- Module 9 -
k-Nearest Neighbors

Outline

- k-Nearest Neighbors
    - Distance Metrics
    - Effects of k
- Classifier Performance Metrics

k-Nearest Neighbors (1)

- k-Nearest Neighbors (k-NN) is a **non-parametric**, **instance-based** learning algorithm
  - non-parametric: it does not make explicit assumptions about the form of the mapping function, $h(X)$: $X \to Y$
    (as opposed to parametric techniques, e.g., linear regression, $h(X)$: $Y = X\beta$)
  - instance-based: it does not explicitly learn a model, i.e., there is no training stage
    (instead, the training samples are only used during the prediction stage)
- k-NN can be used for classification and regression
  ➔ for this course, focus on multiclass classification where the input features can have continuous or discrete values

k-Nearest Neighbors (2)

- Steps to use k-NN for classification, $kNN(x, \mathcal{T}, k)$, given
  - a test sample, x
  - a training set, $\mathcal{T}$ consists of the training samples, (x_i, y_i) for $i = 1, \ldots, N$
  - a positive integer, k
  - a **distance metric**, $d(x, x_i)$ measures the distance between x and x_i $\forall\, i = 1, \ldots, N$

Step 1: For each training sample $x_i \,\forall\, i = 1, \ldots, N$, compute the distance, $d(x, x_i)$, between x and x_i

Step 2: Order the training samples by increasing distance

Step 3: Select the first k training samples that have the smallest $d(x, x_i)$, i.e., k nearest neighbors of x

Step 4: Determine the predicted value $\hat{y}$ of x by majority vote, i.e., most commonly occurring class of the k nearest neighbors

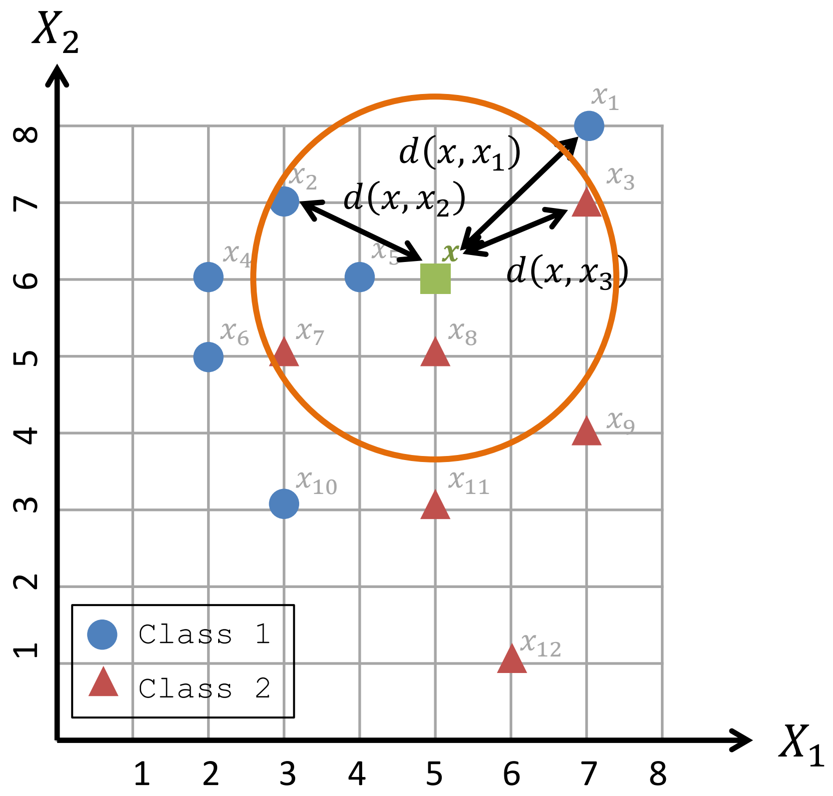
(Note: k is usually an odd number to avoid ties)

k-Nearest Neighbors: Example (3)

Example:

Consider the following training samples.

Classify test sample x using 5-NN.



5 nearest neighbors

$d(x, x_1) = 2.83$	$d(x, x_5) = 1.00$	●
$d(x, x_2) = 2.24$	$d(x, x_8) = 1.00$	▲
$d(x, x_3) = 2.24$	$d(x, x_2) = 2.24$	●
$d(x, x_4) = 3.00$	$d(x, x_3) = 2.24$	▲
$d(x, x_5) = 1.00$	$d(x, x_7) = 2.24$	▲
$d(x, x_6) = 3.16$	$d(x, x_1) = 2.83$	
$d(x, x_7) = 2.24$	$d(x, x_9) = 2.83$	
$d(x, x_8) = 1.00$	$d(x, x_4) = 3.00$	Majority vote
$d(x, x_9) = 2.83$	$d(x, x_{11}) = 3.00$	
$d(x, x_{10}) = 3.61$	$d(x, x_6) = 3.16$	
$d(x, x_{11}) = 3.00$	$d(x, x_{10}) = 3.61$	
$d(x, x_{12}) = 5.10$	$d(x, x_{12}) = 5.10$	▲

sort

$x \rightarrow$ ▲ Class 2

k-Nearest Neighbors: Distance Metrics (4)

