CS57300
PURDUE UNIVERSITY

SEPTEMBER 29, 2021

DATA MINING

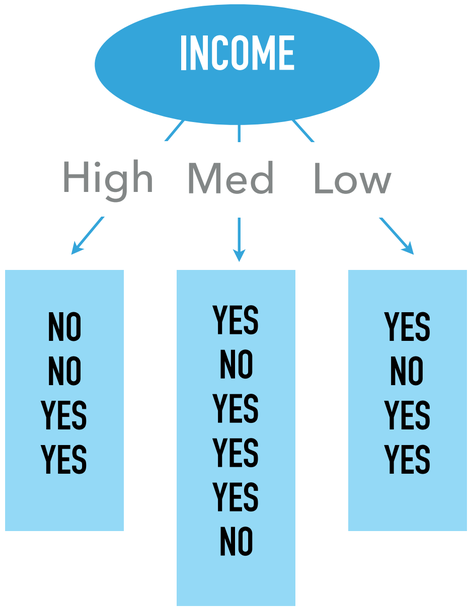
DECISION TREE

HOW TO AVOID OVERFITTING IN DECISION TREES

- Post-pruning
  - Separate the training data into a training set and a validation set (i.e., a pruning set).
  - Fully grow a tree
  - Use the pruning set to evaluate the utility of pruning (i.e. deleting) nodes from the tree
- Pre-pruning
  - Apply a statistical test to decide whether to expand a node
  - Add penalty terms in scoring functions to prefer trees with smaller sizes

PRE-PRUNING METHODS

- Stop growing tree at some point during top-down construction when there is no longer sufficient data to make reliable decisions



	Buy	No buy	
High	2	2	4
Med	4	2	6
Low	3	1	4
	9	5	14

Gain(S,Income)= 0.029

Gini-Gain (S,Income)= 0.020

$\chi^2 = 0.57$

IS THIS SPLIT REALLY MEANINGFUL?

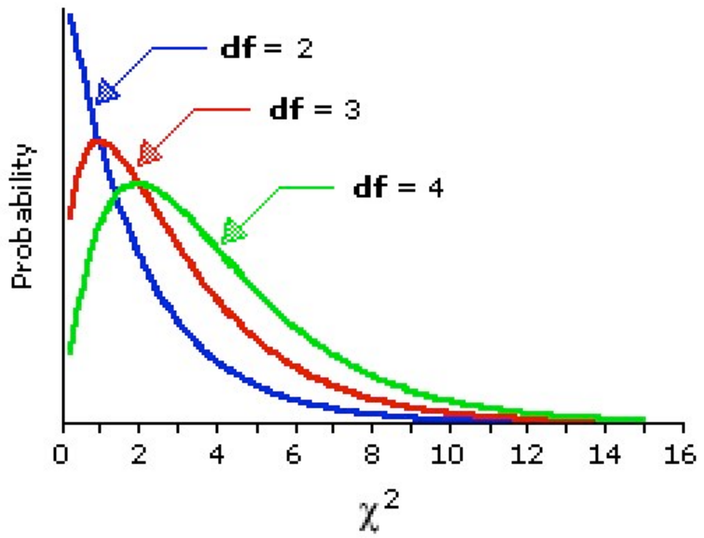
PRE-PRUNING METHODS

- Approach:
  - Choose threshold on feature score (e.g., information gain, gini gain)
  - Stop splitting if the best feature score is below threshold
  - Threshold can be decided through significance in statistical test or cross validation

