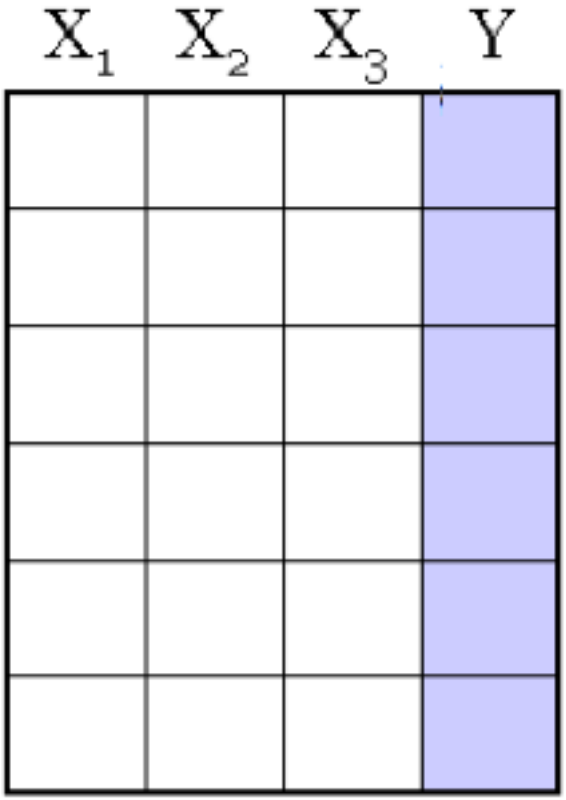
```
KNeighborsRegressor(algorithm='auto', leaf_size=3(
                    metric_params=None, n_jobs=Nor
                    weights='uniform')
```

```
###
Xfit = np.linspace(3, 10, 1000).reshape(-1, 1)
yfit =knn.predict(Xfit)

# Plot outputs
plt.scatter(X, y,  color='red')
plt.plot(Xfit, yfit, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```



Today: **K-nearest-neighbor(regressor or classifier)**





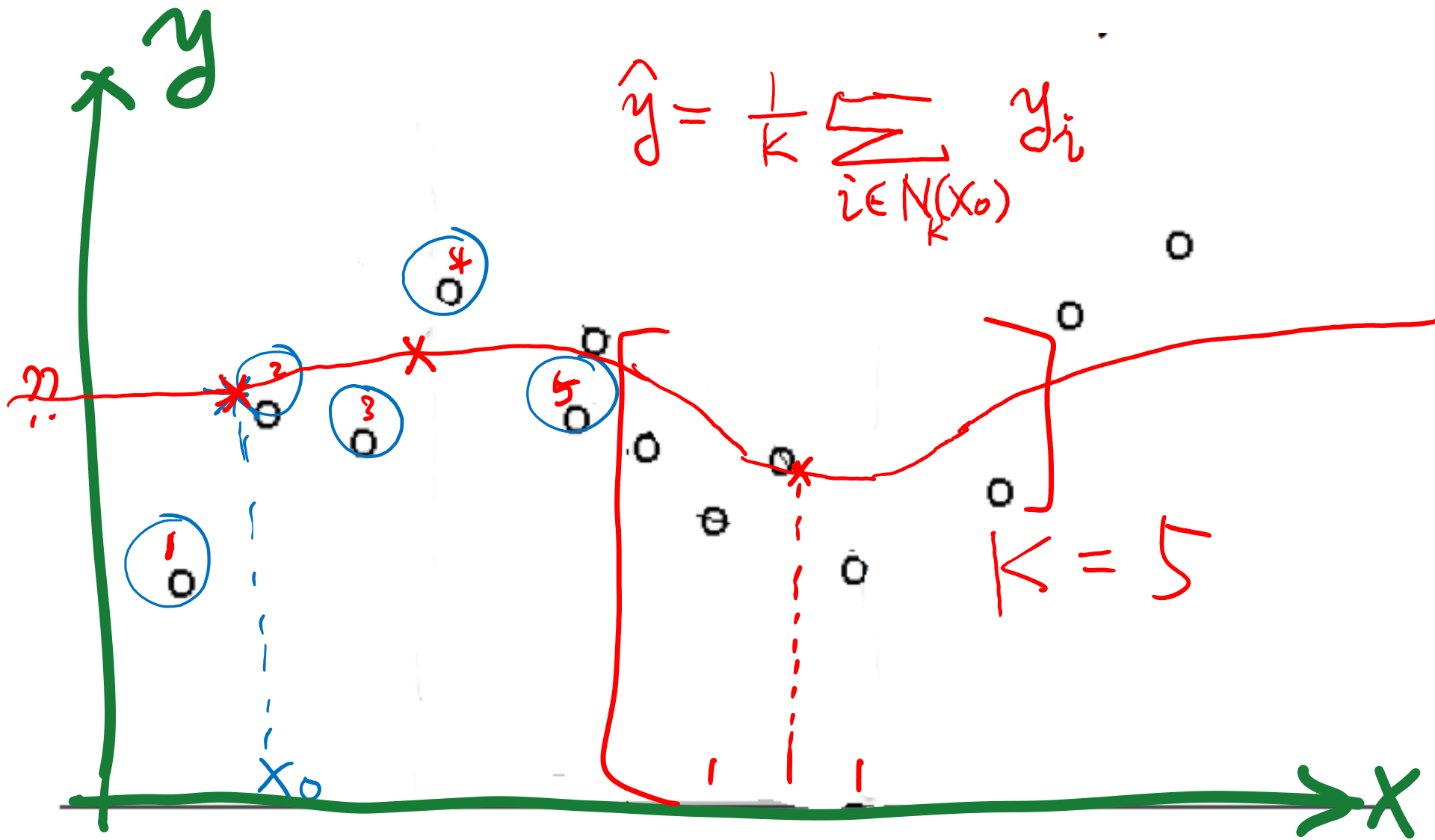
A Tabular Dataset for regression

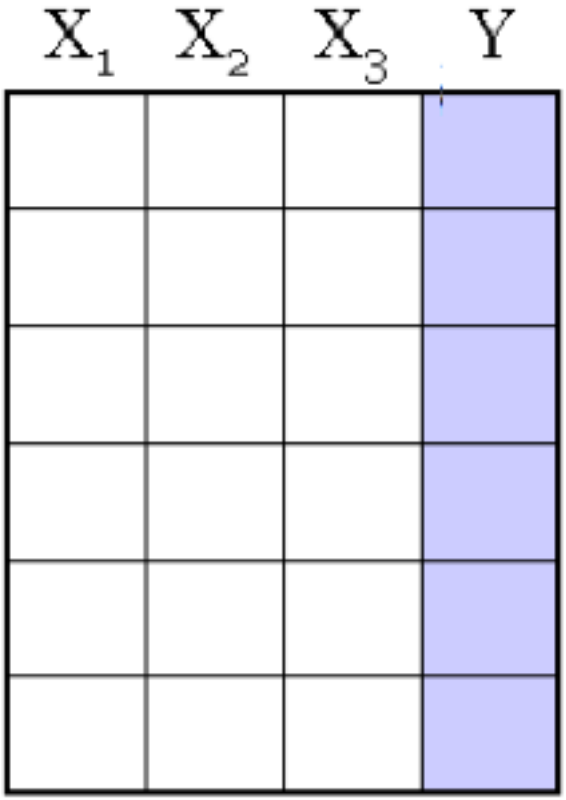
$$f : X \longrightarrow Y$$

Output Class: numerical variable

- **Data***/points/instances/examples/samples/records*: [rows]
- **Features***/attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target***/outcome/response/label/dependent variable*: special column to be predicted [last column in the above matrix]

K=5-Nearest Neighbor (1D input) for Regression





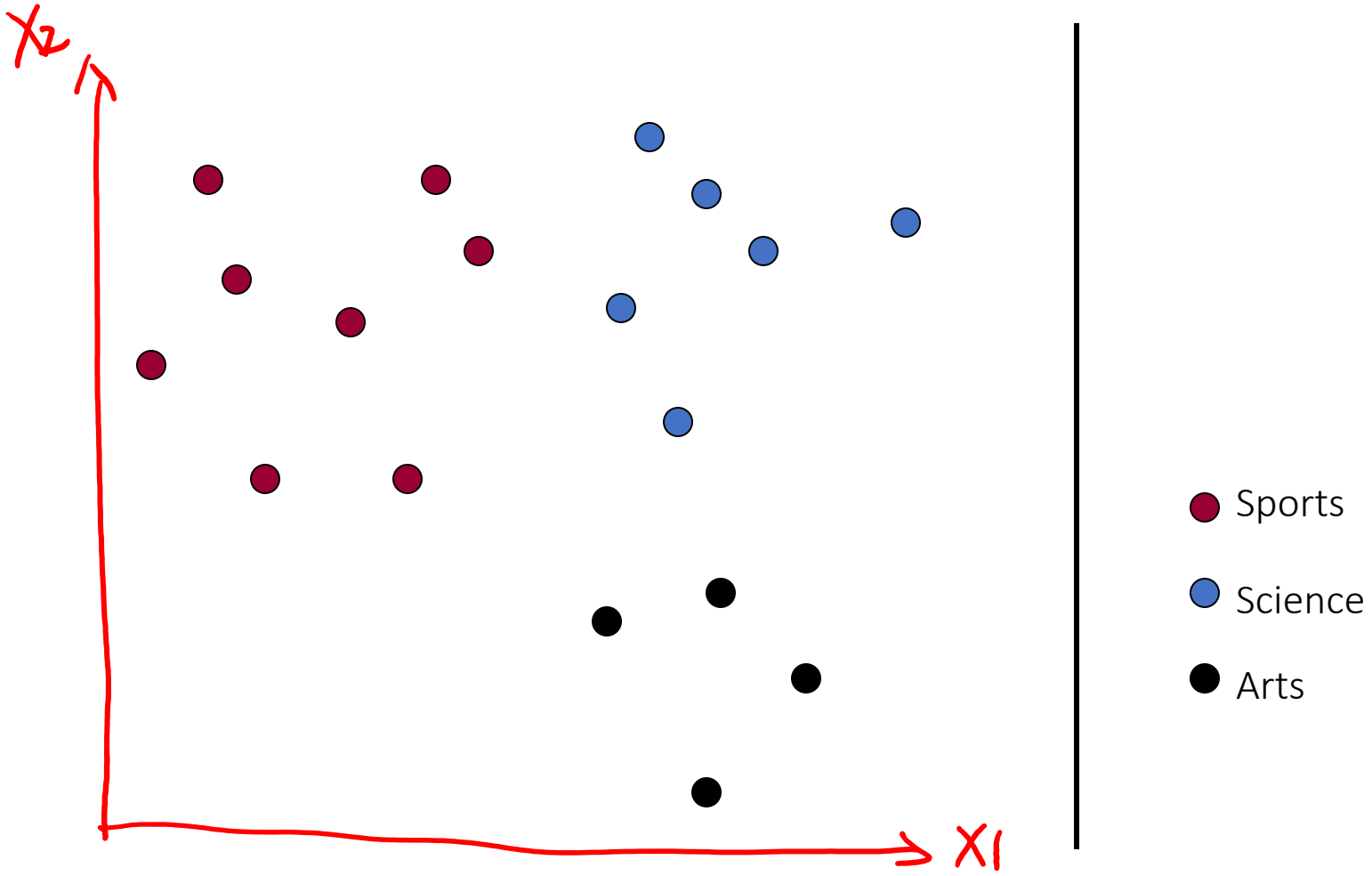
A Tabular Dataset for classification

$$f : X \longrightarrow Y$$

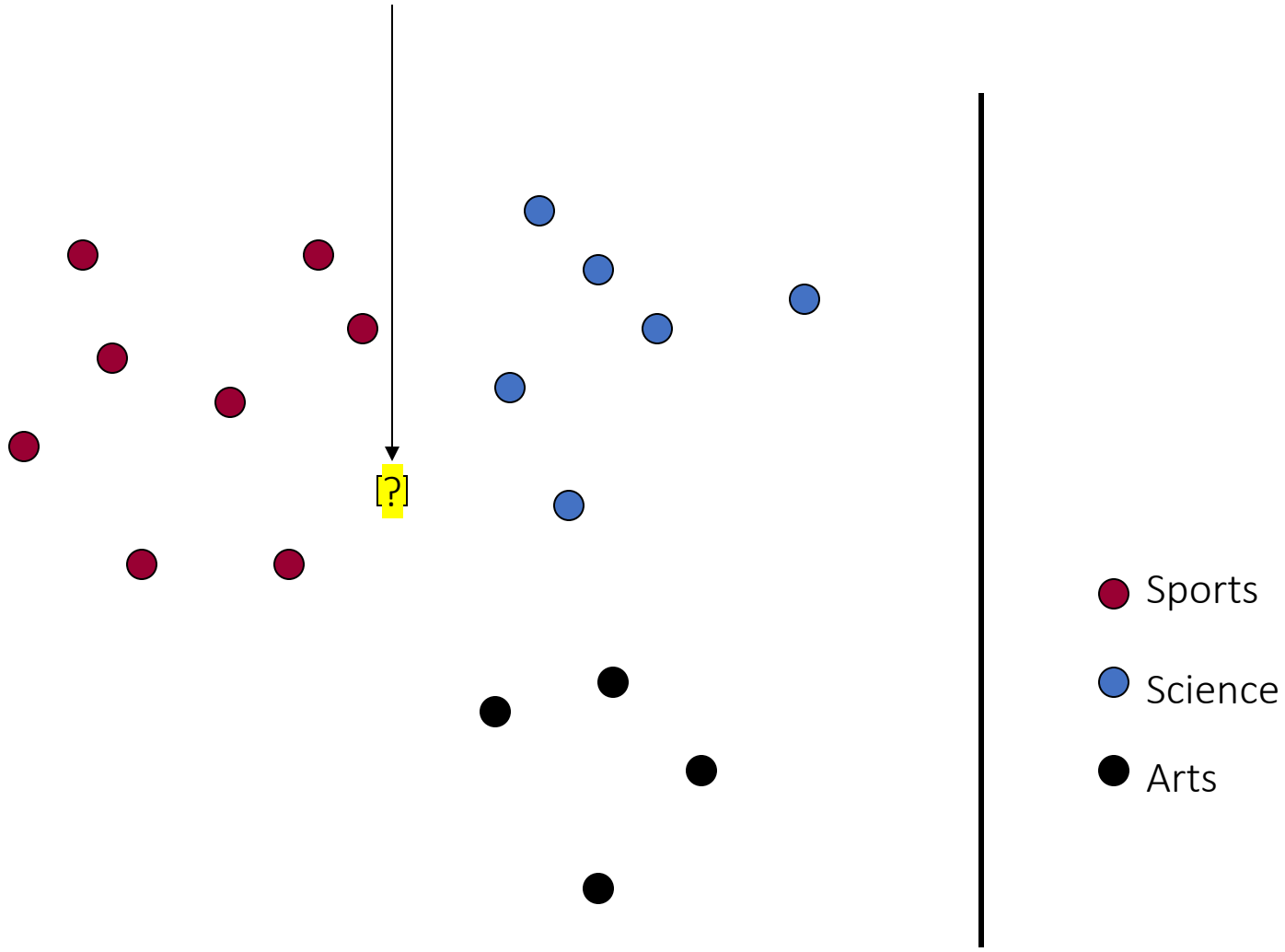
Output Class: categorical variable

- **Data***/points/instances/examples/samples/records*: [rows]
- **Features***/attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target***/outcome/response/label/dependent variable*: special column to be predicted [last column in the above matrix]

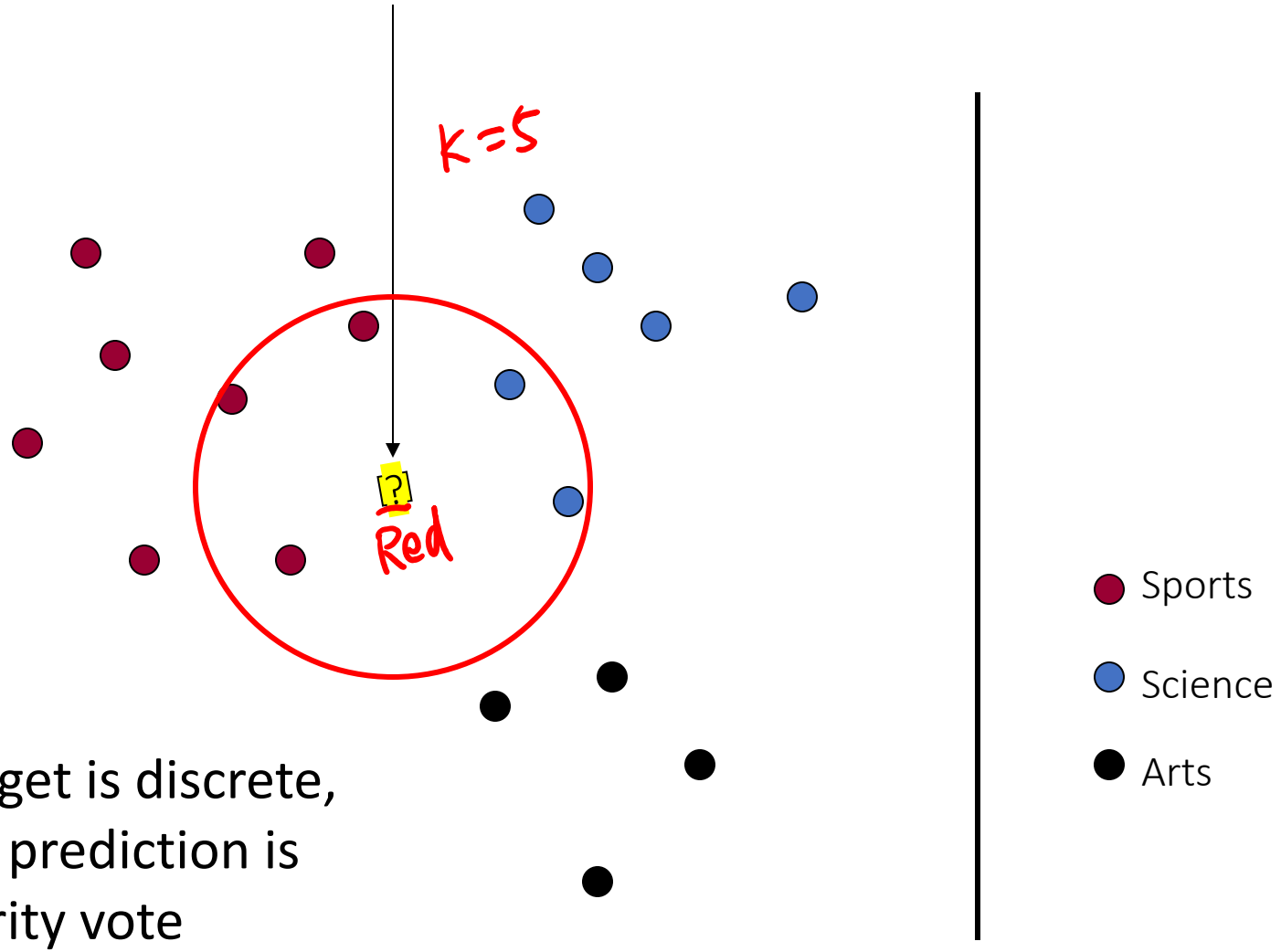
Multiple Target Classes in a Vector Space



Test Document = ?

