Classification

Data Mining for Business and Governance*
12/09/2017

**Formerly known as Social Data Mining*

Course Schedule

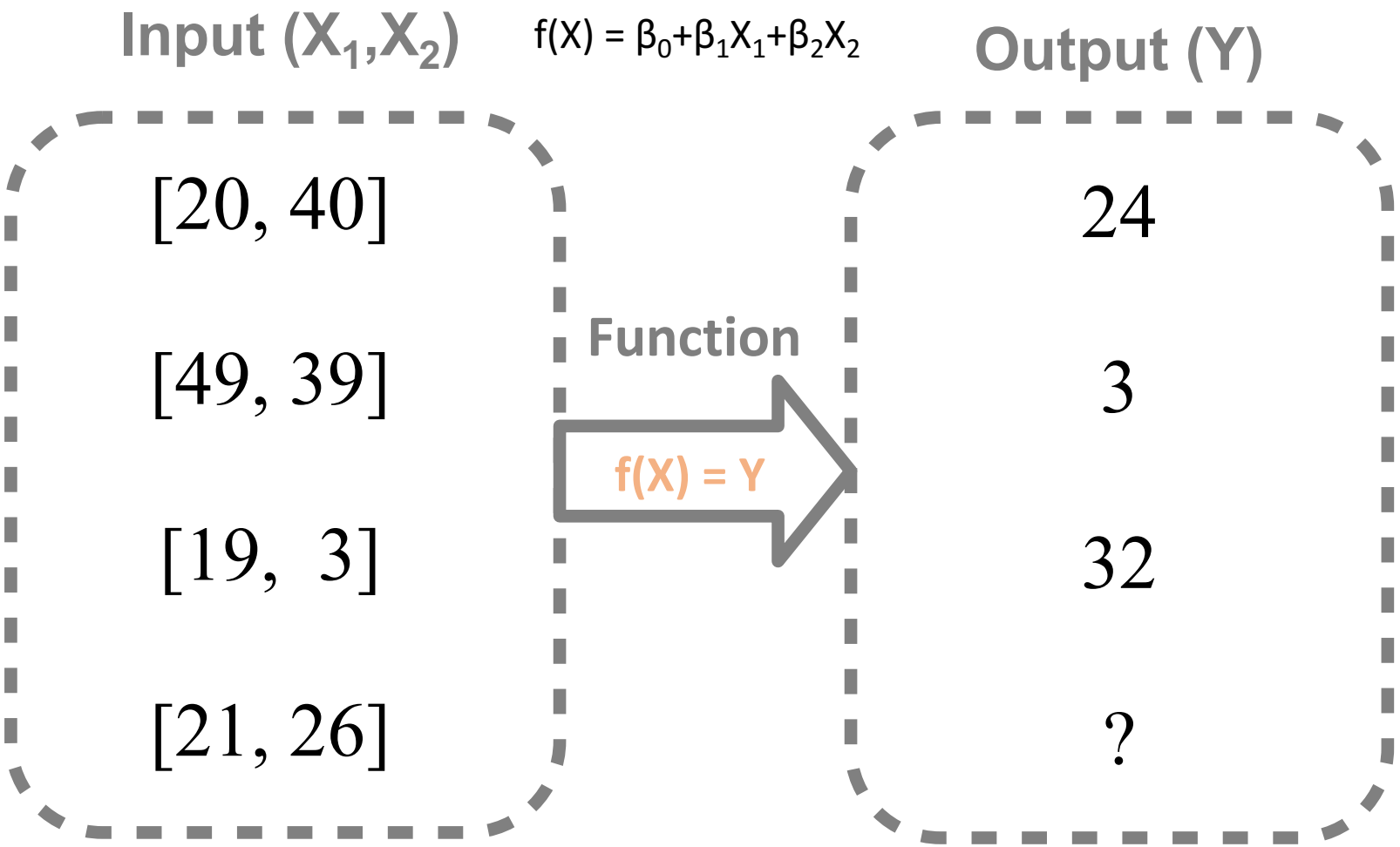
`v05.09.2017` (subject to change – always check the latest version!)

#	Date	Lectures (Theory - Willem)	Date	Video Lectures (Applications - Chris)	Video Practicals & Notebooks
1	29-08	Introduction to Data Mining	31-08	Introduction to Data Science	Introduction to jupyter, pandas, and scikit-learn
2	05-09	Regression	07-09	Representing Data: Vectors, Types, Databases	Handling & Interpreting Data, Plotting
3	12-09	Classification	14-09	Working with Text Data Part 1 (17-09)	DIY Pandas + scikit-learn
4	19-09	Algorithm Fitting & Tuning	21-09	Working with Text Data Part 2	**No practical** -> time to prepare for midterm.
5	26-09	**Midterm**	28-09	Best Practices, Common Pitfalls & Research	Preprocessing + Pipelines, MNIST Challenge
6	03-10	Data Reduction & Decomposition	05-10	Mining Massive Data, Ensemble Methods	Online / Out-of-Core Learning
7	10-10	Time Series Analysis	12-10	Applications of Deep Learning	Social Media and Multi-modal Data
8	17-10	Clustering and Graphs	19-10	Explaining Models, Ethics, Privacy	Unsupervised Learning: Intuitions and Metrics