- The distance metrics, $d(x, x')$, where x and x' are vectors of p features indexed by j, for $j = 1, \dots, p$, are computed as follows:
  - for continuous features
    - L^2 norm (Euclidean distance):

$$d(x, x') = \sqrt{\sum_{j=1}^{p} (x_j - x'_j)^2}$$

    - L^1 norm (Manhattan distance):

$$d(x, x') = \sum_{j=1}^{p} |x_j - x'_j|$$

  - for categorical features
    - Hamming distance:

Indicator function:
$$I(condition) = \begin{cases} 1, & \text{if } condition = true \\ 0, & \text{otherwise} \end{cases}$$

$$d(x, x') = \sum_{j=1}^{p} I(x_j \neq x'_j)$$

k-Nearest Neighbors: Distance Metrics (5)

- Note that the distance metric is dependent on the scales of the features

  (e.g., if a feature changes measurements from centimeters to miles while keeping other features the same, a different set of nearest neighbors will be selected)

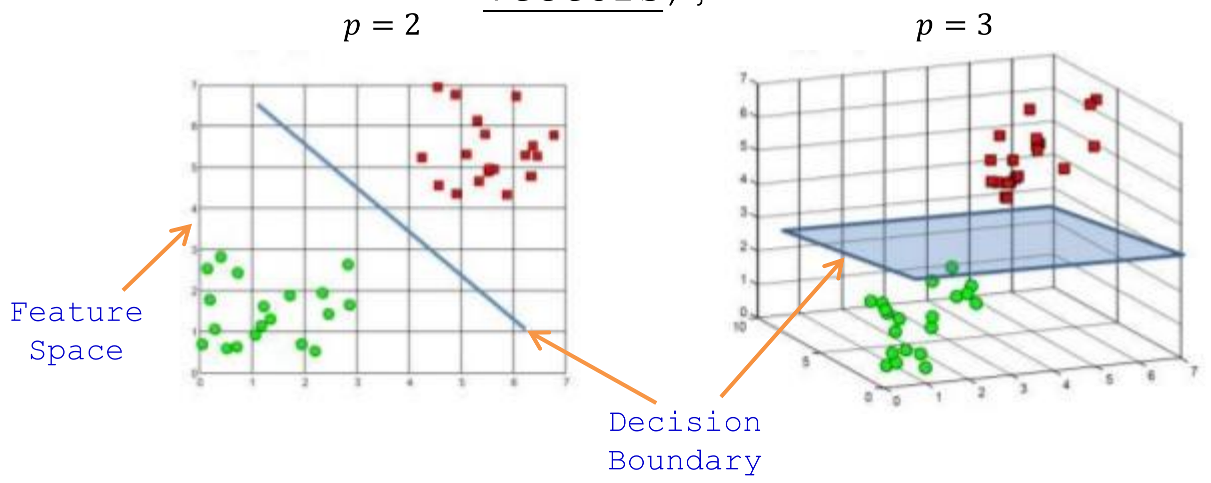
➔ to avoid this, it is common to standardize each feature j, i.e., $x_{i,j}$ is replaced with $(x_{i,j} - \mu_j)/\sigma_j$, before performing k-NN

k-Nearest Neighbors: Effects of k (6)

- In k-NN, the parameter, k, is a tuning parameter that controls the smoothness of the **decision boundary** of a k-NN classifier and its fit for the given dataset

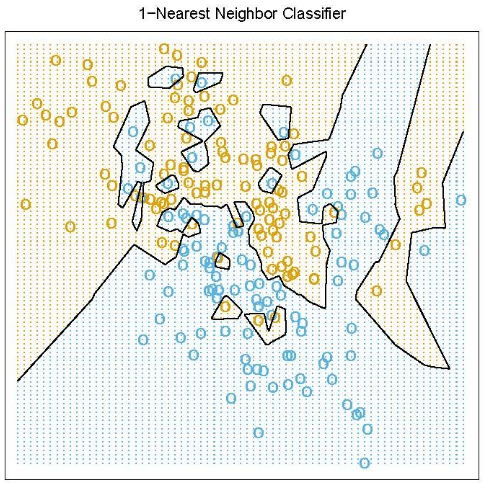
{decision boundary: partition the **feature space** (space spanned by possible values of $X_1, X_2, \ldots, X_p$) into C non-overlapping regions, one for each class

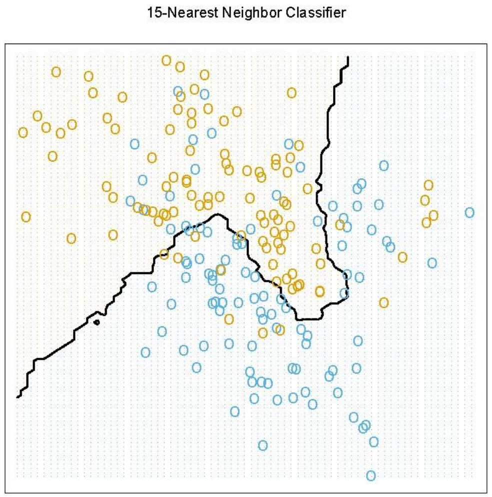
(e.g., lines for two-dimensional ($p = 2$) input vectors or hyper-surfaces for higher-dimensional ($p > 2$) input vectors)}



k-Nearest Neighbors: Effects of k (7)

