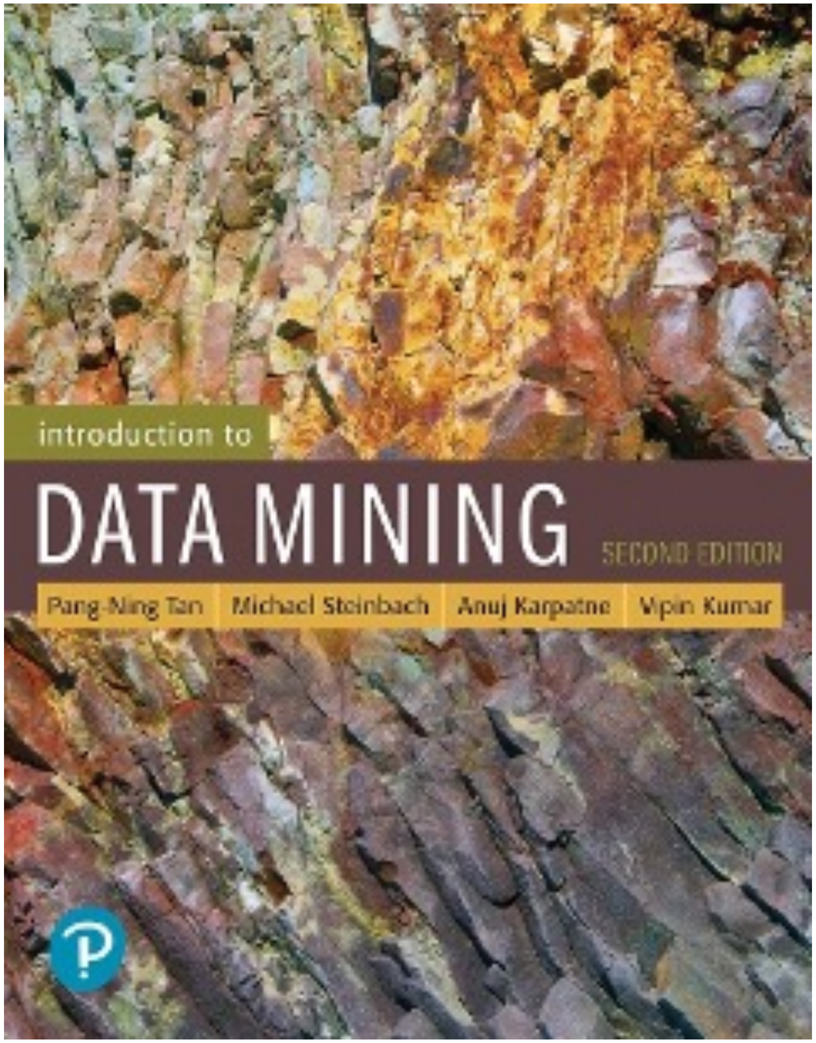
CSCE 5380/4380 – Data Mining



Chapter Four:

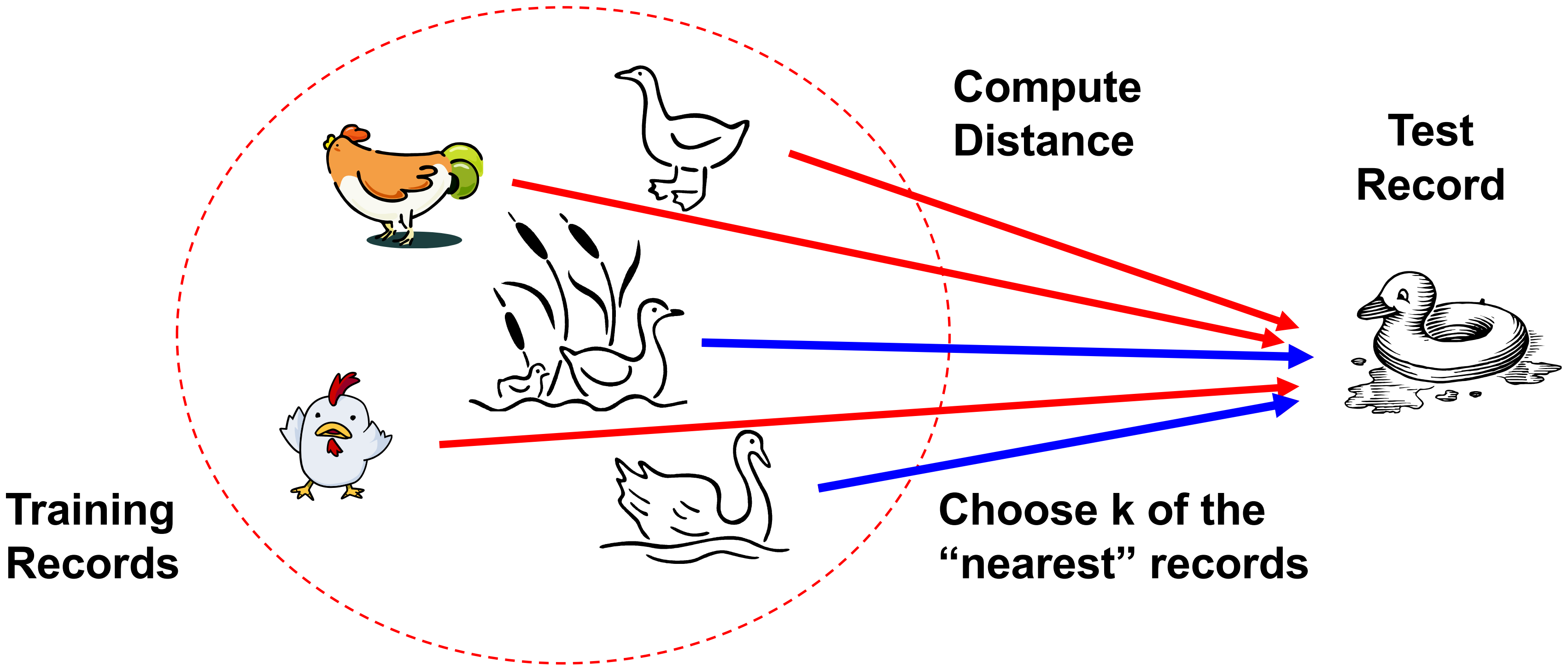
K-Nearest Neighbor & Logistic Regression

