Biostat 202: Opportunities and Challenges of "Big Data".

Machine Learning 3: Supervised learning, contd. [classification]

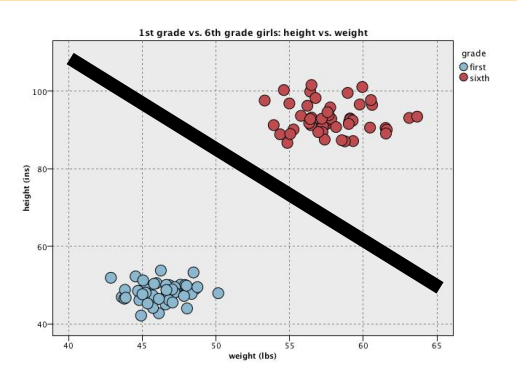
Aaron Wolfe Scheffler

Division of Biostatistics

Department of Epidemiology and Biostatistics

Review of last lecture

1. Introduction to classification



2. Evaluation metrics for classes

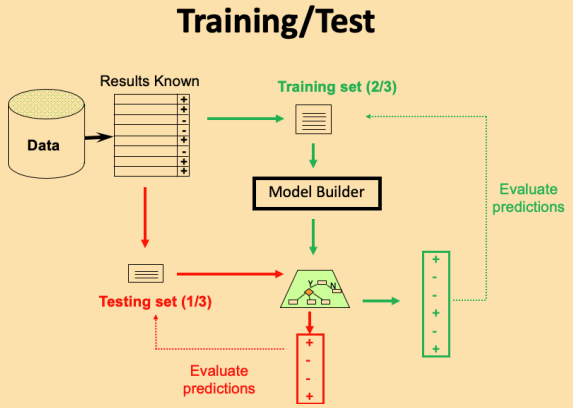
Confusion Matrix

Prediction\Truth	Negative	Positive	Total
Negative	#TN	#FN	#FN + #TN
Positive	#FP	#TP	#TP + #FP
Total	#FP + #TN	#TP + FN	N

Metrics obtained from confusion matrix:

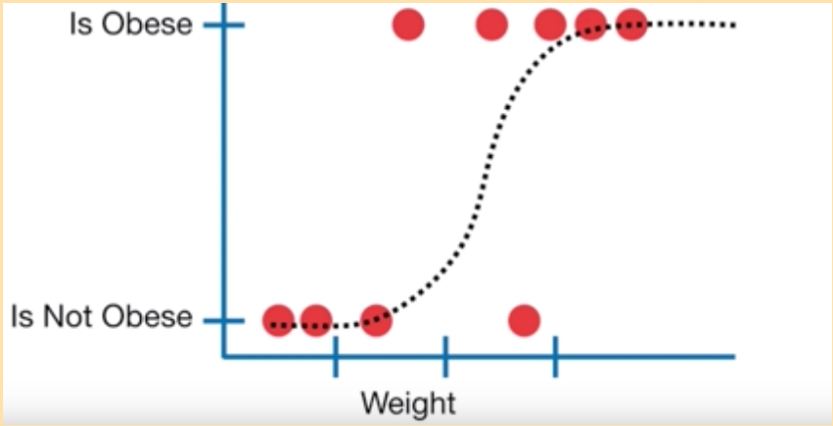
$$\text{Accuracy} = \frac{\#TP + \#TN}{N} \qquad \text{Specificity} = \frac{\#TN}{\#TN + \#FP} \qquad \text{Sensitivity} = \frac{\#TP}{\#TP + \#FN}$$

3. Validation and testing, contd.



4. Logistic regression

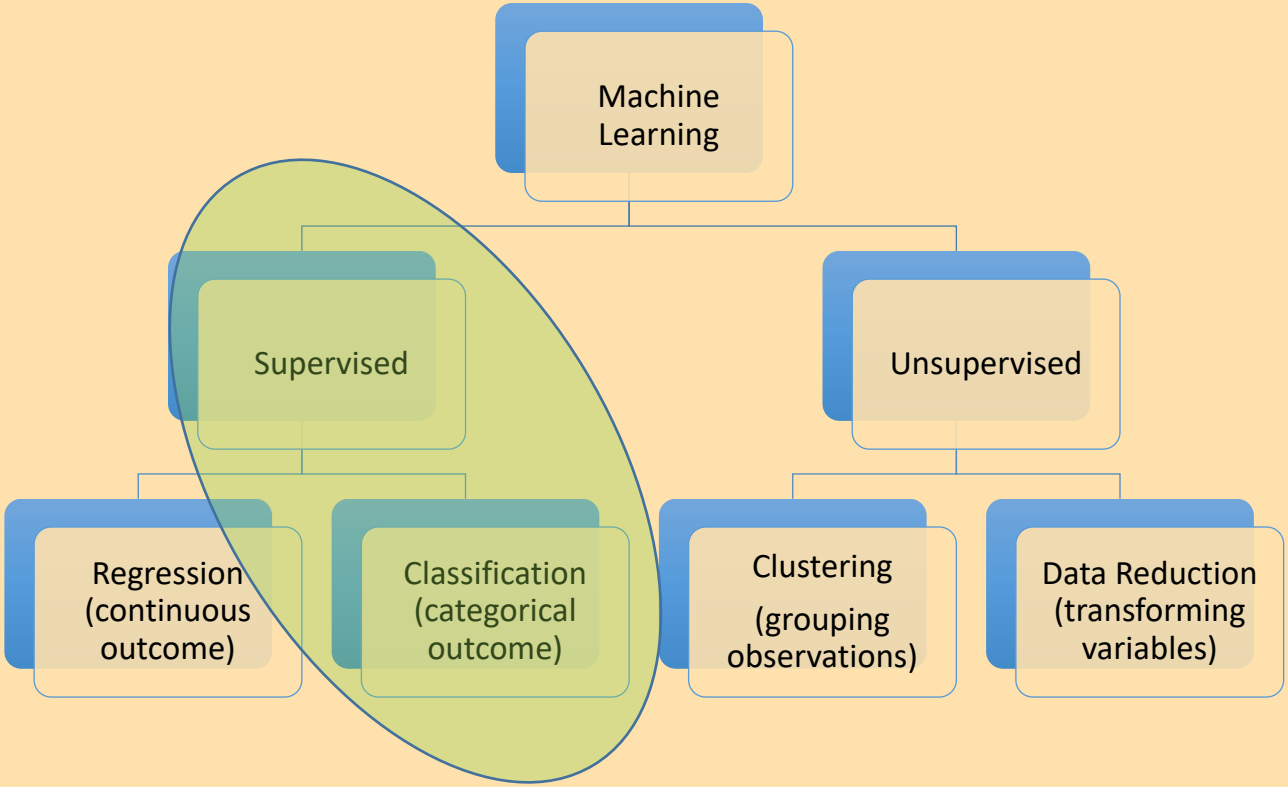
$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = a * weight + c$$



Overview of this lecture - more classification!

- K-nearest neighbors
  - Parameter tuning
  - Training-validation-test
- Classification trees (recursive partitioning)
- Support vector machines
- Artificial neural networks
- Comparison of classifiers

