SC4000/CZ4041/CE4041: Machine Learning

Lesson 6b: *K*-NN Classifiers

Kelly KE

School of Computer Science and Engineering, NTU, Singapore

Acknowledgements: some figures are adapted from the lecture notes of the book "Introduction to Data Mining" (Chap. 5)

Training data

		F1	F2	F3	F4	F5	F6	…	
y_1	$\boxed{+1}$	1	1	0	0	1	0	…	$\boldsymbol{x}_1$
y_2	$\boxed{-1}$	0	0	1	1	0	1	…	$\boldsymbol{x}_2$
					…				
y_N	$\boxed{-1}$	0	1	0	0	1	1	…	$\boldsymbol{x}_N$

Some classification algorithm

$$f: \boldsymbol{x} \rightarrow y$$

Predicted label

$$\boxed{y^*} = f(\boldsymbol{x}^*)$$

Test data

	F1	F2	F3	F4	F5	F6	…	
?	1	1	0	0	1	0	…	$\boldsymbol{x}^*$

Inductive Learning

Training data

		F1	F2	F3	F4	F5	F6	…	
y_1	$\boxed{+1}$	1	1	0	0	1	0	…	$\boldsymbol{x}_1$
y_2	$\boxed{-1}$	0	0	1	1	0	1	…	$\boldsymbol{x}_2$
					…				
y_N	$\boxed{-1}$	0	1	0	0	1	1	…	$\boldsymbol{x}_N$

Predicted label

$\boxed{y^*}$

Test data

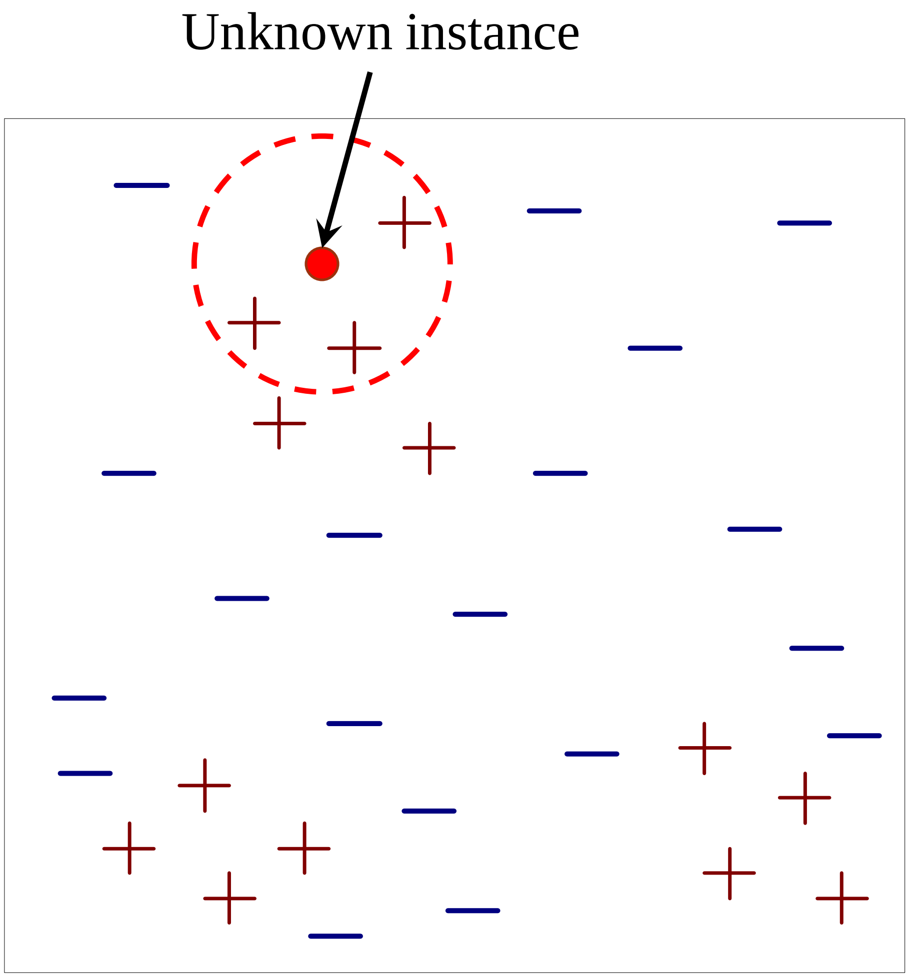
	F1	F2	F3	F4	F5	F6	…	
?	1	1	0	0	1	0	…	$\boldsymbol{x}^*$