Outline

- Nearest Neighbor Classifiers
- K-NN: Algorithms
- Characteristics of K-NN
- Logistic Regression
- Learning a Logistic Regression Model
- Characteristics of Logistic Regression

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it’s probably a duck



Compute Distance

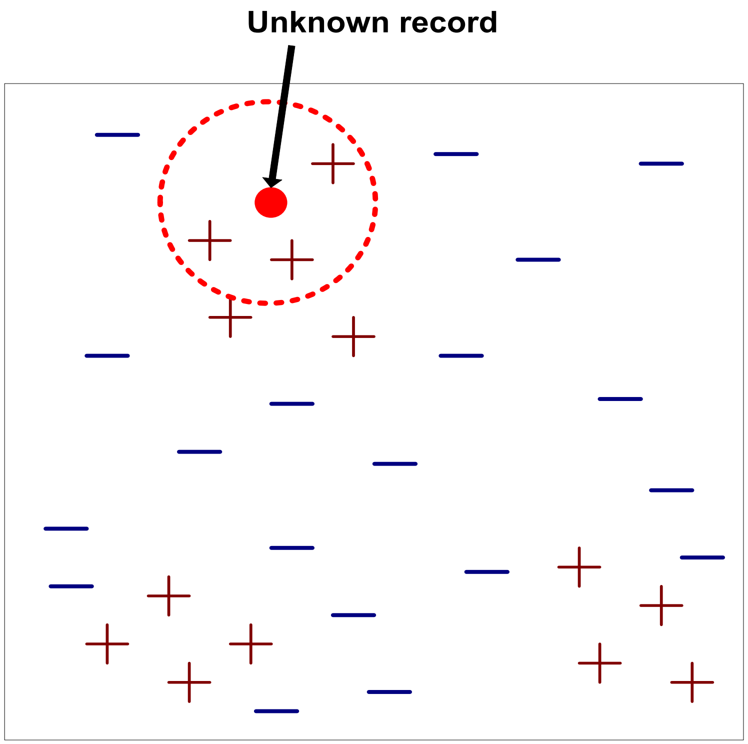
Test Record

Training Records

Choose k of the “nearest” records

Nearest Neighbor Classifiers

The *k*-nearest neighbors of a given test instance *z* refer to the *k* training examples that are closest to *z*



- Requires the following:
  - A set of labeled records
  - Proximity metric to compute distance/similarity between a pair of records
    - e.g., Euclidean distance
  - The value of *k*, the number of nearest neighbors to retrieve
  - A method for using class labels of K nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

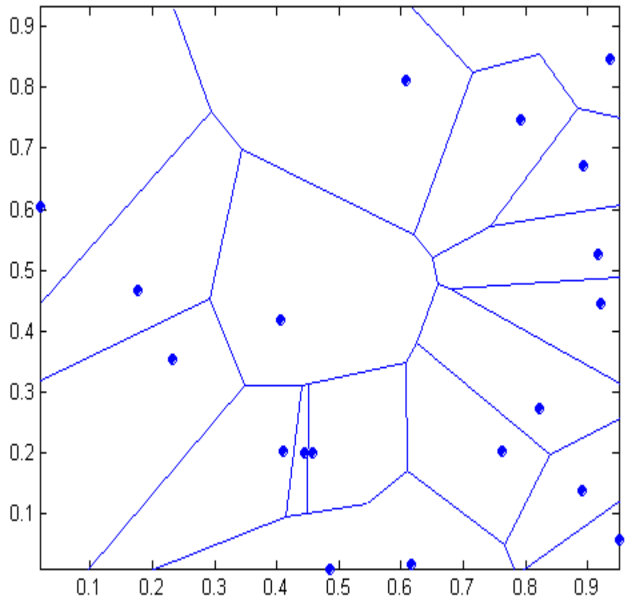
Nearest Neighbor classifiers

- Nearest neighbor classifiers are know as ***Lazy Learners*** (Strategy of delaying the process of modeling the training data until it is needed to classify the test instances)