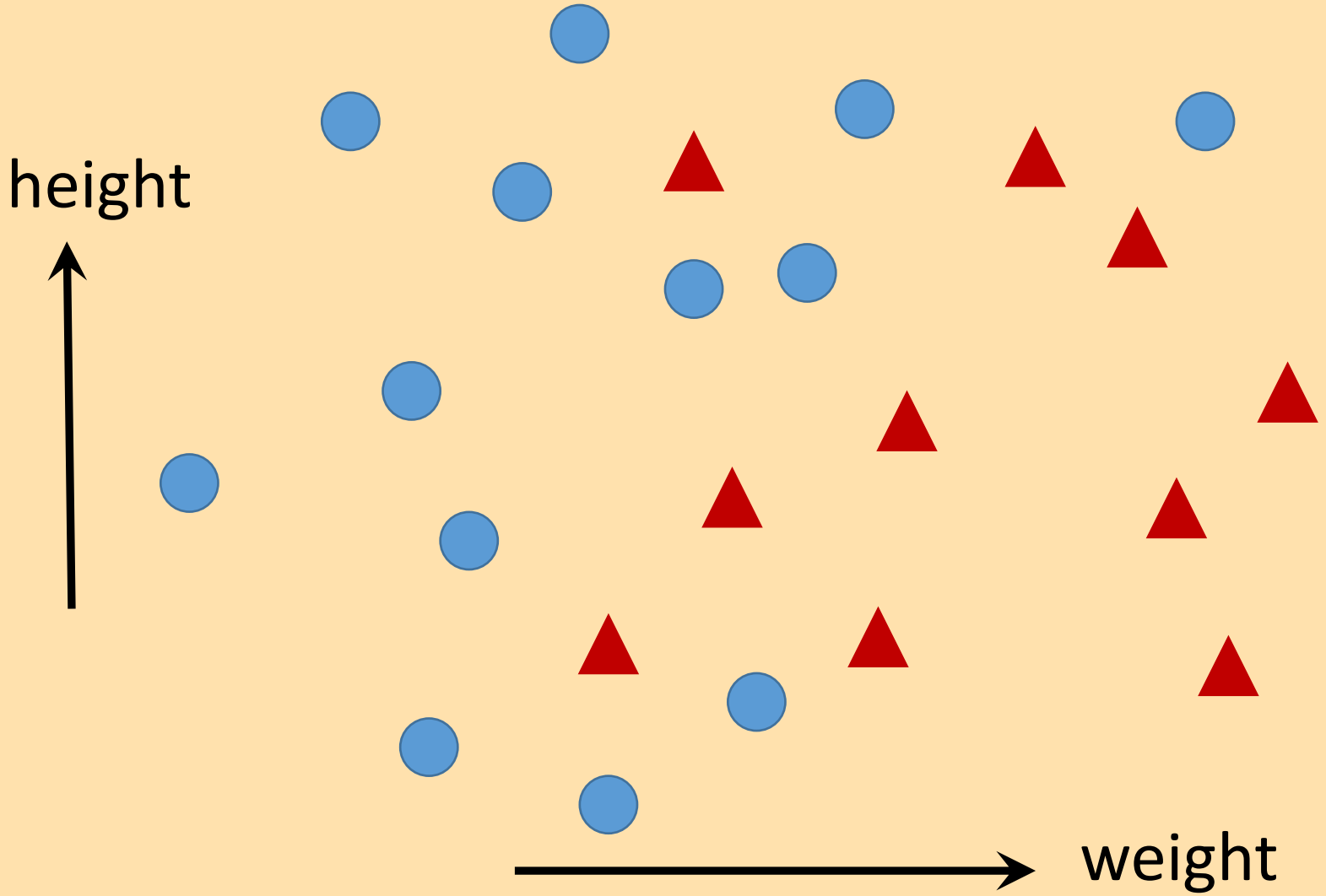
Classification



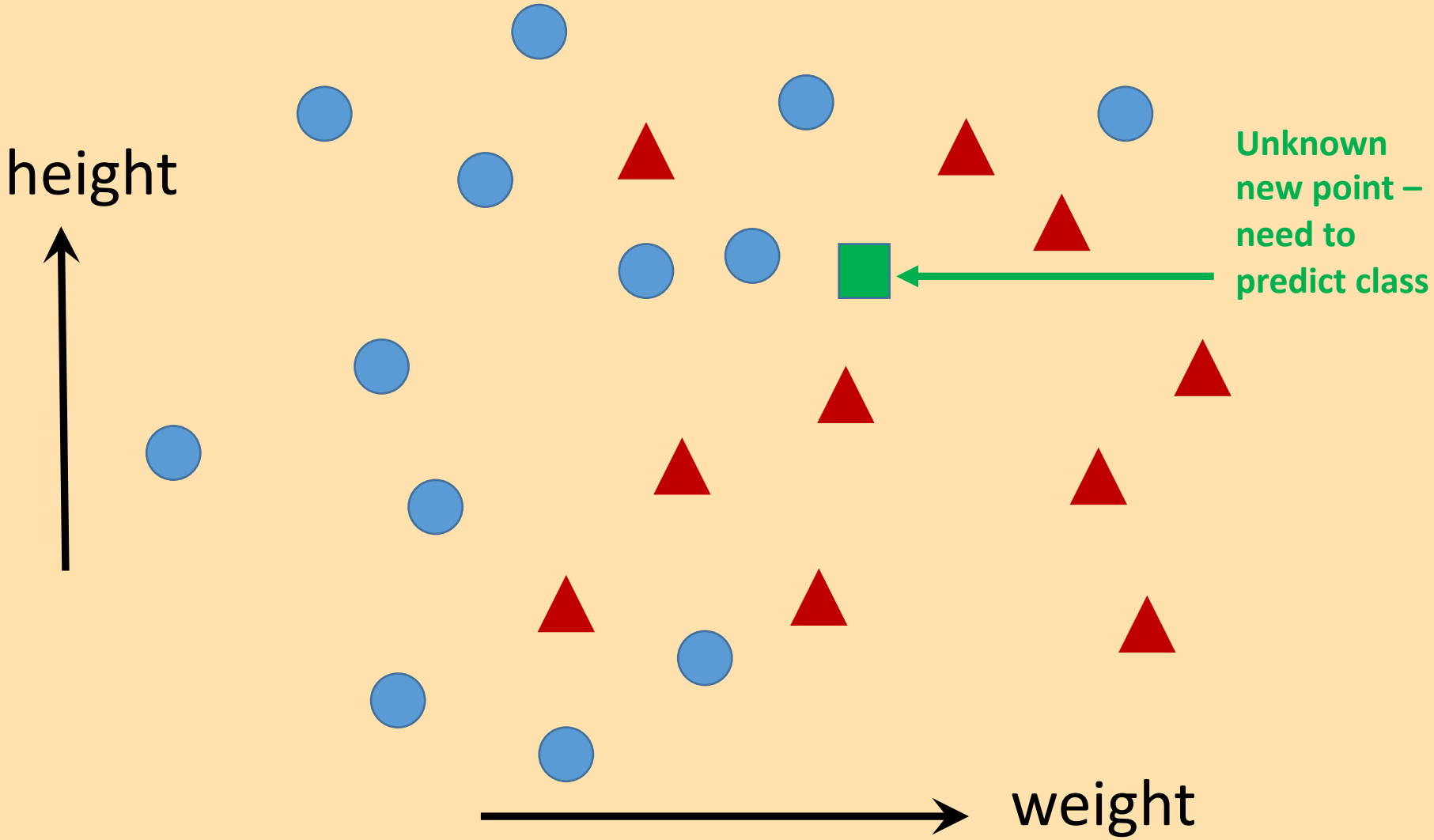
K-nearest neighbors classification

K-nearest neighbors classification

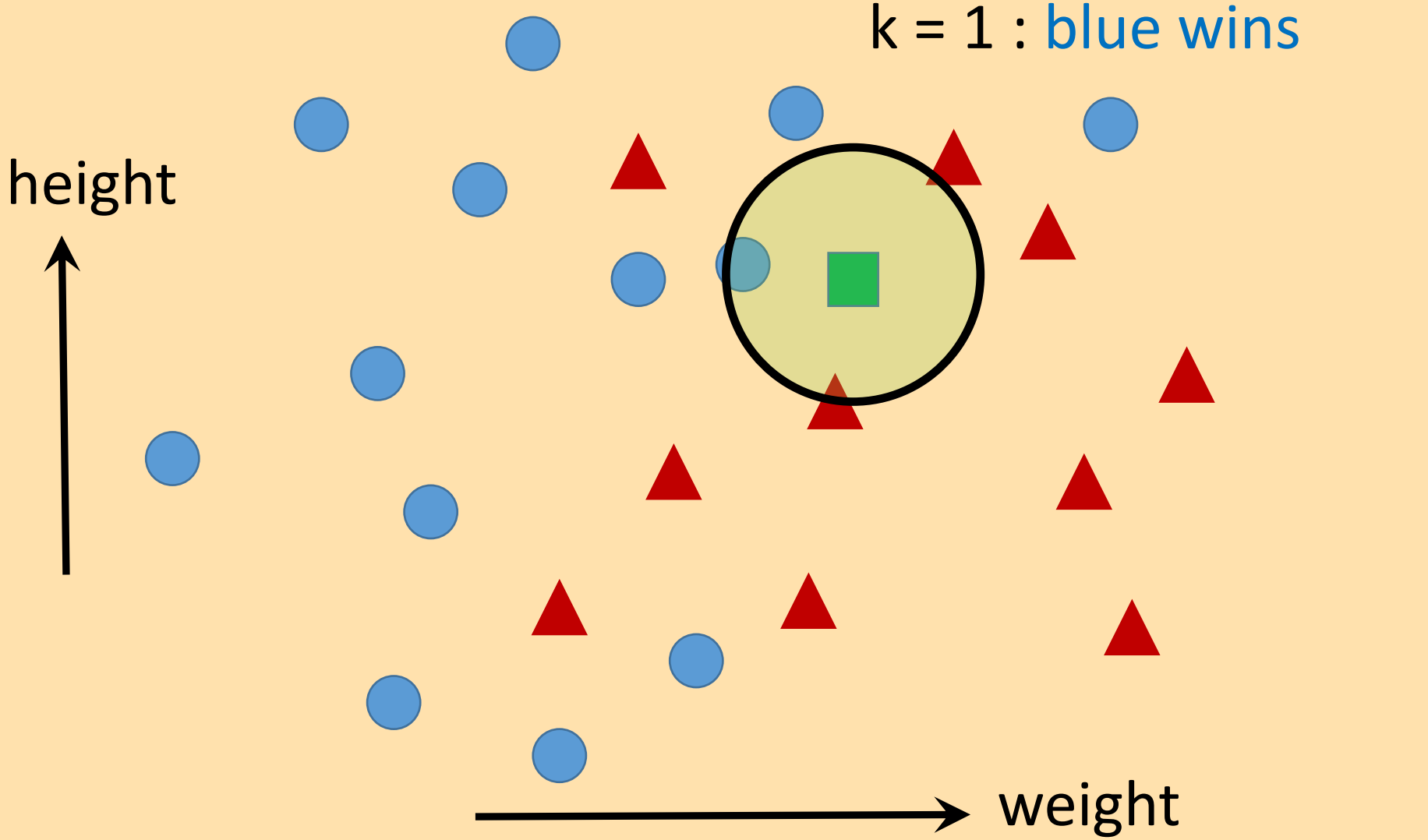
Training data: ● = female, ▲ = male



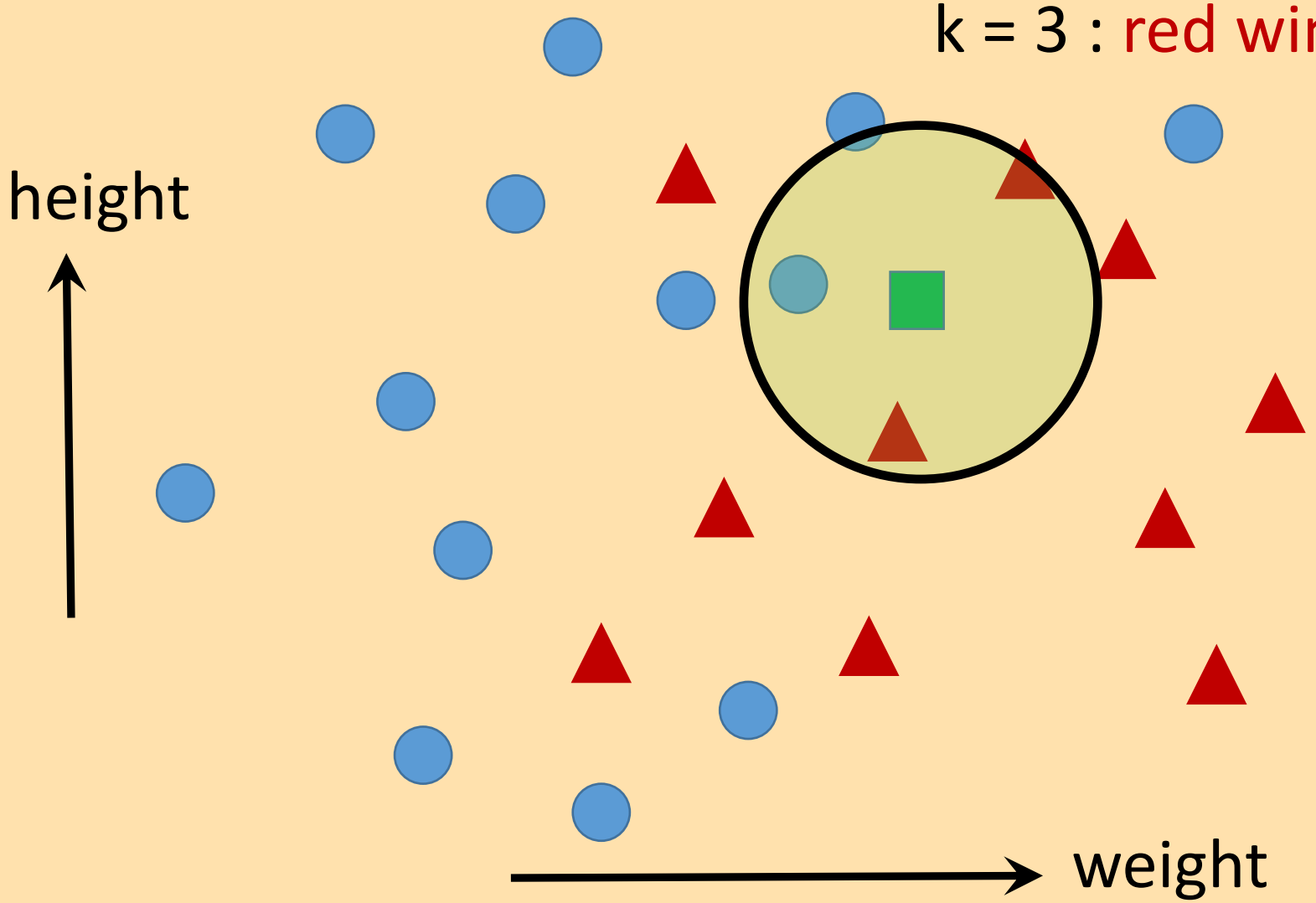
K-nearest neighbors classification



K-nearest neighbors classification

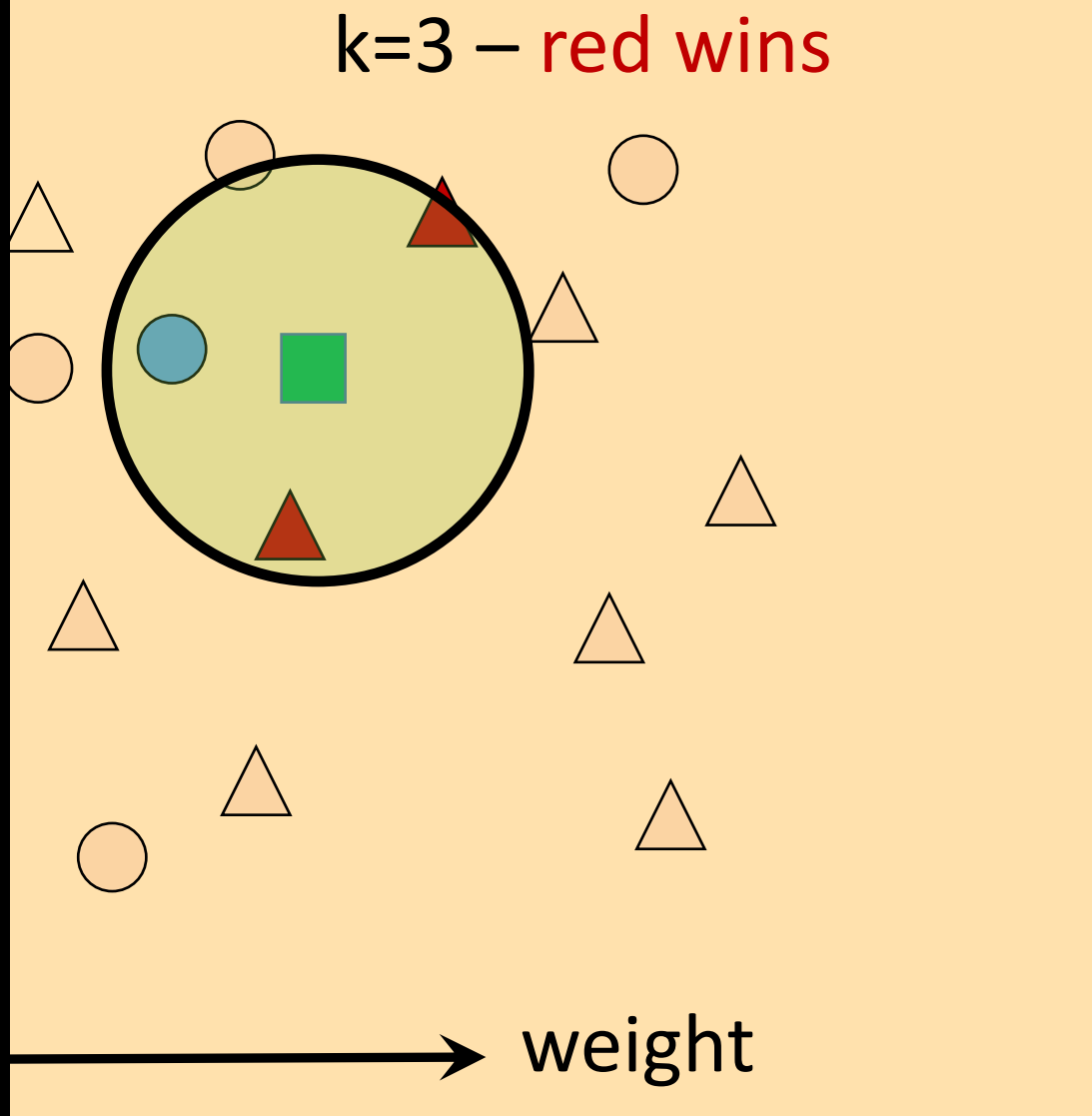


K-nearest neighbors classification



KNN VS. LOGISTIC

- To predict the category for each class, KNN **ONLY** considers the k-nearest points - other points do not matter!
- Contrast with logistic regression which uses all the data to construct a classification rule via the coefficient parameters