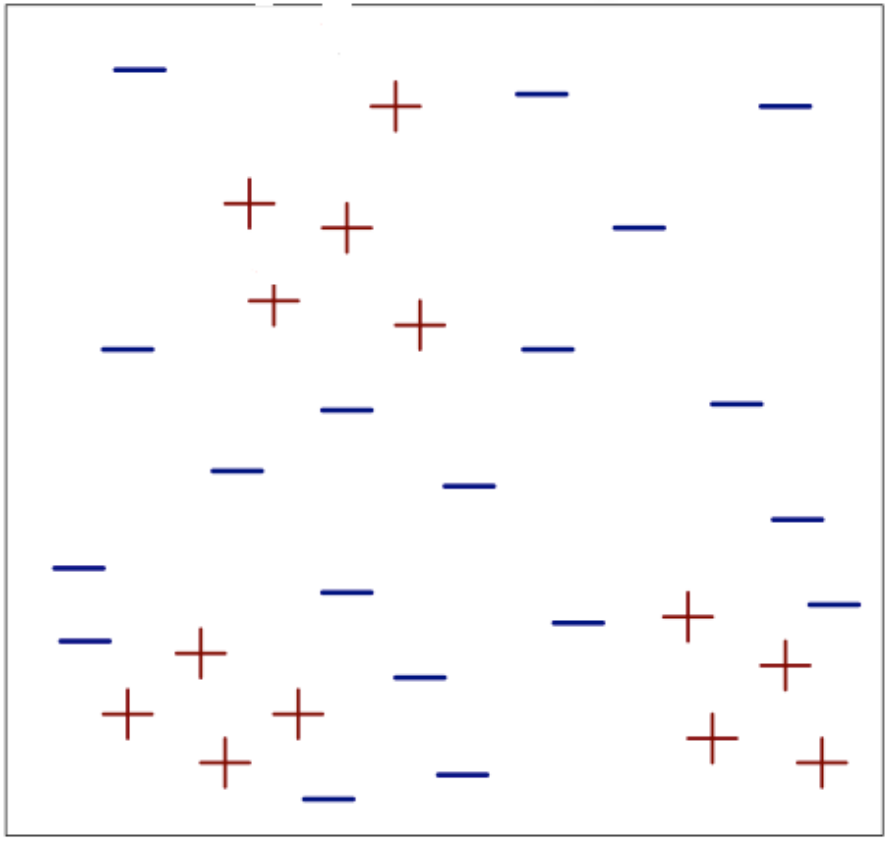


K-Nearest Neighbor (kNN) classifier



When target is discrete, the naïve prediction is the majority vote

K Nearest neighbor (Instance-based method)

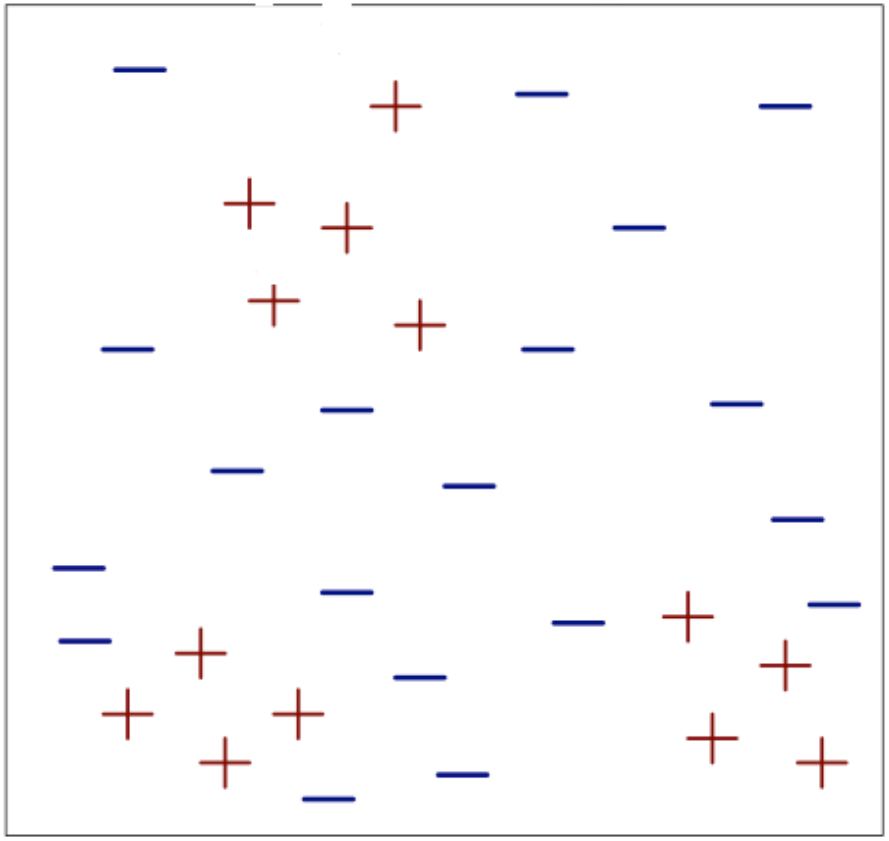


It Needs:

1. The set of stored training samples
2. Distance metric to compute distance between samples
3. The value of k, i.e., the number of nearest neighbors to retrieve

$y \in \{+, -\}$

K Nearest neighbor (Training Mode)

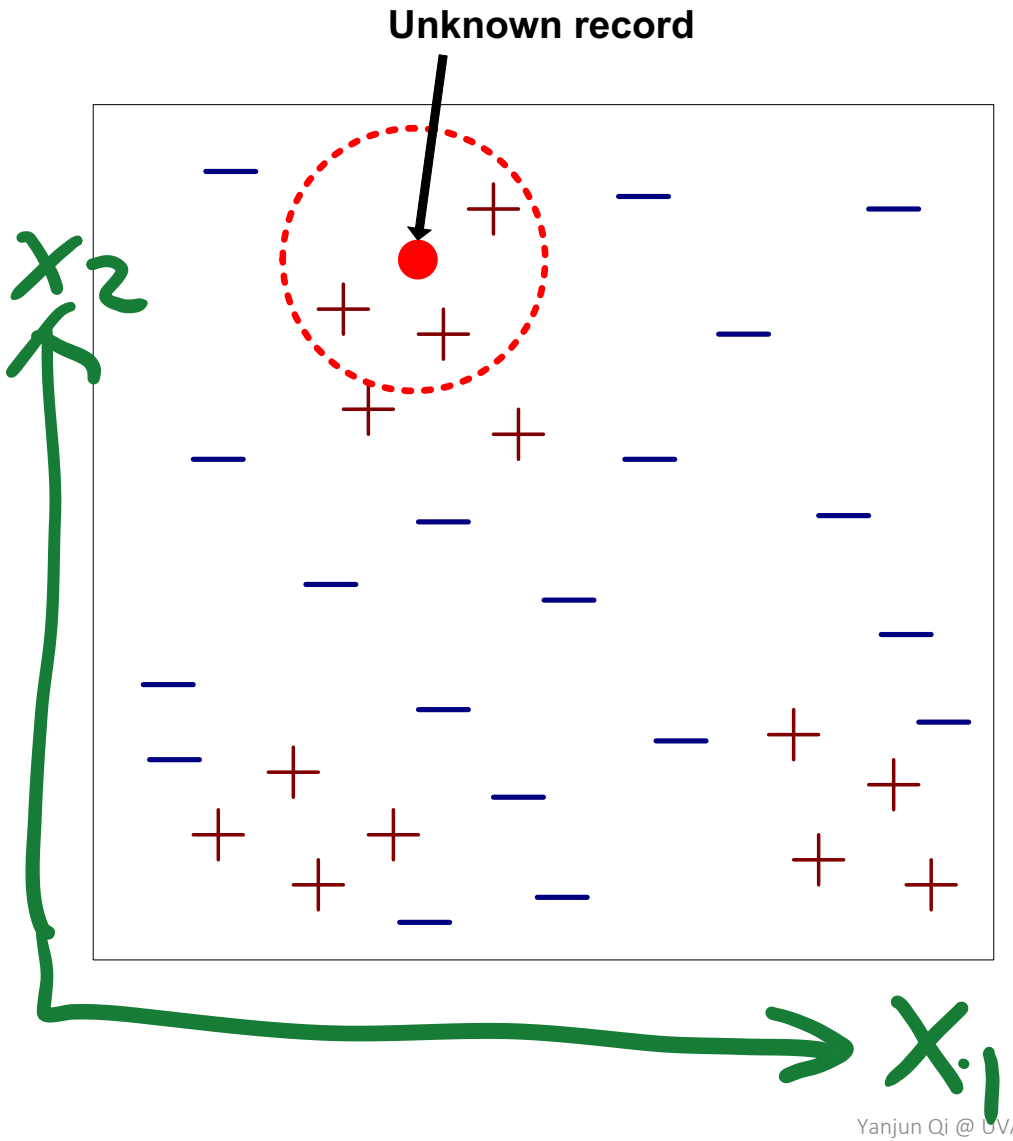


Training Mode:

- (Naïve) version:
  DO NOTHING !!!!

$y \in \{+, -\}$

K Nearest neighbor (Testing/Deploy Mode)



$x_?$

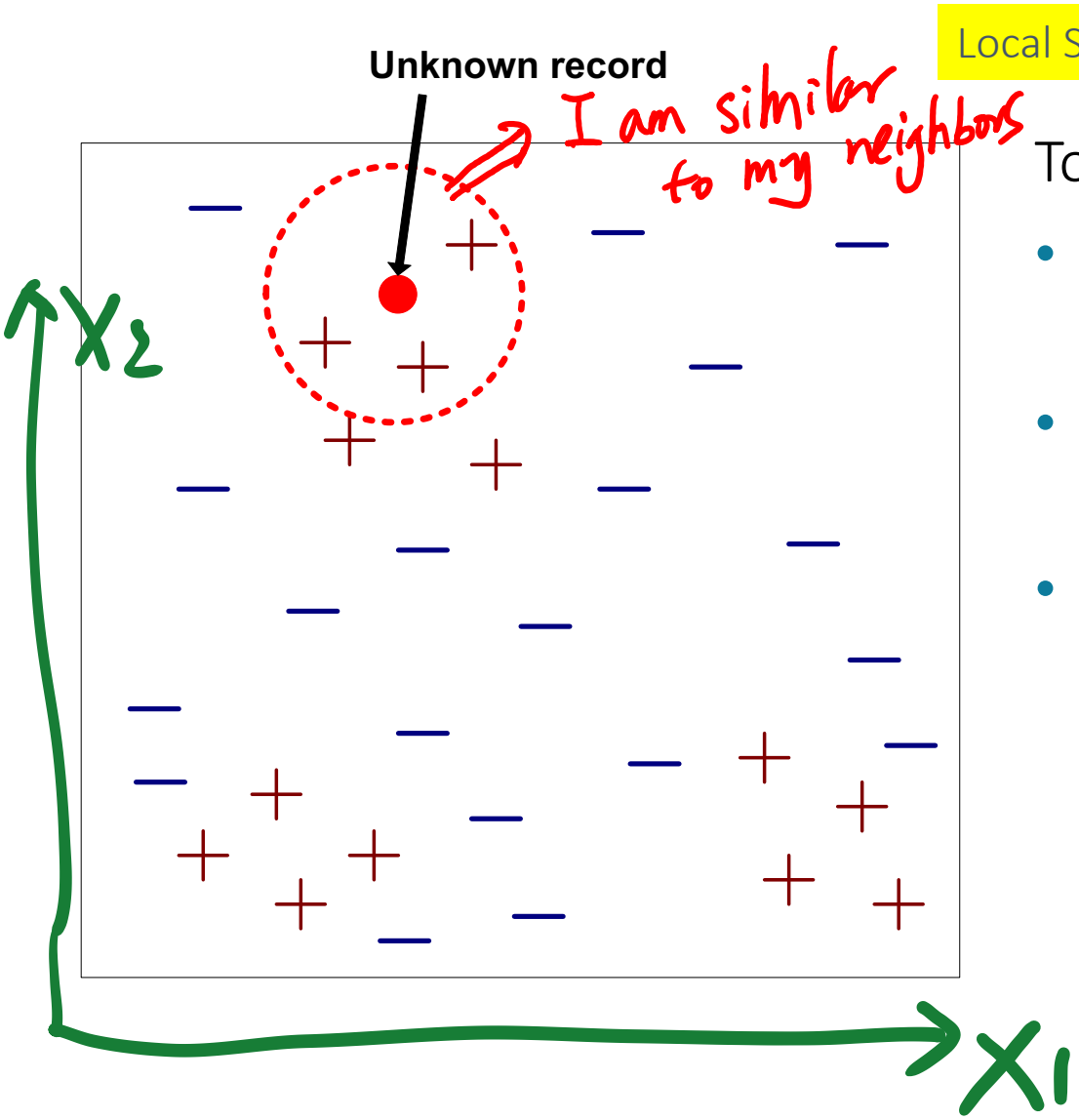
$\begin{cases} d(x_?, x_1) \\ d(x_?, x_2) \\ \vdots \\ d(x_?, x_n) \end{cases}$

$y_?$

① distance ② sort ③

$y \in \{+, -\}$

K Nearest neighbor (Testing/Deploy Mode)

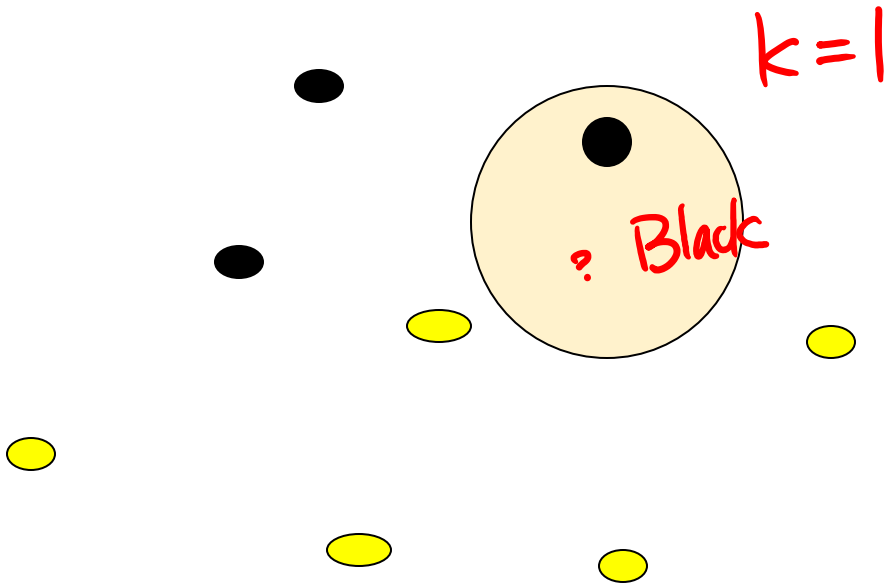


Local Smoothness

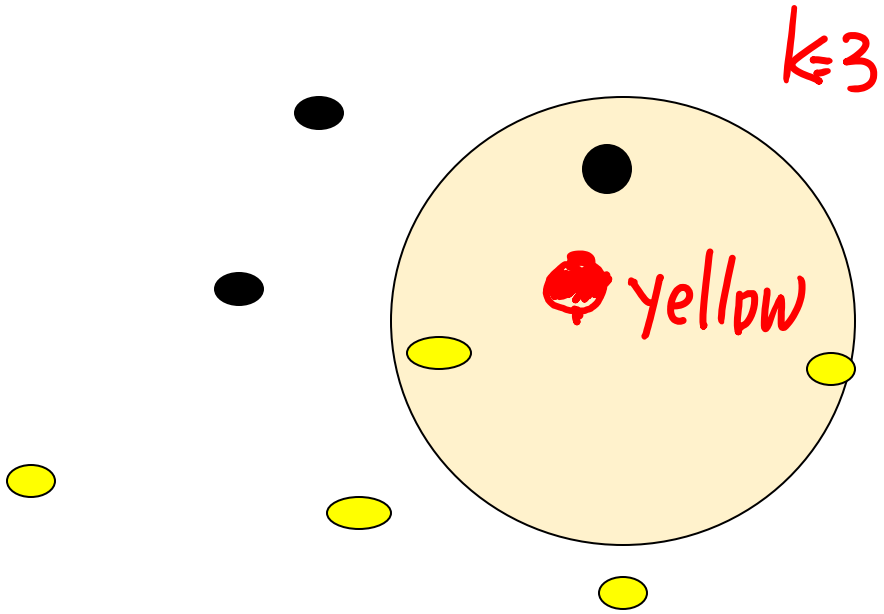
To classify unknown sample:

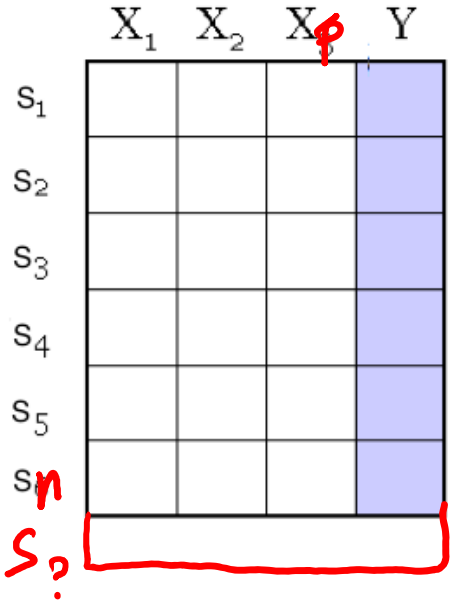
- Step1: Compute distance to all training records
- Step2: Identify k nearest neighbors
- Step3: Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

1-Nearest Neighbor



3-Nearest Neighbor





Step 1: $d(\vec{S_1}, \vec{S_?})$ $O(p)$

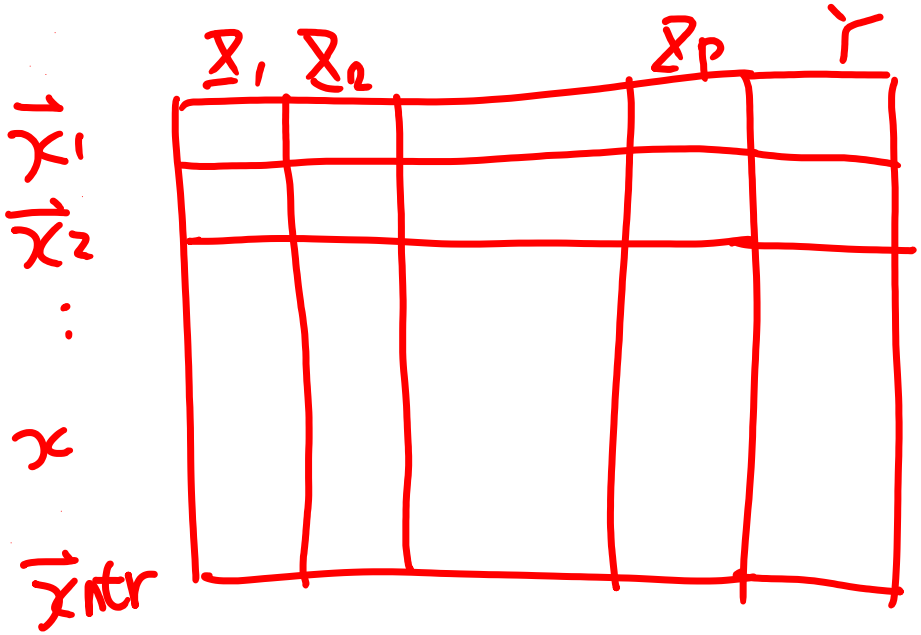
$\vec{S_1}$: $p \times 1$, $\vec{S_?}$: $p \times 1$

$$\sum_{j=1}^{p} (x_j - x_j)^2$$

$$\begin{cases} d(\vec{S_1}, \vec{S_?}) \\ d(\vec{S_2}, \vec{S_?}) \\ d(\vec{S_{tr}}, \vec{S_?}) \end{cases} \Rightarrow O(np)$$

Step 2: Sort(n select k) $\Rightarrow O(n \log n)$

m test $\Rightarrow$ $O(mnp + mn\log n)$



→ Step1: $O(n_{tr} \cdot p)$

$d(\vec{x}_?, \vec{x}_1)$

$d(\vec{x}_?, \vec{x}_2)$

$\vdots$

$d(\vec{x}_?, \vec{x}_{n_{tr}})$

① $d(\vec{x}_i, \vec{x}_j)$

② k

→ Step2:
pick top k from n_{tr}

$O(n_{tr})$

→ step3 y_γ

$y_? = \frac{1}{k} \sum_{j \in NN(x_?)} y_\gamma$

K-Nearest Neighbor: How to decide:

- Decision of output value is delayed till a new instance arrives
- Target variable may be discrete or real-valued
  - When target is discrete, the naïve prediction is the majority vote

$$y_? = \frac{1}{K} \sum_{j \in KNN(x_?)} y_j$$

$y \in \{0, 1\}$

① Majority vote: If $(y_i > t)$ → $t = 0.5$ Decision Boundary

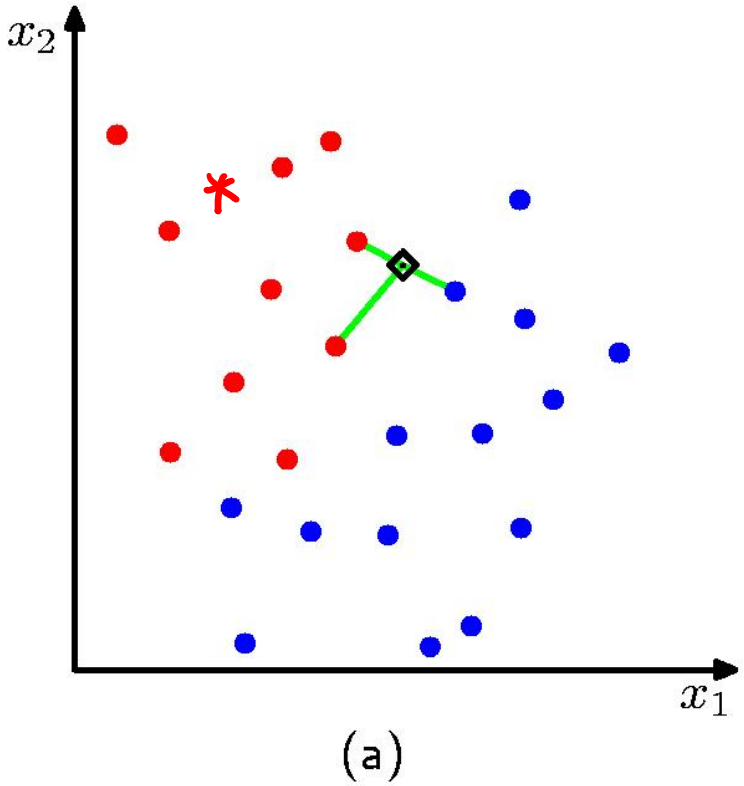
② $p_r(y_i | x_?) \approx 0.5$ tie

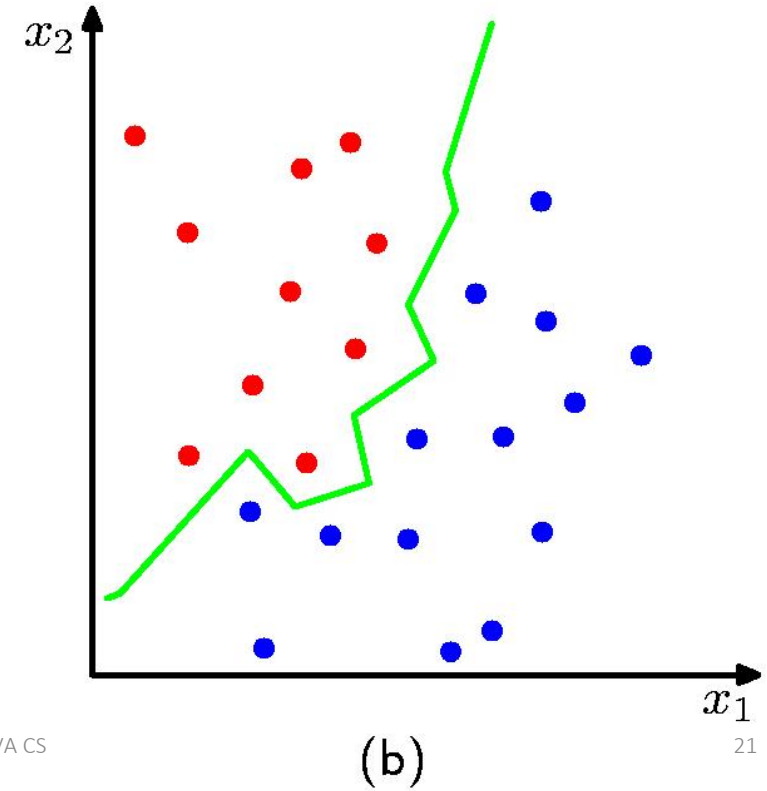
prob.