EXAMPLE: DETERMINE CHI-SQUARE THRESHOLD ANALYTICALLY

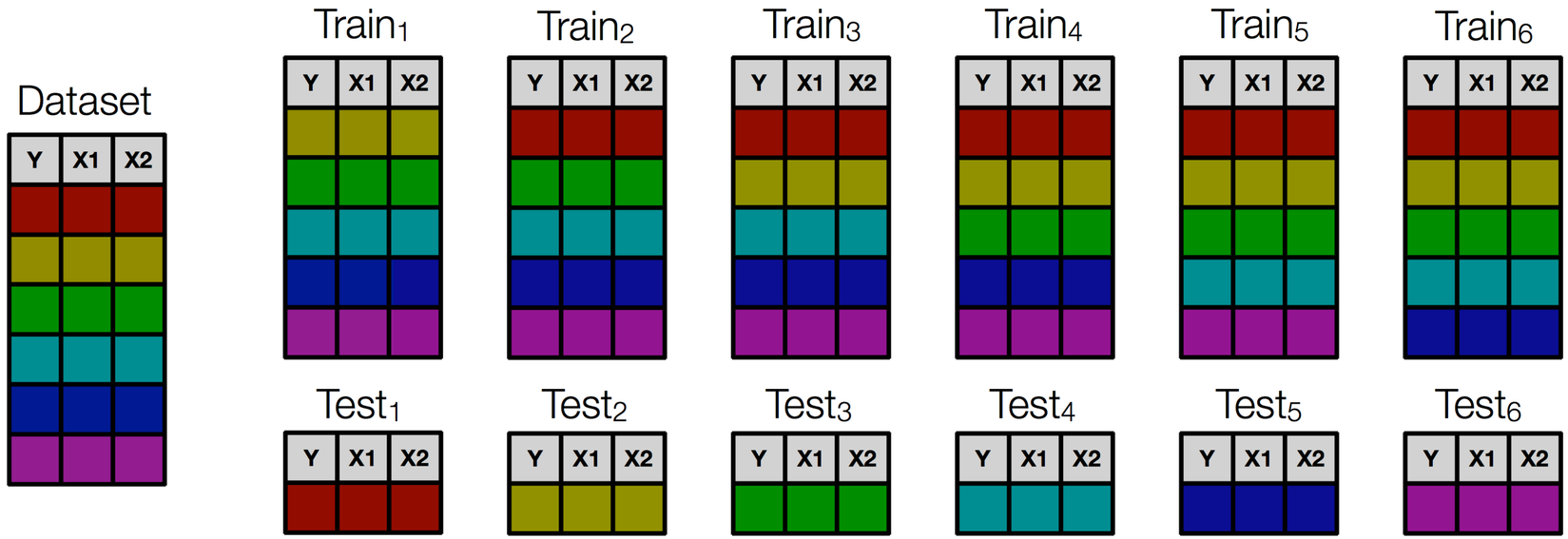
- Chi-square has known sampling distribution, can look up significance threshold
  - Degrees of freedom= (#rows-1)(#cols-1)
  - 3*2 table:
    5.99 is 95% critical value
- Stop growing when chi-square feature score is not **statistically significant**

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$



K-FOLD CROSS VALIDATION

- Randomly **partition** training data into k folds
- For i=1 to k
  - Learn model on D - i^{th} fold; evaluate model on i^{th} fold
- Average results from all k trials



EXAMPLE: CHOOSING A GINI THRESHOLD WITH CROSS VALIDATION

- For i in 1.. k
  - For t in threshold set (e.g, [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8])
    - Learn decision tree on $Train_i$ with Gini gain threshold t
      (i.e. stop growing when max Gini gain is less than t)
    - Evaluate learned tree on $Test_i$ (e.g., with accuracy)
  - Set $t_{max,i}$ to be the t with best performance on $Test_i$
- Set t_{max} to the average of $t_{max,i}$ over the k trials
- Relearn the tree on all the data using t_{max} as Gini gain threshold