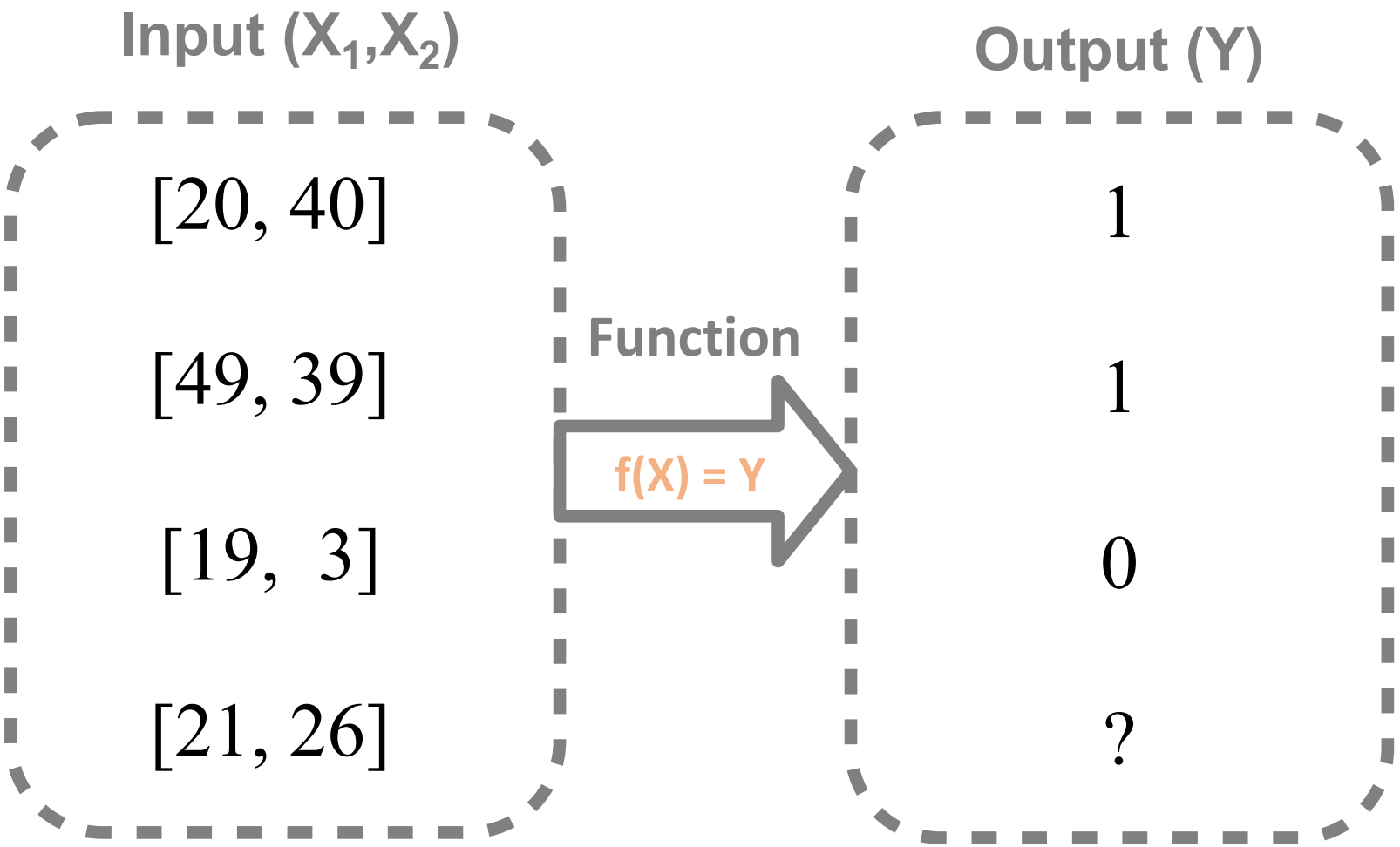
Overview: Classification

- **Classification**
- Decision Boundaries
- Learning by Example
- Logistic Regression
- K Nearest Neighbor (KNN)
- Decision Tree

Regression Task



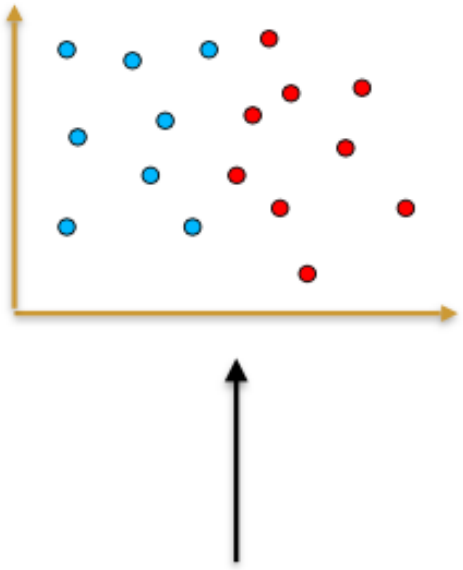
Classification Task



Classification

Examples of classification tasks:

- Stock Market features → {keep, sell}
- Transaction → {legit, fraudulent}
- Blogpost features → {female, male}
- Tumor Diagnosis → {benign, malignant}
- Iris Species → {setosa, versicolor, virginica}



Strongly idealised!
We almost never see such clear separation in real datasets.

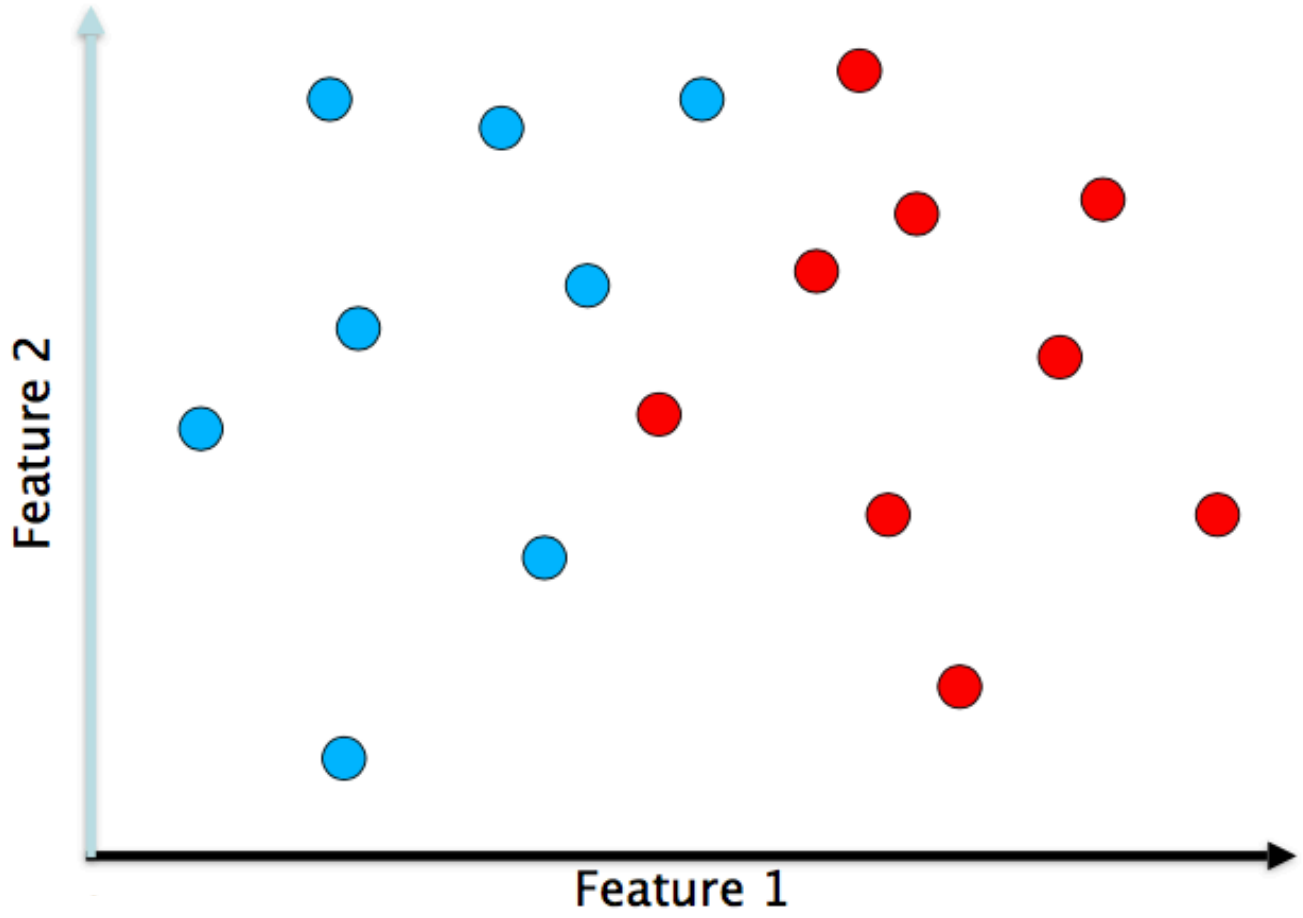
Assignment to a class: $y \varepsilon \{0,1\}$