- When k is small, k-NN has high flexibility but highly sensitive to outliers
  - exhibit overfitting (i.e., low bias and high variance) and the decision boundary is more jagged
- When k is large, k-NN is more resilient to outliers but has lower flexibility
  - exhibit underfitting (i.e., high bias and low variance) and the decision boundary is more smooth





➔ the best value of k is chosen using cross-validation, i.e., select k that gives the lowest average cross-validation estimate of prediction error

k-Nearest Neighbors: Cross-Validation (8)

- K-fold cross-validation for k-NN:

Step 1: Sub-divide the training set into K non-overlapping subsets of equal size

Step 2: For each trial $q = 1, \dots, K$

- all subsets except the q-th subset is used as the training set, $\mathcal{T}_q$, and the q-th subset is used as the validation set, $\mathcal{V}_q$
- for each sample x_i in $\mathcal{V}_q$
  - perform $kNN(x_i, \mathcal{T}_q, k)$ to obtain the predicted value, $\hat{y}_i$
- compute the cross-validation estimate of prediction error (or **misclassification rate**) for trial q:

$$Err_{CV}^{q} = \frac{\sum_{i=1}^{|\mathcal{V}_q|} I(y_i \neq \hat{y}_i)}{|\mathcal{V}_q|},$$

where $|\mathcal{V}_q|$ denotes the number of samples in $\mathcal{V}_q$

k-Nearest Neighbors: Cross-Validation (9)

Step 3: Compute the average cross-validation estimate of prediction error, Err_{CV}, across K trials

$$Err_{CV} = \frac{1}{K}\sum_{q=1}^{K} Err_{CV}^{q}$$

- To determine the best value of k (in k-NN), repeat steps 1 to 3 for all considered k values and select k that gives the lowest average cross-validation estimate of prediction error, Err_{CV}

k-Nearest Neighbors: Advantages and Limitations (10)

- **Advantages** of k-NN:
  - easy to implement
  - ability to achieve good classification performance when the training set is large
  - works with multiclass datasets

- **Limitations** of k-NN:
  - prediction stage is computationally intensive as the algorithm has to iterate through all samples in the training set

    ➔ often an issue where the training set is very large
  - high memory requirement as it has to store all of the training samples all the time
  - unable to provide information as to whether a feature is more informative than another

Classifier Performance Metrics (1)

- How to assess the performance of a classifier?
  - **Accuracy:** percentage of correct classifications
  - **Error Rate:** percentage of misclassifications (1 – **Accuracy**)

- However, accuracy or error rate assume equal costs for misclassification, but this assumption is often not valid for real applications
  - **Example:** Spam filter
    - cost of misclassifying a spam e-mail as a legitimate e-mail ➔ **LOW**
    - cost of misclassifying a legitimate e-mail as a spam e-mail ➔ **HIGH**
  - **Example:** Medical screening
    - cost of misclassifying a healthy person as a potential cancer patient ➔ **LOW**
    - cost of misclassifying a potential cancer patient as a healthy person ➔ **HIGH**

- Need to differentiate different kinds of misclassifications ➔ **Confusion Matrix**

Classifier Performance Metrics (2)

- Confusion matrix
  - **True Positives (TP):** actual class of a sample is positive and the predicted class is also positive

    (e.g., the person has cancer and the classifier also indicates that the person has cancer)
  - **True Negatives (TN):** actual class of a sample is negative and the predicted class is also negative

    (e.g., the person does not have cancer and the classifier also indicates that the person does not have cancer)
  - **False Positives (FP):** actual class of a sample is negative but the predicted class is positive

    (e.g., the person does not have cancer but the classifier indicates that the person has cancer)
  - **False Negatives (FN):** actual class of a sample is positive but the predicted class is negative

    (e.g., the person has cancer but the classifier indicates that the person does not have cancer)

		Predicted	
		Positive	Negative
Actual	Positive	# of **True Positives** (TP)	# of **False Negatives** (FN)
	Negative	# of **False Positives** (FP)	# of **True Negatives** (TN)

(false positive)

(false negative)

Classifier Performance Metrics (3)

- Different performance metrics can be calculated from the confusion matrix:
  - $Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{TP+TN}{total}$
  - $Precision = \frac{TP}{TP+FP} = \frac{TP}{predicted\ positive}$
  - $Sensitivity/Recall = \frac{TP}{TP+FN} = \frac{TP}{actual\ positive}$
  - $Specificity = \frac{TN}{TN+FP} = \frac{TN}{actual\ negative}$
  - $False\ Positive\ Rate = \frac{FP}{TN+FP} = \frac{FP}{actual\ negative}$
    $= 1 - specificity$
  - $False\ Negative\ Rate/MissRate = \frac{FN}{FN+TP}$
    $= \frac{FN}{actual\ positive}$