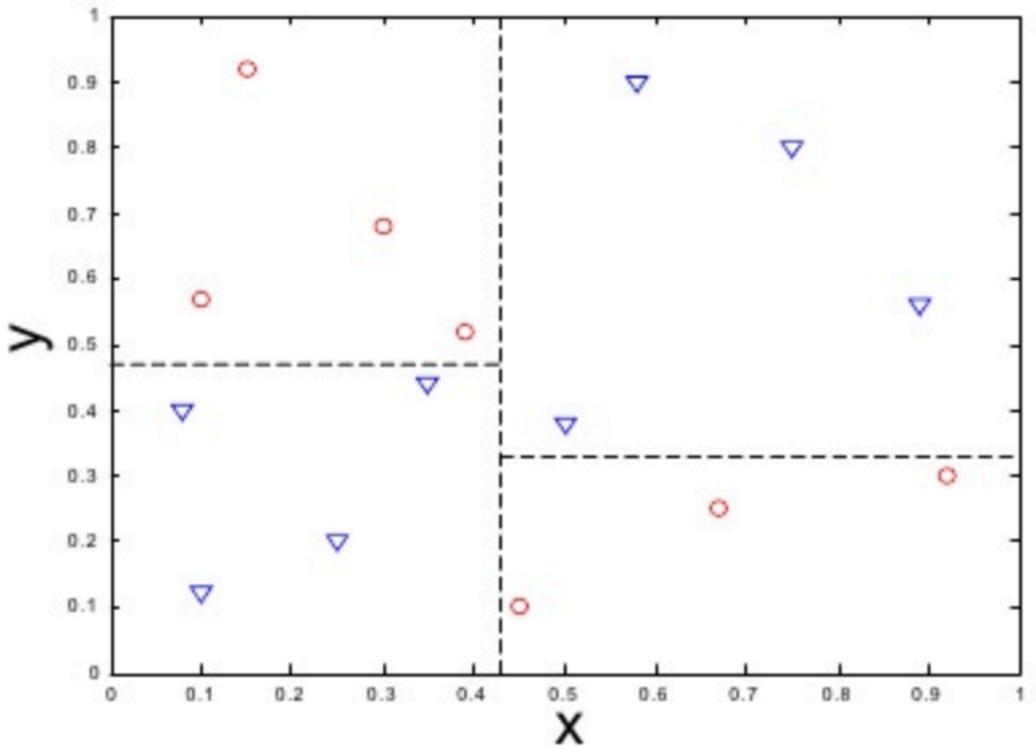
ALGORITHM COMPARISON

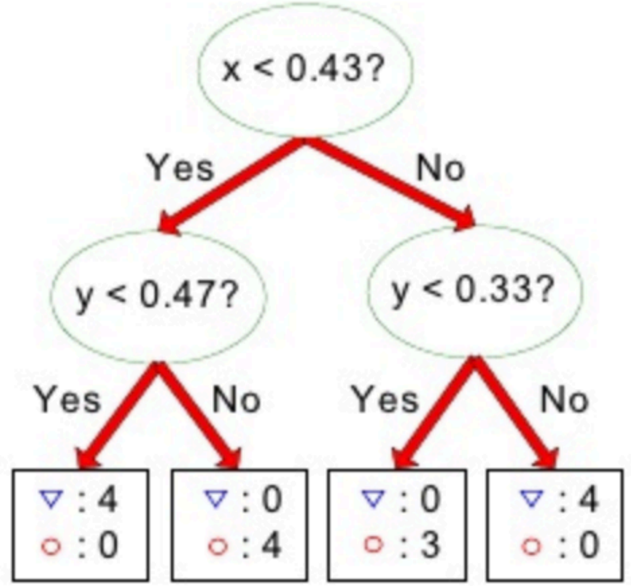
- CART
  - Evaluation criterion:
    Gini gain
  - Search algorithm:
    Heuristic, greedy search
  - Pruning mechanism:
    Cross-validation to select gini threshold
- C4.5
  - Evaluation criterion:
    Information gain
  - Search algorithm:
    Heuristic, greedy search
  - Pruning mechanism:
    Reduce error pruning

NAIVE BAYES VS. DECISION TREES

- Naive Bayes
  - **Probabilistic classification**: output posterior class distribution p(y|**x**), and model the underlying probability distributions
  - Parametric model
  - Model space: parameters in prior distributions p(y) and conditional distributions p(**x**|y)
  - Scoring function: likelihood function / posterior probability of observing the data
  - Search: Convex optimization
- Decision trees
  - **Discriminative classification**: output class labels and model the decision boundary directly
  - Non-parametric model
  - Model space: all possible trees that can be generated from the set of attributes: different attribute to use on each node, different ways to split continuous variables into intervals, different depth of the tree, etc.
  - Scoring function: misclassification rate
  - Search: Greedy, heuristic search