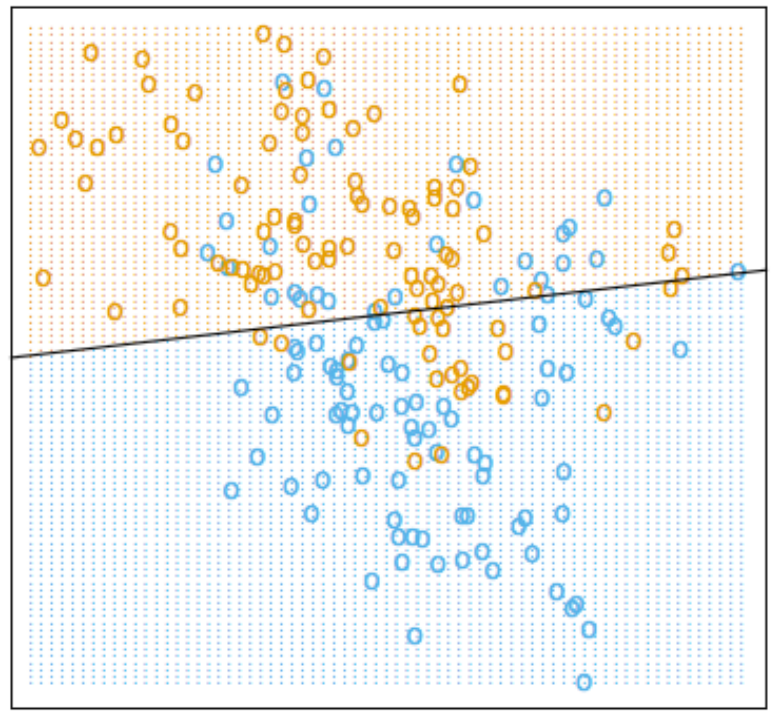


K-nearest neighbors classification

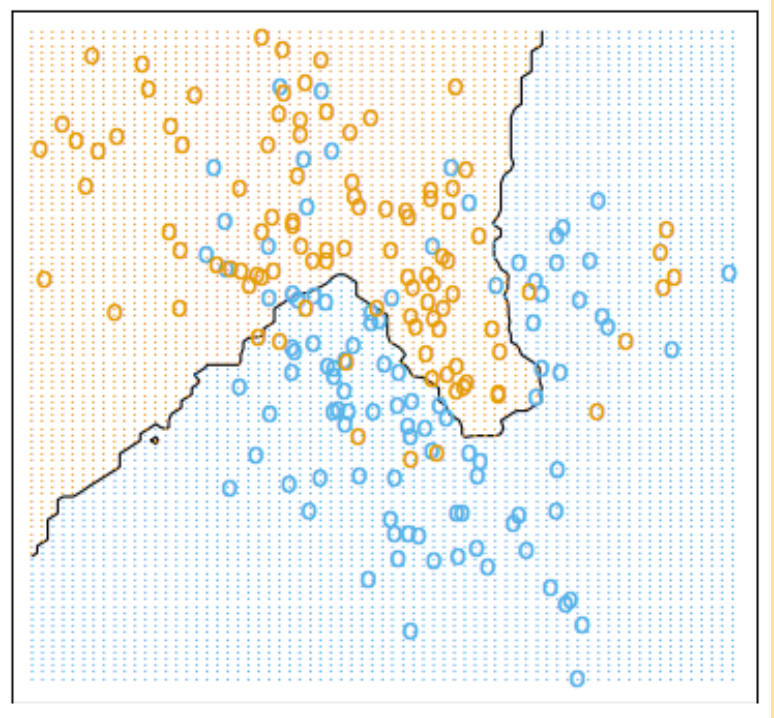
- Captures local behavior, a nosy neighbor 👀
- Neighborhood defined by “closest points” (distance)
- Common to standardize predictors to have mean 0 and standard deviation of 1 – allows equity among variables of different scales
- Classification based on majority vote among k-nearest neighbors, vote also used to calculate probability of each class in neighborhood
- Ties are broken at random

Classification/Decision Boundaries

Linear decision boundary (LDA)



15-nearest neighbor boundary

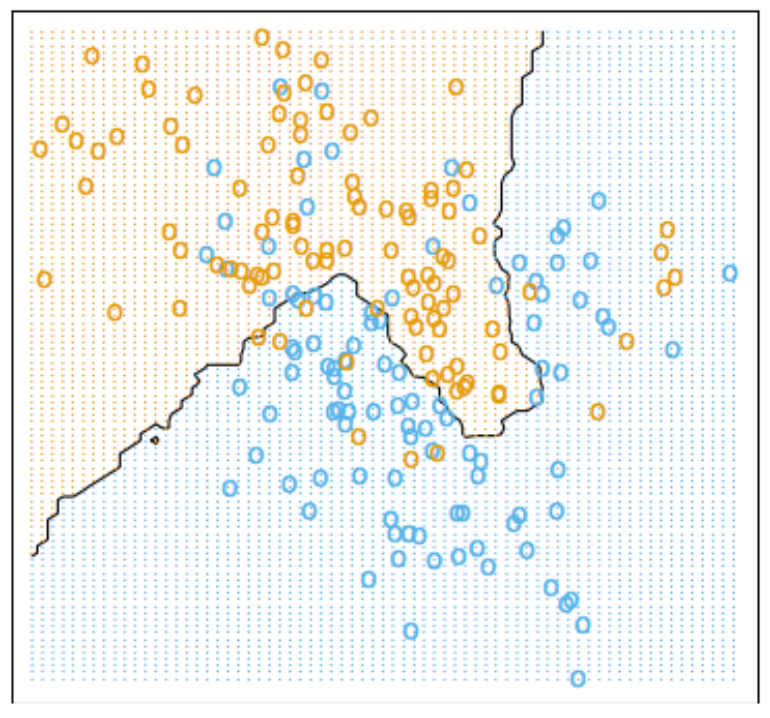


Example taken from Elements of Statistical Learning – Hastie et al. 2nd Ed. pp. 13-17

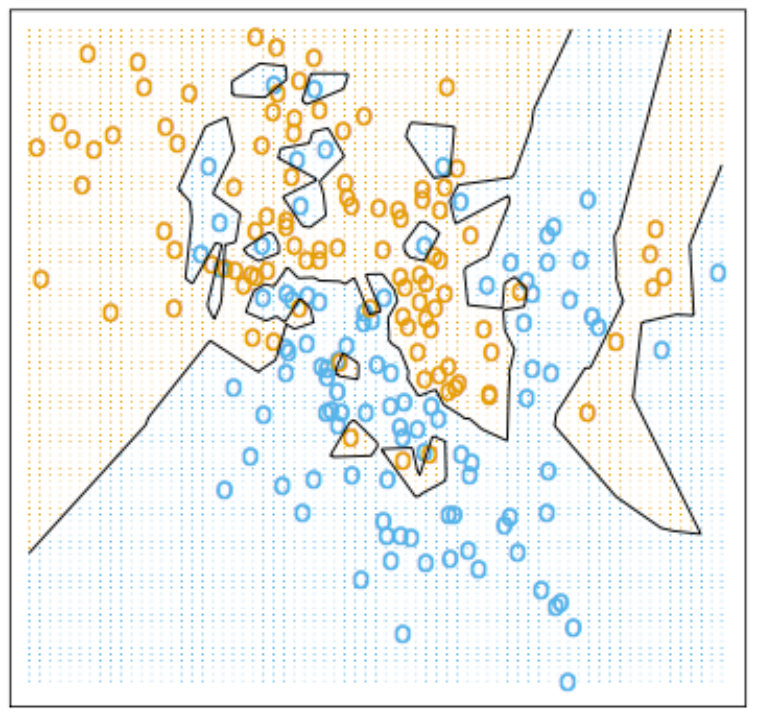
Choice of k affects boundary

SMALLER K = MORE LOCAL/FLEXIBLE DECISION BOUNDARY!!!

15 nearest-neighbor boundary



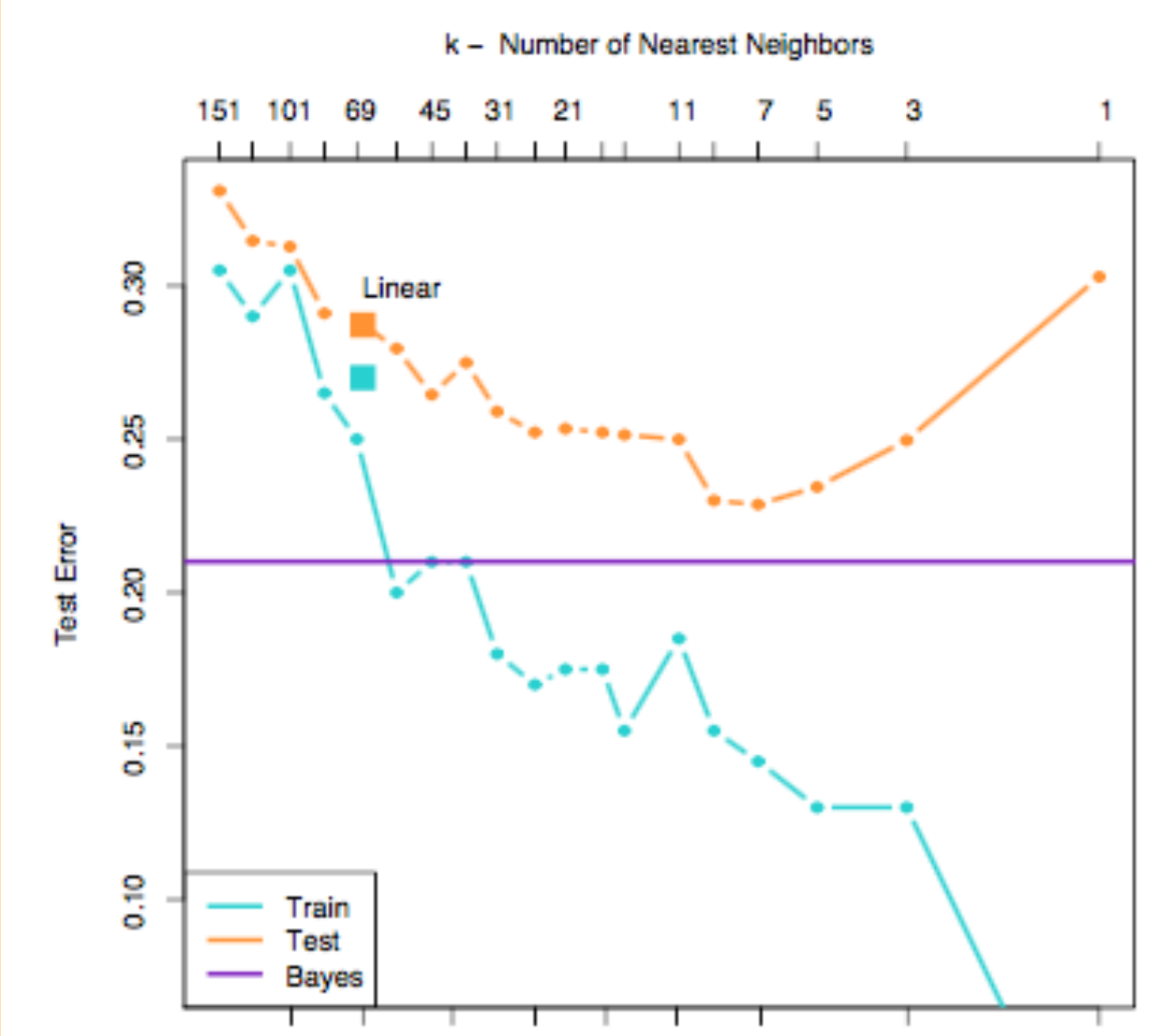
1-nearest neighbor boundary



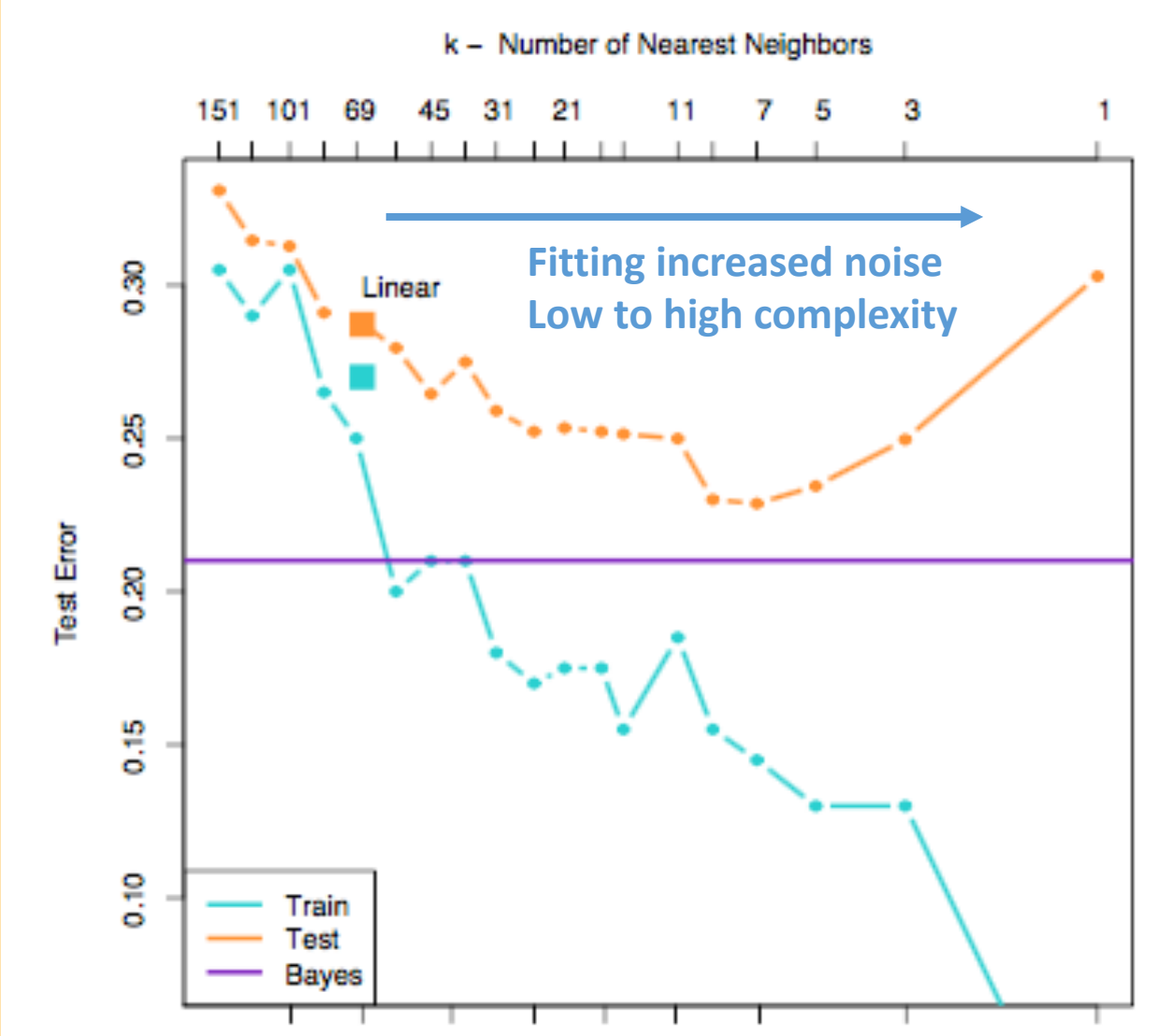
Parameter tuning

- Parameter tuning is the process of adjusting model parameters in order to find the “best” model
- For example, we tune the K-nearest neighbors by adjusting the parameter k, the number of neighbors
- Analogy: tuning the strings of a guitar to play the proper note is like parameter tuning to get the best prediction