e.g. Decision boundary implemented by 3NN

k-nearest neighbors of a sample x :

➔ datapoints that have the k smallest distances to x among all training points





K-Nearest Neighbor: How to decide:

- Decision of output value is delayed till a new instance arrives
- Target variable may be discrete or real-valued
  - When target is discrete, the naïve prediction is the majority vote
  - When target is continuous, the naïve prediction is the mean value of the k nearest training examples

$$\hat{y}_{x_0} = \frac{1}{k} \sum_{i \in N_k(x_0)} y_i$$

$N_k(x_0)$ → k neighbor set

Summary of Nearest neighbor methods

- For regression, average the predictions of the K nearest neighbors.

- For classification, pick the class with the most votes.
  - How should we break ties?
  - E.g., Let the k'th nearest neighbor contribute a count that falls off with k. For example,

$$1+\frac{1}{2^k}$$


Thank You

UVA CS 4774: Machine Learning

Lecture 8: K-nearest-neighbor (Instance-based Learning)

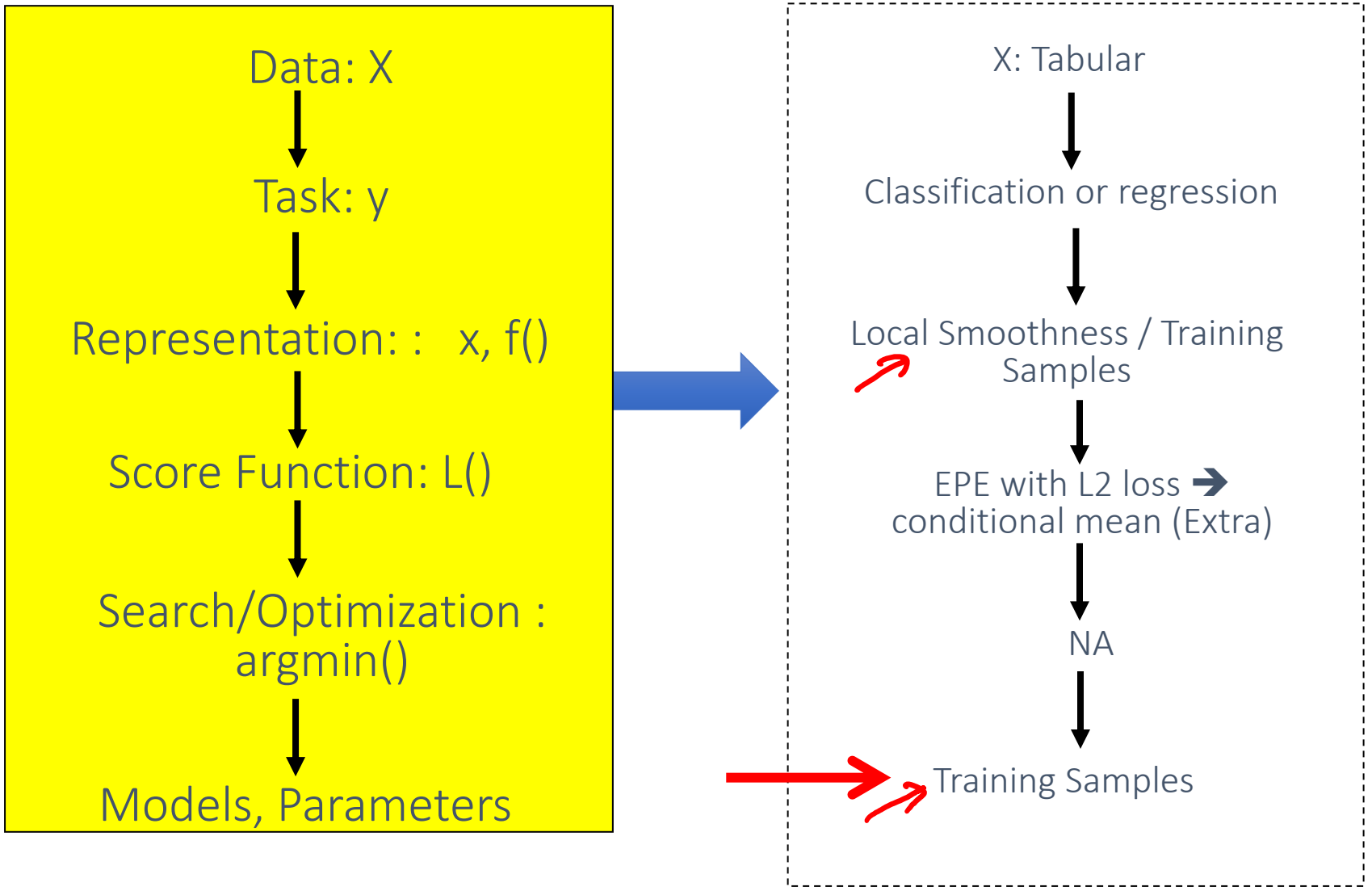
Module II

Dr. Yanjun Qi

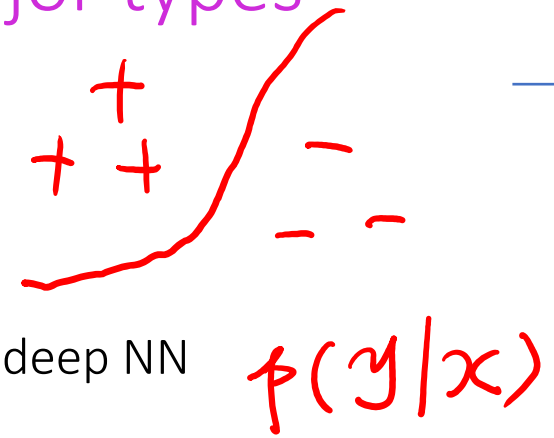
University of Virginia

Department of Computer Science

Today: **K-nearest-neighbor(regressor or classifier)**



We can divide the large variety of supervised classifiers into roughly three major types



1. Discriminative

   directly estimate a decision rule/boundary

   e.g., support vector machine, decision tree,

   e.g. logistic regression, neural networks (NN), deep NN

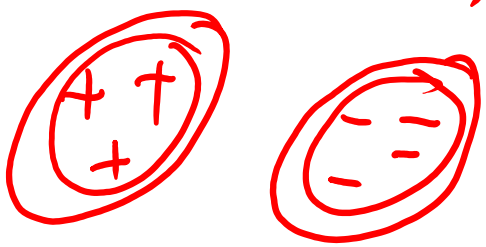
$p(y|x)$

2. Generative:

   build a generative statistical model

   e.g., Bayesian networks, Naïve Bayes classifier

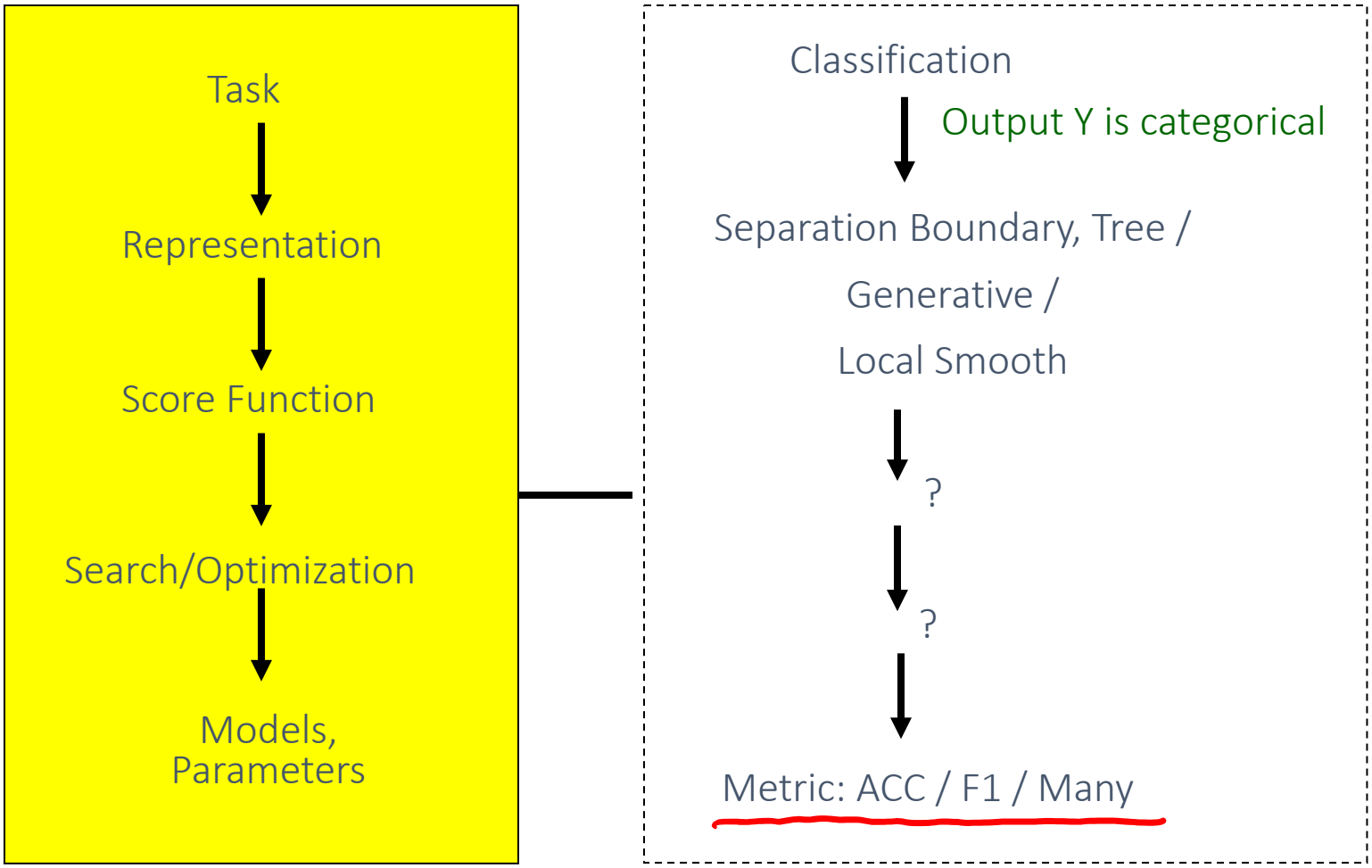
$p(x|y=c)$



3. Instance based classifiers

   - Use observation directly (no models)
   - e.g. K nearest neighbors

Supervised Classifiers



Instance-based Learning

- Simplest form of learning:
  - Training instances are searched for those that most closely resembles new instance
  - The instances themselves represent the knowledge

- Similarity function defines what's "learned"
- Instance-based learning is *lazy* learning

From witten&eibe

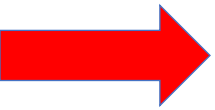
Instance-based Learning Examples:

- K-Nearest Neighbor Algorithm
- Weighted Regression
- Case-based reasoning

(deep) metric learning ⇒ to learn distance fun from data

From witten&eibe

What makes an Instance-Based Learner?



- A distance metric
- How many nearby neighbors to look at?
- A weighting function (optional)

Popular Distance Metric

- Euclidean $$D(x, x') = \sqrt{\sum_i \sigma_i^2 (x_i - x_i')^2}$$
- Or equivalently,

$$D(x, x') = \sqrt{(x - x')^T \Sigma (x - x')}$$

- Other metrics:
  - L_1 norm: |x-x'|
  - L_∞ norm: max |x-x'| (elementwise …)
  - Mahalanobis: where S is full, and symmetric
  - Correlation
  - Angle
  - Hamming distance, Manhattan distance
  - …

String Search

Feature Scaling in Nearest neighbor method

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5 m to 1.8 m
    - weight of a person may vary from 90 lb to 300 lb
    - income of a person may vary from \$10K to \$1M

$k=1$

$k=2$

$k=3$

negative inner product

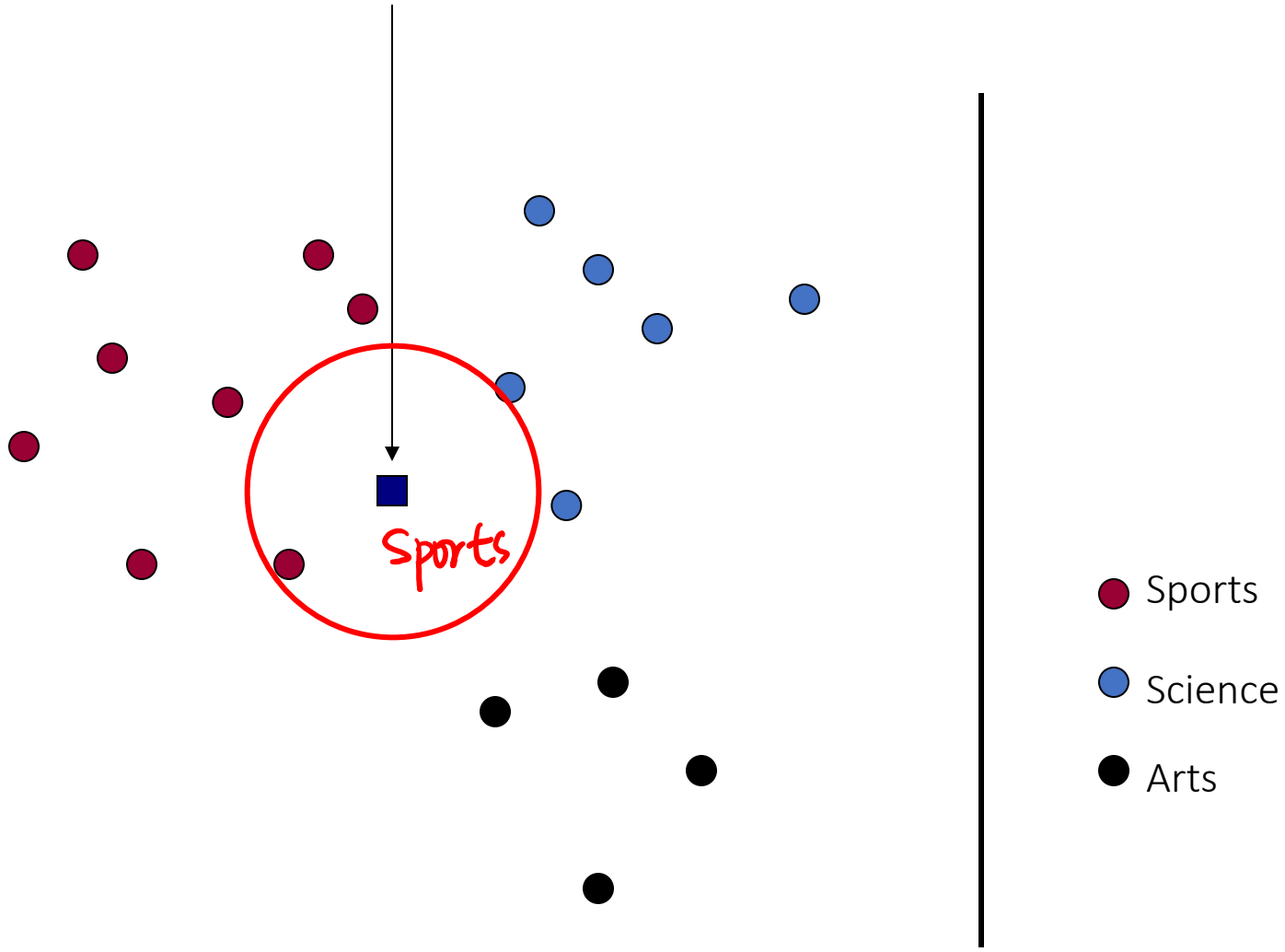
$$-\sum_{k=1}^{p}(\vec{x}_{ik}\cdot\vec{x}_{jk}) = d(\vec{x}_i, \vec{x}_j)$$

The relative scaling in the distance metric affect region shapes.