Lazy Learning

Instance Based Classifiers

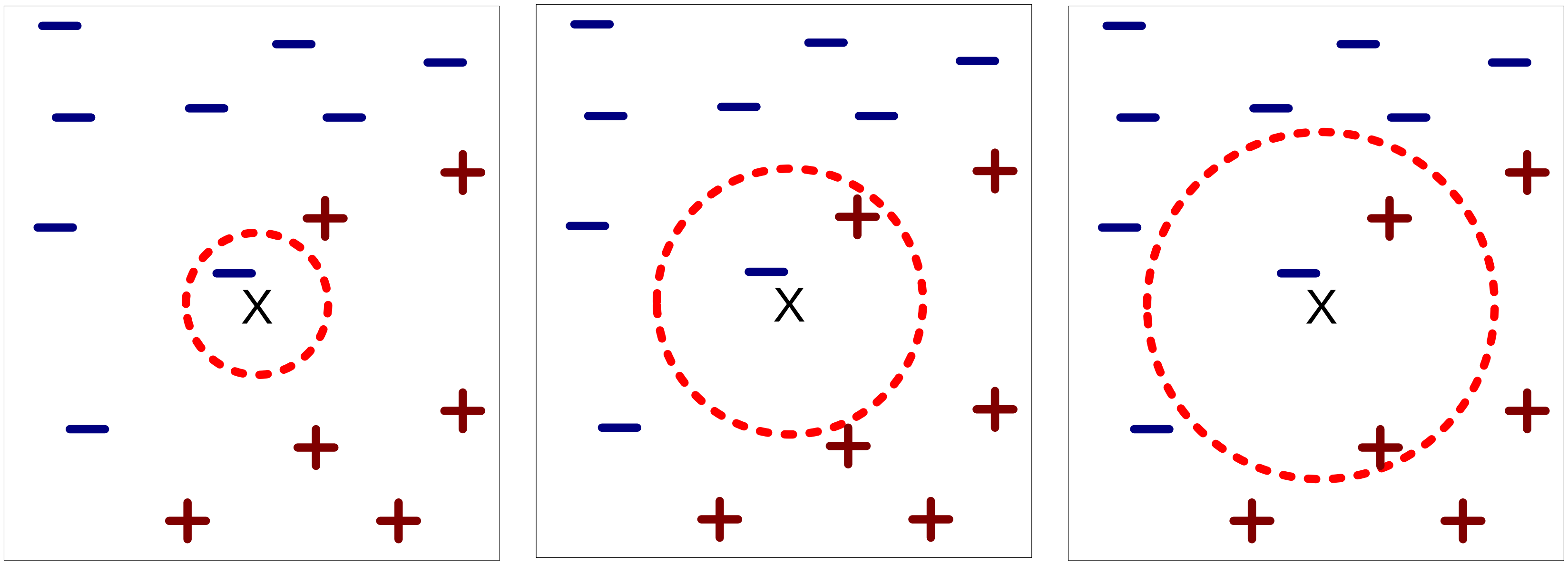
- *K*-Nearest Neighbors classifier
  - Uses *K* “closest” points (nearest neighbors) for performing classification

Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored labeled instances
  - Distance metric to compute distance between instances
  - The value of *K*, the number of nearest neighbors to retrieve
- To classify an unknown instance:
  - Compute distance to all the training instances
  - Identify *K* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of the unknown instance (e.g., by taking majority vote)