0 = "negative class" (e.g legit, benign}

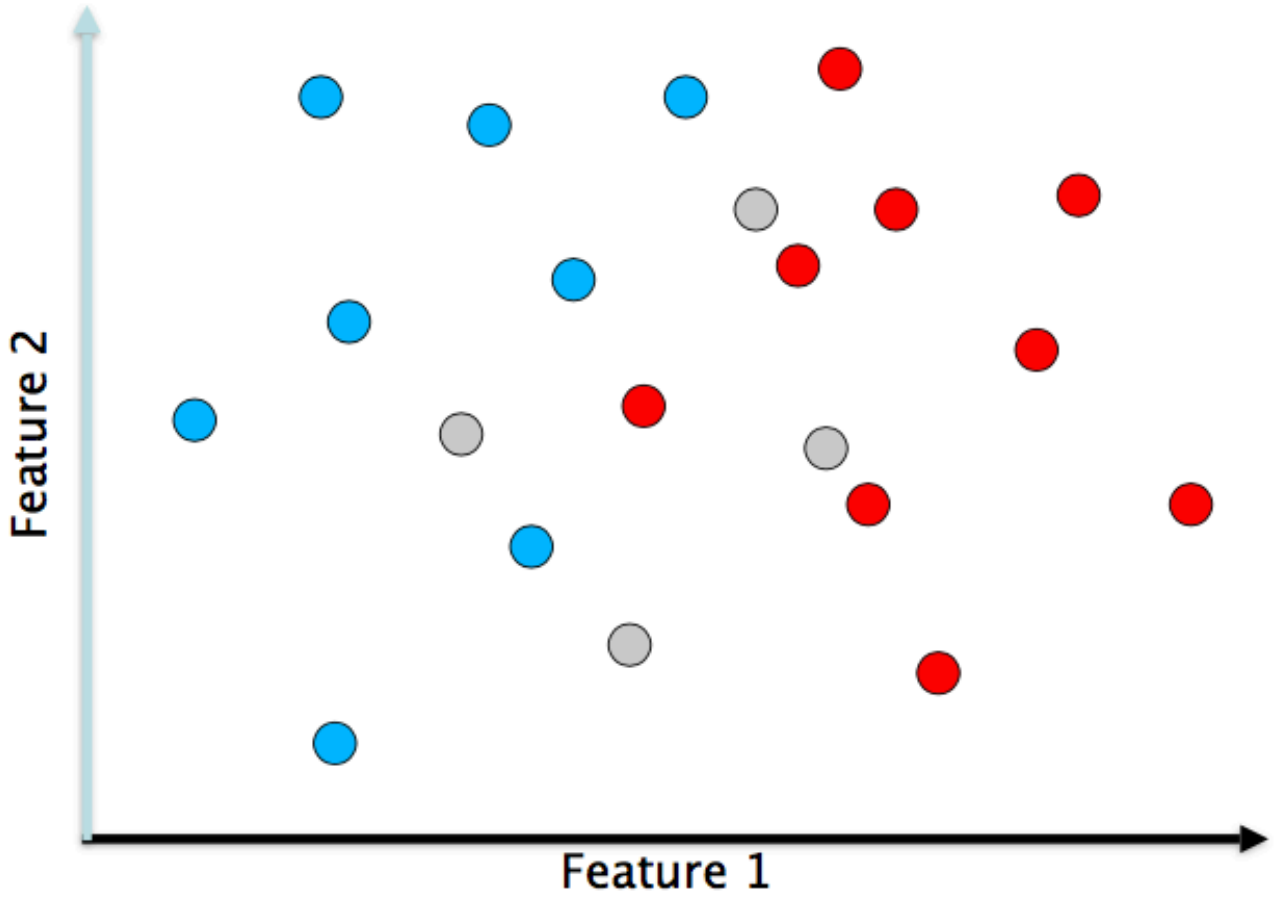
1 = "positive class" (e.g. fraudulent, malignant)

Each instance (point) has a class label, which we represent by colors {blue, red}, and can be represented by feature vectors $[X_1 X_2]$

Example of two classes defined by two features



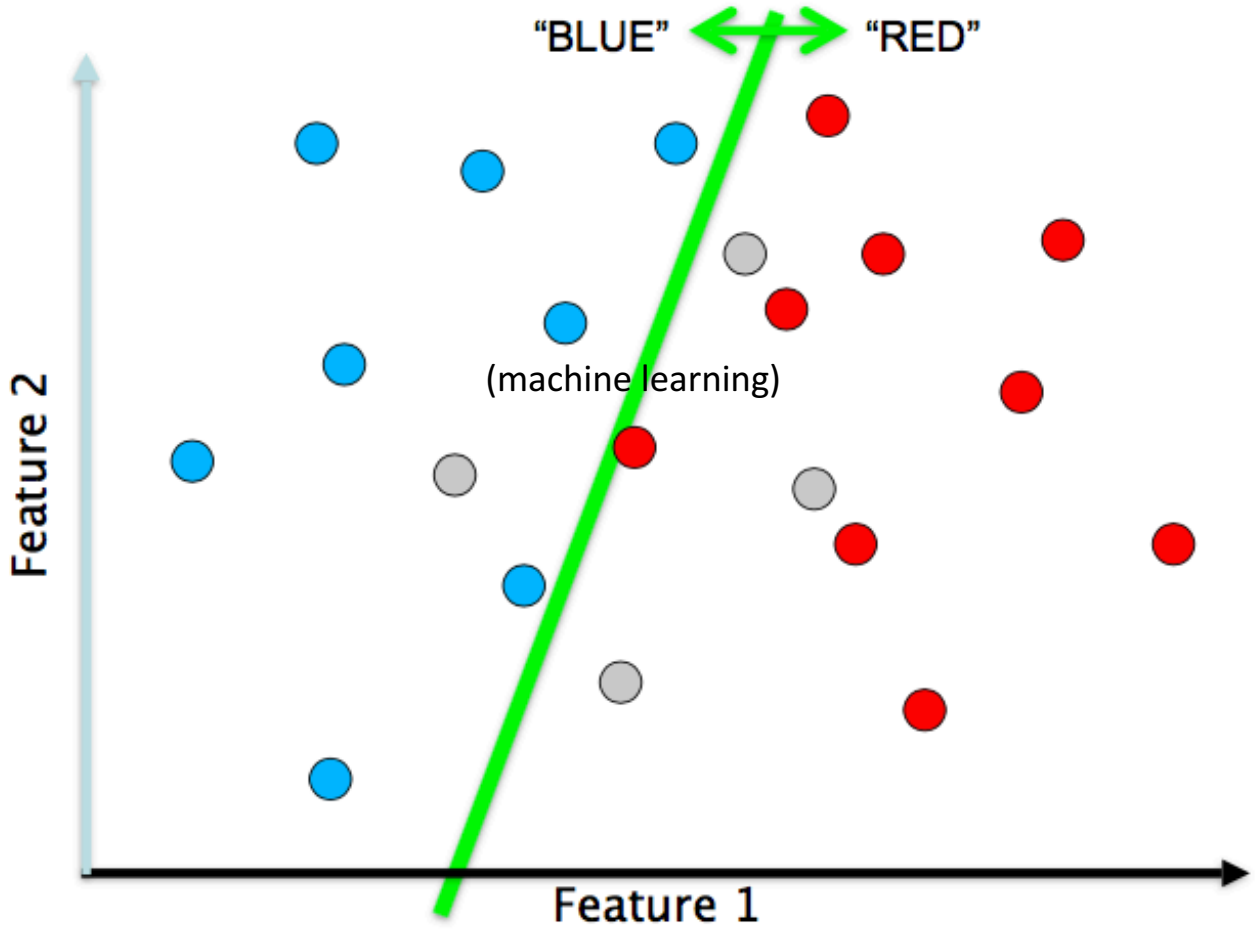
What are the class labels assigned to the grey instances?