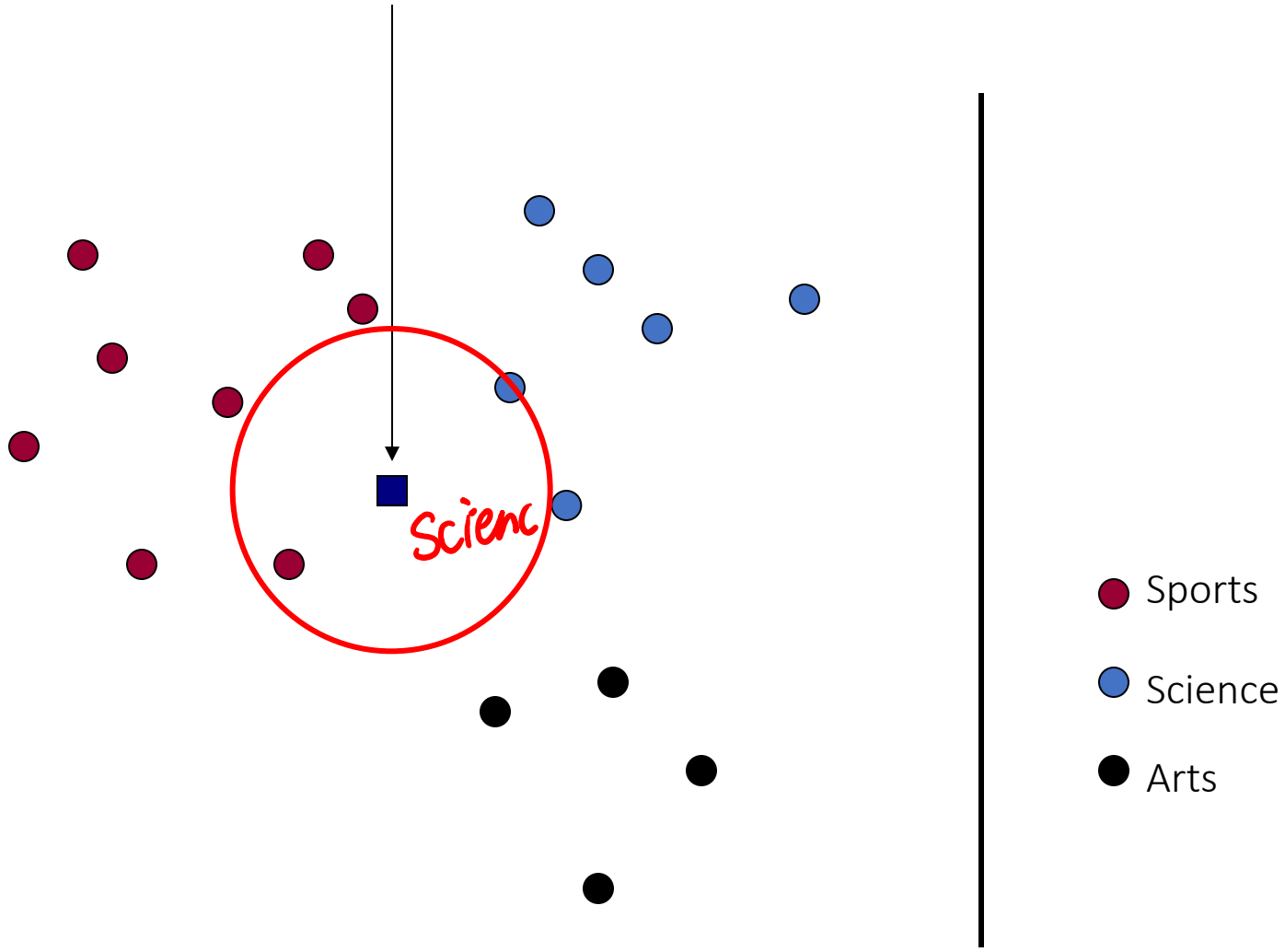
What makes an Instance-Based Learner?

- A distance metric
- How many nearby neighbors to look at?
- A weighting function (optional)

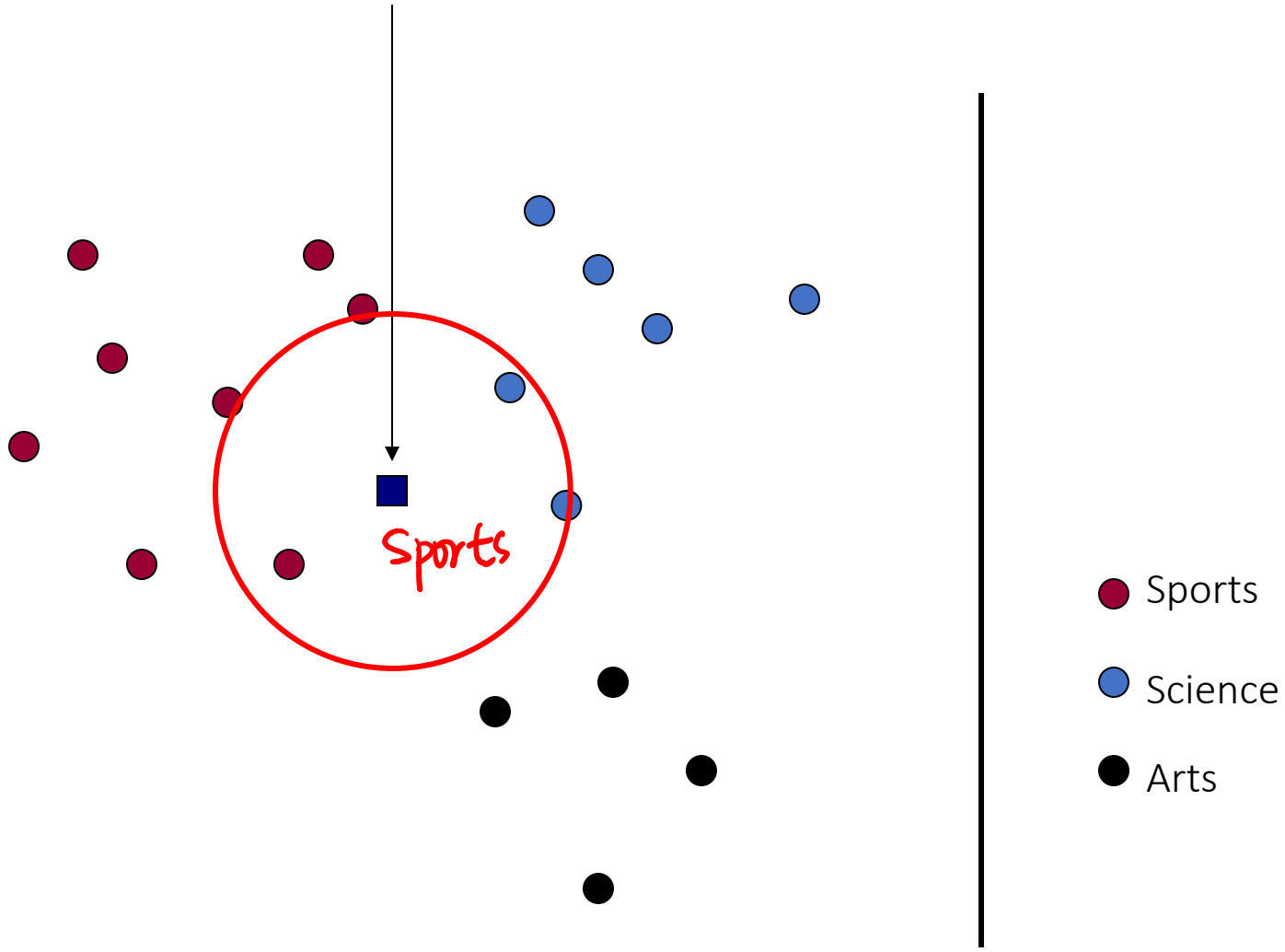
1-Nearest Neighbor (kNN) classifier



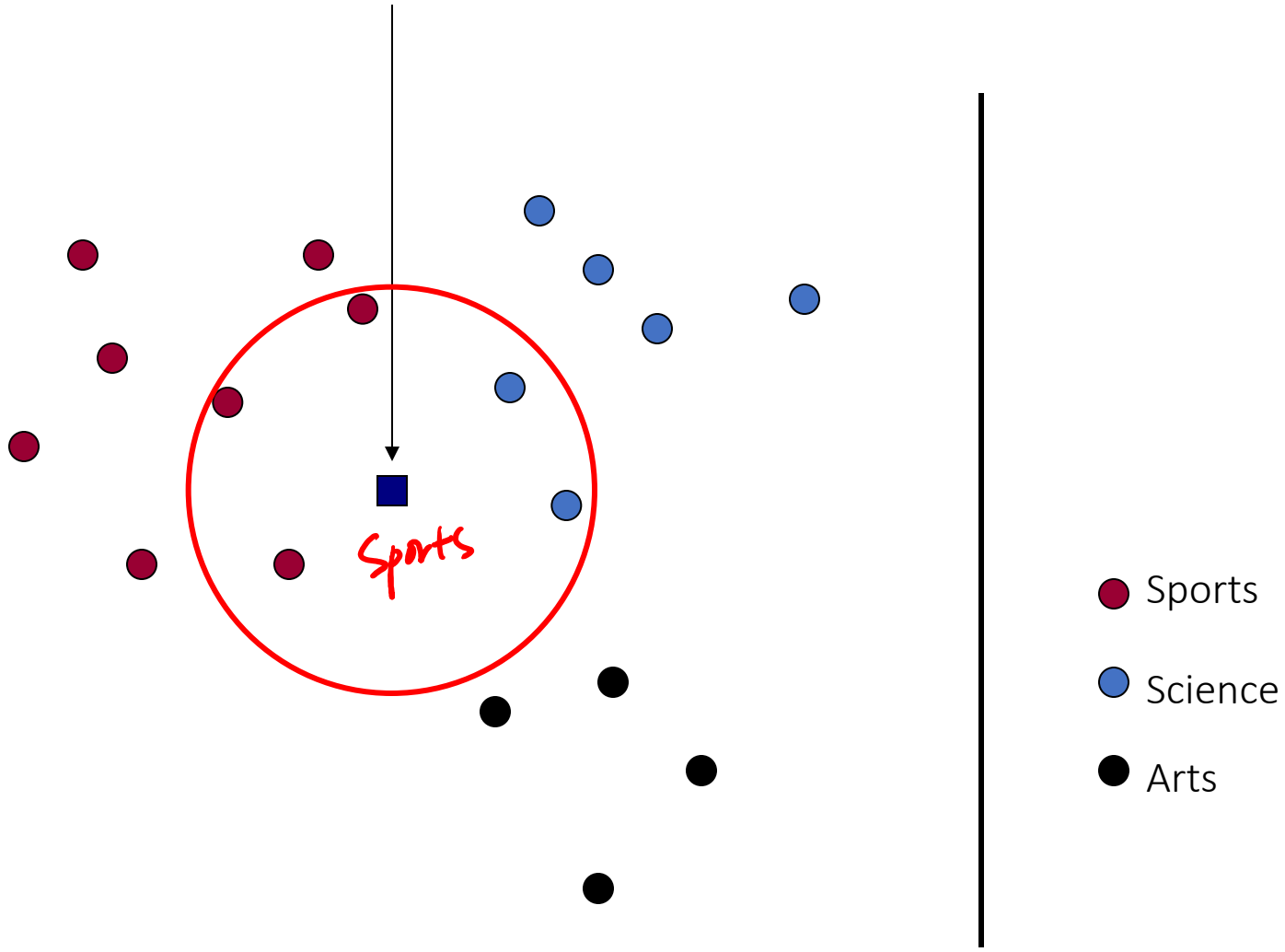
2-Nearest Neighbor (kNN) classifier



3-Nearest Neighbor (kNN) classifier



5-Nearest Neighbor (kNN) classifier

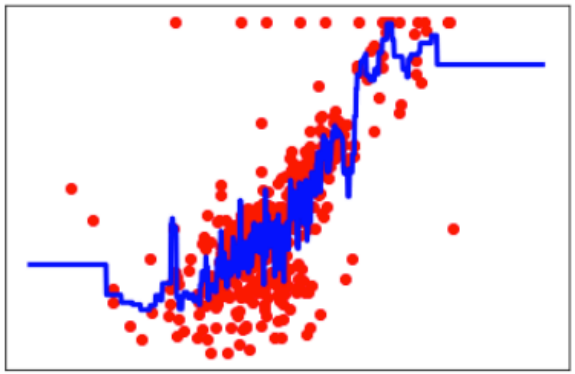


Our code run:
https://colab.research.google.com/drive/1Wzmg5ziMBMvVvLHx_0C4u0x7syF8jPT9?usp=sharing

```
[42] # import regressor
     from sklearn.neighbors import KNeighborsRegressor
     # instantiate with K=5
     knn = KNeighborsRegressor(n_neighbors=5)
     # fit with data
     knn.fit(X, y)
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=3
                    metric_params=None, n_jobs=No
                    weights='uniform')
```

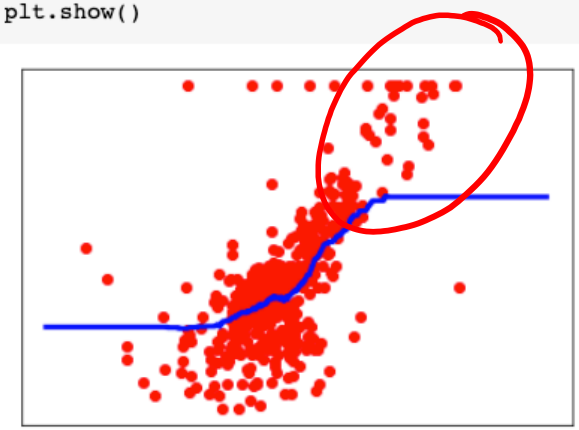
```
###
Xfit = np.linspace(3, 10, 1000).reshape(-1, 1)
yfit =knn.predict(Xfit)

# Plot outputs
plt.scatter(X, y,  color='red')
plt.plot(Xfit, yfit, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```



```
[44] # import regressor
     from sklearn.neighbors import KNeighborsRegressor
     # instantiate with K=5
     knn = KNeighborsRegressor(n_neighbors=100)
     # fit with data
     knn.fit(X, y)
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=3(
                    metric_params=None, n_jobs=Nor
                    weights='uniform')
```

```
###
Xfit = np.linspace(3, 10, 1000).reshape(-1, 1)
yfit =knn.predict(Xfit)

# Plot outputs
plt.scatter(X, y,  color='red')
plt.plot(Xfit, yfit, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

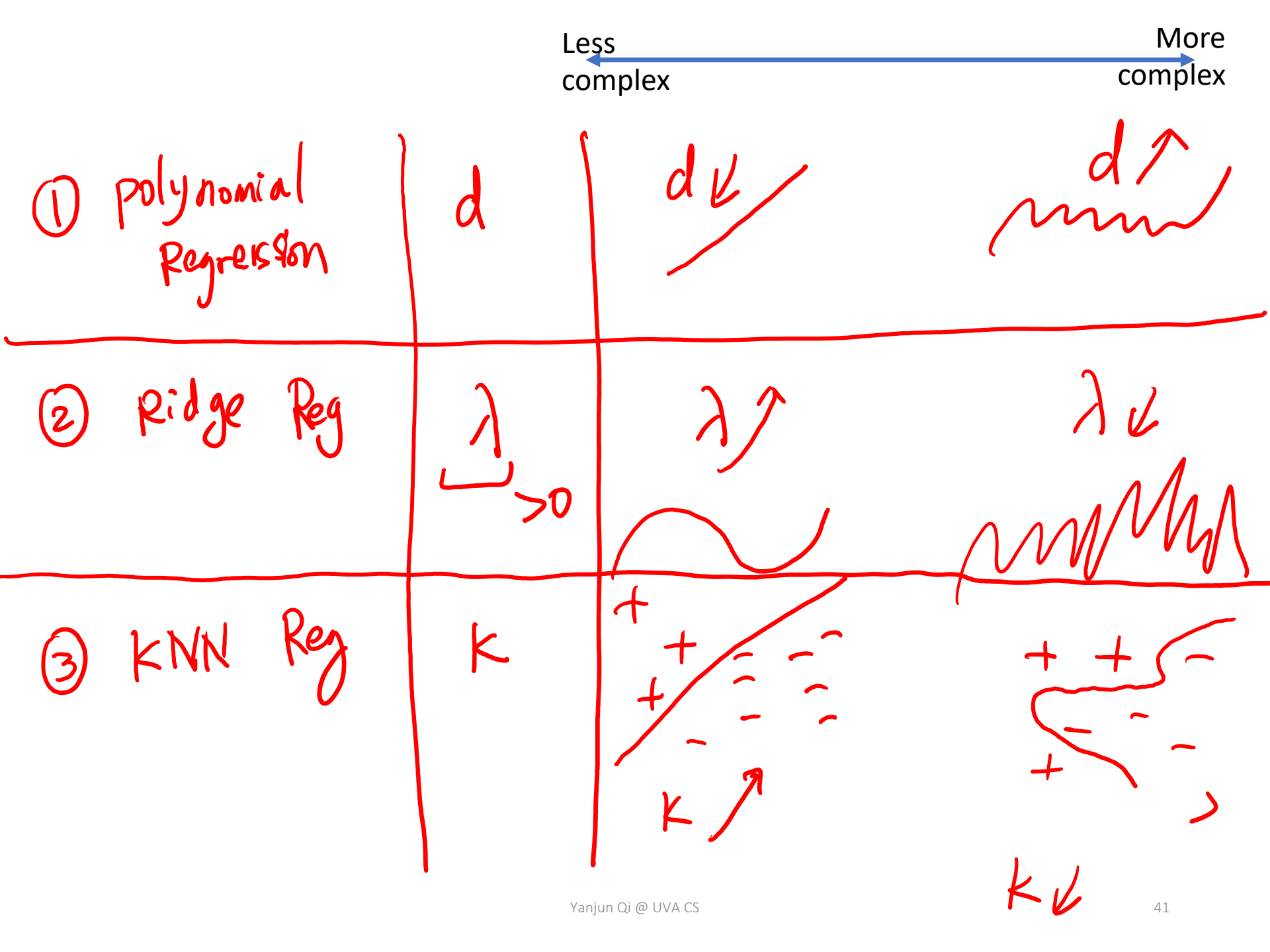


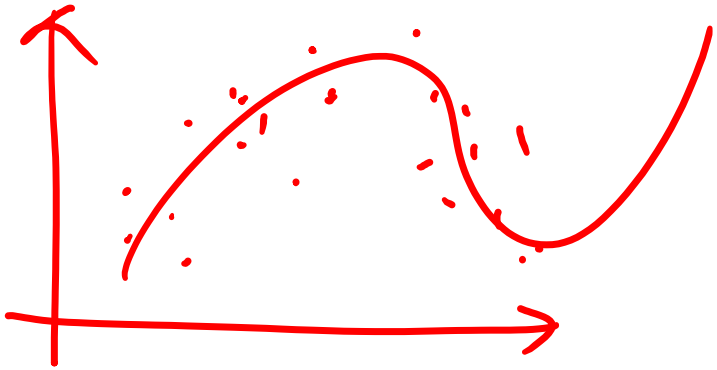
Model Selection for Nearest neighbor classification

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

k ↓ flexible / varies a lot

k ↑ Smooth / varies little


Less complex
More complex
① polynomial Regression
d
d↙
d↗
② Ridge Reg
λ
>0
λ↗
λ↙
③ KNN Reg
k
k↗
k↙



Polynomial Regression

d ↑ Complexity ↑

d ↓ Complexity ↓

Ridge Regression

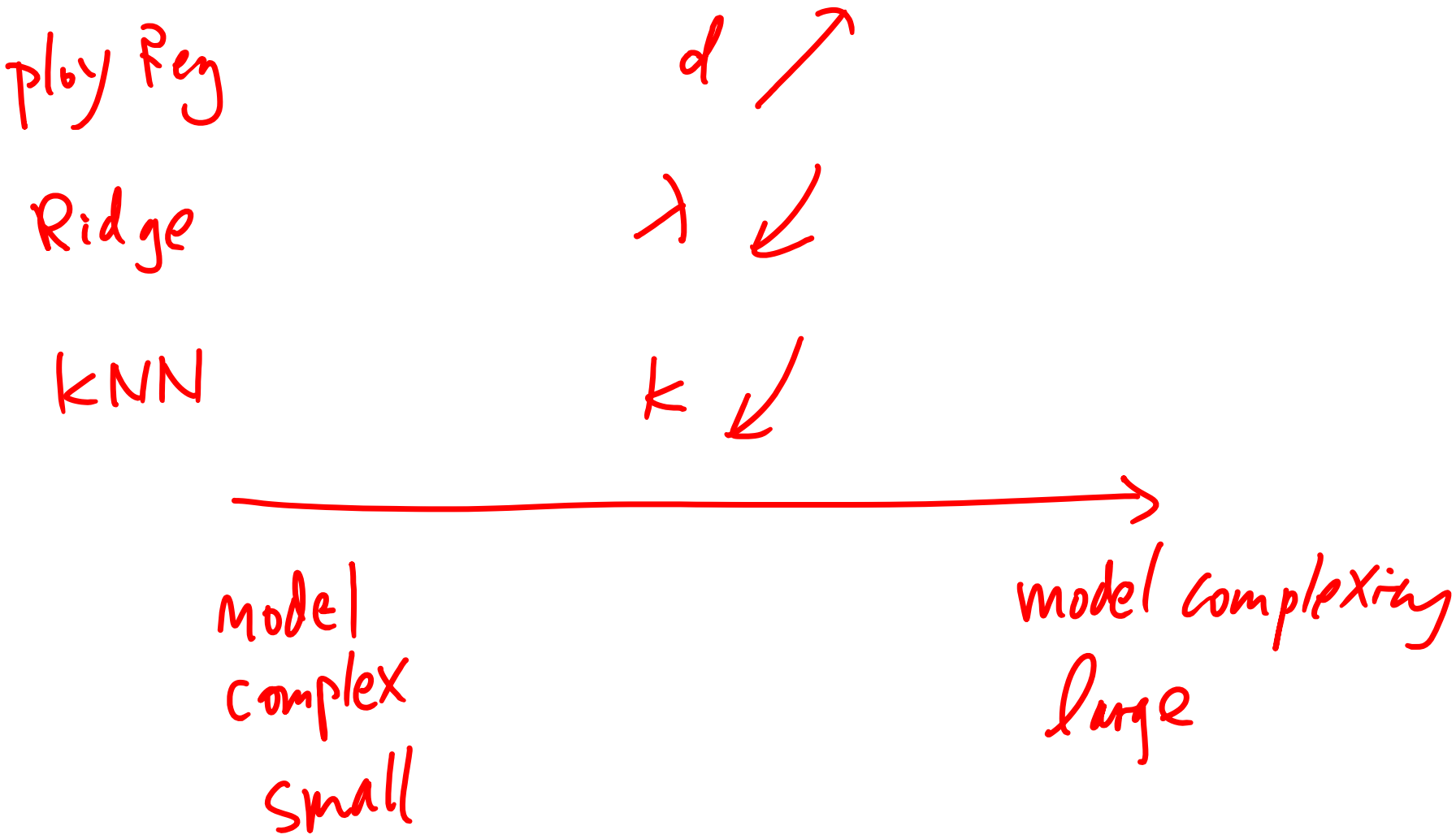
λ ↑ complexity ↓

λ ↓ complexity ↑

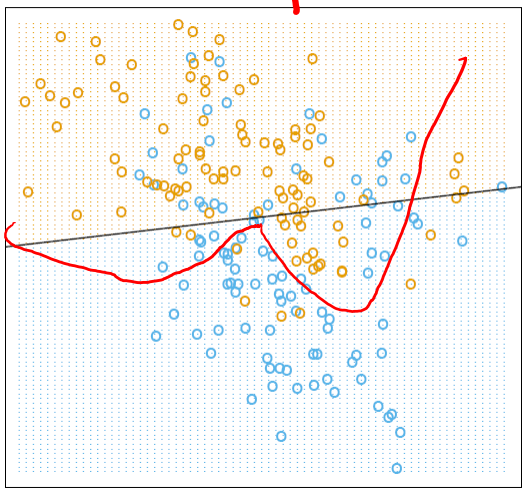
KNN

k ↑ complexity ↓

k ↓ complexity ↑

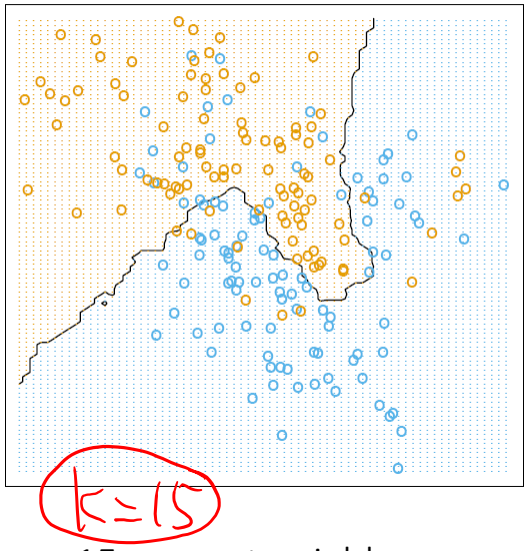

poly Reg
d ↗
Ridge
λ ↙
KNN
k ↙
model complex small
model complexity large

Decision boundaries in Linear vs. kNN models (Later)

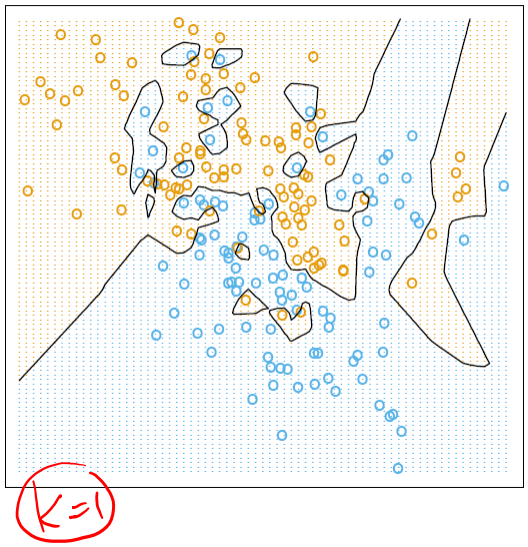


Low Variance / High Bias

linear regression
- global
- stable
- can be inaccurate



15-nearest neighbor



1-nearest neighbor

KNN
- local
- accurate
- unstable

Low Bias / High Variance

What ultimately matters: GENERALIZATION

Model Selection for Nearest neighbor classification

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

- Bias and variance tradeoff
  - A small neighborhood → large variance → unreliable estimation
  - A large neighborhood → large bias → inaccurate estimation

overfit

underfit

What makes an Instance-Based Learner?