DECISION TREES MODEL DECISION BOUNDARIES





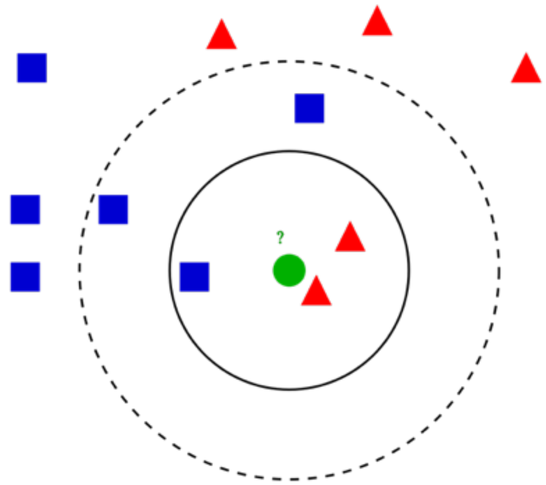
NEAREST NEIGHBOR

NEAREST NEIGHBOR

- Discriminative classification, non-parametric, instance-based method
- Assumes that all points are represented in *p*-dimensional space
- Learning
  - Stores (i.e., memorizes) all the training data
- Prediction
  - Look for “nearby” training examples
  - Classification is made based on class labels of neighbors

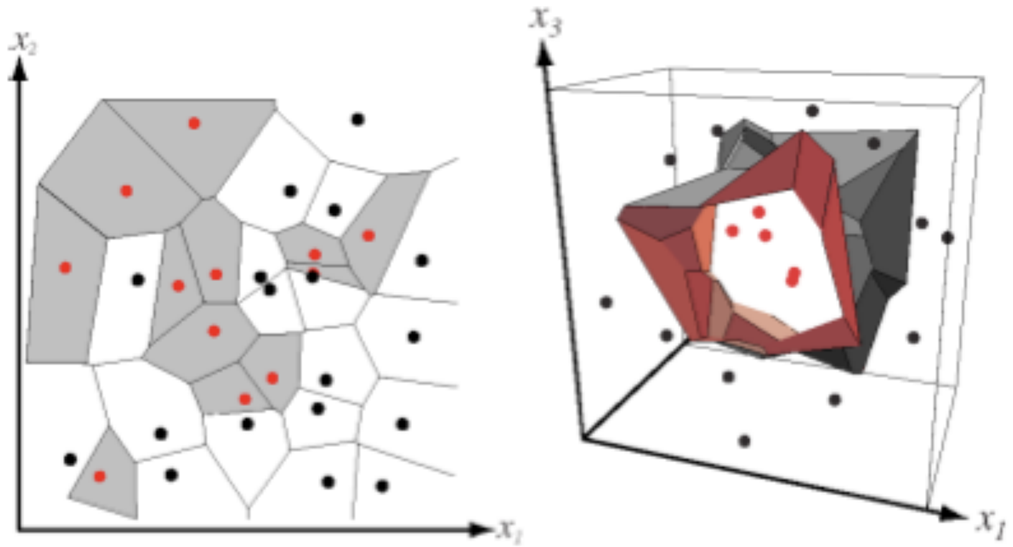
FROM 1NN TO KNN

- Training set: $(\mathbf{x}_1,y_1)$, $(\mathbf{x}_2,y_2)$, …, $(\mathbf{x}_n,y_n)$ where $\mathbf{x}_i = [x_{i1}, x_{i2}, …, x_{ip}]$ is a feature vector of *p* attributes and y_i is a discrete class label
- To predict a class label for new instance *j*: Find the training instance point $\mathbf{x}_i$ such that $d(\mathbf{x}_i, \mathbf{x}_j)$ is minimized; Let $f(\mathbf{x}_j)= y_i$
- Key idea: Find instances that are "similar" to the new instance and use their class labels to make prediction for the new instance
  - 1NN generalizes to KNN when labels of the *K* nearest neighbors are considered;
    Let $f(\mathbf{x}_j)=g(\{y_i\})$, g() is an aggregation function such as majority vote.