Decision Boundaries

- Classifiers are trained on the dataset (labelled data points) and automatically “draw” a decision boundary between the two classes

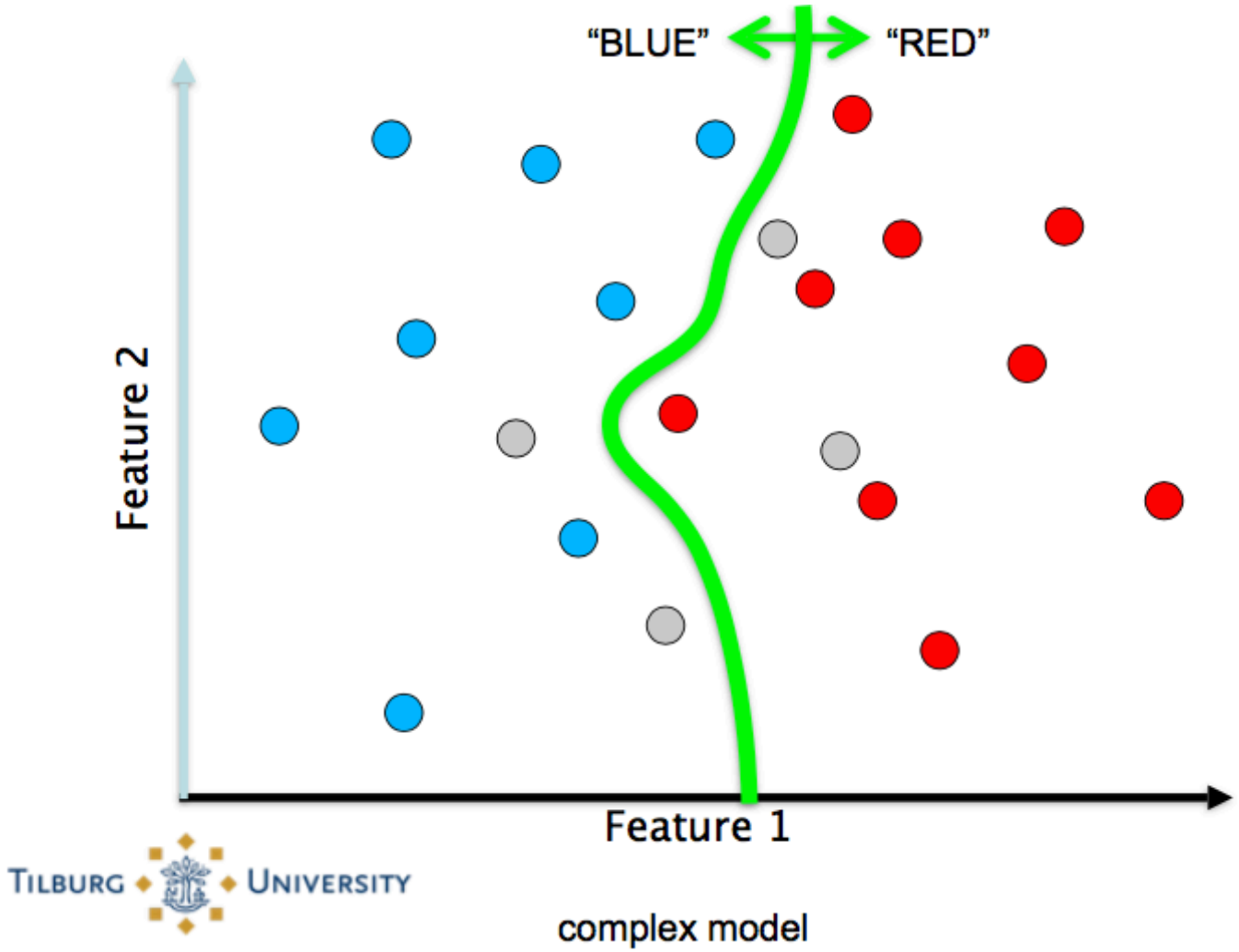
Decision Boundaries: linear



Decision Boundaries

- Classifiers are trained on the dataset (labelled data points) and automatically “draw” a decision boundary between the two classes
- The decision boundary can be a straight line (“stiff”) or a wiggly line (“flexible”). Depends on the number of parameter

Decision Boundaries: non-linear



Decision Boundaries

- Classifiers are trained on the dataset (labelled data points) and automatically “draw” a decision boundary between the two classes
- The decision boundary can be a straight line (“stiff”) or a wiggly line (“flexible”). Depends on the number of parameter
- The decision boundary is considered to be a model of the separation between the two classes $f(X) = Y$, $h_\theta(x) = y$
- The model is “learned” from the data (iterative process)