- A distance metric
- How many nearby neighbors to look at?
- A weighting function (optional)

Nearest neighbor Variation (1)

- Options for determining the class from nearest neighbor list:
  Weight the votes according to distance
  - example: weight factor $w = 1 / d^2$

Spurious or less relevant points need to be downweighted

$$y_? = \frac{1}{K} \sum_{j \in kNN(x_?)} w_j \, y_j$$

$$w_j = \frac{1}{d(x_j, x_?)^2}$$

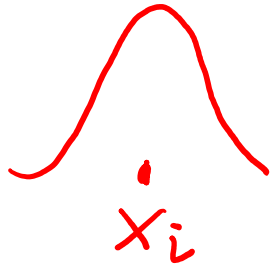
Distance-Weighted k-Nearest Neighbor Algorithm

- Assign weights to the neighbors based on their "distance" from the query point
  - Weight "may" be inverse square of the distances $w = 1 / d^2$
- Extreme Option: All training points may influence a particular instance
  - E.g., Shepard's method/ Modified Shepard, … by Geospatial Analysis

Nearest neighbor Variation (2): RBF kernel Weighted kNN

- Weight the contribution via Weight function

$$w(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\lambda |\mathbf{x} - \mathbf{x}_i|_2^2\right)$$



- Prediction

$$\Pr(y|\mathbf{x}) = \frac{\sum_{i=1}^{n} w(\mathbf{x}, \mathbf{x}_i)\delta(y, y_i)}{\sum_{i=1}^{n} w(\mathbf{x}, \mathbf{x}_i)}$$

$$\delta(y, y_i) = \begin{cases} 1 & y = y_i \\ 0 & y \neq y_i \end{cases}$$

$$\sum_{c=1}^{C} p(y=c|x)$$

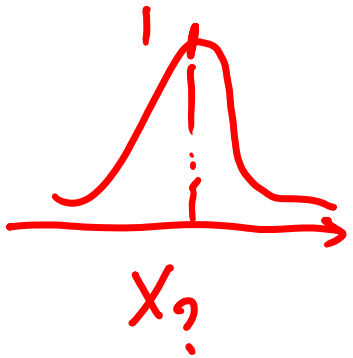
Another: RBF kernel Weighted kNN

$$w(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\lambda |\mathbf{x} - \mathbf{x}_i|_2^2\right)$$

For Classification

$$\Pr(y|\mathbf{x}) = \frac{\sum_{i=1}^{n} w(\mathbf{x}, \mathbf{x}_i)\delta(y, y_i)}{\sum_{i=1}^{n} w(\mathbf{x}, \mathbf{x}_i)}$$

$$\delta(y, y_i) = \begin{cases} 1 & y = y_i \\ 0 & y \neq y_i \end{cases}$$

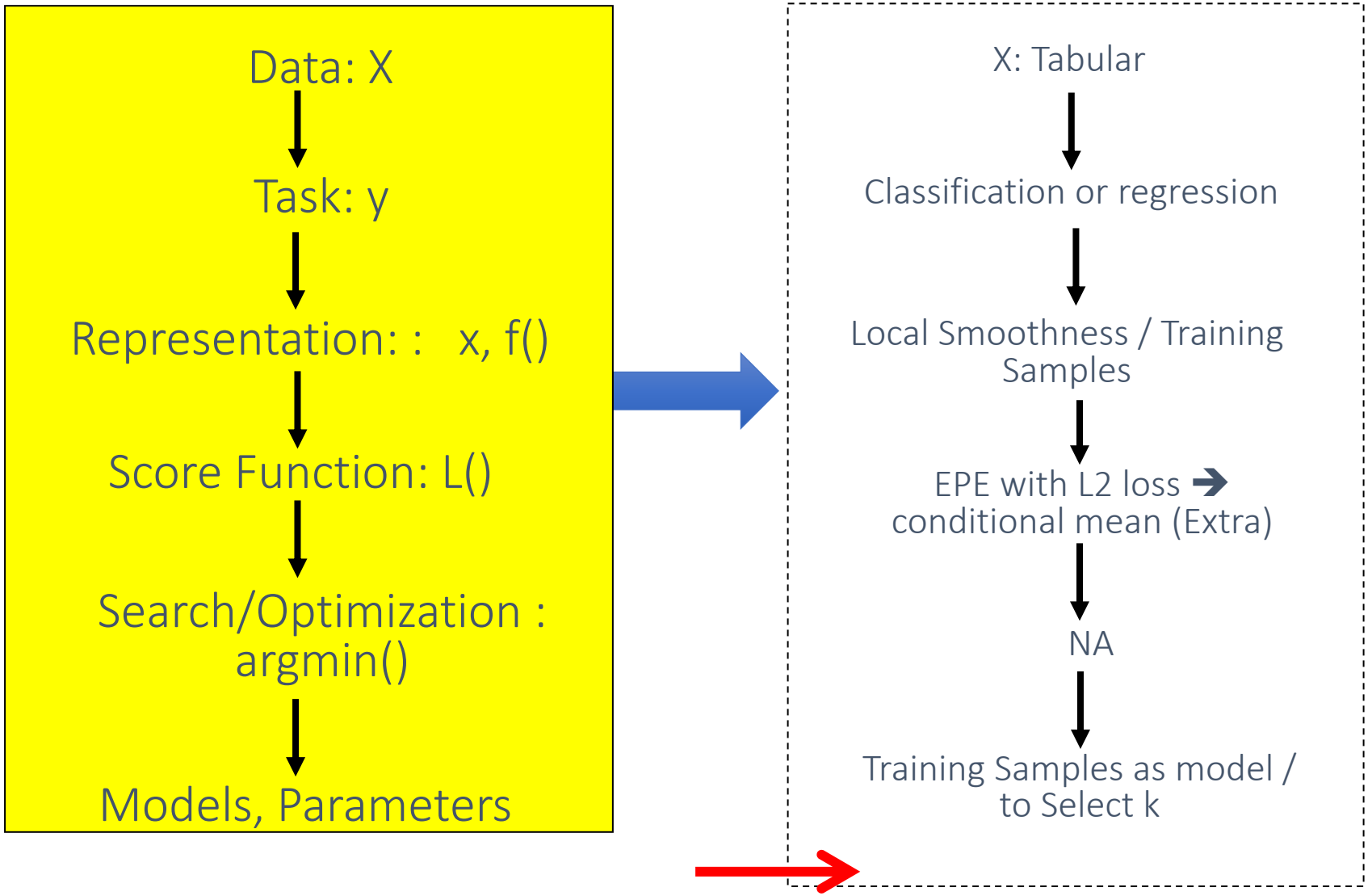


For Regression

$$\hat{y} = \frac{\sum w(x, x_i)\, y_i}{\sum w(x, x_i)}$$

weighted regression

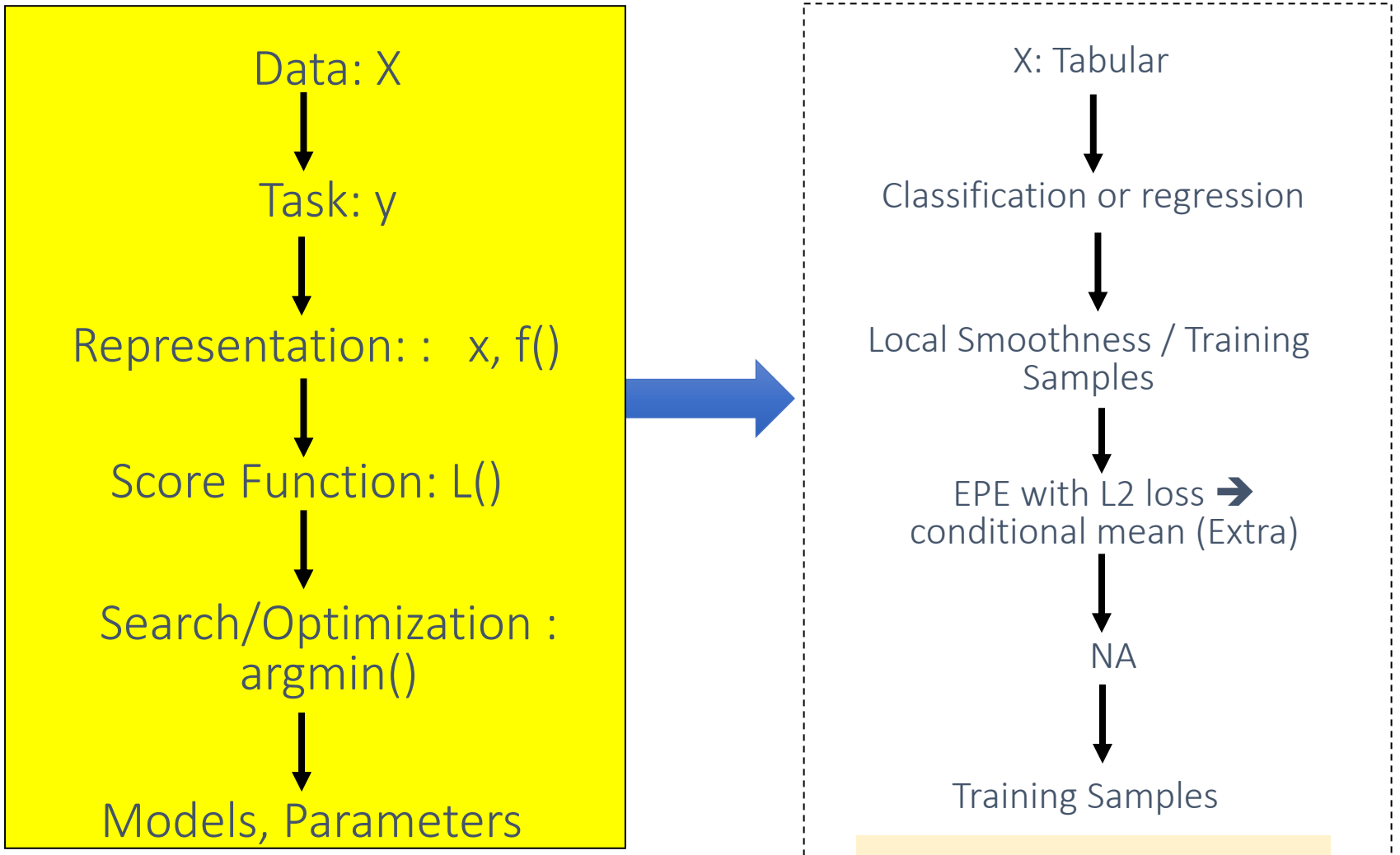
Today: **K-nearest-neighbor(regressor or classifier)**



We aim to make our trained model

- 1. Generalize Well
- 2. Computational Scalable and Efficient
- 3. Trustworthy: Robust / Interpretable
  - Especially for some domains, this is about trust!

Today: **K-nearest-neighbor(regressor or classifier)**



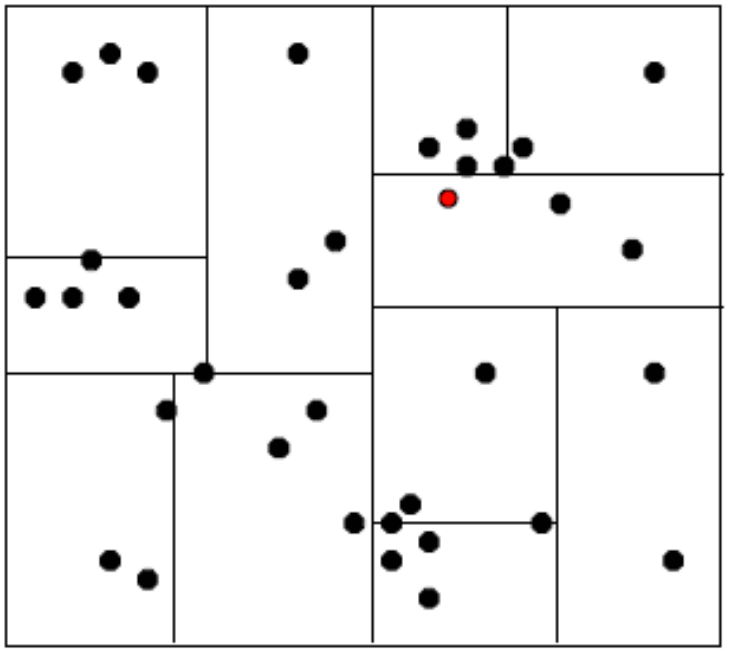
Computational Time Cost

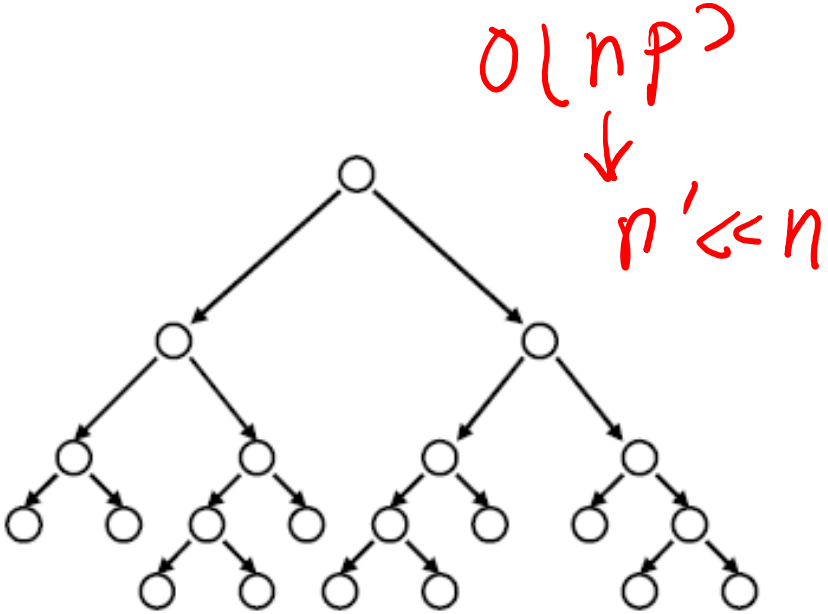
	Train (n)	Test (m=1)
Linear Regression	$O(np^2+p^3)$	$O(p)$ $\hat{y}=\beta^T x$
KNN Reg	$O(1)$	$O(np)+$ $O(\text{sort } n\text{-}k)$???