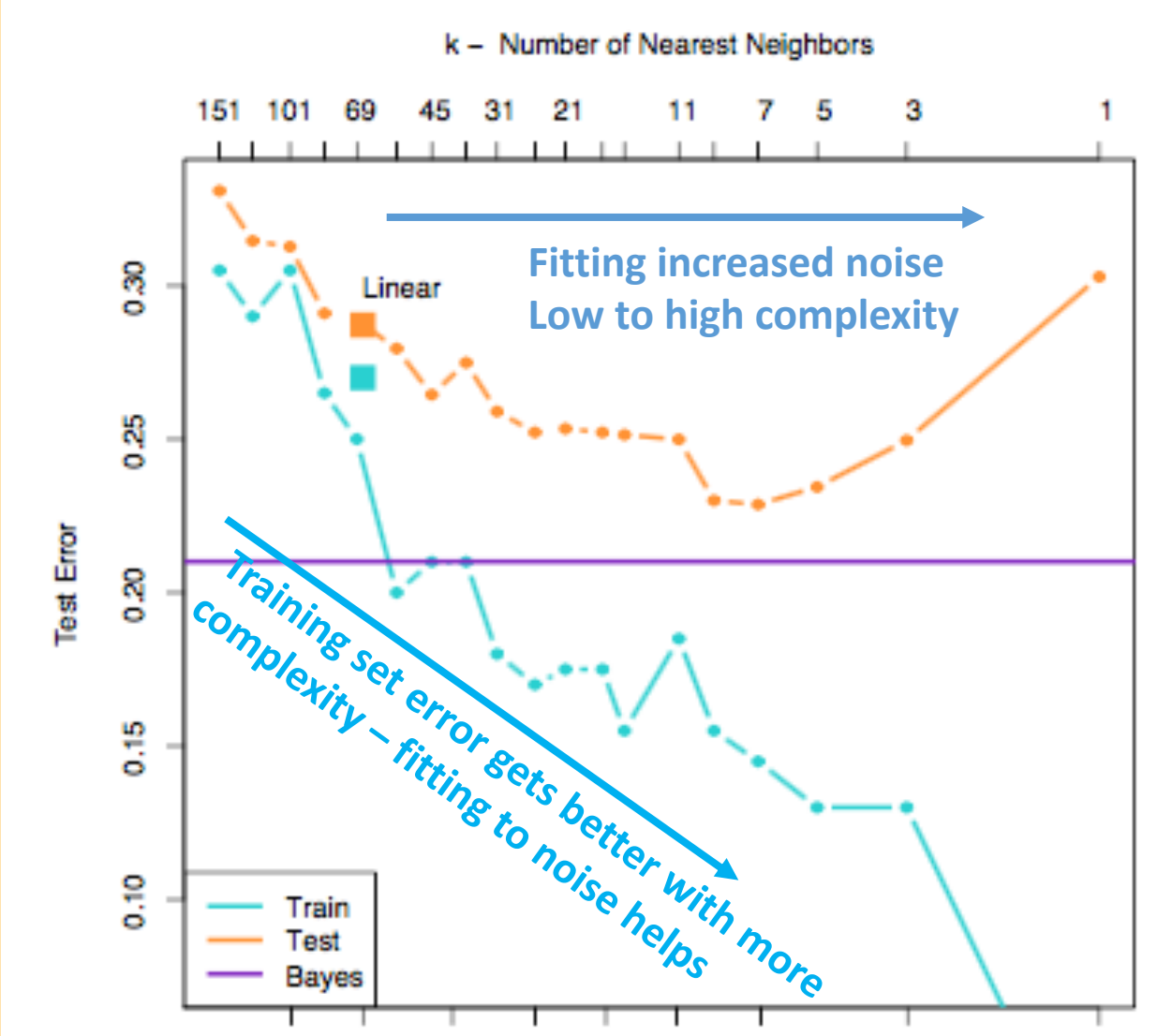
Choice of k


k – Number of Nearest Neighbors
151
101
69
45
31
21
11
7
5
3
1
Linear
Test Error
0.30
0.25
0.20
0.15
0.10
Train
Test
Bayes

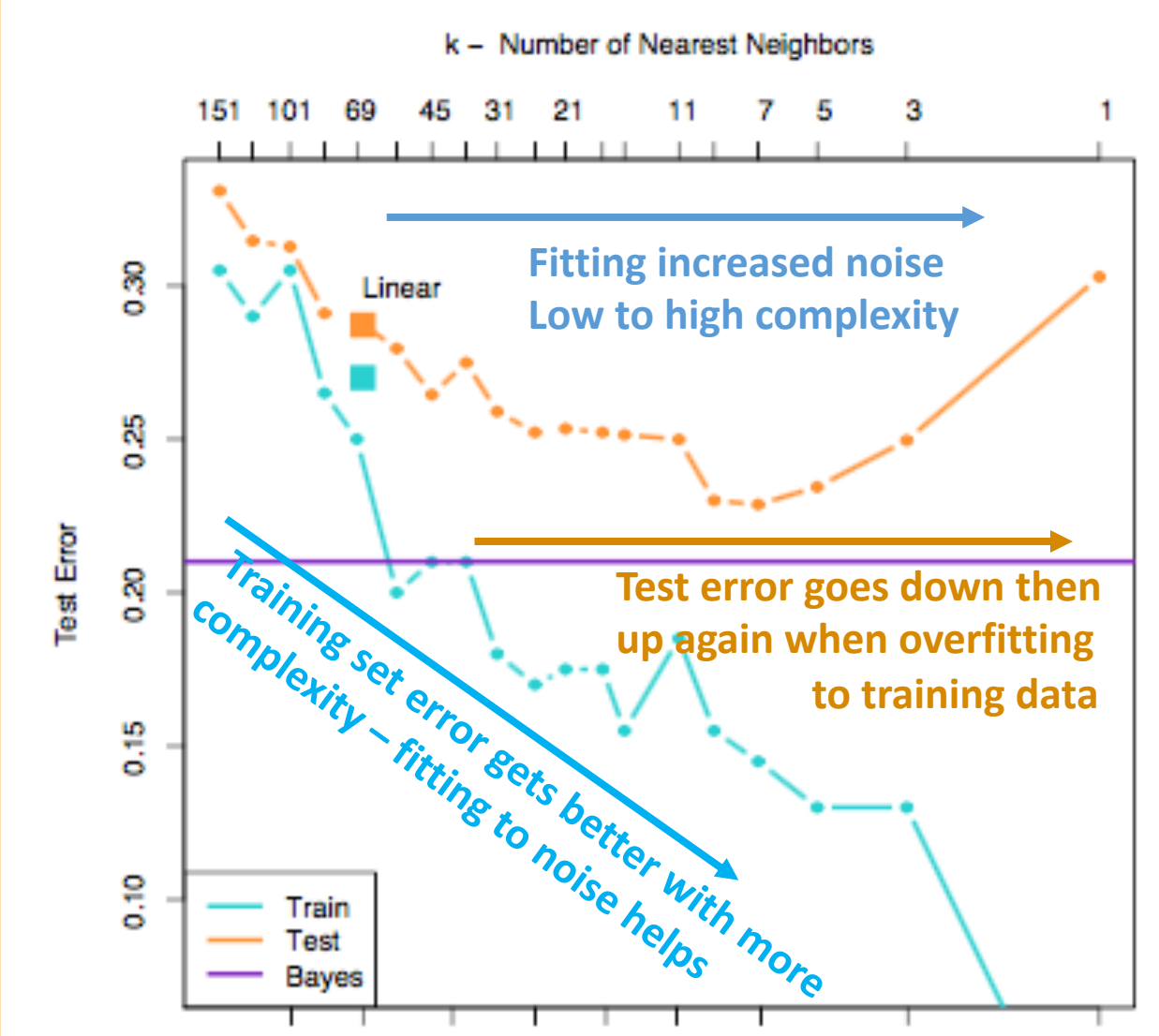
Choice of k


k – Number of Nearest Neighbors
151
101
69
45
31
21
11
7
5
3
1
Linear
Fitting increased noise
Low to high complexity
Test Error
0.30
0.25
0.20
0.15
0.10
Train
Test
Bayes

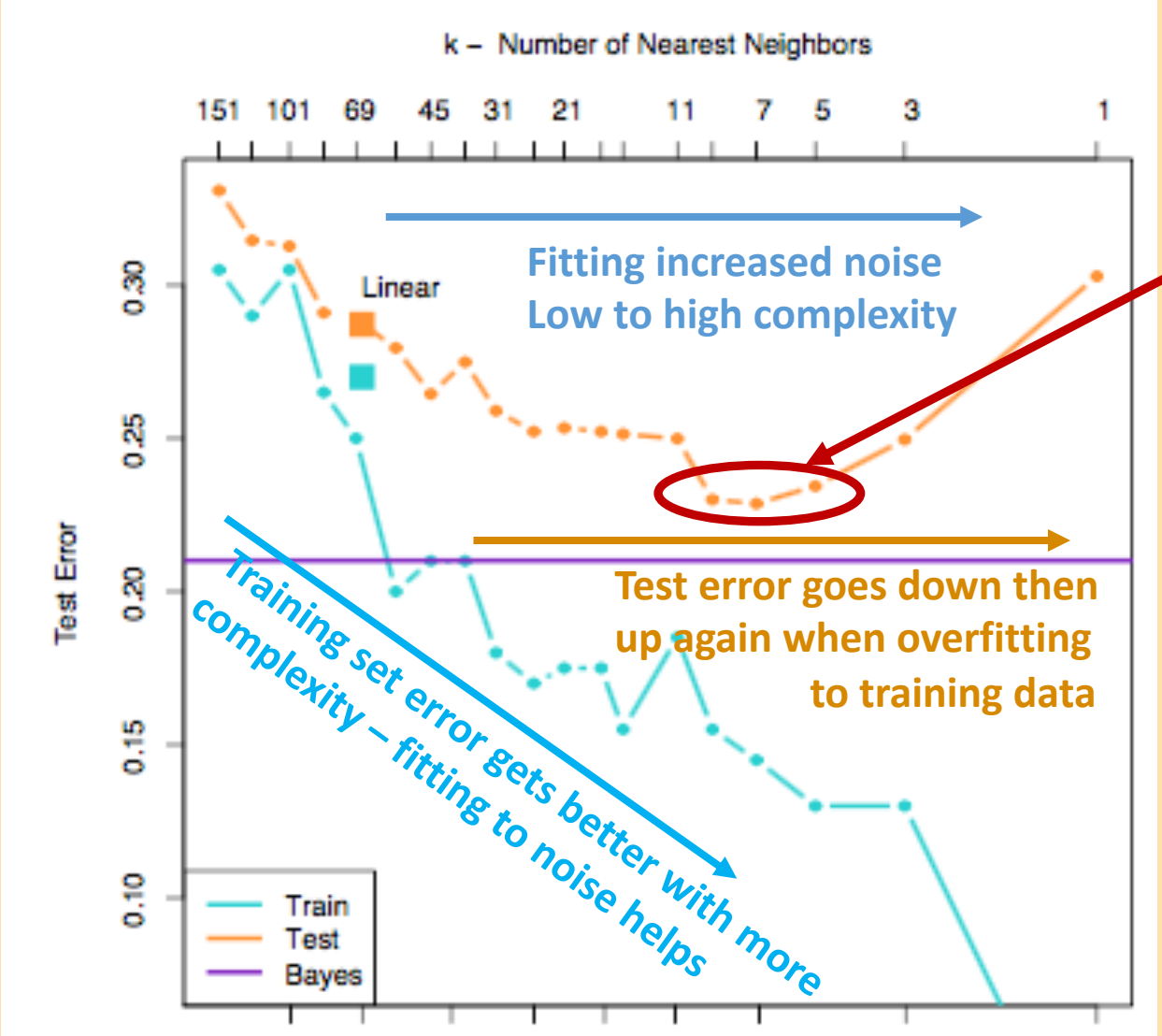
Choice of k


k – Number of Nearest Neighbors
151
101
69
45
31
21
11
7
5
3
1
Test Error
0.30
0.25
0.20
0.15
0.10
Linear
Fitting increased noise
Low to high complexity
Training set error gets better with more complexity – fitting to noise helps
Train
Test
Bayes

Choice of k

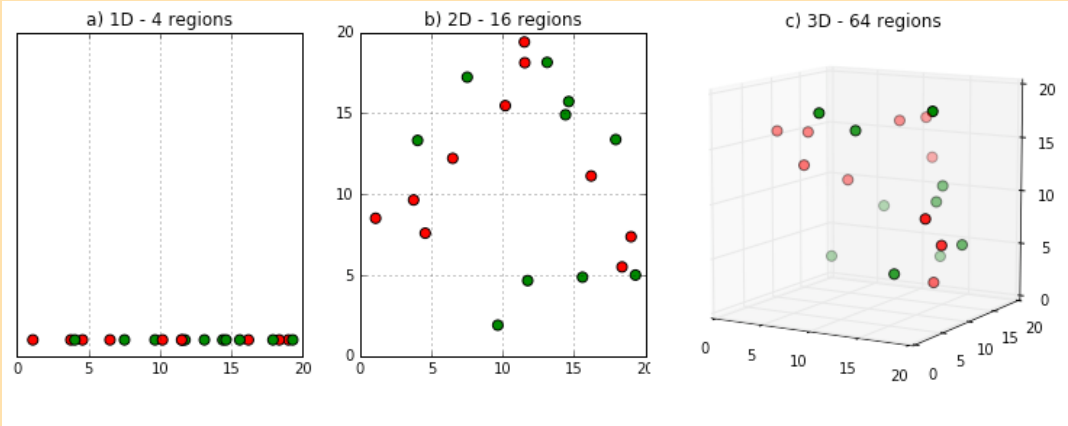

k – Number of Nearest Neighbors
151
101
69
45
31
21
11
7
5
3
1
Test Error
0.10
0.15
0.20
0.25
0.30
Linear
Fitting increased noise
Low to high complexity
Test error goes down then
up again when overfitting
to training data
Training set error gets better with more
complexity – fitting to noise helps
Train
Test
Bayes