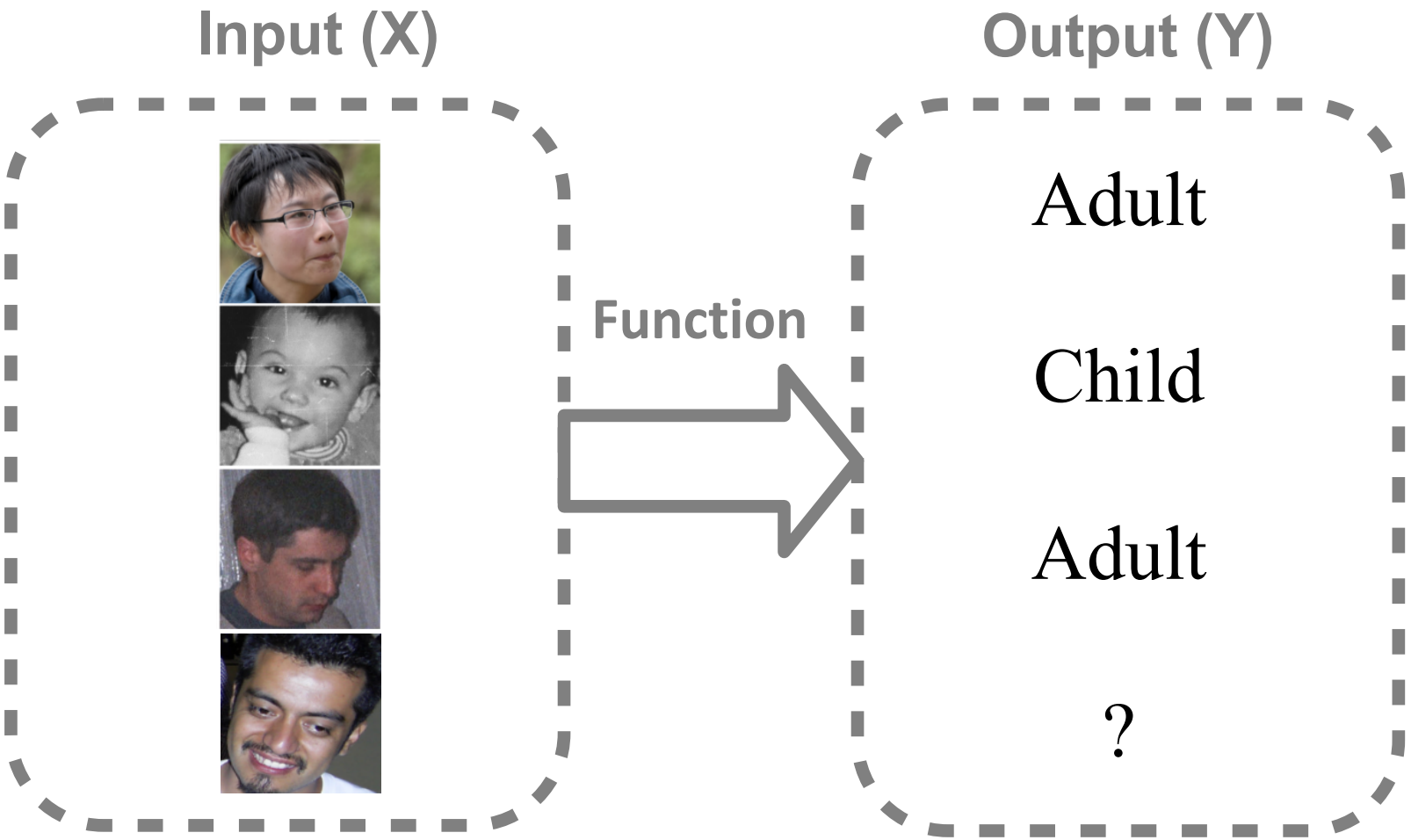
Overview: Classification

- Classification
- Decision Boundaries
- **Learning by Example**
- Logistic Regression
- K Nearest Neighbor (KNN)
- Decision Tree

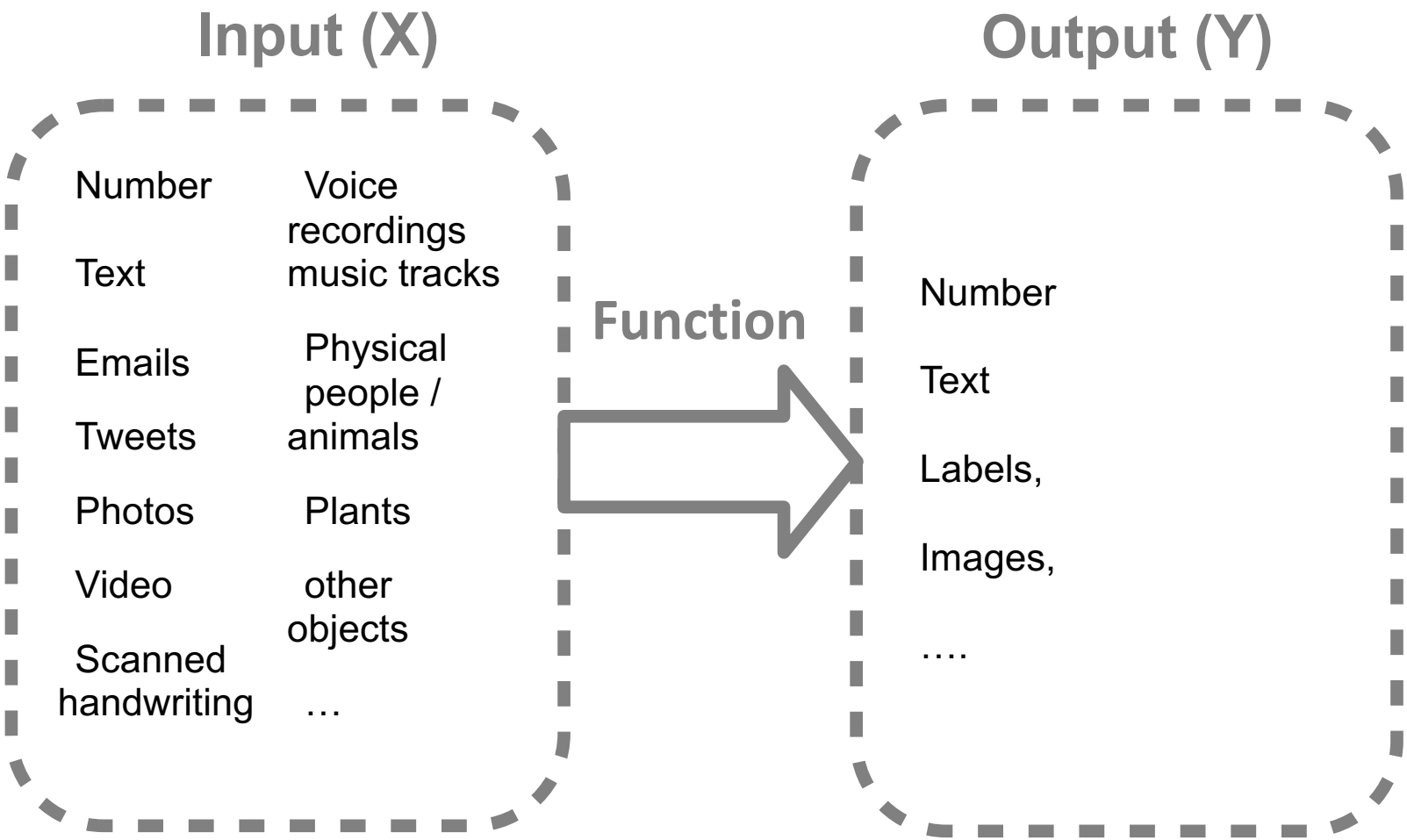
What is a learning example?

- Generic algorithms require common representations
- Decompose objects into Features
- We decompose inputs into features
- A feature is a measurable aspect of an object
- Features are often extracted before learning
- Some learning algorithms can extract features from some types of input (e.g. images or text)