1NN DECISION BOUNDARY

- For each training example *i*, we can calculate its **Voronoi cell**, which corresponds to the space of points for which i is their nearest neighbor
- All points in such a Voronoi cell are labeled by the class of the training point, forming a Voronoi tessellation of the feature space



Source: http://www.cs.bilkent.edu.tr/~saksoy/courses/cs551-Spring2008/slides/cs551_nonbayesian1.pdf

NEAREST NEIGHBOR: MODEL SPACE

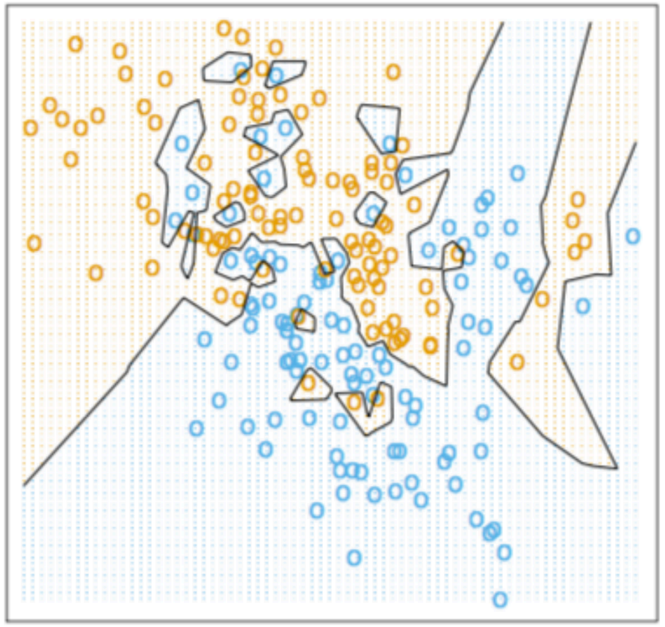
- How many neighbors to consider (i.e., choice of K)?
  ... Usually a small value is used, e.g. $K<10$
- What distance measure $d()$ to use?
  ... Euclidean L_2 distance is often used
- What function $g()$ to combine the neighbors' labels into a prediction?
  ... Majority vote is often used

NEAREST NEIGHBOR: SEARCH

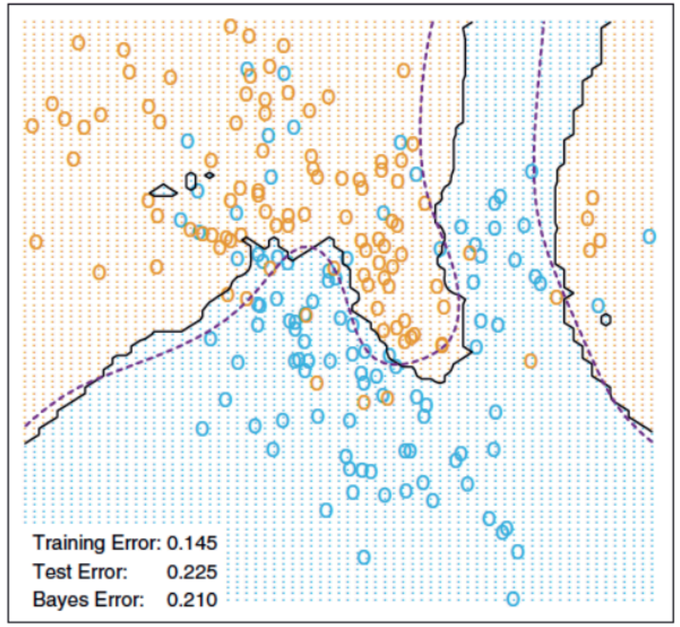
- Scoring function: Misclassification rate

K=1, training error = 0!