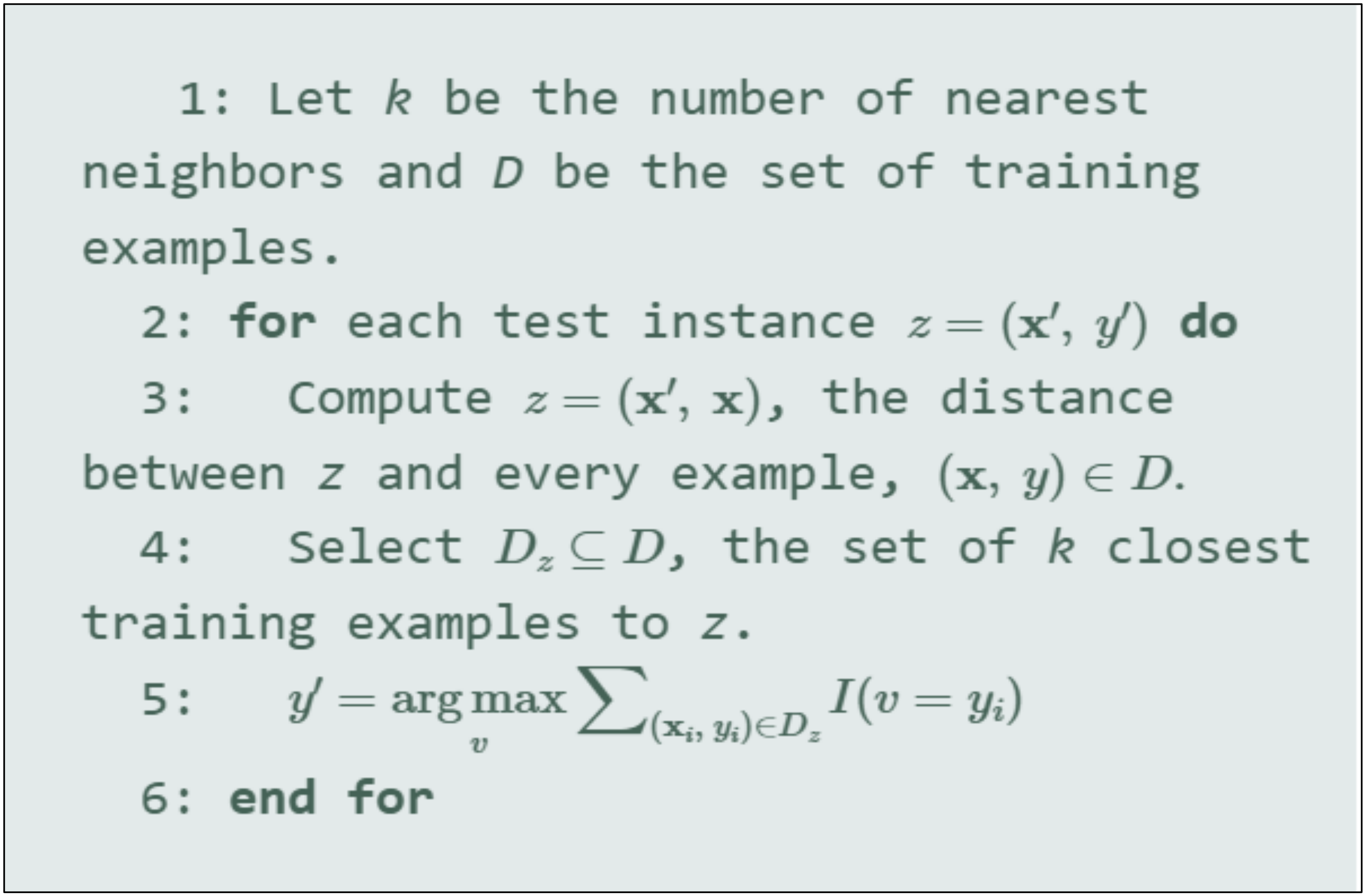
- Nearest neighbor classifiers are local classifiers

- They can produce decision boundaries of arbitrary shapes.

1-nn decision boundary is a Voronoi Diagram



K-NN: Algorithms



```
Let $k$ be the number of nearest neighbors and $D$ be the set of training examples.
for each test instance $z = (\mathbf{x}', y')$ do
    Compute $z = (\mathbf{x}', \mathbf{x})$, the distance between $z$ and every example, $(\mathbf{x}, y) \in D$.
    Select $D_z \subseteq D$, the set of $k$ closest training examples to $z$.
    $y' = \arg\max_v \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$
end for
```

How to Determine the class label of a Test Sample?