$p = 30,000$

$n = 20,000$

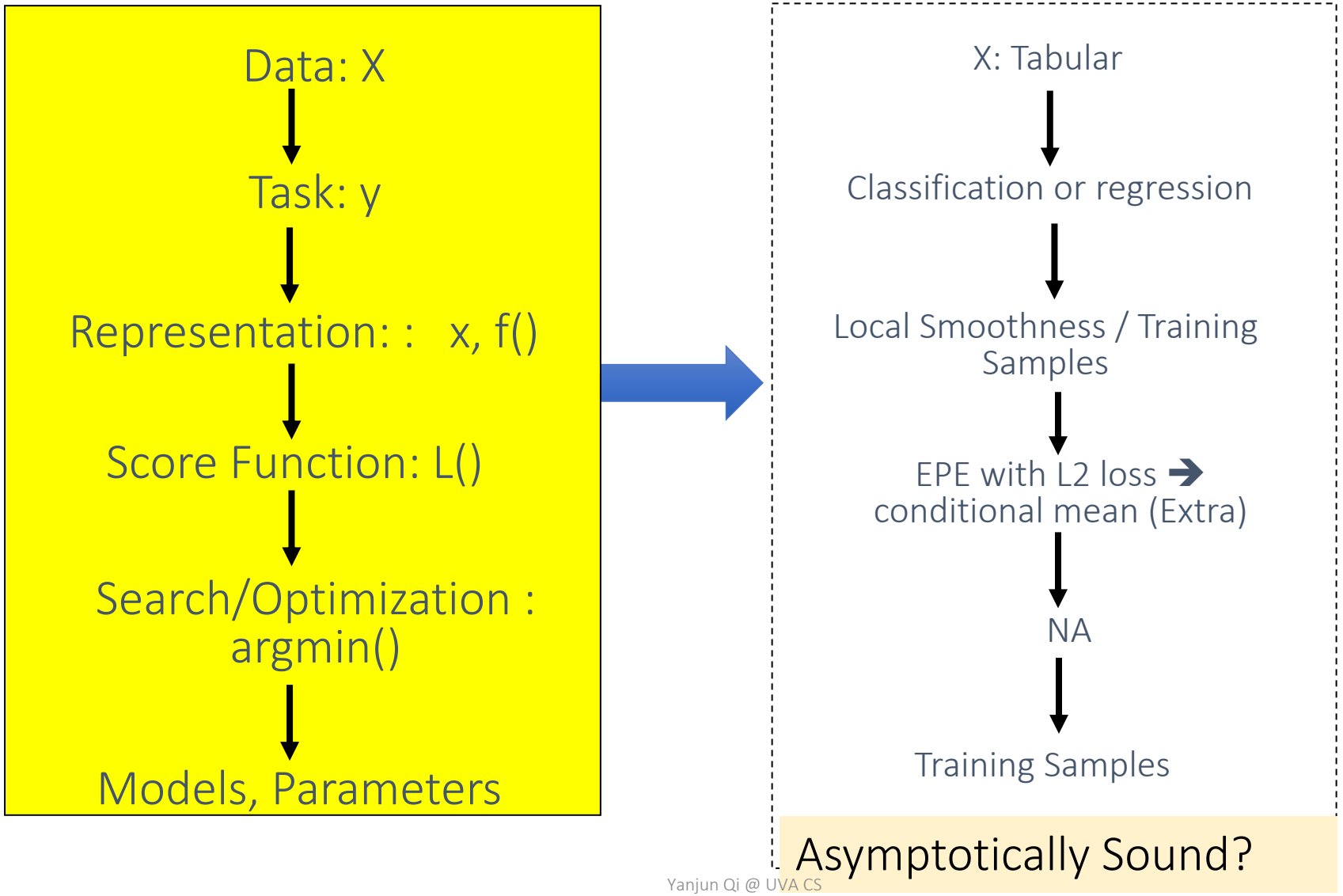
NN Search by KD Tree





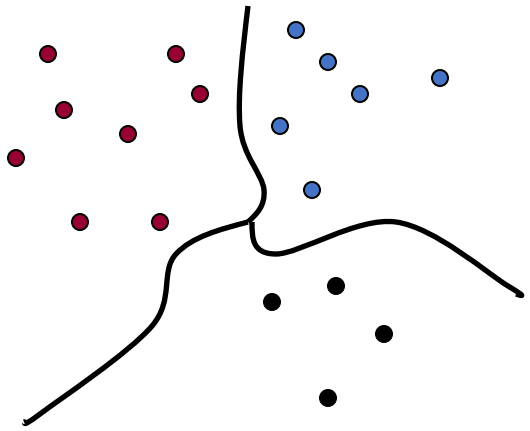
We traverse the tree looking for the nearest neighbor of the query point.

Today: **K-nearest-neighbor(regressor or classifier)**

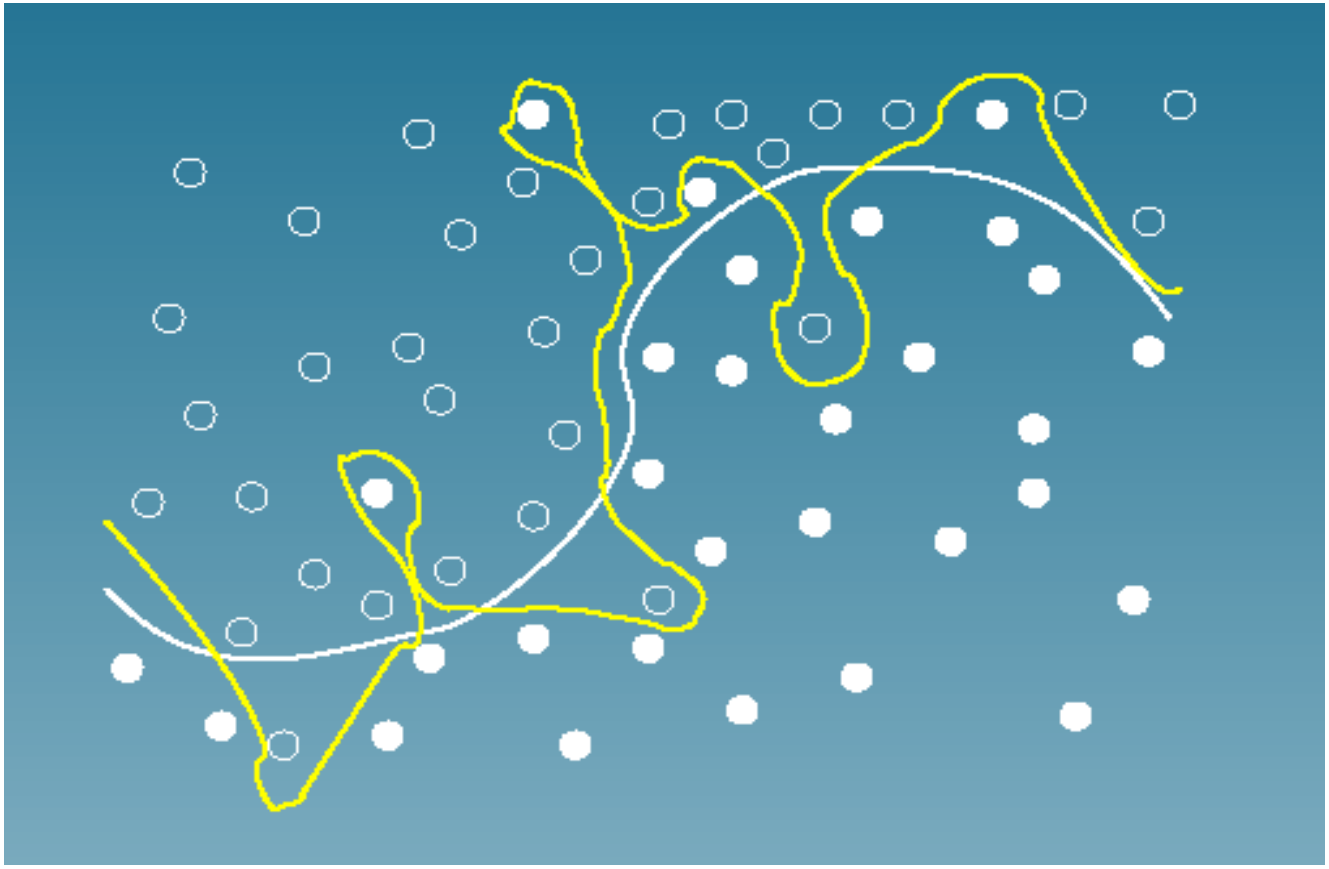


kNN Is Close to Optimal Asymptotically (EXTRA, when infinite training)

- Cover and Hart 1967
- Asymptotically, the error rate of 1-nearest-neighbor classification is less than twice the Bayes rate [error rate of classifier knowing model that generated data]
- In particular, asymptotic error rate is 0 if Bayes rate is 0.
- Decision boundary:



Is kNN ideal? … See Extra



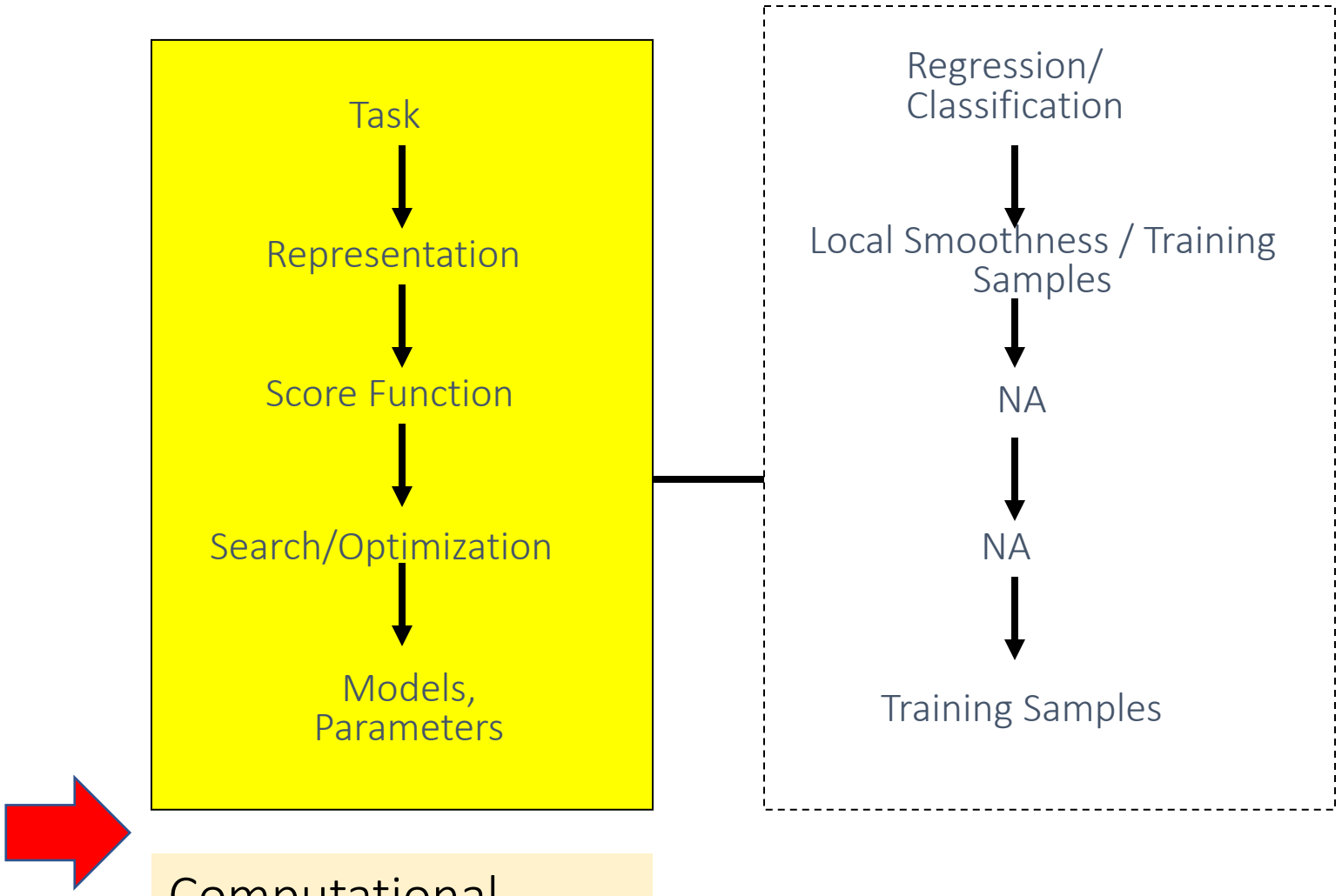

Thank You

References

- Prof. Tan, Steinbach, Kumar's "Introduction to Data Mining" slide
- Prof. Andrew Moore's slides
- Prof. Eric Xing's slides
- Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.

Extra: More about K-nearest-neighbor

K-Nearest Neighbor



Computational Scalable?

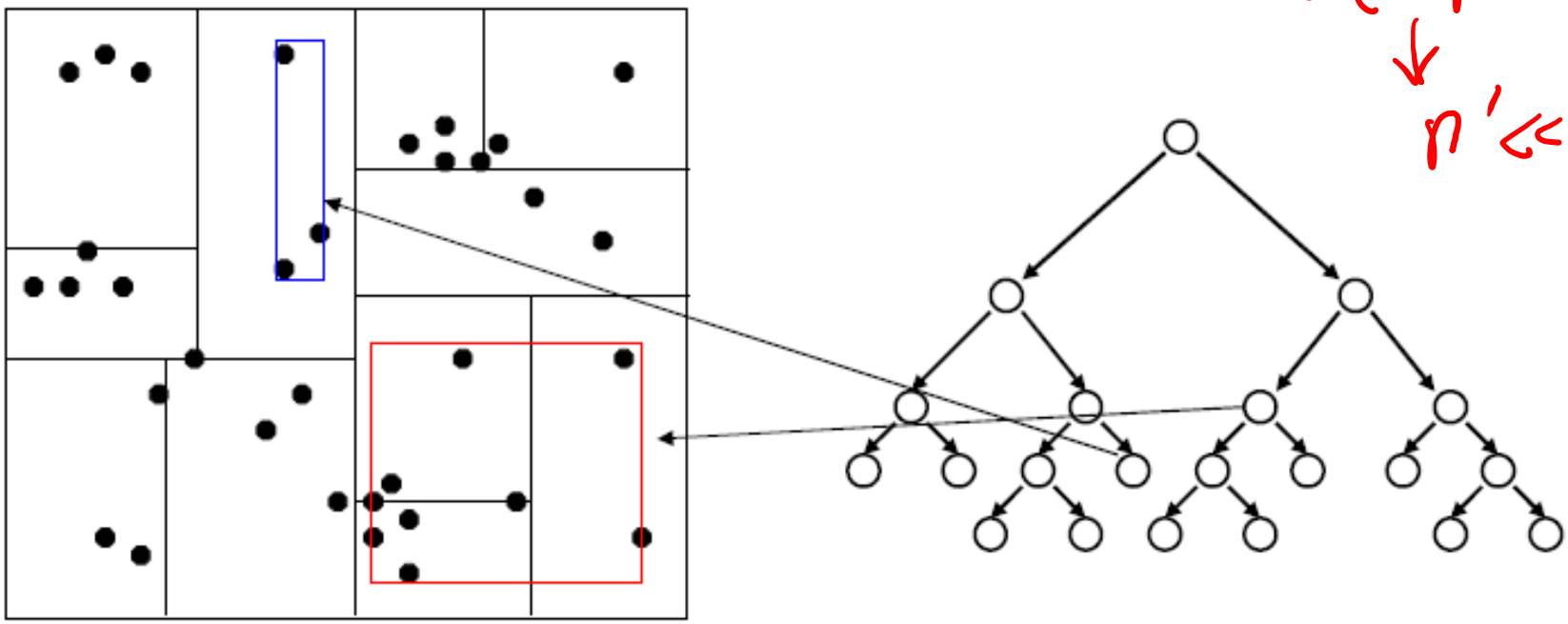
Computational Time Cost

	Train (n)	Test (m=1)
Linear Regression	$O(np^2+p^3)$	$O(p)$
KNN	$O(1)$	$O(np)+$ $O(\text{sort } n\text{-}k)$???

$p = 30,000$

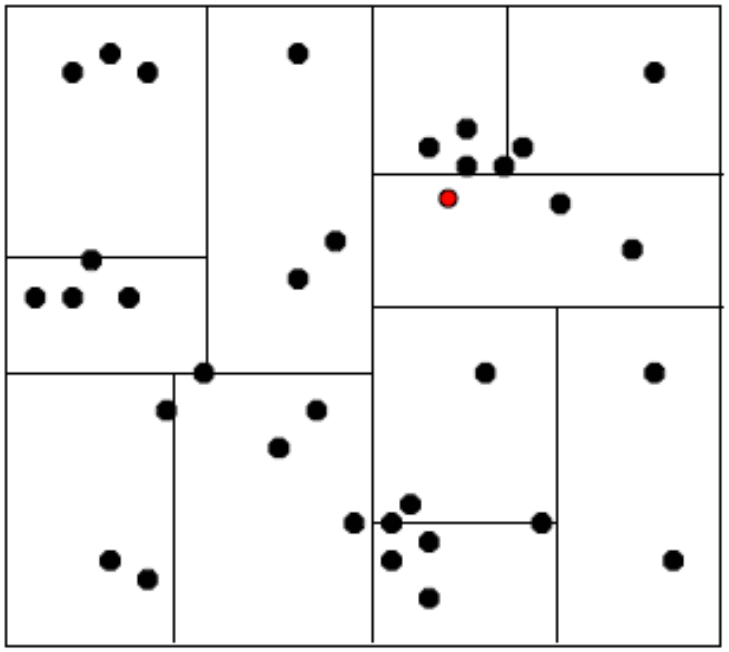
$n = 20,000$

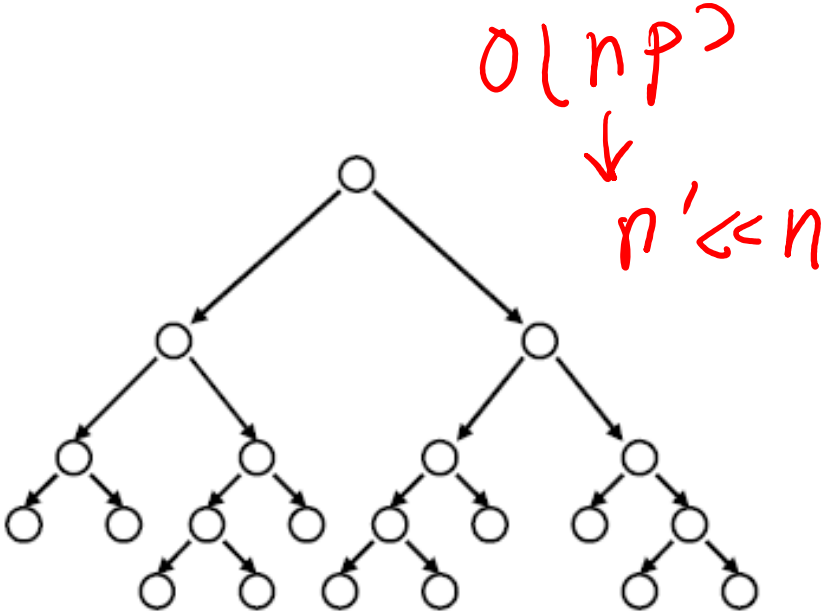
KD Tree for NN Search



- Each node contains
  - Children information
  - The tightest box that bounds all the data points within the node.