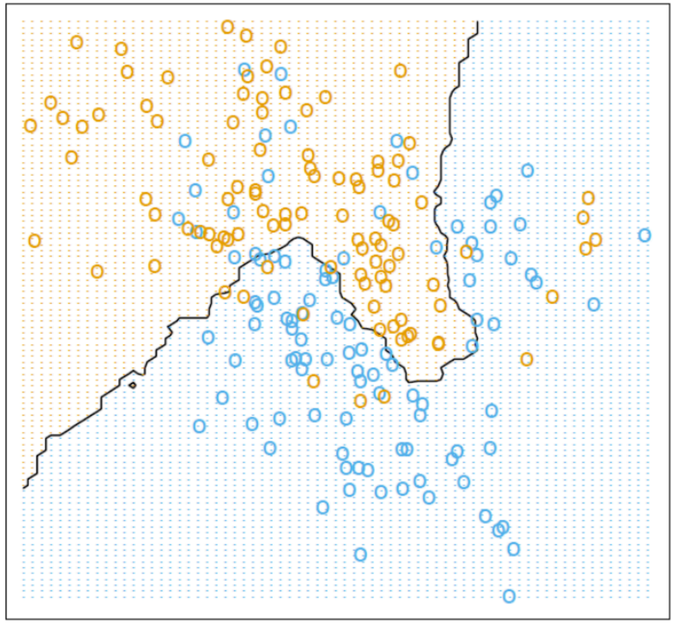
Is this a good choice of K?



K=7

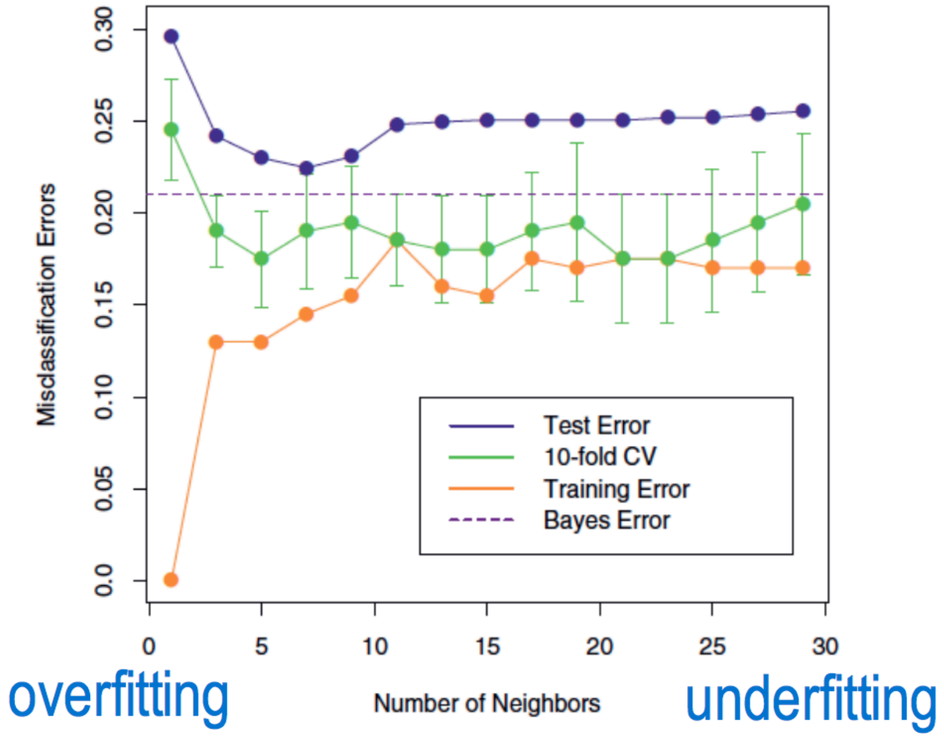


K=15



NEAREST NEIGHBOR: CHOOSE K THROUGH CROSS VALIDATION

- Divide the training dataset into *k* folds and conduct *k*-fold cross validation using different values of *K* for the KNN model (*k* and *K* here are different things!)



Choose K=5!

NEAREST NEIGHBOR: SUMMARY

- Strengths:
  - Simple model, easy to implement
  - Very efficient learning: Only need to memorize all training data points
- Weaknesses:
  - Inefficient inference: need to compute distance to all training data points and select the nearest *k* ones.
  - Curse of dimensionality:
    - As number of features increase, you need an exponential increase in the size of the data to ensure that you have nearby examples for any given data point

LOGISTIC REGRESSION

LOGISTIC REGRESSION

- Probabilistic classification
  - Output is the posterior (positive) class probability $P(y=1|\mathbf{x})$
  - Output is in the range [0, 1]
- Can we map the posterior class probability to another range that is easier to process?