Definition of Nearest Neighbors



(a) 1-nearest neighbor

(b) 2-nearest neighbor

(c) 3-nearest neighbor

K-nearest neighbors of an instance x are data points that have the K smallest distance to x

Distance Metric

- Compute distance between two data points in a d-dimensional space:

$\boldsymbol{x}_i$

x_{i1}	x_{i2}	…	x_{id}

$\boldsymbol{x}_j$

x_{j1}	x_{j2}	…	x_{jd}

- Euclidean distance

Inner product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$:

$\boldsymbol{x}_i \cdot \boldsymbol{x}_j = \sum_{k=1}^{d} x_{ik} x_{jk}$

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{k=1}^{d} \left(x_{ik} - x_{jk}\right)^2} = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j) \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j)}$$

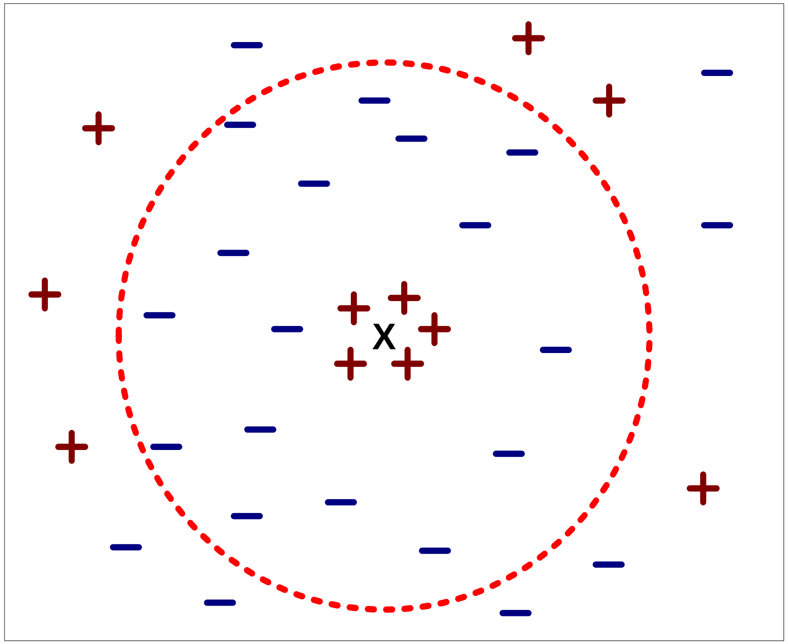
$$= \left\|(\boldsymbol{x}_i - \boldsymbol{x}_j)\right\|_2$$

L_2 norm distance

L_2 norm of $\boldsymbol{x}_i$: $\|\boldsymbol{x}_i\|_2 = \sqrt{\sum_{k=1}^{d} x_{ik}^2}$

Value of *K*

- 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



Determine Class Label

- Determine the class from nearest neighbor list
  - Take the majority vote of class labels among the K-nearest neighbors
- Given test data $\boldsymbol{x}^*$, majority voting:

Indicator function that returns 1 if its input is true, otherwise 0

$$y^* = \arg\max_c \sum_{\boxed{(\boldsymbol{x}_i, y_i) \in \mathcal{N}_{\boldsymbol{x}^*}}} \boxed{I}(c = y_i)$$

Nearest neighbors of the test instance $\boldsymbol{x}^*$

  - Every neighbor has the same impact on the classification
  - This indeed makes the algorithm sensitive to the choice of K