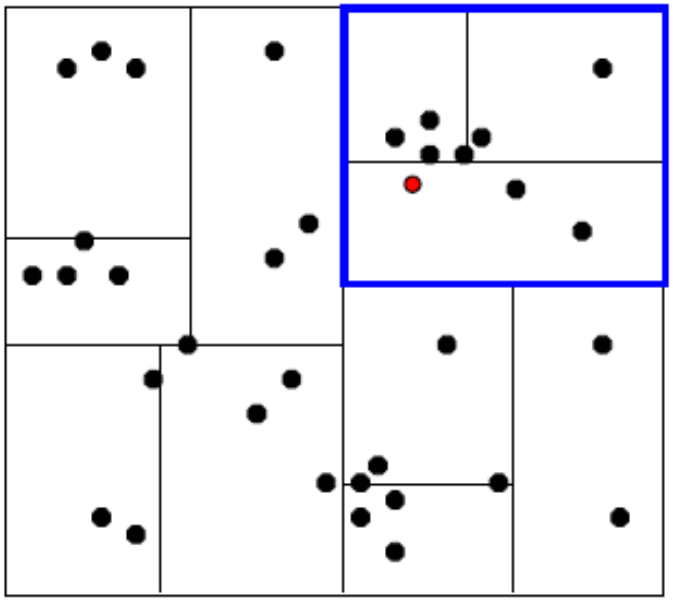
NN Search by KD Tree

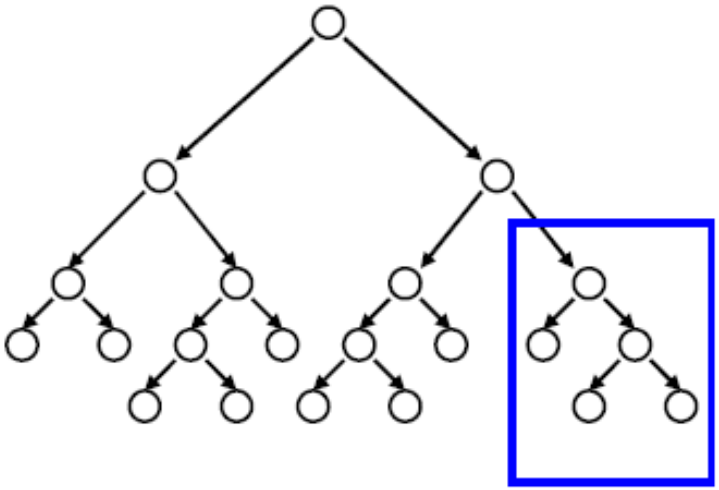




We traverse the tree looking for the nearest neighbor of the query point.

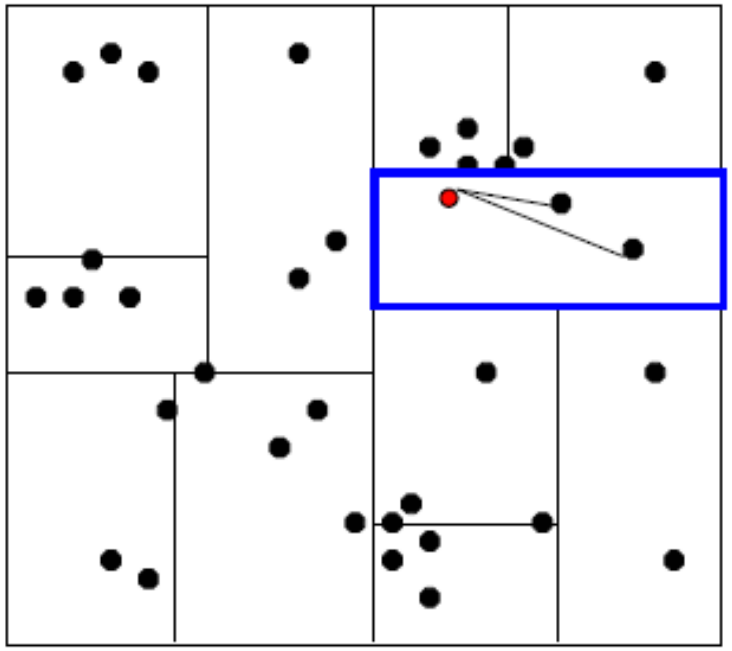
NN Search by KD Tree

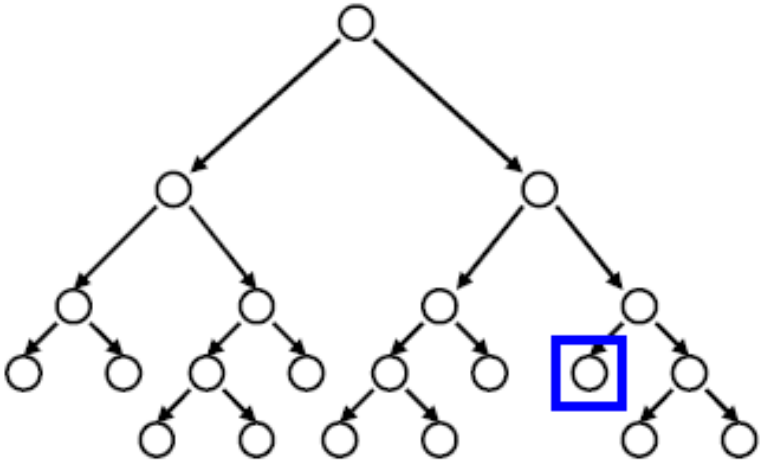




Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

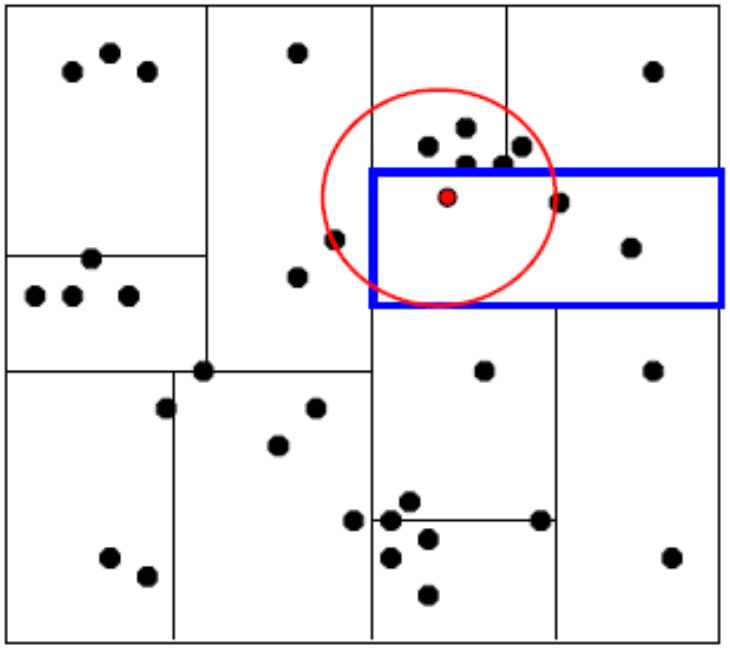
NN Search by KD Tree

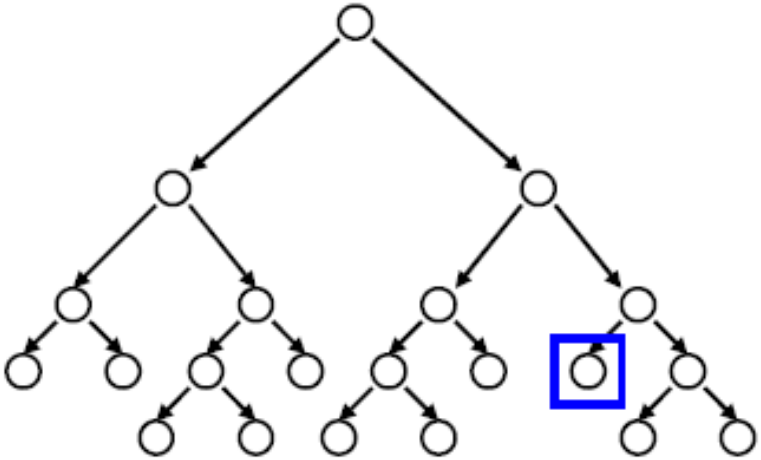




When we reach a leaf node: compute the distance to each point in the node.

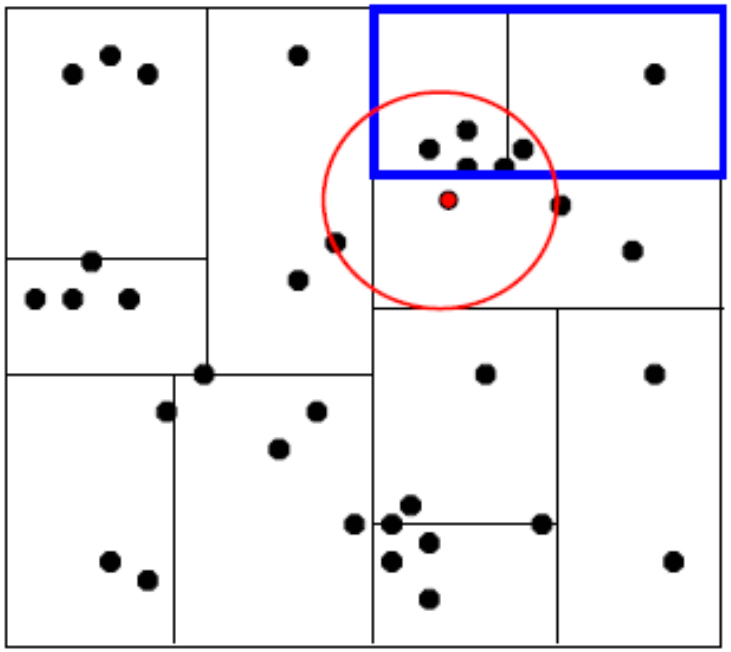
NN Search by KD Tree

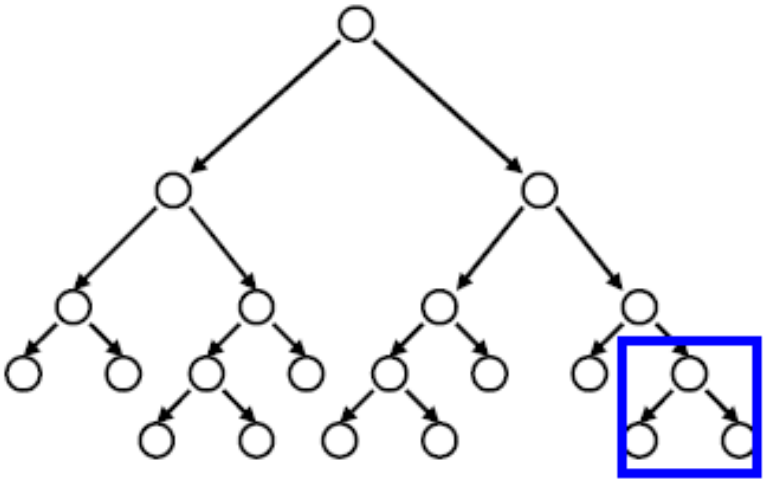




When we reach a leaf node: compute the distance to each point in the node.

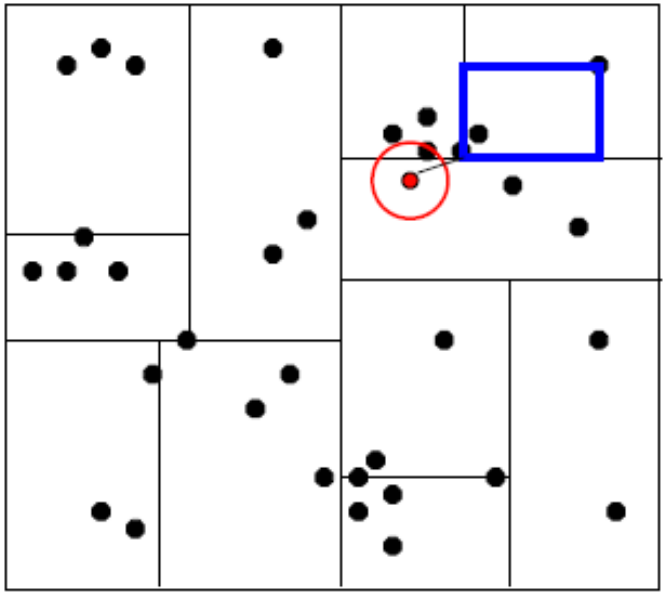
NN Search by KD Tree

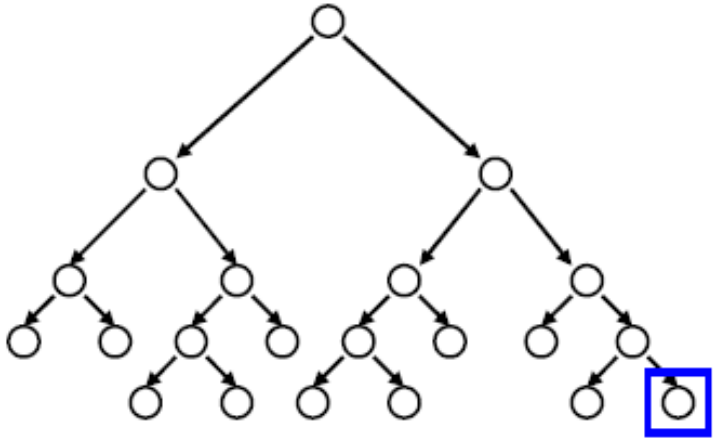




Then we can backtrack and try the other branch at each node visited.

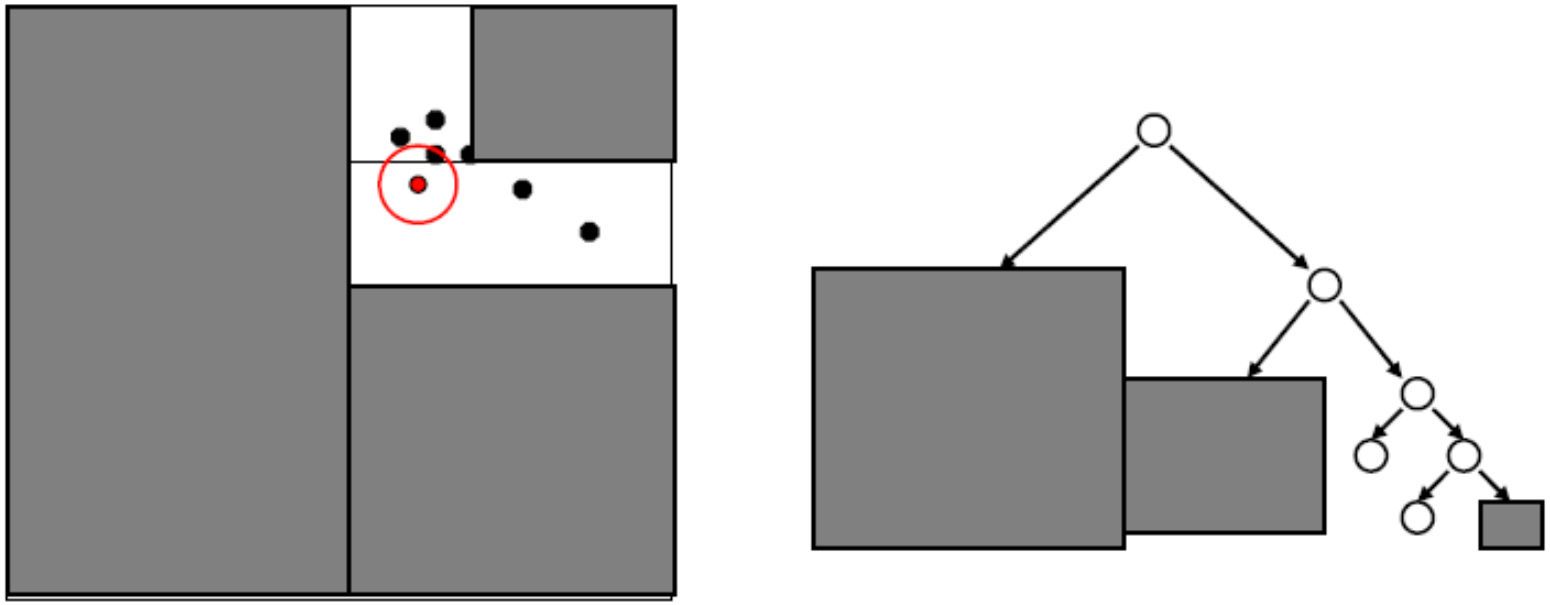
NN Search by KD Tree





Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

NN Search by KD Tree



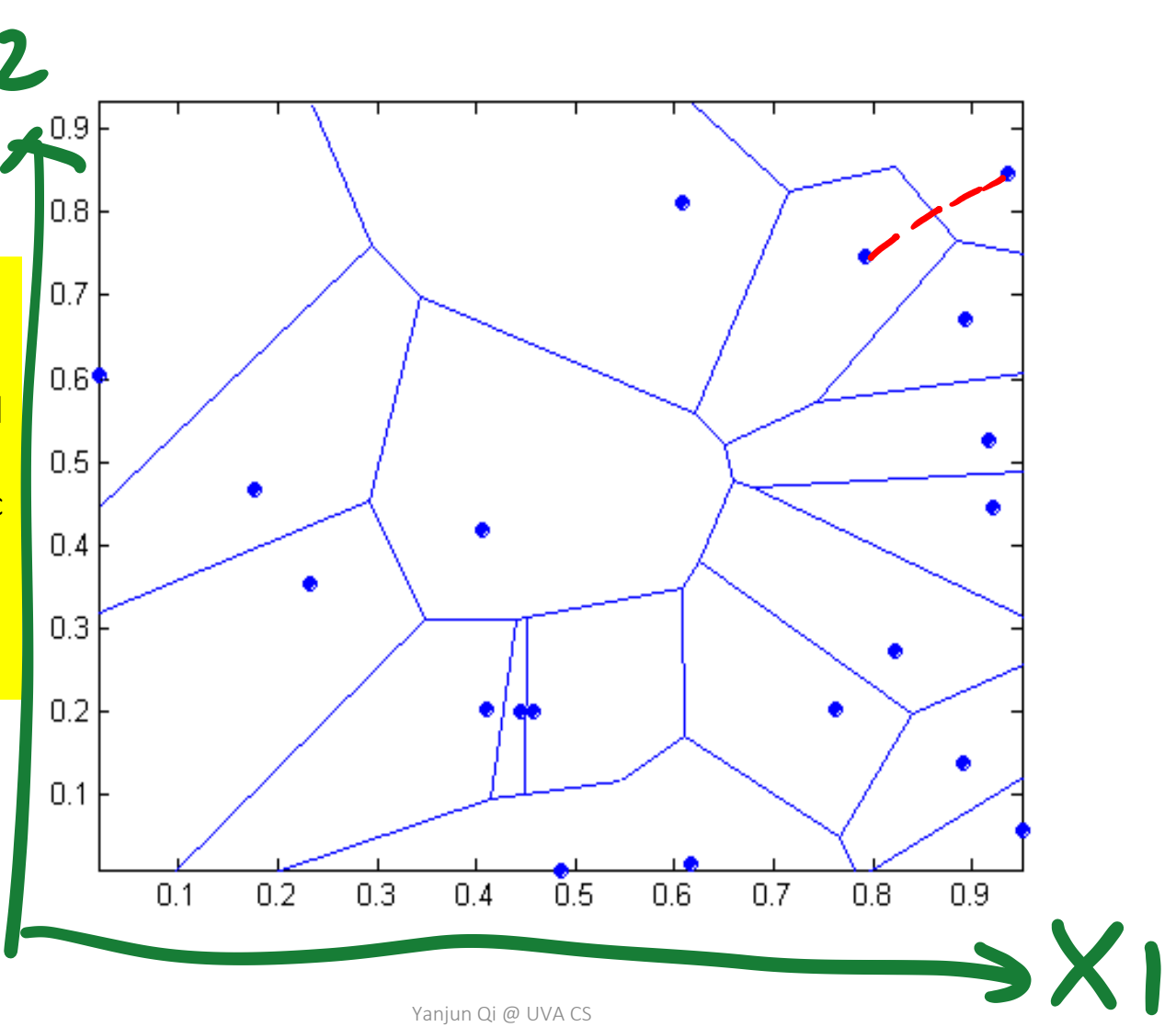
Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Curse of Dimensionality

- Imagine instances described by 20 attributes, but only 2 are relevant to target function
- Curse of dimensionality: nearest neighbor is easily mislead when high dimensional X
- Consider N data points uniformly distributed in a p-dimensional unit ball centered at origin. Consider the nn estimate at the original. The mean distance from the origin to the closest data point is:

$$d(p,N) = \left(1 - 2^{-1/N}\right)^{1/p} \approx 1 - \frac{\log N}{p}$$

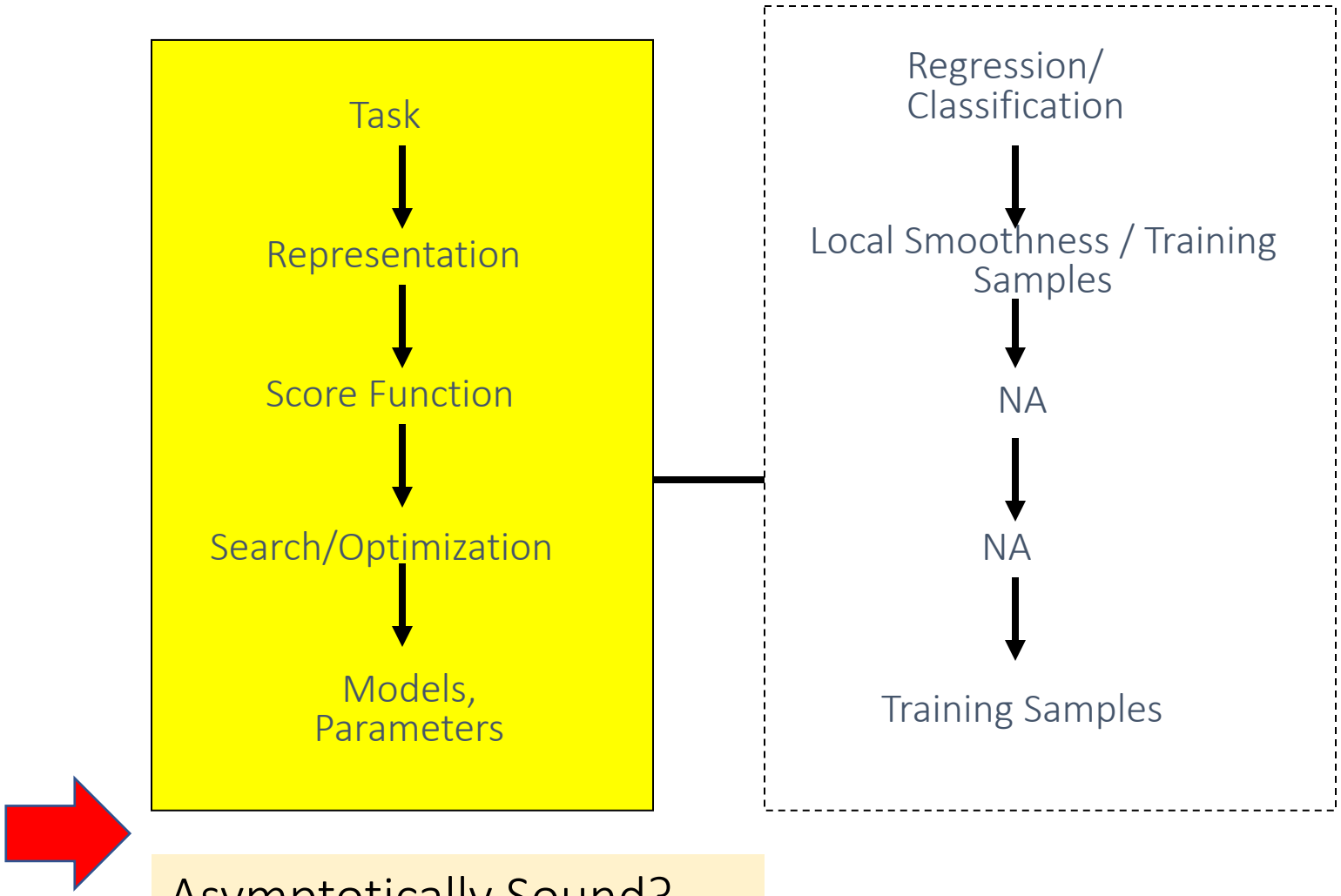
e.g., 1-nearest neighbor



Voronoi diagram:

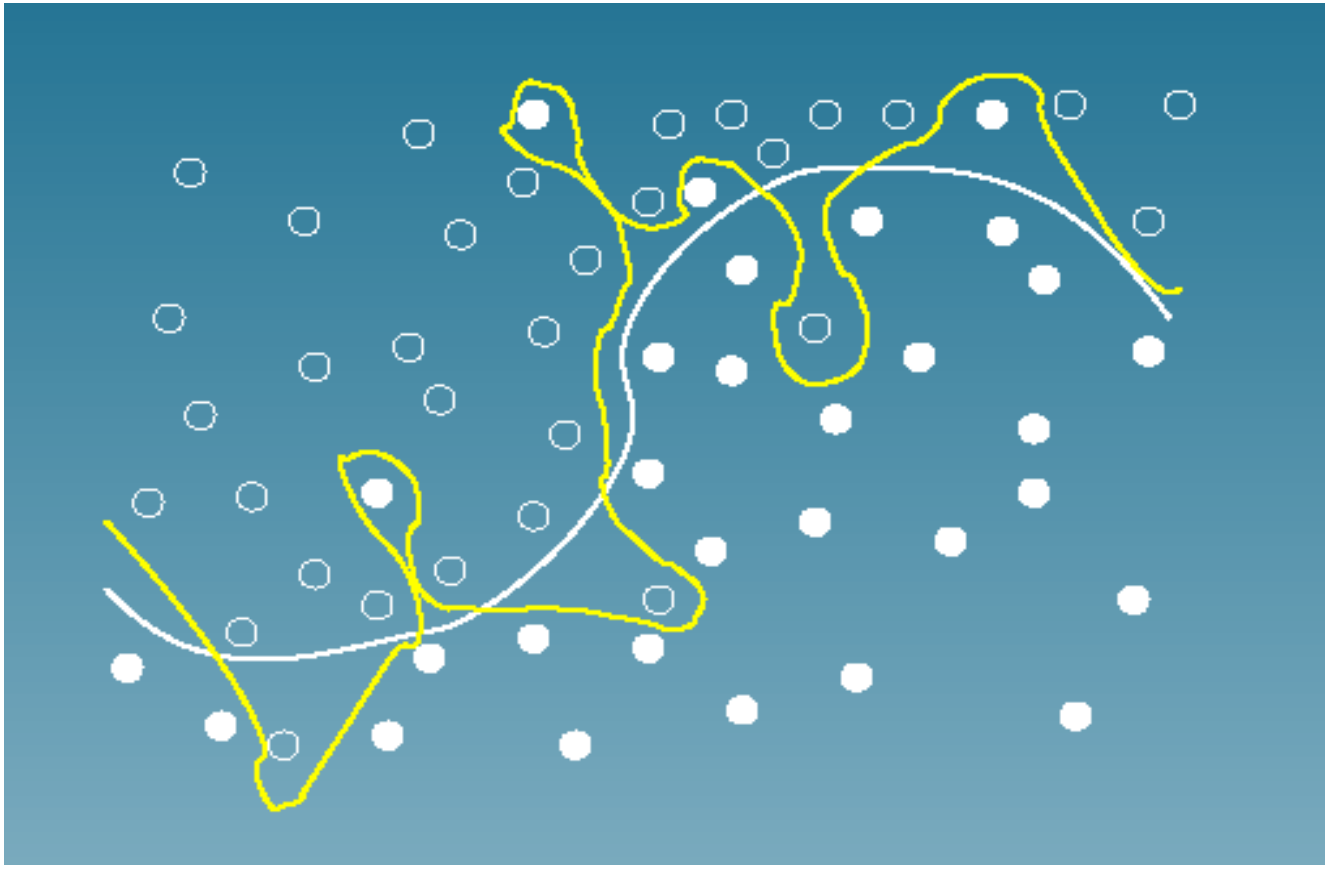
partitioning of a plane into regions based on distance to points in a specific subset of the plane.

K-Nearest Neighbor



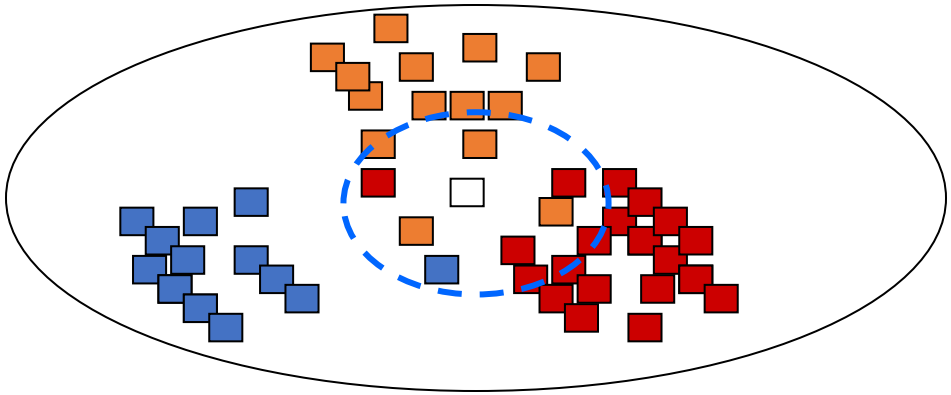
Asymptotically Sound?

Is kNN ideal? … See Extra



Probabilistic Interpretation of KNN

- Estimate conditional probability Pr(y|x)
  - Count of data points in class y in the neighborhood of x



Bayes Classifier

- ***Bayes classifier*** is the best classifier which minimizes the probability of classification error.
- A classifier becomes the ***Bayes classifier*** if the density estimates converge to the true densities
  - when an infinite number of samples are used
  - The resulting error is the ***Bayes error,*** the smallest achievable error given the underlying distributions.

The Bayes Rule

- What we have just did leads to the following general expression:

$$P(Y \mid X) = \frac{P(X \mid Y)p(Y)}{P(X)}$$

This is Bayes Rule

Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London,* **53:370-418**

The Bayes Decision Rule for Minimum Error

- The *a posteriori* probability of a sample

$$P(Y=i \mid X) = \frac{p(X \mid Y=i)P(Y=i)}{p(X)} = \frac{\pi_i p_i(X)}{\sum_i \pi_i p_i(X)} \equiv q_i(X)$$

- Bayes Test:

- Likelihood Ratio:

$$\ell(X) =$$

- Discriminant function:

$$h(X) =$$

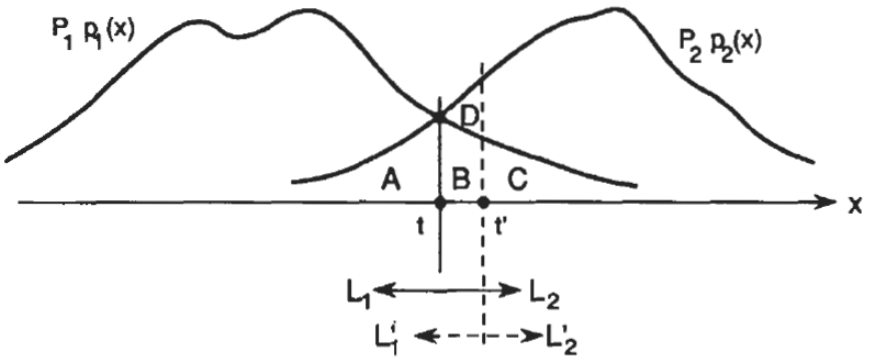
Bayes Error

- We must calculate the *probability of error*
  - the probability that a sample is assigned to the wrong class
- Given a datum X, what is the *risk*?

$$r(X) = \min[q_1(X), q_2(X)]$$

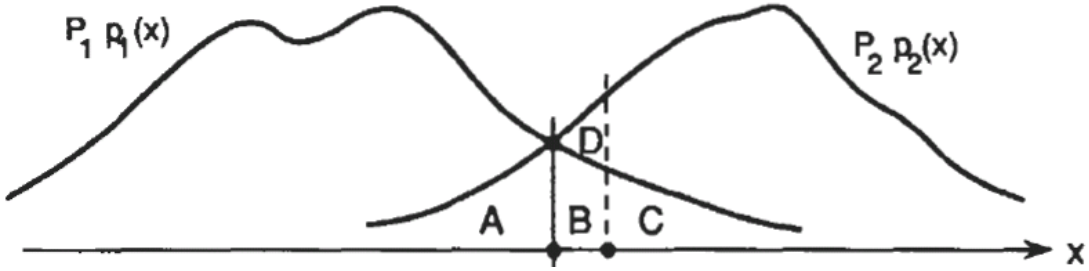
- The Bayes error (the expected risk):

$$\begin{aligned}
\epsilon &= E[r(X)] = \int r(x)p(x)dx \\
&= \int \min[\pi_i p_1(x), \pi_2 p_2(x)]dx \\
&= \pi_1 \int_{L_1} p_1(x)dx + \pi_2 \int_{L_2} p_2(x)dx \\