DIFFERENT WAYS OF EXPRESSING PROBABILITY

- Suppose p=P(y=1|**x**), q=1-p=P(y=0|**x**)

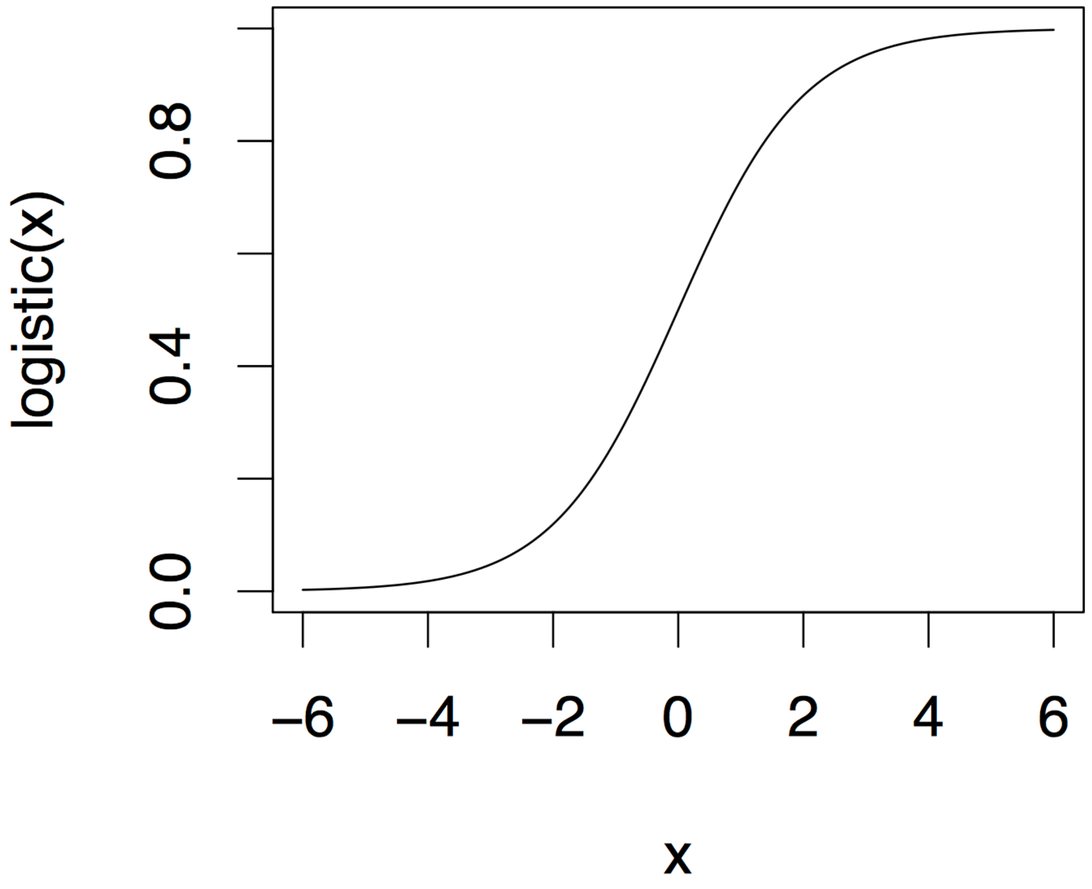
		min		max
standard probability	p	0	0.5	1
odds	p / q	0	1	+ ∞
log odds (logit)	log(p / q)	- ∞	0	+ ∞

$$log(p/q) = \mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0$$

LOGISTIC REGRESSION KNOWLEDGE REPRESENTATION

- p= $P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+w_0)}}$
- Logistic function:

$$\text{logistic}(x) := \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$



HOW ABOUT CATEGORICAL VARIABLES?

- Ordinal variable
  - Categorical variables for which the possible values are ordered
  - GPA: A, B, C, D, E, F
  - Map sorted ordinal variable values to an increasing sequence of numbers, e.g., A=1, B=2, C=3, D=4, E=5, F=6
- Nominal variable
  - Categorical variable for which the possible values have no natural order
  - Eye color: blue, green, brown
  - One-hot encoding: Use N-1 binary variables to represent the N possible values of a nominal variable, e.g., blue = [1, 0], green = [0, 1], brown=[0,0]