Revised Voting Scheme

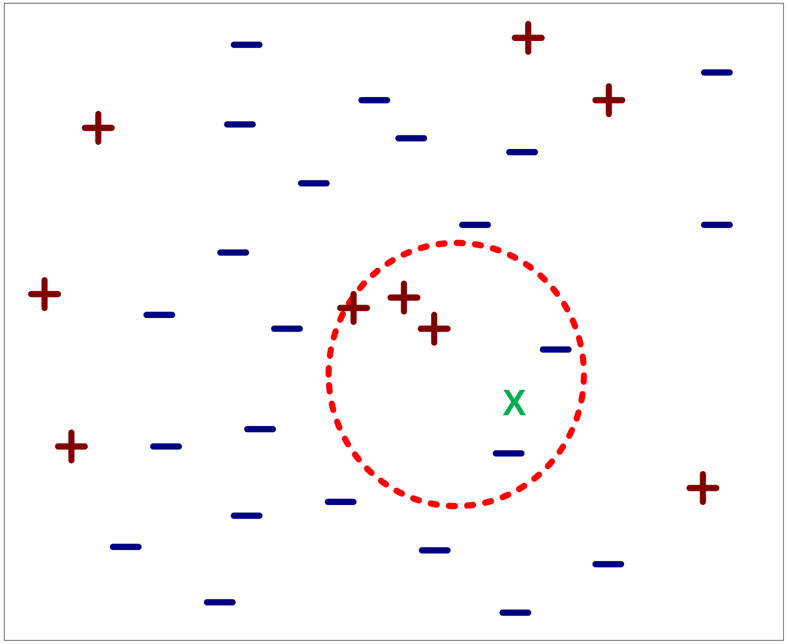
- Solution: distance-weighted voting
  - Weight the influence of each nearest neighbor $\boldsymbol{x}_i$ according to its distance to the test instance $\boldsymbol{x}^*$:

$$w_i = \frac{1}{d(\boldsymbol{x}^*, \boldsymbol{x}_i)^2}$$

$$y^* = \arg\max_c \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{N}_{\boldsymbol{x}^*}} w_i \times I(c = y_i)$$

Example

Consider a binary classification problem, and a 5-NN classifier



Training record	Class label	Distance to test instance
1	+	3
2	+	3.5
3	+	4
4	-	1.5
5	-	2

- Majority voting:

$$+ : 3 > - : 2$$

- Distance-weighted voting:

Tutorial

Other Issues

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{k=1}^{d} \left(x_{ik} - x_{jk}\right)^2}$$

- Scaling issues
  - Feature may need to be scaled to prevent distance from being dominated by some features
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 40kg to 200kg
    - income of a person may vary from \$10K to \$1M
  - Solution: normalization on features of different scales

Normalization

- Min-max normalization: to $[min_{new}, max_{new}]$
  - Example: To normalize income ranging from \$12,000 to \$98,000 to [0.0, 1.0], what is the value for \$73,600 after normalization?

$$v_{new} = \frac{v_{old} - min_{old}}{max_{old} - min_{old}}(max_{new} - min_{new}) + min_{new}$$

73,600 $\Rightarrow$ $\frac{73600 - 12000}{98000 - 12000}(1.0 - 0) + 0 = 0.716$

Normalization (cont.)

- Standardization (z-score normalization) (μ: mean, σ: standard deviation):

$$v_{new} = \frac{v_{old} - \mu_{old}}{\sigma_{old}} \quad \Rightarrow \quad \mu_{new} = 0, \text{and } \sigma_{new} = 1$$

  - Example: Let $\mu = 54{,}000$, $\sigma = 16{,}000$. What is the value for \$73,600 after standardization?

$$\frac{73600 - 54000}{16000} = 1.225$$

Summary of NN Classifier

- The *K*-NN classifier is a lazy learner
  - It does not build models explicitly.
  - “Training” is very efficient.
  - Classifying unknown test instances is relatively expensive.

Implementation

```
>>> from sklearn.neighbors import KNeighborsClassifier

…

>>> knnC = KNeighborsClassifier(n_neighbors=3)
>>> knnC.fit(X, y)
>>> pred= knnC.predict(X)
```

set number of neighbors

Build indices s.t. it is more efficient
when making predictions on test data

Thank you!