&= \pi_1\epsilon_1 + \pi_2\epsilon_2
\end{aligned}$$



More on Bayes Error

- Bayes error is the lower bound of probability of classification error

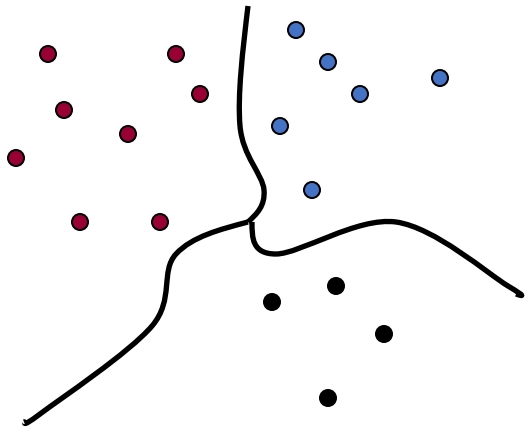


- Bayes classifier is the theoretically best classifier that minimizes probability of classification error
- Computing Bayes error is in general a very complex problem. Why?
  - Density estimation:
  - Integrating density function:

$$\epsilon_1 = \int_{\ln(\pi_1/\pi_2)}^{+\infty} p_1(x)dx \qquad \epsilon_2 = \int_{-\infty}^{\ln(\pi_1/\pi_2)} p_2(x)dx$$

kNN Is Close to Optimal

- Cover and Hart 1967
- Asymptotically, the error rate of 1-nearest-neighbor classification is less than twice the Bayes rate [error rate of classifier knowing model that generated data]
- In particular, asymptotic error rate is 0 if Bayes rate is 0.
- Decision boundary:



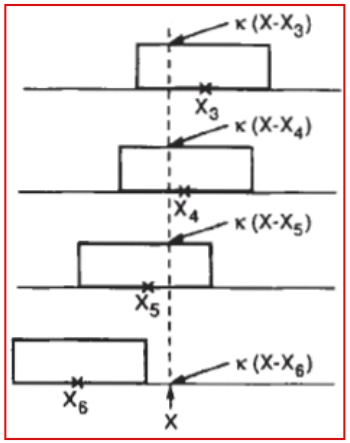
Where does kNN come from?

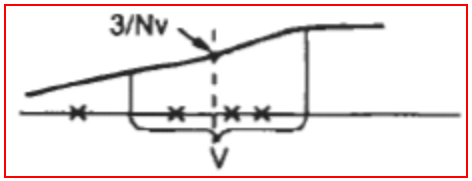
- How to estimation p(X) ?
- Nonparametric density estimation
  - Parzen density estimate

    E.g. (Kernel density est.):

$$\hat{p}(X) = \frac{1}{N}\sum_{i=1}^{N}\kappa(X - x_i)$$

    More generally: $\hat{p}(X) = \frac{1}{N}\frac{k(X)}{V}$





Where does kNN come from?

- Nonparametric density estimation
  - Parzen density estimate $\hat{p}(X) = \frac{1}{N}\frac{k(X)}{V}$
  - kNN density estimate $\hat{p}(X) = \frac{1}{N}\frac{(k-1)}{V(X)}$
- Bayes classifier based on kNN density estimator:

$$h(X) \quad = \quad -\ln\frac{p_1(X)}{p_2(X)} = -\ln\frac{(k_1-1)N_2V_2(X)}{(k_2-1)N_1V_1(X)} \begin{matrix} > \\ < \end{matrix} \ln\frac{\pi_1}{\pi_2}$$

Voting kNN classifier

Pick K_1 and K_2 implicitly by picking $K_1+K_2=K$, $V_1=V_2$, $N_1=N_2$

Asymptotic Analysis

- Condition risk: $r_k(X,X_{NN})$
  - Test sample X
  - NN sample X_{NN}
  - Denote the event X is class I as X«I

  - Assuming k=1

$$r_1(X, X_{NN}) = Pr\Big\{\{X \leftrightarrow 1 \ \& \ X_{NN} \leftrightarrow 2\} \text{ or } \{X \leftrightarrow 2 \ \& \ X_{NN} \leftrightarrow 1\} | X, X_{NN}\Big\}$$
$$= Pr\Big\{\{X \leftrightarrow 1 \ \& \ X_{NN} \leftrightarrow 2\}\Big\} + Pr\Big\{\{X \leftrightarrow 2 \ \& \ X_{NN} \leftrightarrow 1\} | X, X_{NN}\Big\}$$
$$= q_1(X)q_2(X_{NN}) + q_2(X)q_1(X_{NN})$$

  - When an infinite number of samples is available, X_{NN} will be so close to X

$$r_1^*(X) = 2q_1(X)q_2(X) = 2\xi(X)$$

Asymptotic Analysis, cont.

- Recall conditional Bayes risk:

$$r^*(X) = \min[q_1(X), q_2(X)]$$

$$= \frac{1}{2} - \frac{1}{2}\sqrt{1 - 4\xi(X)}$$

$$= \sum_{i=1}^{\infty} \frac{1}{i} \binom{2i-2}{i-1} \xi^i(X)$$

This is called the MacLaurin series expansion

- Thus the asymptotic condition risk

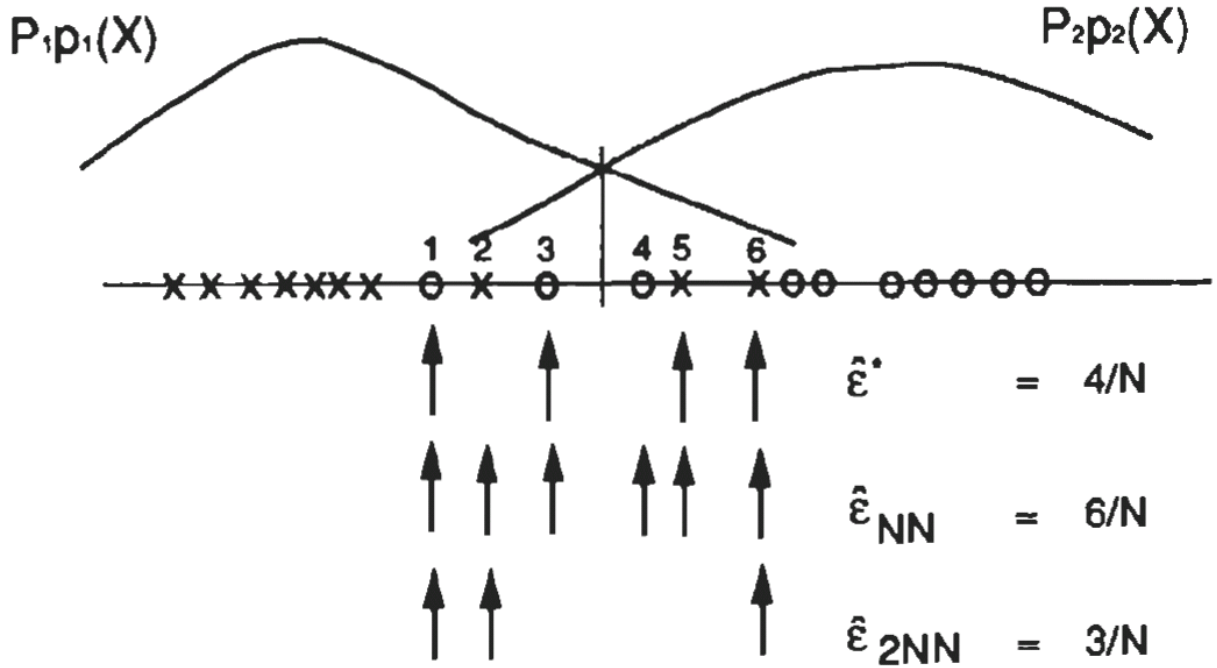
$$r_1^*(X) = 2\xi(X) \leq 2r^*(X)$$

- It can be shown that $\epsilon_1^* \leq 2\epsilon^*$
  - This is remarkable, considering that the procedure does not use any information about the underlying distributions and only the class of the single nearest neighbor determines the outcome of the decision.

In fact

$$\frac{1}{2}\epsilon^* \leq \epsilon^*_{2NN} \leq \epsilon^*_{4NN} \leq \ldots \leq \epsilon^* \leq \ldots \leq \epsilon^*_{3NN} \leq \epsilon^*_{NN} \leq 2\epsilon^*$$

- Example:



References

- Prof. Tan, Steinbach, Kumar's "Introduction to Data Mining" slide
- Prof. Andrew Moore's slides
- Prof. Jin Rong's slides about kNN
- Prof. Eric Xing's slides
- Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: Springer, 2009.