Choice of k


k – Number of Nearest Neighbors
151 101 69 45 31 21 11 7 5 3 1
Test Error
0.10 0.15 0.20 0.25 0.30
Linear
Fitting increased noise
Low to high complexity
Best predictive models (k = 7)
Test error goes down then up again when overfitting to training data
Training set error gets better with more complexity – fitting to noise helps
Train
Test
Bayes

K-nearest neighbors classifier

- Simple technique
- No modeling, just use the training data to “vote”
- Has been applied to many machine learning problems successfully
- Have to “carry the data” to make a new prediction
- Not good for a large number of predictors – suffers from the “curse of dimensionality”



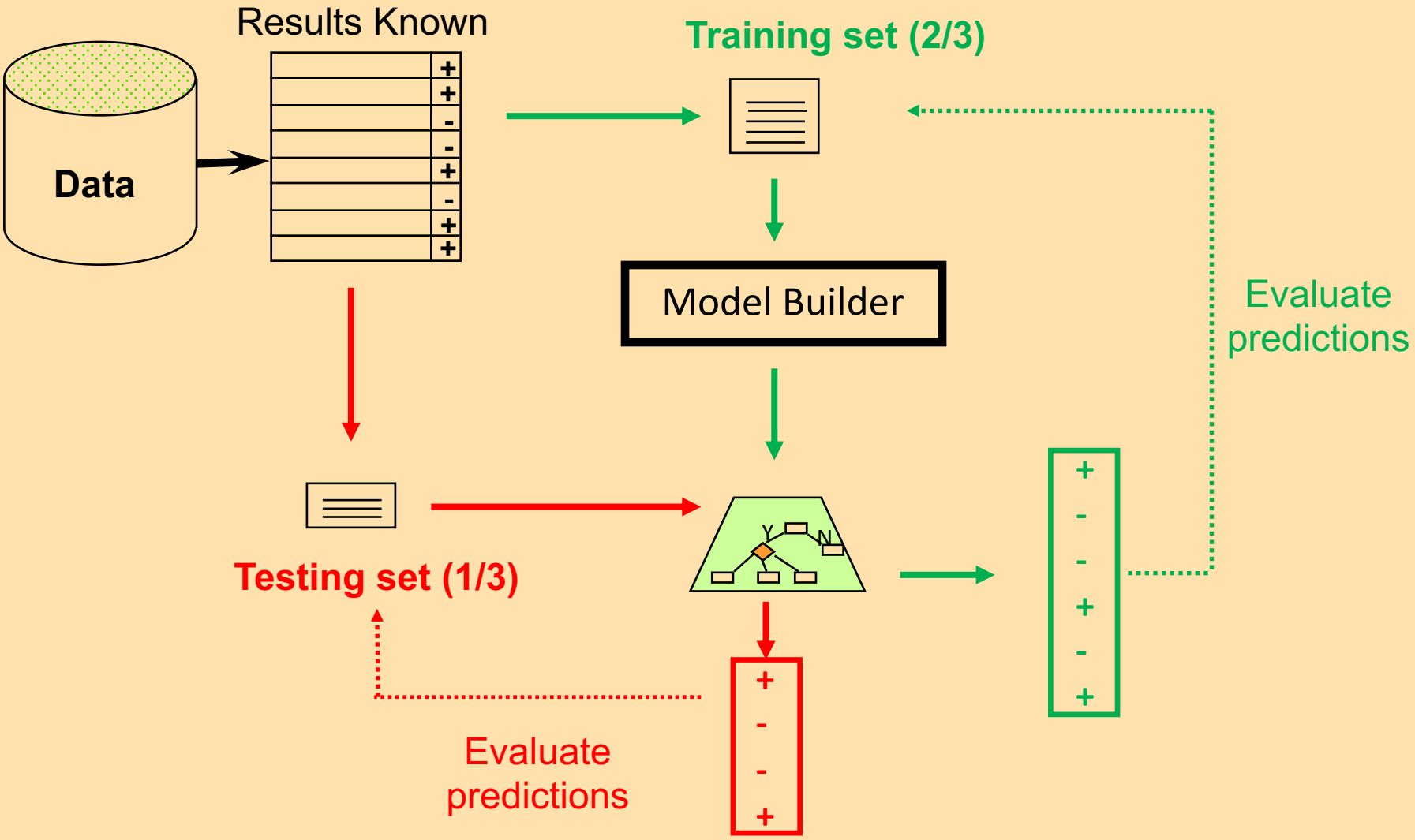
Parameter tuning

- Many machine learning algorithms have “tuning parameters” (hyperparameters) that control behavior of model (e.g. k in KNN)
- Parameter tuning occurs in three stages:
  - Stage 1: builds the model for several parameter values
  - Stage 2: calculate performance for models using evaluation metrics
  - Stage 3: select the model with optimal parameter settings, i.e. optimizes an evaluation metric in independent data set

Parameter tuning

- How do we assess performance for models across parameters? We do not want to use our test data until the very end…
- Proper procedure divides the data into three sets:
  - **Training (1/2 data):** used to train several models with different parameter values (e.g. k = $3 - 6$ for KNN)
  - **Validation (1/4 data):** used to assess predictive performance of all models, one optimal model is selected (e.g. k = 5)
  - **Test (1/4 data):** final validation performed on testing data for optimal model (i.e. k = 5)

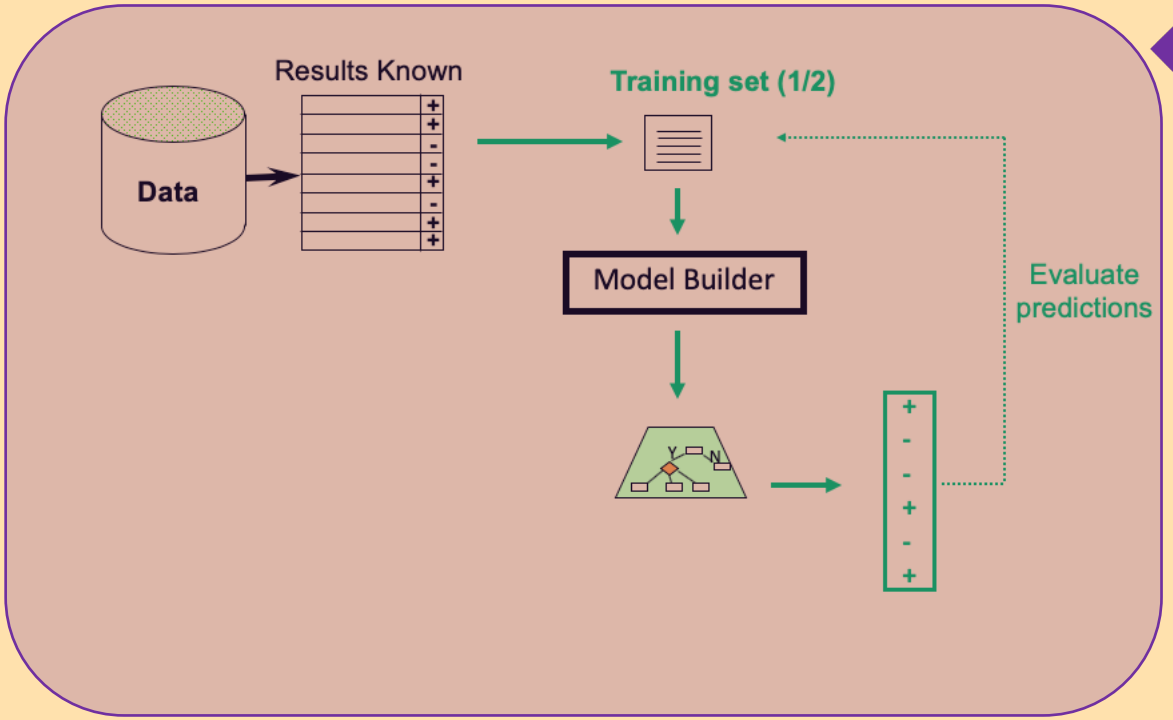
Training/Test



Train/Validate/Test

k = 3, 4, 5, 6

1. Train: repeatedly fit model to training data for multiple values of the procedures parameter(s), e.g. k = 3, 4, 5, 6 for K-nearest neighbors

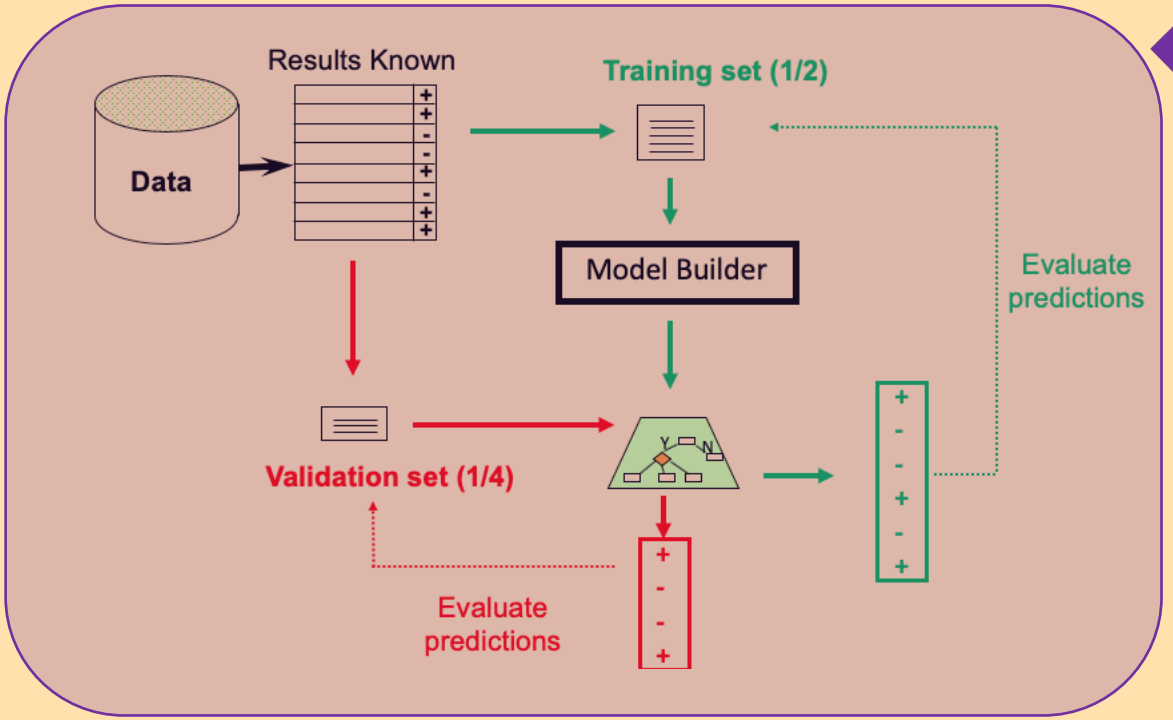


Train/Validate/Test

k = 3, 4, 5, 6

1. Train: repeatedly fit model to training data for multiple values of the procedures parameter(s), e.g. k = 3, 4, 5, 6 for K-nearest neighbors

2. Validate: each model based on training set has potentially been overfit… so utilize a validation set to determine best fitting model



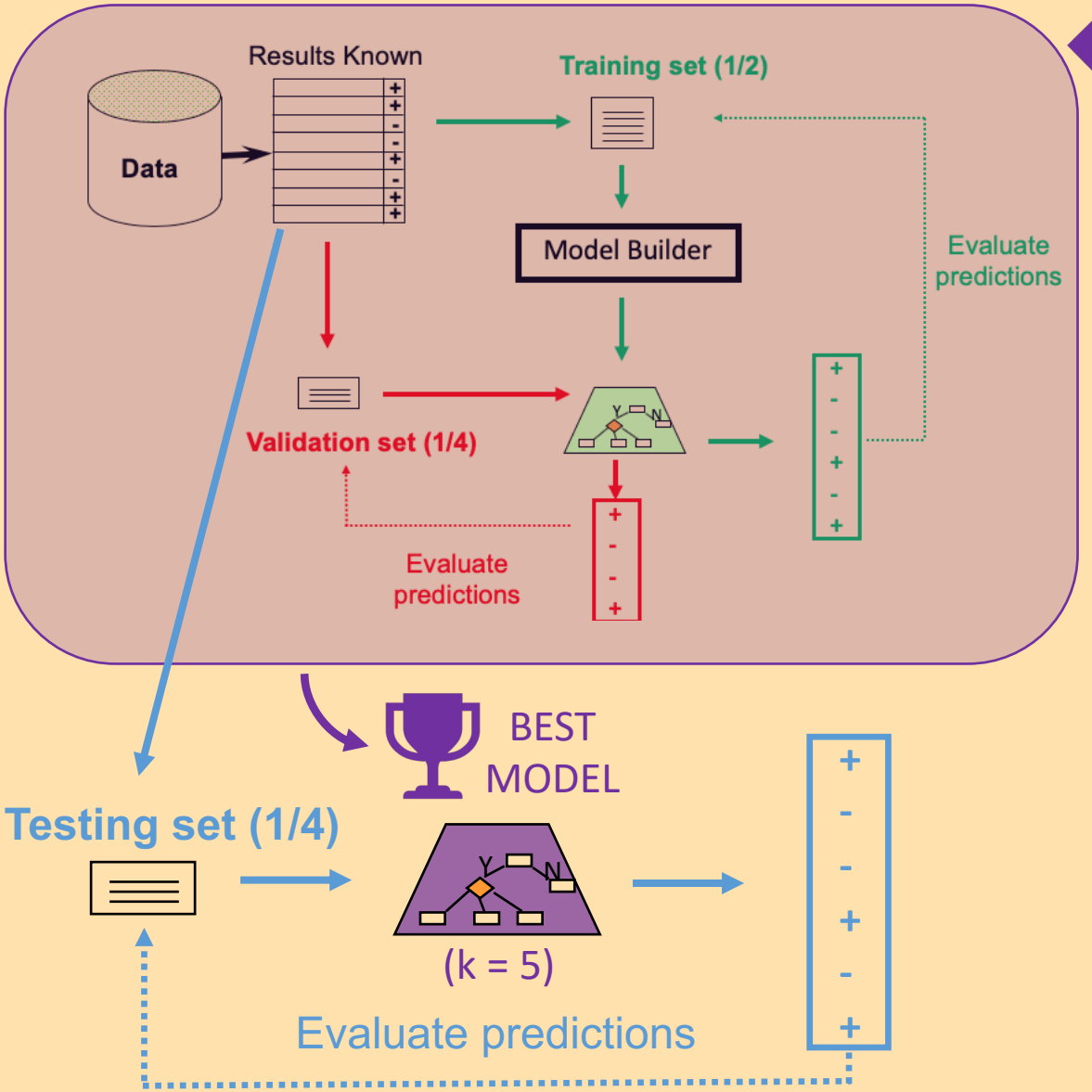
Train/Validate/Test

k = 3, 4, 5, 6

1. **Train**: repeatedly fit model to training data for multiple values of the procedures parameter(s), e.g. k = 3, 4, 5, 6 for K-nearest neighbors