- Take the majority vote of class labels among the k-nearest neighbors

$$\text{Majority Voting}: \; y' = \underset{v}{\arg\max} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in D_z} I(v = y_i),$$

- To reduce the impact of K neighbors, Weight the vote according to distance
  - weight factor, $w = 1/d^2$

Choice of proximity measure matters

- For documents, cosine is better than correlation or Euclidean

1 1 1 1 1 1 1 1 1 1 1 0

0 1 1 1 1 1 1 1 1 1 1 1

vs

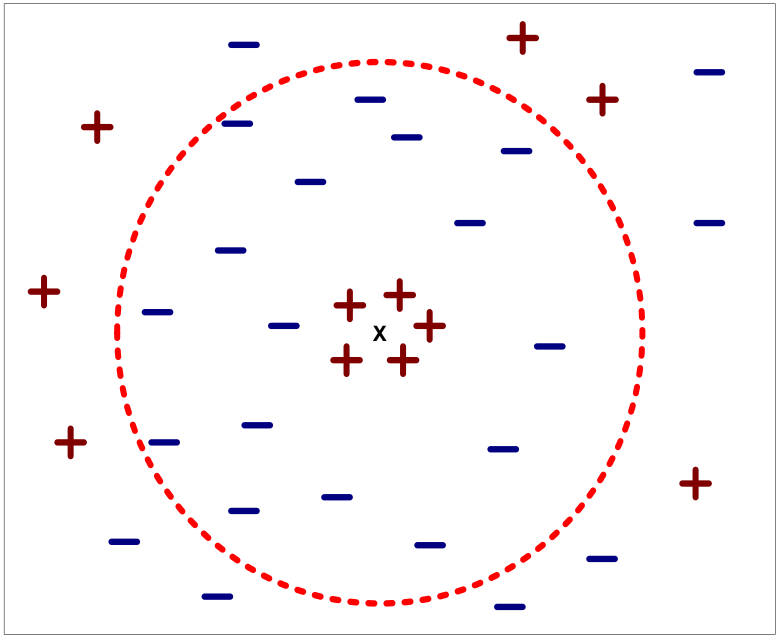
0 0 0 0 0 0 0 0 0 0 0 1

1 0 0 0 0 0 0 0 0 0 0 0

Euclidean distance = 1.4142 for both pairs, but the cosine similarity measure has different values for these pairs.

Nearest Neighbor Classification…

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



Nearest Neighbor Classification…

- **Data preprocessing is often required**
  - 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 1.5m to 1.8m
      - weight of a person may vary from 90lb to 300lb
      - income of a person may vary from $10K to $1M
  - Time series are often standardized to have 0 means a standard deviation of 1

Nearest Neighbor Classification…

- **How to handle missing values in training and test sets?**
  - Proximity computations normally require the presence of all attributes
  - Some approaches use the subset of attributes present in two instances
    - This may not produce good results since it effectively uses different proximity measures for each pair of instances
    - Thus, proximities are not comparable

Characteristics of K-NN

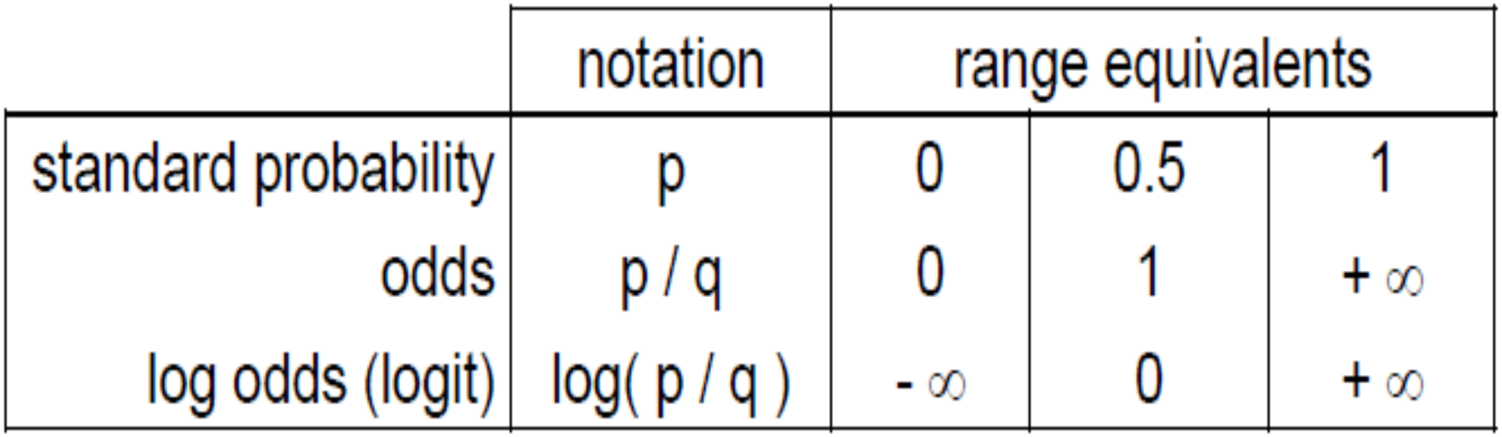
- K-NN is part of instance-based learning, which uses the training examples to make predictions for a test instance.
- Nearest neighbor classifiers make their predictions based on local information and can produce decision boundaries of arbitrary shape.
- Nearest neighbor classifiers have difficulty handling missing values in both the training and test sets since proximity computations normally require the presence of all attributes.
- Nearest neighbor classifiers can handle the presence of interacting attributes, but the presence of irrelevant and redundant attributes can adversely affect the performance of nearest neighbor classifiers.
- K-NN can produce wrong predictions unless the appropriate proximity measure and data preprocessing steps are taken.