Classification Task



Input / Output Mapping



We want to distinguish between three species of the iris plant

Iris setosa

Iris versicolor

Iris virginica

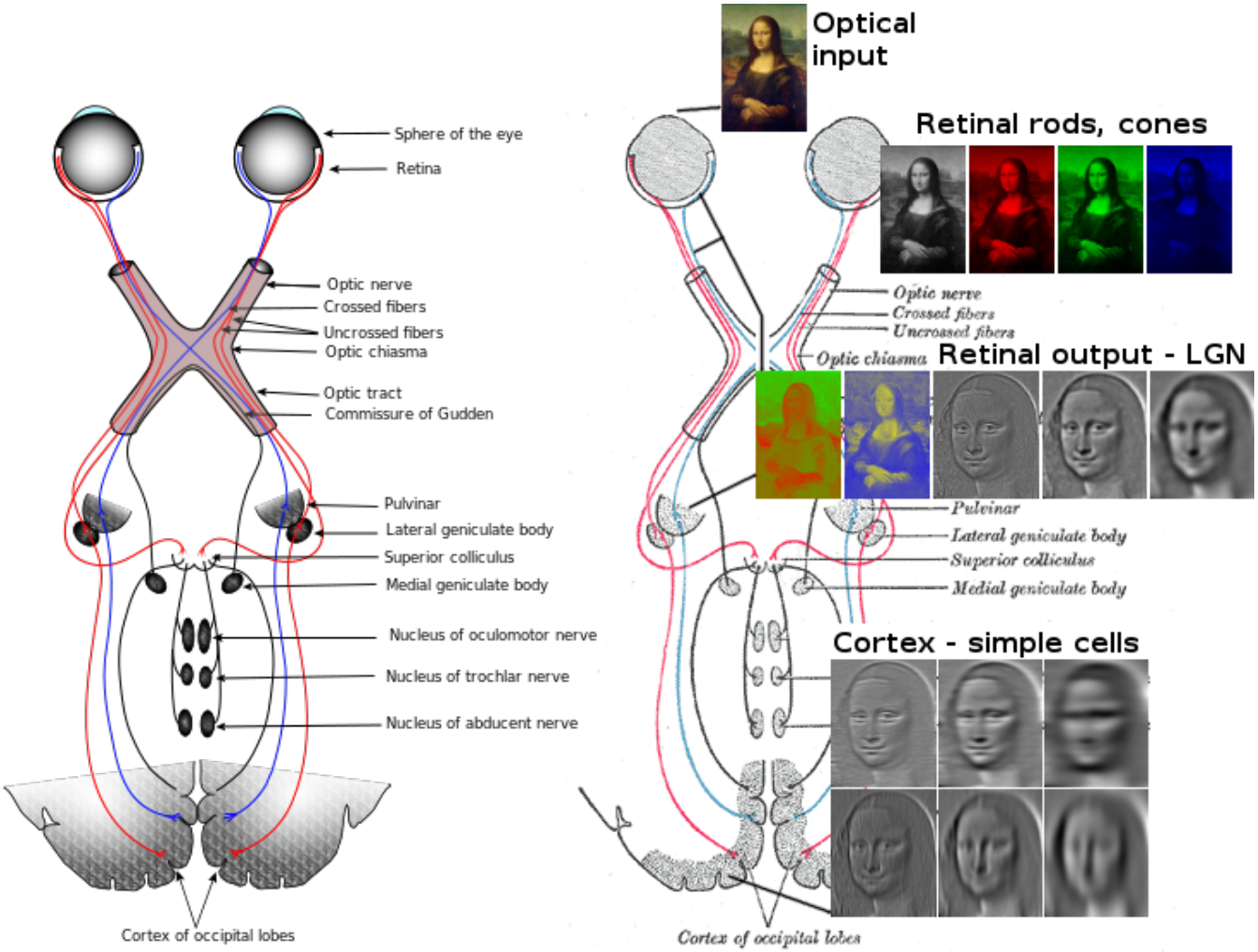
How do we extract features?

- If inputs are physical samples of flowers
  - Manual or automatic measurements
  - Size of petals, leaves, color, weight, ..

If inputs are photographs of flowers

- Image processing: edges, color, gradients, ...
- Automatic learning of features from pixels

Our brain extracts features



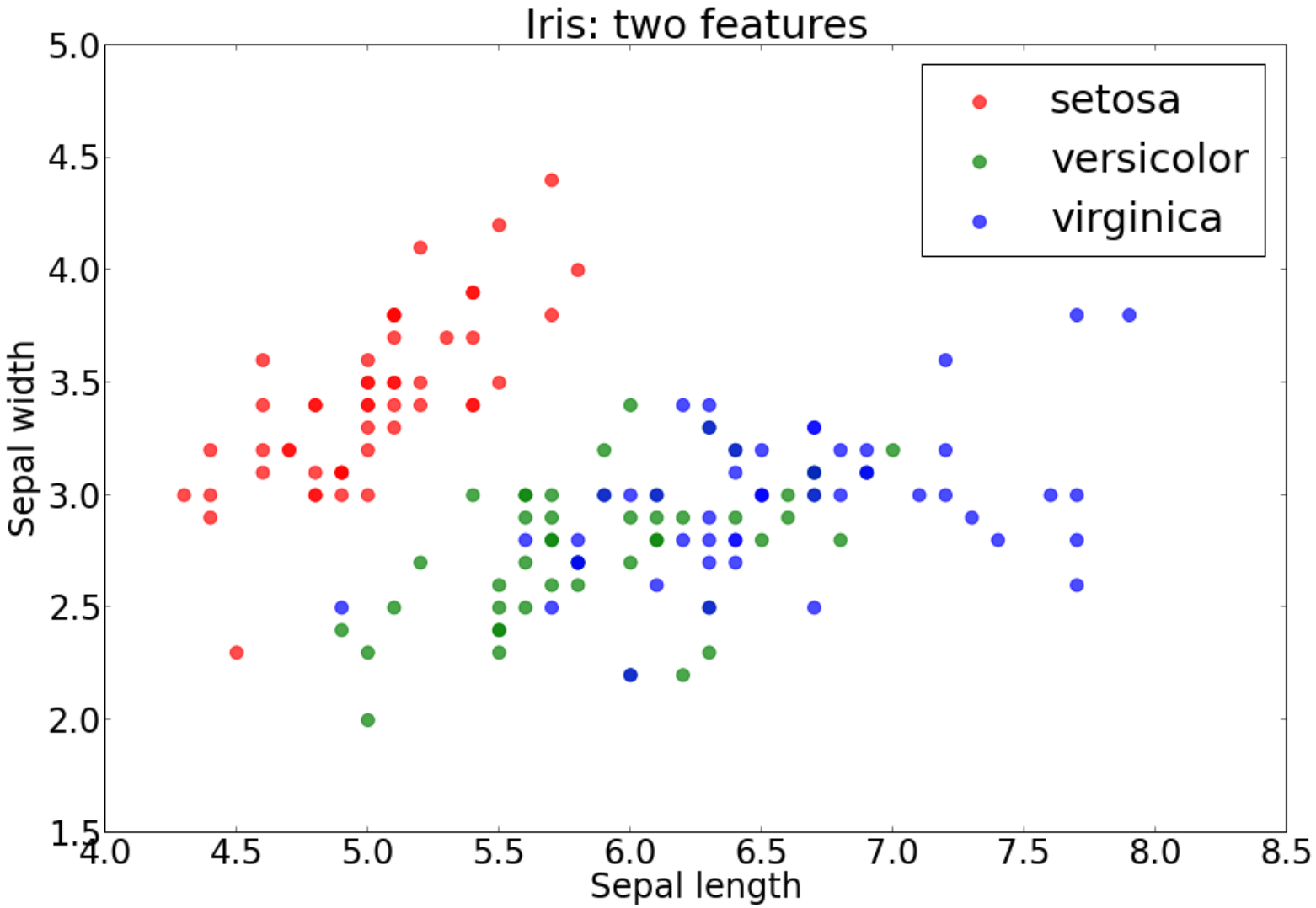
The iris dataset:

physical samples → numerical features

INPUT				OUTPUT
6.9	3.2	5.7	2.3	virginica
5.4	3.4	1.5	0.4	setosa
7.2	3.0	5.8	1.6	virginica
6.3	3.3	4.7	1.6	versicolor
5.8	2.7	3.9	1.2	versicolor
7.2	3.6	6.1	2.5	virginica
5.4	3.9	1.7	0.4	setosa