2. **Validate**: each model based on training set has potentially been overfit… so utilize a validation set to determine best fitting model

3. **Test**: we are choosing from many models so we may have over-fit…utilize a test set to determine predictive performance of the best fitting model from the purple box

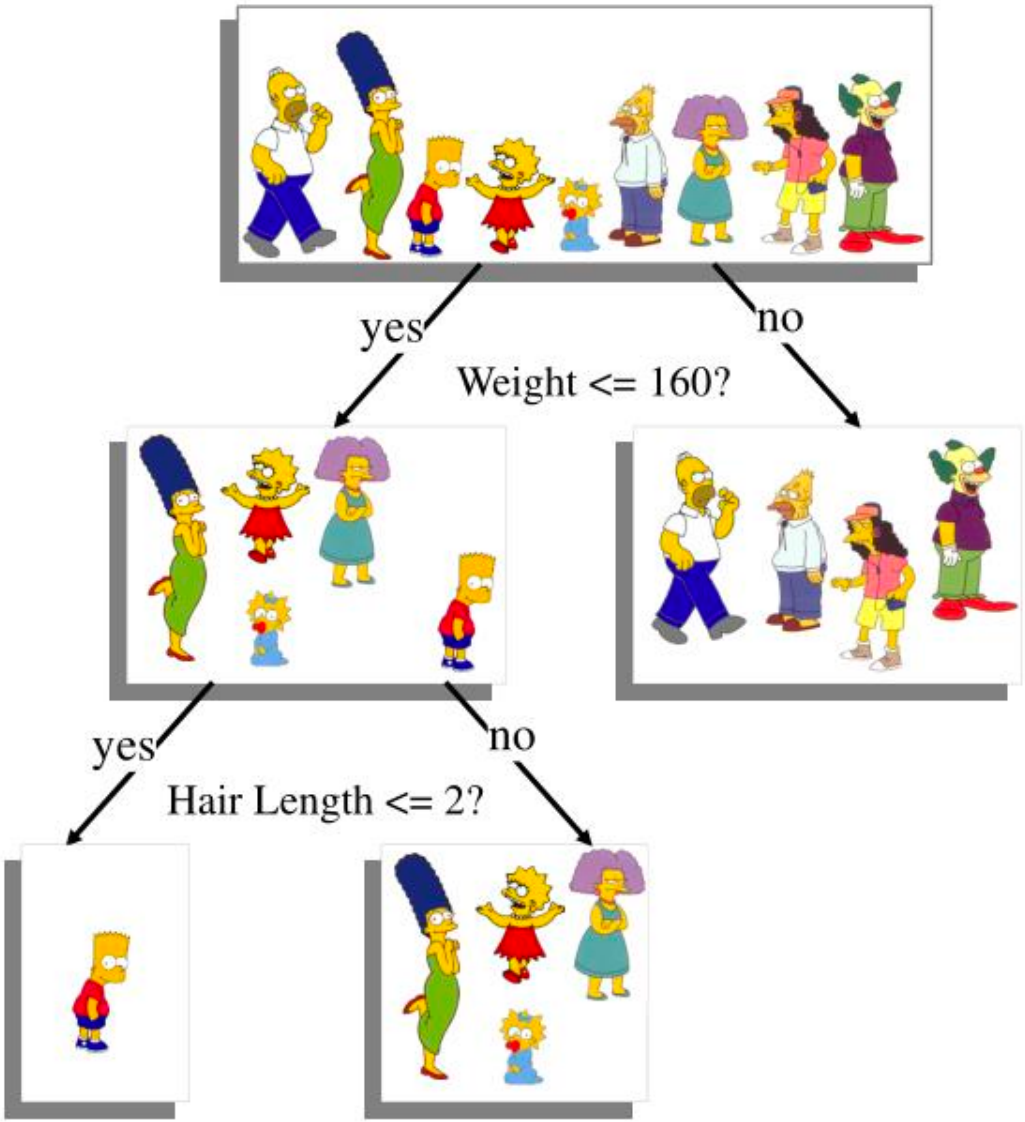


Decision Trees
(Recursive Partitioning)

Recursive partitioning – how trees work

- **Recursive partitioning**: Classification based on a series of decision cut-points along a single variable at a time
- At the end of the series of questions a *decision node* is reached
- Example: at risk or heart disease?
  - [?] Is systolic blood pressure greater that 140?
  - [?] If yes, is LDL Cholesterol greater than 190?
  - [DECISION] If yes, at risk for heart disease!
- Recursive portioning is successful because branches act like interaction terms (from Machine Learning Lecture 1)

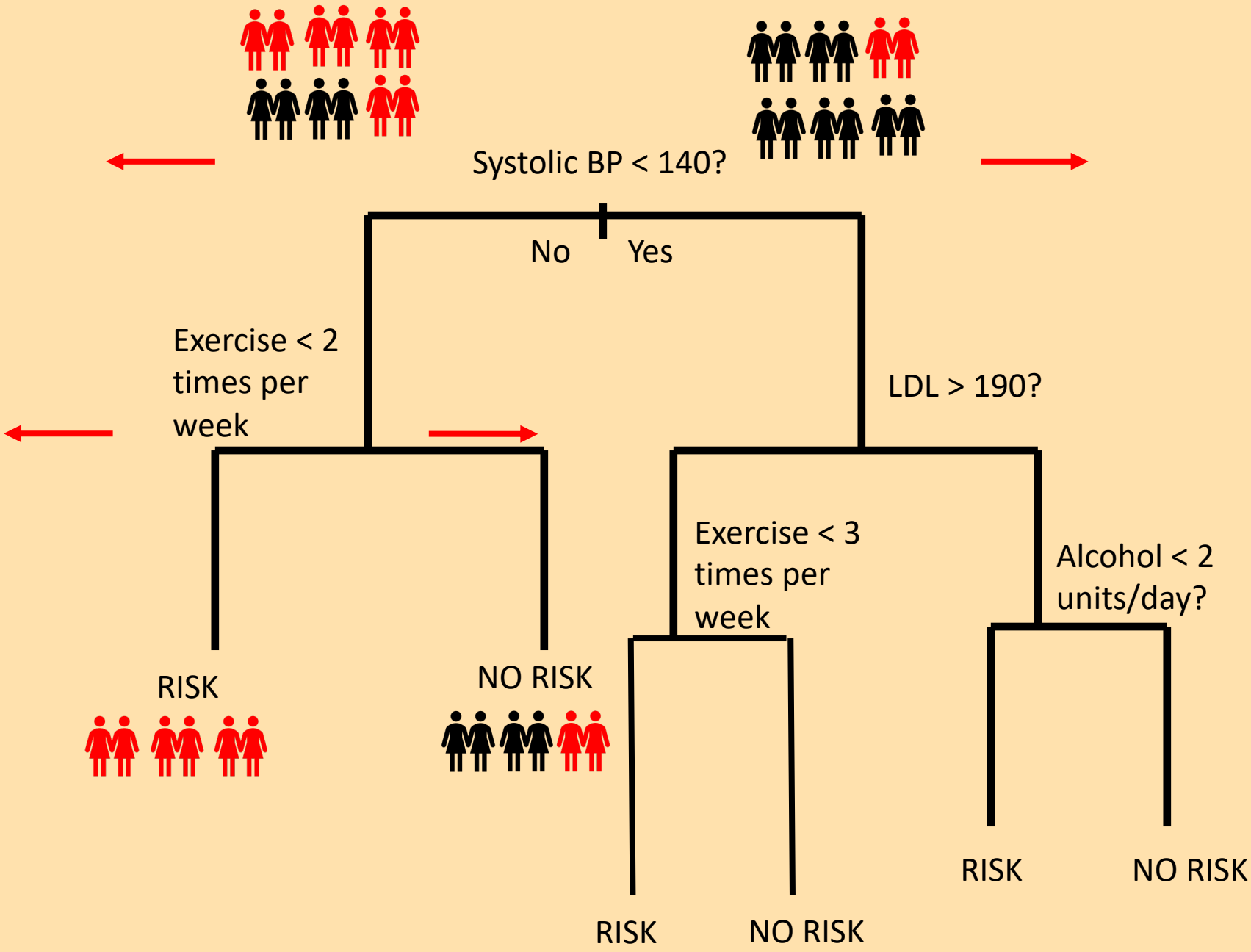
Recursive partitioning logic



At risk for heart disease?

- **Target**: risk for heart disease (RISK/NO RISK)
- **Features**:
  - Systolic BP
  - LDL cholesterol
  - Exercise (times/week)
  - Alcohol (units/day)

At risk for heart disease? RISK vs NO RISK



How are trees “grown”?

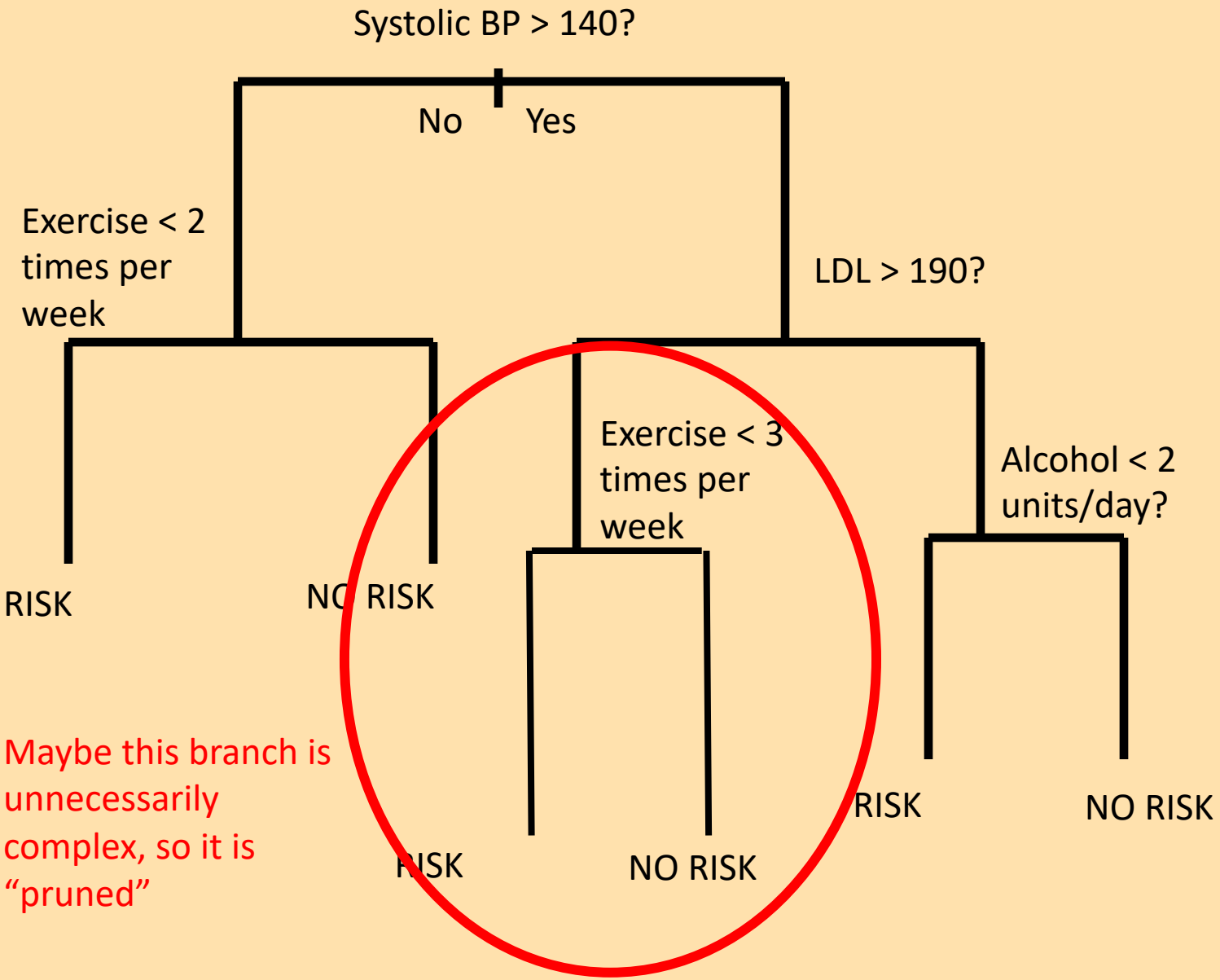
- Greedy approach - only considers one variable at a time for split

1. Start with first best split on a single variable based on some criterion – different implementations of decision trees use different criterion
   - *Information gain* is on such criterion - favors *splits that induce purity in the resulting node* [used in Orange]
   - *Gini Index* is related to information gain – slightly biased towards splitting along features with many categories
2. Then within each branch look for the best split on a single variable for the observations within that branch
3. Keep going until each branch is pure or has less than some number of observations (e.g. 5) – majority vote