Logistic Regression

- Logistic regression is a classification algorithm used to assign observations to a discrete set of classes (e.g. to classify instances, in some classification problems, to Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign).
- Logistic regression transforms its output using the *logistic (sigmoid) function* to return a probability value.
- logistic regression is a *probabilistic discriminative model*, which directly estimates the *odds* of a data instance *x* using its attribute values.

Different ways of expressing probability

- Consider a two-outcome probability space, where:
  - $p(O_1) = p$
  - $p(O_2) = 1 - p = q$
- Can express probability of O_1 as:



	notation	range equivalents		
standard probability	p	0	0.5	1
odds	p / q	0	1	+ ∞
log odds (logit)	log(p / q)	- ∞	0	+ ∞

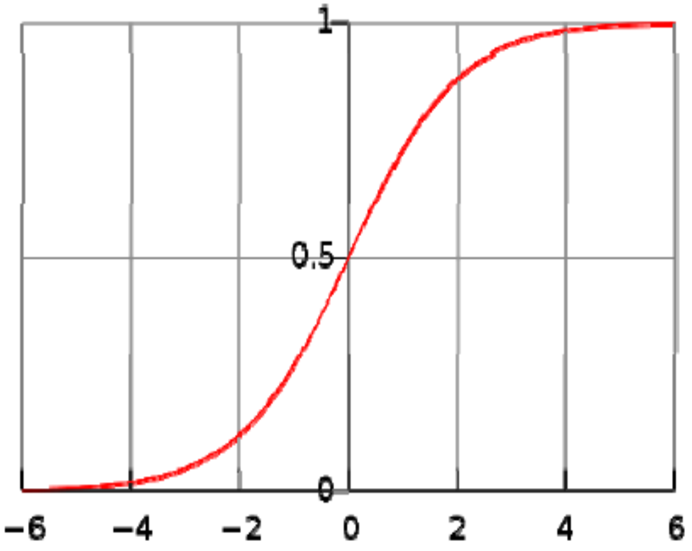
Log odds (logit function) & logistic function

- Numeric treatment of outcomes O_1 and O_2 is equivalent
  - If neither outcome is favored over the other, then log odds = 0.
  - If one outcome is favored with log odds = x, then other outcome is disfavored with log odds = $-x$.

$$z = \log\left(\frac{p}{1-p}\right) \qquad \text{logit function}$$

$$\frac{p}{1-p} = e^z$$

$$p = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}} \qquad \text{logistic function}$$



Logistic Regression: The Scenario

- A multidimensional feature space (features can be categorical or continuous).
- Outcome is discrete, not continuous.
  - We'll focus on case of two classes.
- It seems plausible that a linear decision boundary (hyperplane) will give good predictive accuracy.

Logistic Regression: The Idea

- Model consists of a vector θ in *n*-dimensional feature space (Model Parameters)
- For a point ***x*** in feature space, project it onto θ to convert it into a real number *z* in the range - ∞ to + ∞

$$z = \theta^{\top} x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

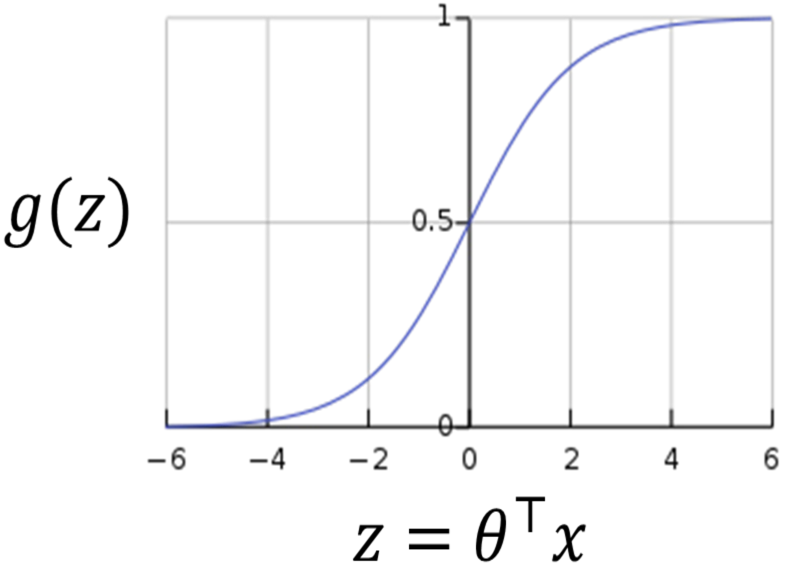
- Map *z* to the range 0 to 1 using the logistic function (Hypothesis Representation)