Features:

Sepal_Length, Sepal_Width, Petal_Length, Petal_Width


Iris: two features
setosa
versicolor
virginica
Sepal width
Sepal length
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5

Census income

INPUT					TARGET
age	edu	occupation	race	sex	income
39	13	Adm-clerical	White	Male	<=50K
50	13	Exec-managerial	White	Male	<=50K
38	9	Handlers-cleaners	White	Male	<=50K
53	7	Handlers-cleaners	Black	Male	<=50K
28	13	Prof-specialty	Black	Female	<=50K
37	14	Exec-managerial	White	Female	<=50K
49	5	Other-service	Black	Female	<=50K
52	9	Exec-managerial	White	Male	>50K
31	14	Prof-specialty	White	Female	>50K
42	13	Exec-managerial	White	Male	>50K
37	10	Exec-managerial	Black	Male	>50K
30	13	Prof-specialty	Asian	Male	>50K
23	13	Adm-clerical	White	Female	<=50K
32	12	Sales	Black	Male	<=50K

Preprocessing: text-mining etc.

Feature transformation: Hollingshead score ~ education * occupation

Categorical features

Some algorithms can easily use categorical features such as `occupation` or `race` or `sex` (`in R implemented as “factor”`)

In many cases we'll convert them to numerical features

Categorical → Numerical

race	sex
White	Male
Black	Male
Black	Female
White	Female
White	Male
Asian	Male

White	Black	Asian	Male	Female
1	0	0	1	0
0	1	0	1	0
0	1	0	0	1
1	0	0	0	1
1	0	0	1	0
0	0	1	1	0

White=1, Black=2, Asian=3 implies that Black/Asian is some similar than White/Asian

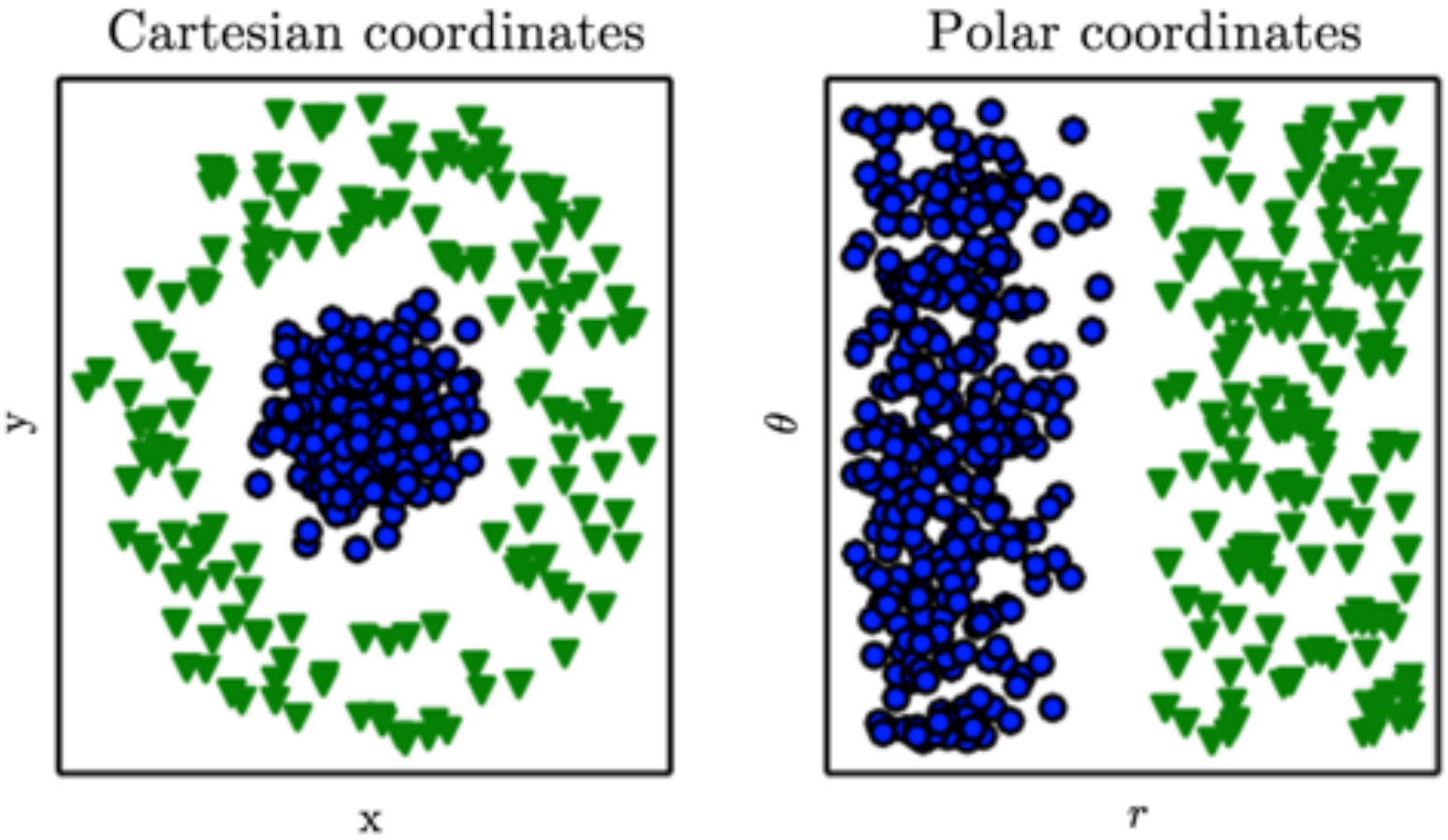
Feature Transformation

Dummy variables, Indicator features, Binarized features

New features often include transformations features. $\{X_i^2, \log(X_i), X_iX_j, X_i/X_j, |X_i-X_j|, \ldots\}$

These feature transformation are part of preprocessing, but can make a problem much easier (in regression you can include these terms in your model)