Pruning trees

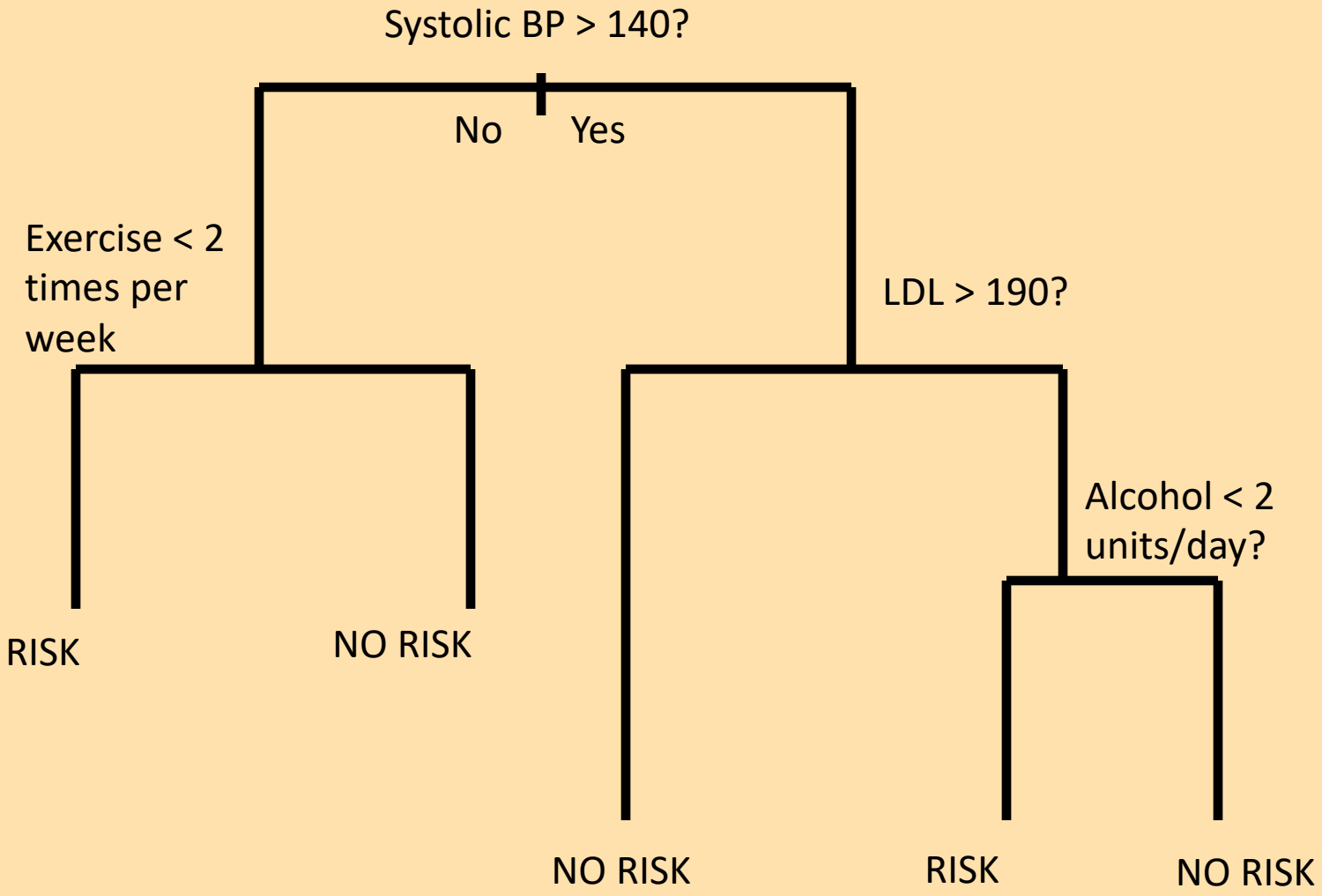
- Full tree after growing is typically too complex
- Goal is to find the sub-tree (contained in the fully grown tree) that minimizes a cost-complexity criterion (regularization!)
- The cost-complexity criterion is a trade-off between the classification error rate and the complexity (size) of the tree.
- Cut branches away that are not necessary and do not substantially improve prediction
- This trade-off can be determined train/validate/test to minimize the classification error rate

Pruning a tree



Maybe this branch is unnecessarily complex, so it is "pruned"

Pruned Tree


Systolic BP > 140?
No
Yes
Exercise < 2 times per week
LDL > 190?
Alcohol < 2 units/day?
RISK
NO RISK
NO RISK
RISK
NO RISK

Recursive Partitioning

- Recursive partitioning methods are a very popular classification approach and come in many flavors:
  - Classification and Regression Trees (CART), C5.0, CHAID, QUEST
  - Methods differ based on method for both growing and pruning
- Partitioning on cut-points approach is very popular in medicine, because mimics decision approach of clinicians
- Cutting-edge *ensemble* extensions such as Random Forests average over a bunch of trees – hence forest (more on this later)

Decision tree methods

Decision tree	Method
CART	– Gini coefficient – Twoing criteria
C5.0	Information Gain (Entropy)
CHAID	Chi-squared statistics
QUEST	Significance statistics

C5.0 is generally accepted as the best stand alone recursive partitioning method

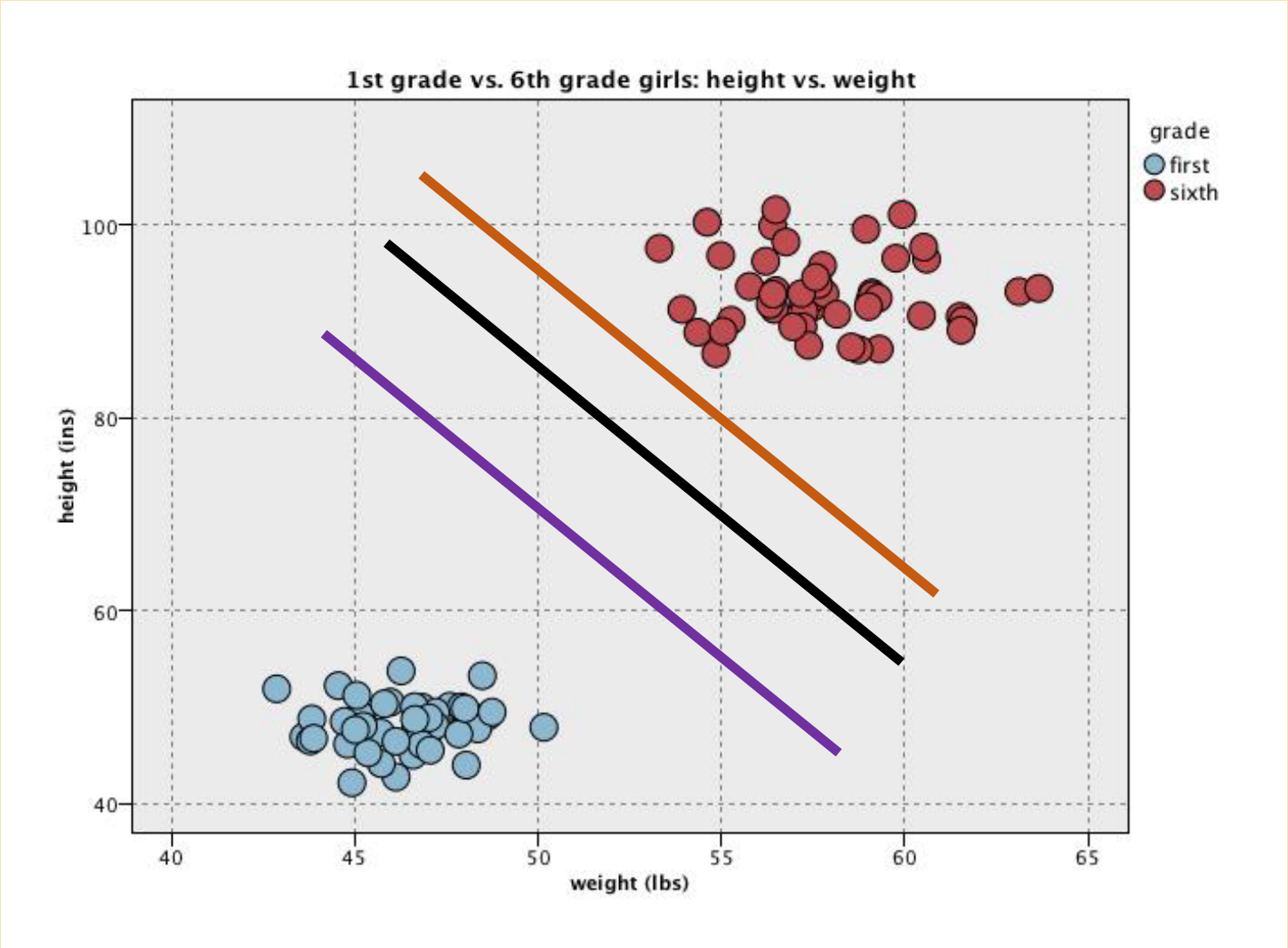
Support Vector Machines (SVM)

Support vector machines

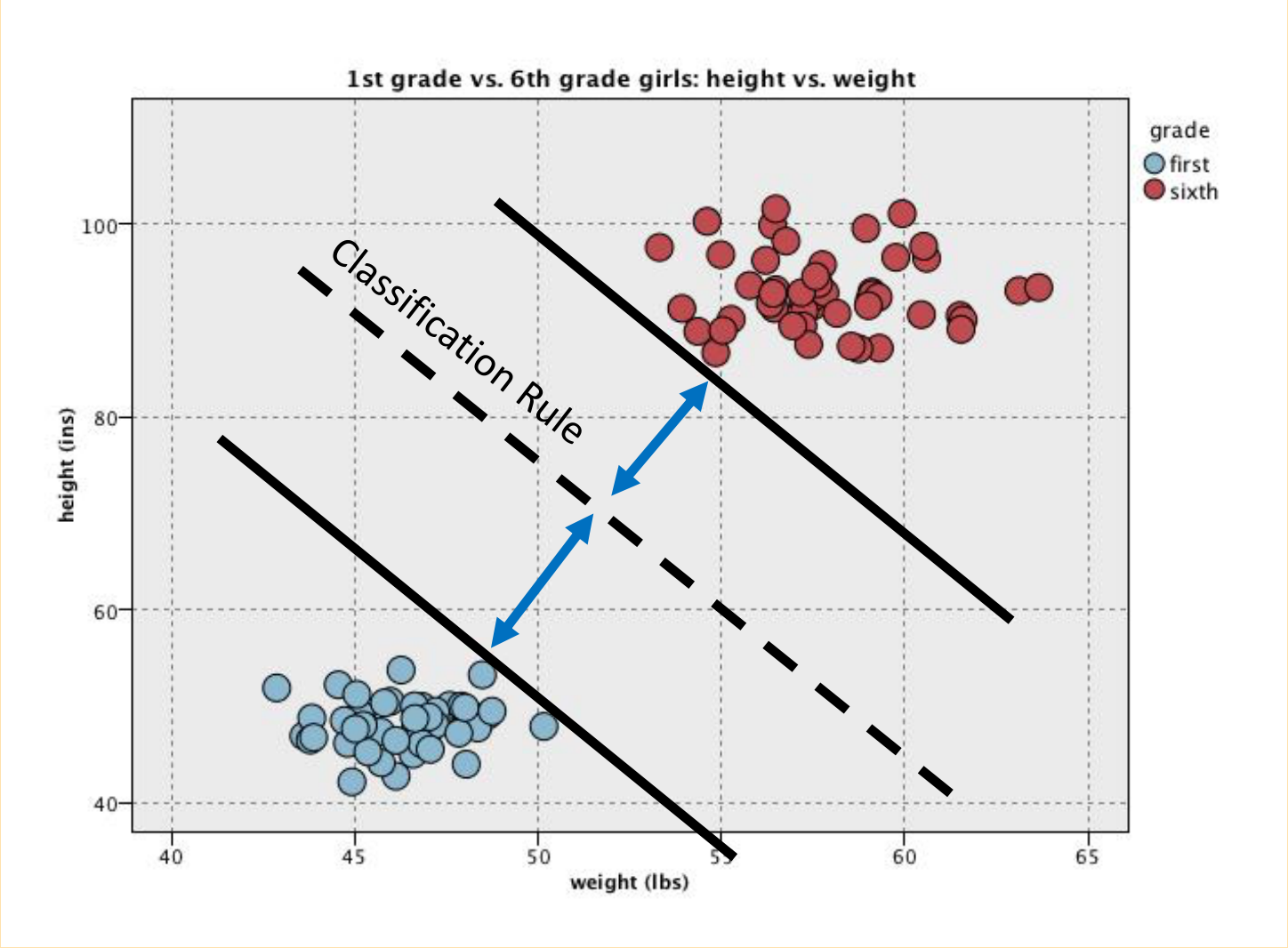
- Developed in computer science community in 1990s
- Idea is to generate a line/plane that optimally separates points from different classes in a feature space
- In SVM, this line/plane forms our classification rule and optimal separation is achieved when classes are as “far” apart as possible (i.e. have the “maximal margin”)

SVM – what is best line to separate categories?