$$h_\theta(x) = g(\theta^{\top} x), \text{where } g(z) = \frac{1}{1+e^{-z}}$$

- Overall, logistic regression maps a point ***x*** in *n*-dimensional feature space to a value in the range 0 to 1

Logistic Regression: The Model

$$h_\theta(x) = g(\theta^\top x)$$
$$g(z) = \frac{1}{1 + e^{-z}}$$



Suppose predict “y = 1” if $h_\theta(x) \geq 0.5$

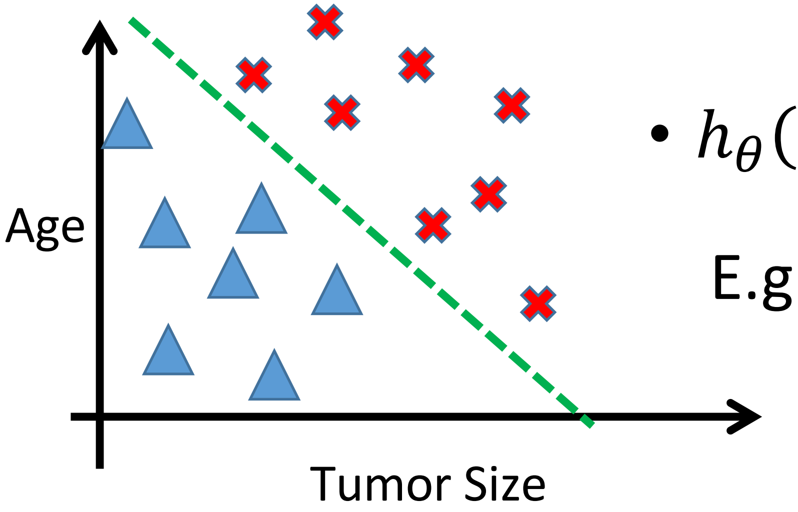
$$z = \ \theta^\top x \geq 0$$

predict “y = 0” if $h_\theta(x) < 0.5$

$$z = \ \theta^\top x < 0$$

Logistic Regression: Decision Boundary

Example:
Classification of Tumor Malignant (y=1) or Benign (y=0), based on two features (Age and Tumor Size)



- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

  E.g., $\boldsymbol{\theta_0 = -3,\ \theta_1 = 1,\ \theta_2 = 1}$

- Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Learning a Logistic Regression Model

- Need to optimize θ so the model gives the best possible reproduction of training set labels
  - Done by numerical approximation of maximum likelihood

  or

  - By using stochastic gradient descent

Logistic Regression: Cost Function

How to learn a *logistic regression model* $\boldsymbol{h_\theta}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{x})$, where $\boldsymbol{\theta} = [\boldsymbol{\theta}_{\boldsymbol{0}}, \dots, \boldsymbol{\theta}_{\boldsymbol{n}}]$ and $\boldsymbol{x} = [\boldsymbol{x}_{\boldsymbol{0}}, \dots, \boldsymbol{x}_{\boldsymbol{n}}]$?

- By minimizing the following cost function:

$$\text{Cost}(\boldsymbol{h_\theta}(\boldsymbol{x}), \mathrm{y}) = -y \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right) - (1-y)\log\left(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right)$$

- That is:

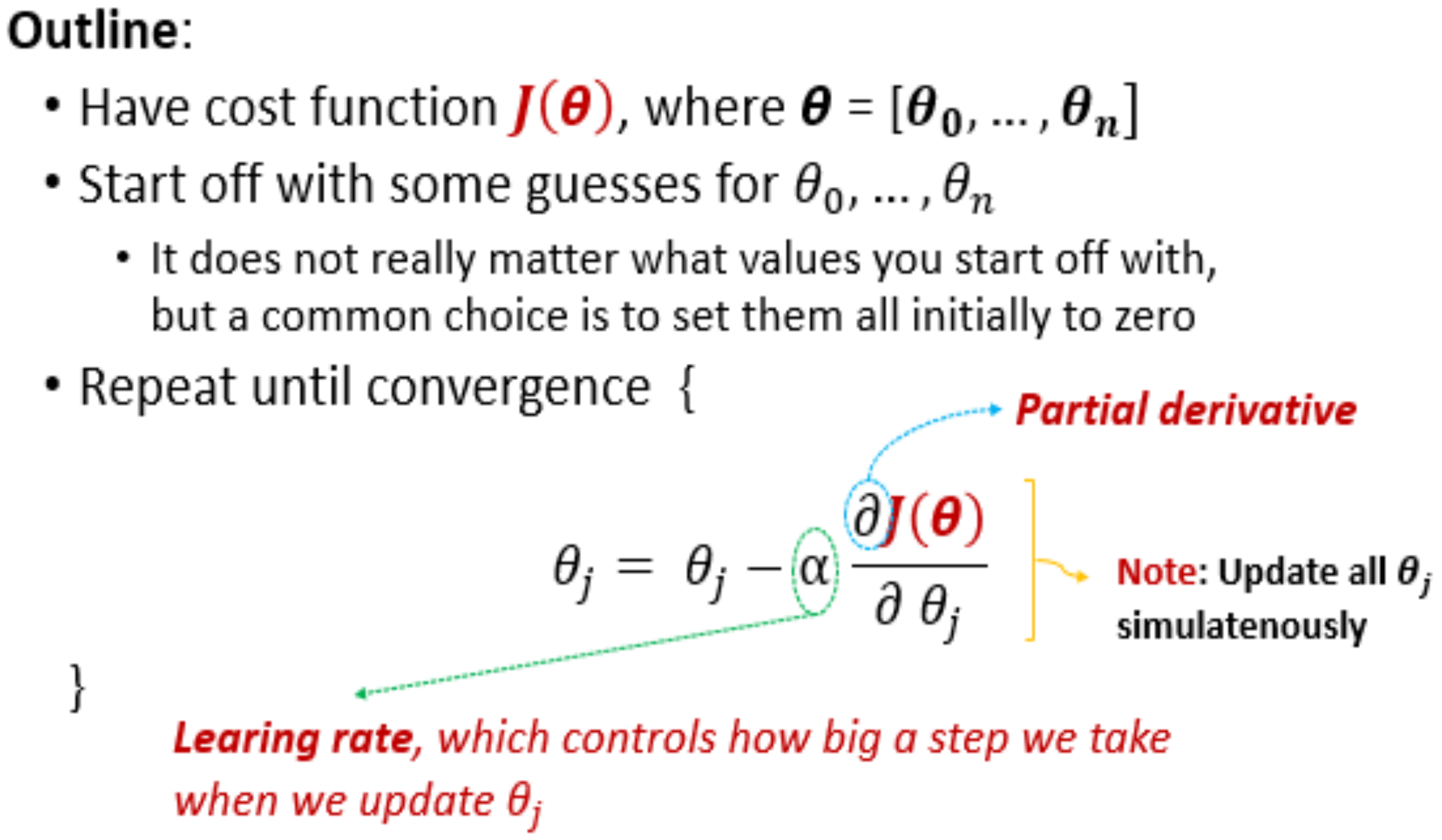
$$\underset{\theta}{\text{minimize}} \frac{1}{n}\sum_{i=1}^{m} \text{Cost}(\boldsymbol{h_\theta}(\boldsymbol{x})^{(i)}, y^{(i)})$$

Cost function

$\boldsymbol{J(\theta)}$

$$\underset{\theta}{\text{minimize}} \frac{1}{n}\sum_{i=1}^{m} \boxed{-y^{(i)} \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}}\right) - (1-y)\log\left(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}}\right)}$$

Gradient Descent For Logistic Regression



Outline:

- Have cost function $\boldsymbol{J}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ = $[\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_n]$
- Start off with some guesses for $\theta_0, \ldots, \theta_n$
  - It does not really matter what values you start off with, but a common choice is to set them all initially to zero
- Repeat until convergence {