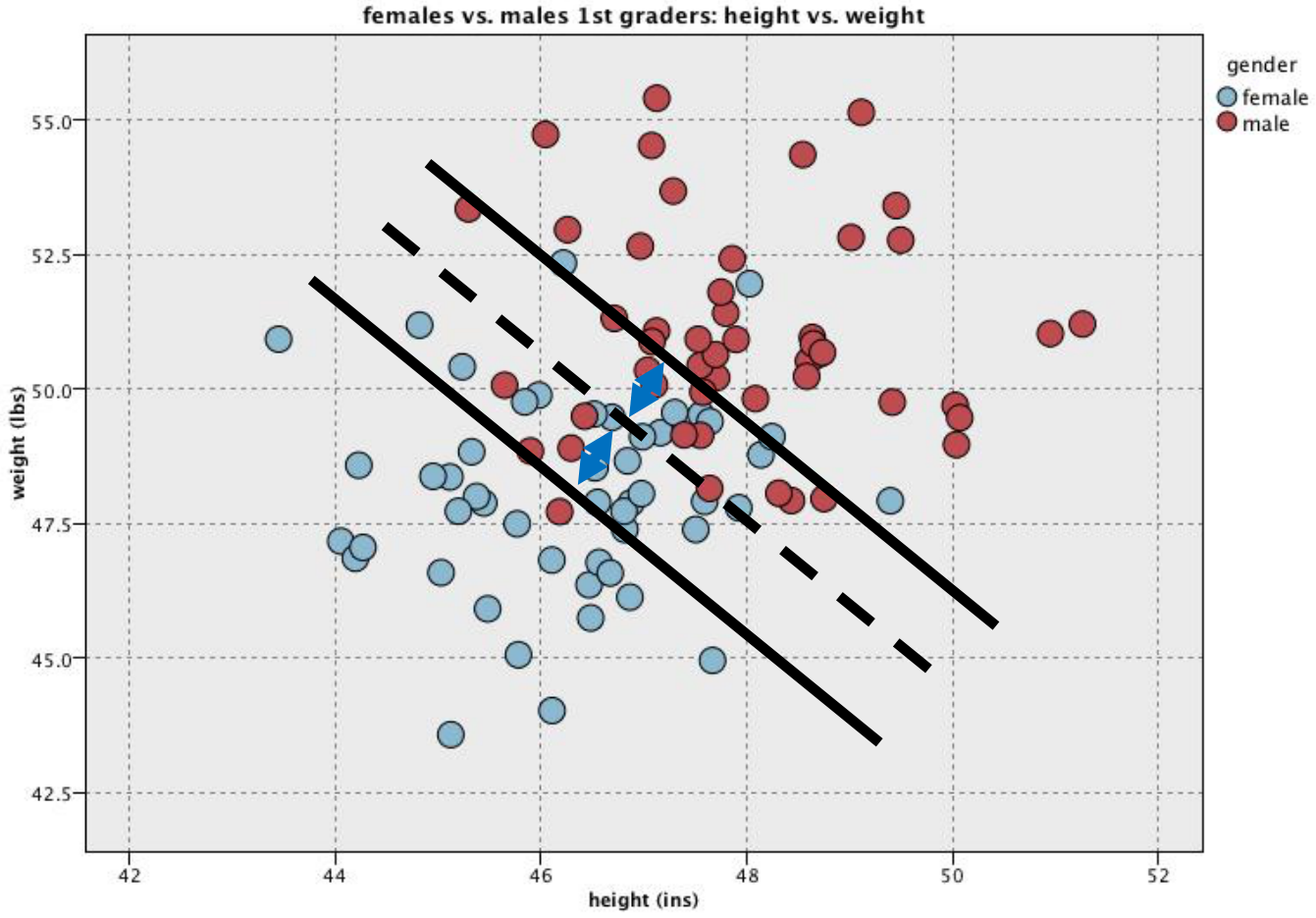
Many classification rules are possible…



Maximizing margins

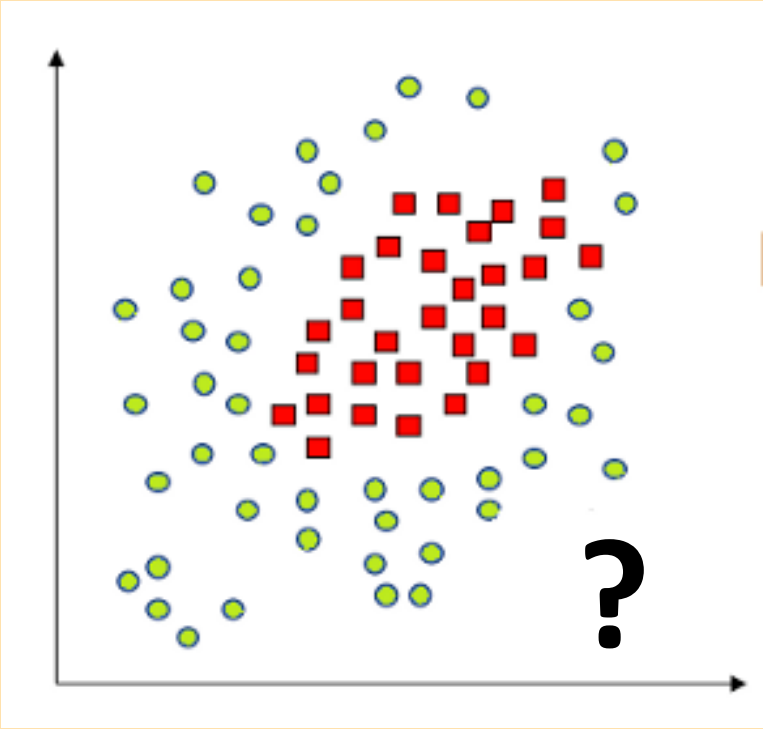


Soft margin

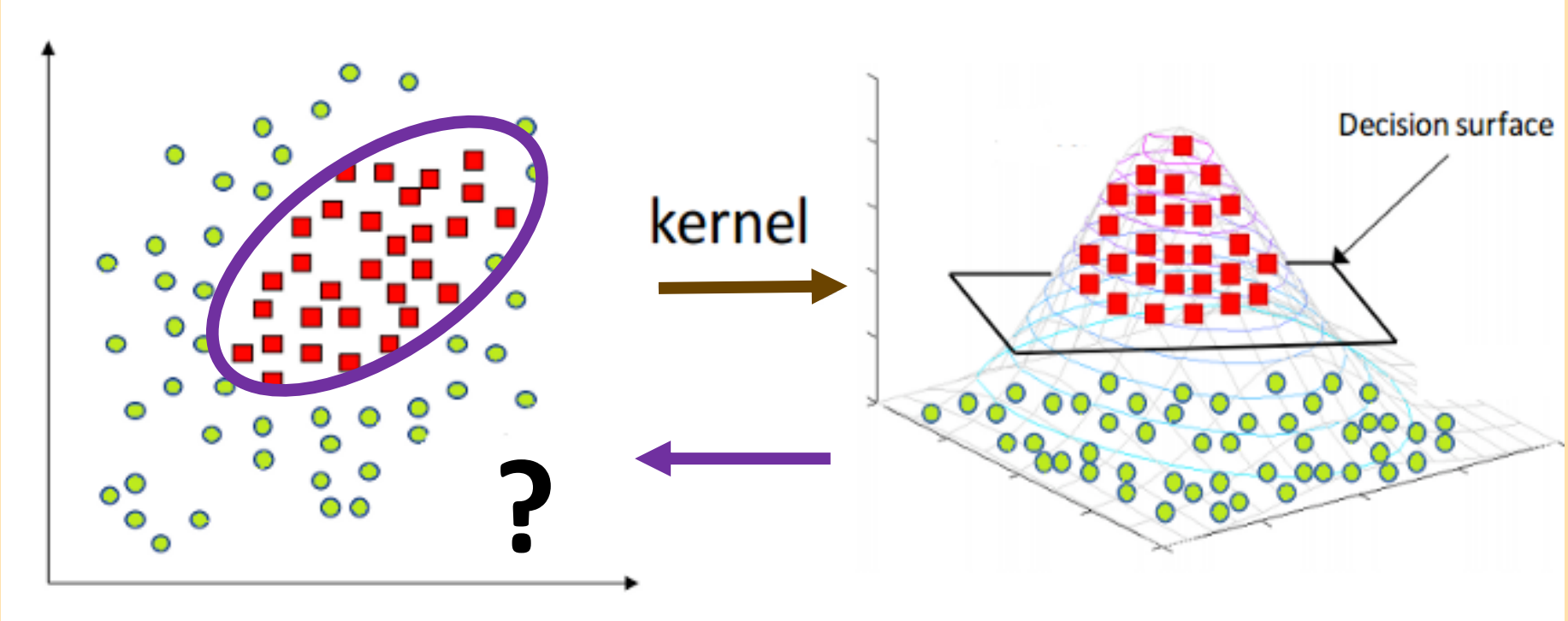


We cannot construct a maximal margin because of overlap in the data. Thus, we use a “soft margin” which balances potential misclassification with margin width

Linearity can be limited, may need "non-linear" decision boundary



Linearity can be limited, may need "non-linear" decision boundary



The trick of SVM is not to have non-linear models for the separation boundaries directly, but rather to **warp the space** that the data is in. That way, linear boundaries in the warped space become **non-linear** back in the original space.

Support vector machines

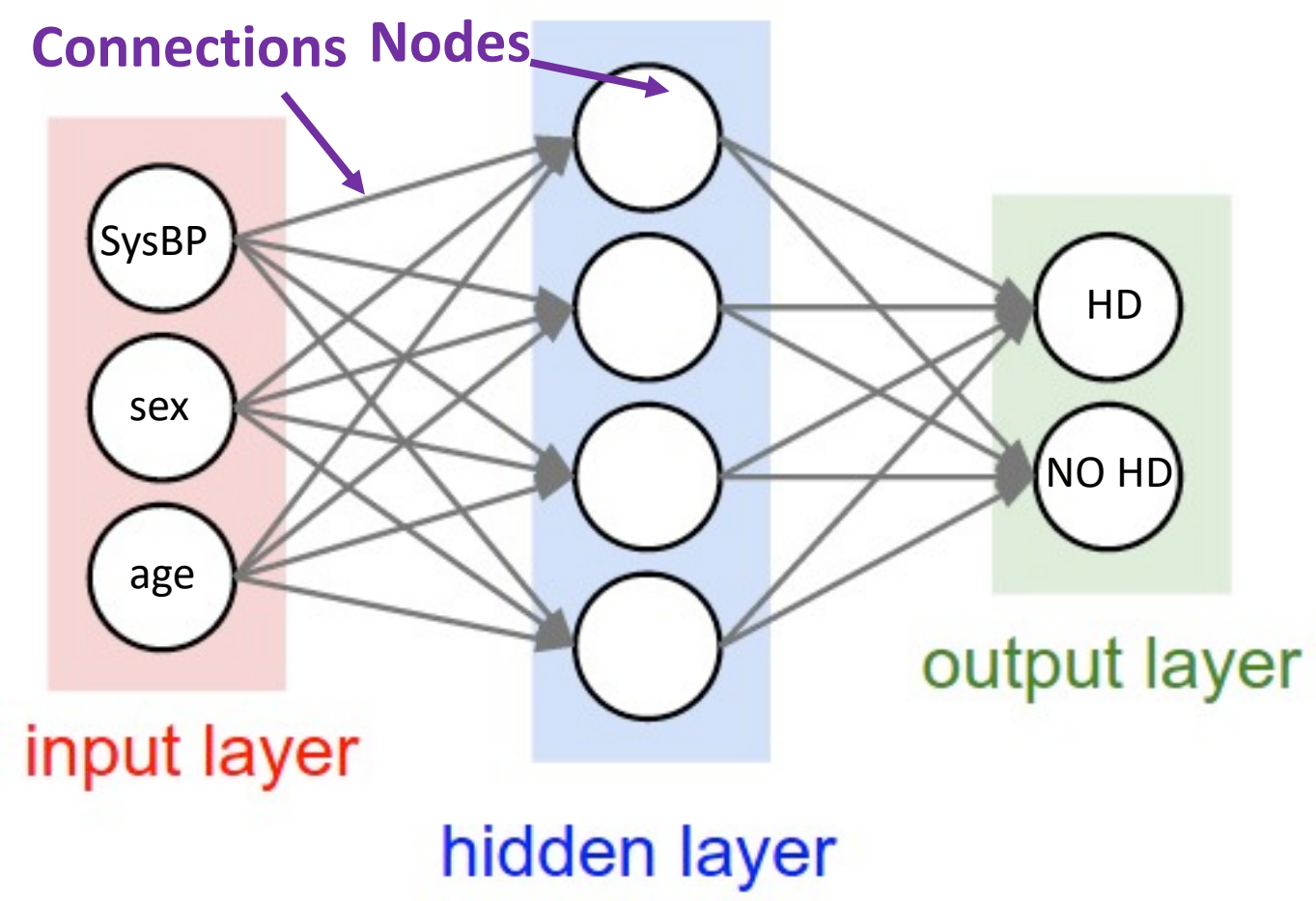
- Considers data closest to the classification boundary, robust to outliers
- Black box predictor, but not so good for interpretation
- Was considered one of the best “out of the box” classifiers for complex big data problems
- Was very popular, declining due to new methods
- It may still work well

Artificial Neural Networks

Artificial neural networks

- “A computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.”
- Inputs are fed into the neural network and processed by hundred of interconnected nodes which each perform some operation on the data (activation)
- The combined activation of all nodes throughout the network and their interactions are reduced to a simple classification rule
- The ultimate “black box” machine learning algorithm

Artificial neural networks



Underlying math is complex, but every node is a complex function of it's inputs, and the parameters/weights (connections) of the function are learned from the training data. To expand model complexity, add more hidden layers.

Artificial neural networks

- Very good when we have lots and lots of data that is complex (e.g. unstructured data)
- Allows for interactions (like decision trees) and non-linear classification rules (like K-nearest neighbors or support vector machines with kernel trick)
- Can be VERY prone to overfitting and models must be tuned carefully – have **billions** of parameters

Choosing a classifier

- **Logistic regression**
  - Good for interpretation
  - Not highly flexible
- **K-nearest neighbors classification**
  - No “model” required
  - Not so good for interpretation
  - Not so good if you have very large number of predictors
- **Decision trees**
  - Easy to interpret and explain
  - No “model” required
- **Support vector machines**
  - Can be very accurate
  - Can deal with highly non-linear separation of features
  - Requires effort to tune and difficult to interpret
- **Neural networks**
  - Very flexible structure, good for unstructured data, requires a lot of data to train
  - Prone to overfitting, requires effort to tune and difficult to interpret
- **If your primary goal is good prediction, then you can run multiple classifiers to find the best, or even combine them to get an even better model overall (*ensemble* techniques of *boosting*, *bagging* and *stacking* – *discuss later in the course*)**

Many of the models discussed today can be used for regression!

- With some adaptation, the techniques we have discussed today can be applied to target continuous outcomes, i.e. regression problems
- We will not discuss but Orange offers regression versions of the following algorithms:
  - K-nearest neighbors
  - Regression trees (recursive partitioning)
  - Support vector machines
  - Artificial neural networks

Review of this lecture

- K-nearest neighbors
  - Parameter tuning
  - Training-validation-test
- Classification trees (recursive partitioning)
- Support vector machines
- Neural networks
- Comparison of classifiers

Next Thursday—supervised learning fin

- Cross validation
- Ensemble methods
  - Bagging
  - Boosting
  - Stacking
- Feature importance