Partial derivative

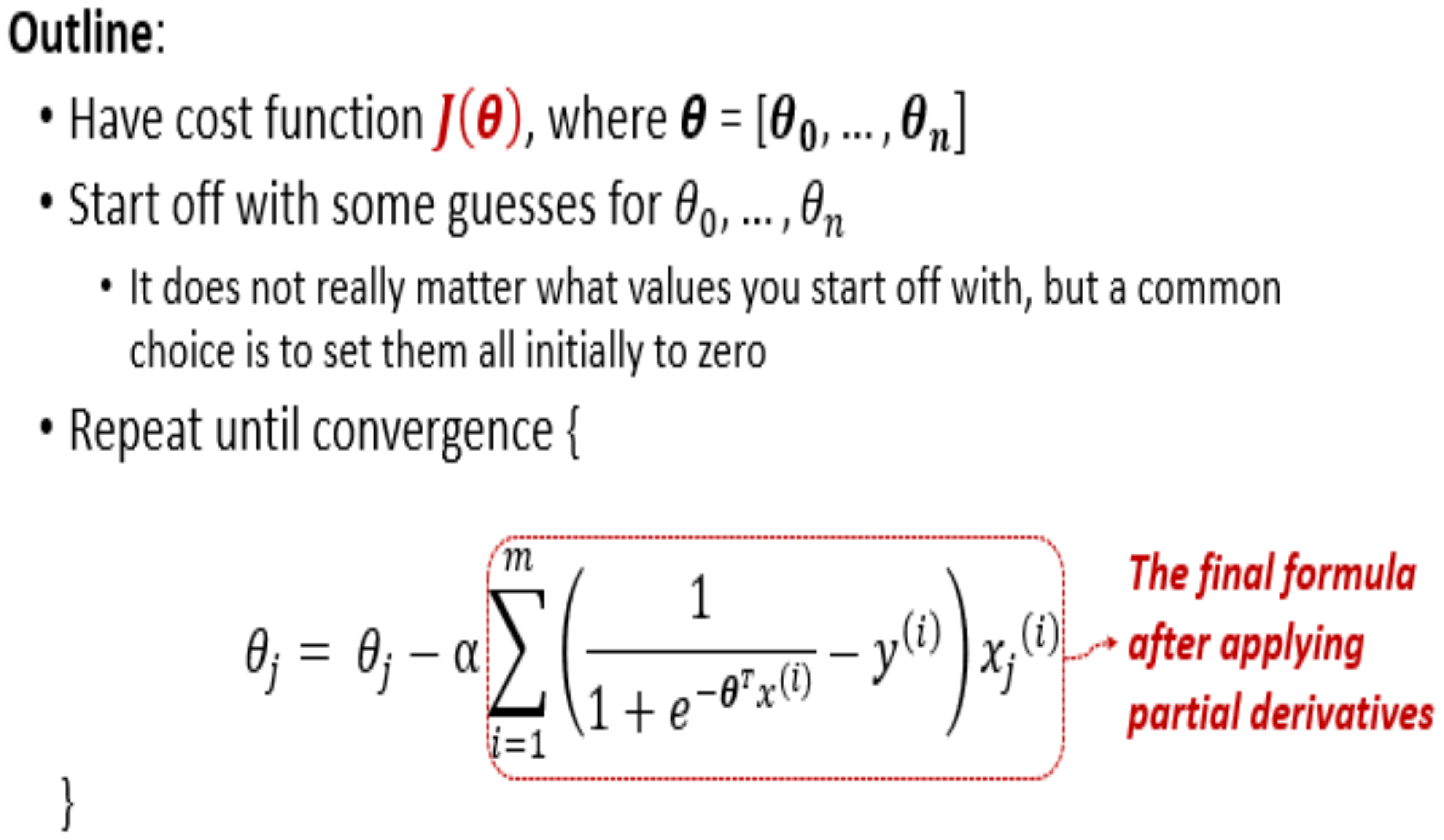
$$\theta_j = \theta_j - \alpha \frac{\partial \boldsymbol{J}(\boldsymbol{\theta})}{\partial \theta_j}$$

Note: Update all $\boldsymbol{\theta}_j$ simulatenously

}

***Learing rate**, which controls how big a step we take when we update θ_j*

Gradient Descent For Logistic Regression



Outline:

- Have cost function $J(\boldsymbol{\theta})$, where $\boldsymbol{\theta} = [\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_n]$
- Start off with some guesses for $\theta_0, \dots, \theta_n$
  - It does not really matter what values you start off with, but a common choice is to set them all initially to zero
- Repeat until convergence {

$$\theta_j = \theta_j - \alpha \boxed{\sum_{i=1}^{m} \left(\frac{1}{1 + e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)} \right) x_j^{(i)}} \dashrightarrow$$

The final formula after applying partial derivatives

}

Logistic Regression: Inference after learning

- After learning the parameters $\boldsymbol{\theta} = [\theta_0, \dots, \theta_n]$, we can predict the output of any new unseen $\boldsymbol{x} = [x_0, \dots, x_n]$ as follows:

$$\begin{cases} if\ h_\theta(x) = \dfrac{1}{1+e^{-\theta^T x}} < 0.5 \text{ predict } 0 \\ \\ Else\ if\ h_\theta(x) = \dfrac{1}{1+e^{-\theta^T x}} \geq 0.5 \text{ predict } 1 \end{cases}$$

Characteristics of Logistic Regression

- The learned parameters of logistic regression can be analyzed to understand the relationships between attributes and class labels.
- Because logistic regression does not involve computing densities and distances in the attribute space, it can work more robustly even in high-dimensional settings.
- Logistic regression can handle irrelevant attributes by learning weight parameters close to 0 for attributes that do not provide any gain in performance during training. It can also handle interacting attributes since the learning of model parameters is achieved in a joint fashion by considering the effects of all attributes together.
- Logistic regression cannot handle data instances with missing values, since the posterior probabilities are only computed by taking a weighted sum of all the attributes.