Example of feature transformation

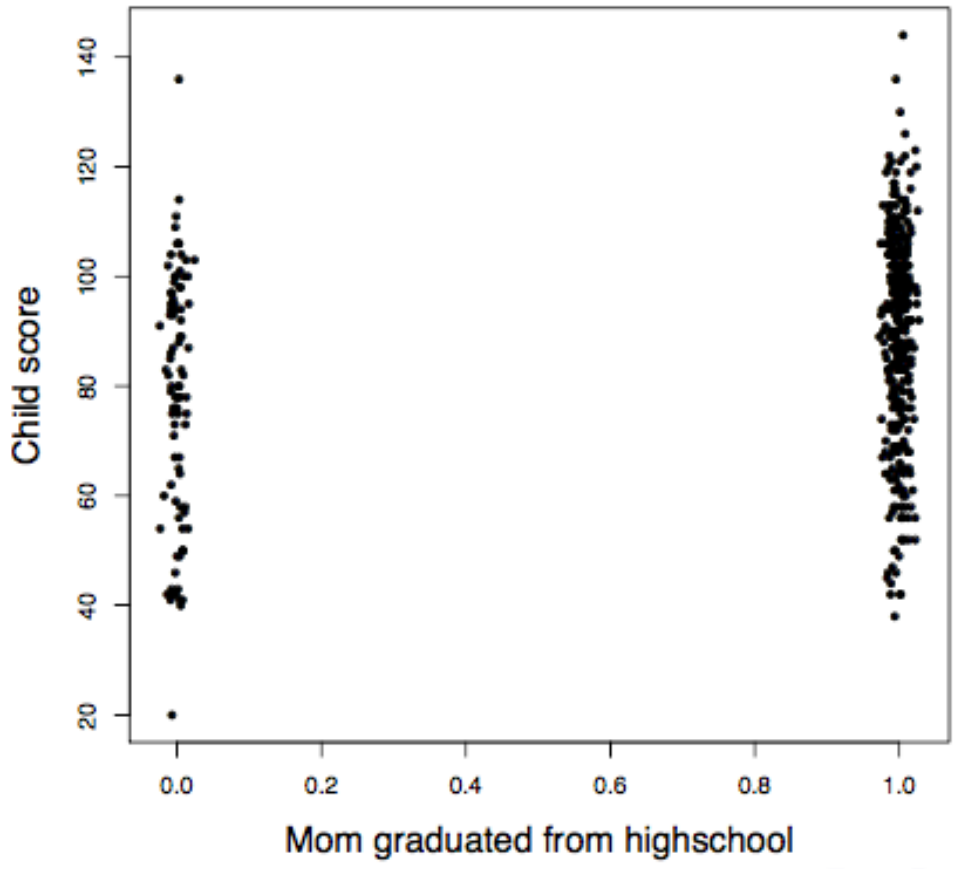


Overview: Classification

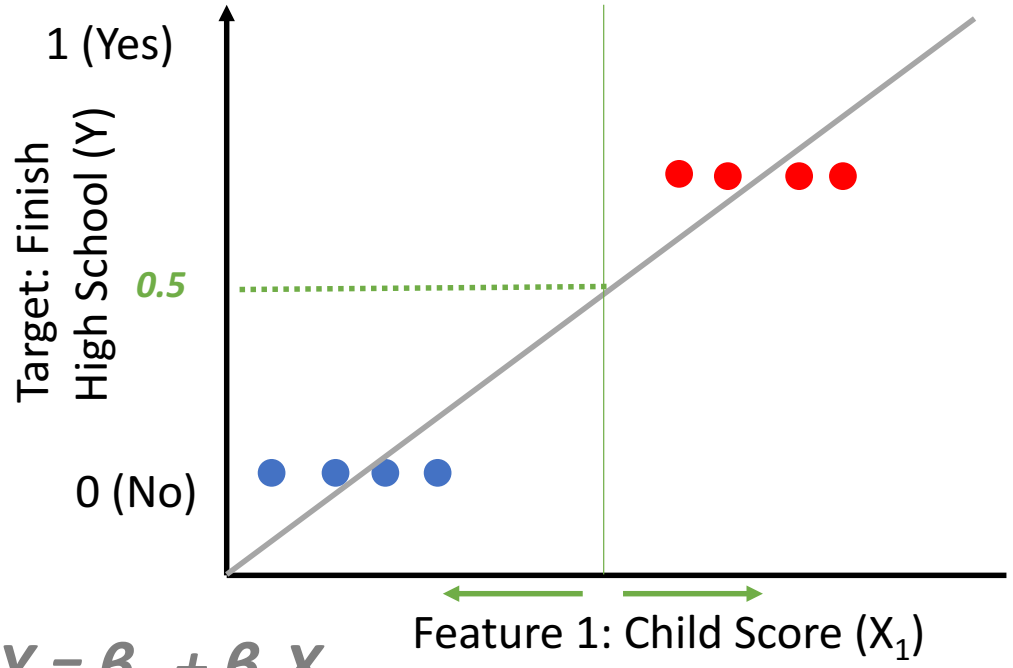
- Classification
- Decision Boundaries
- Learning by Example
- **Logistic Regression**
- K Nearest Neighbor (KNN)
- Decision Tree

Can we use regression for classification?

Binary Prediction



Can we use regression for classification?



$Y = \beta_0 + \beta_1 X$

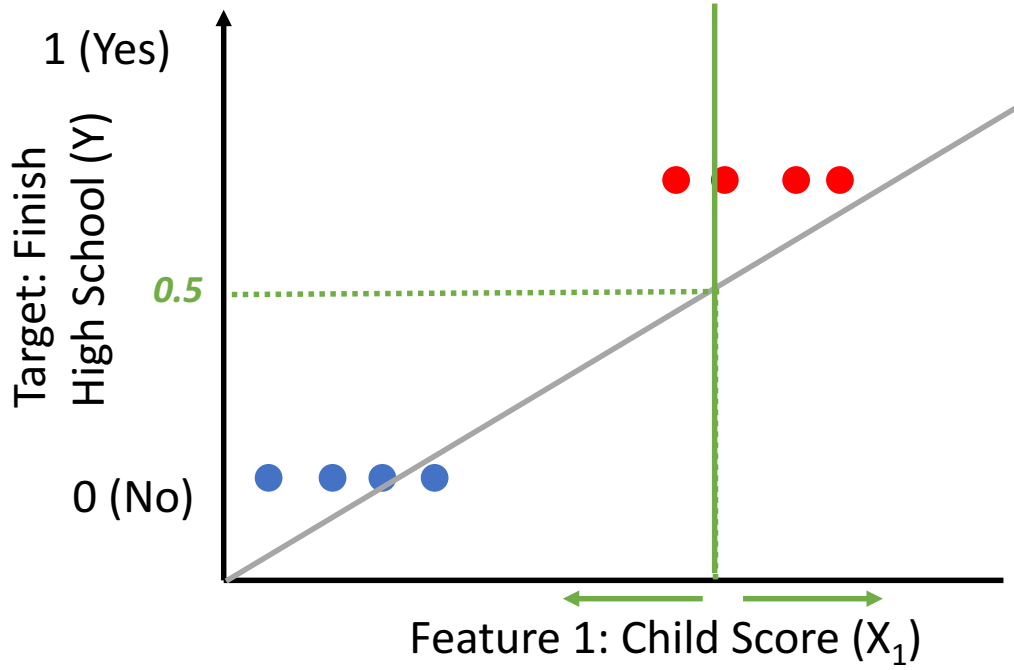
$$X = \frac{0.5 - \beta_0}{\beta_1}$$

Threshold the classifies f(X) at 0.5

If f(x) > 0.5 predict $\hat{y} = 1$

If f(x) $\leq$ 0.5 predict $\hat{y} = 0$

Can we use regression for classification?



$Y = \beta_0 + \beta_1 X$

$X = \frac{0.5 - \beta_0}{\beta_1}$

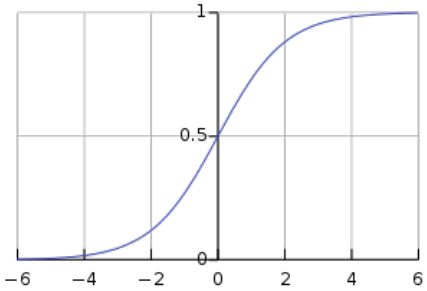
Threshold the classifies f(X) at 0.5

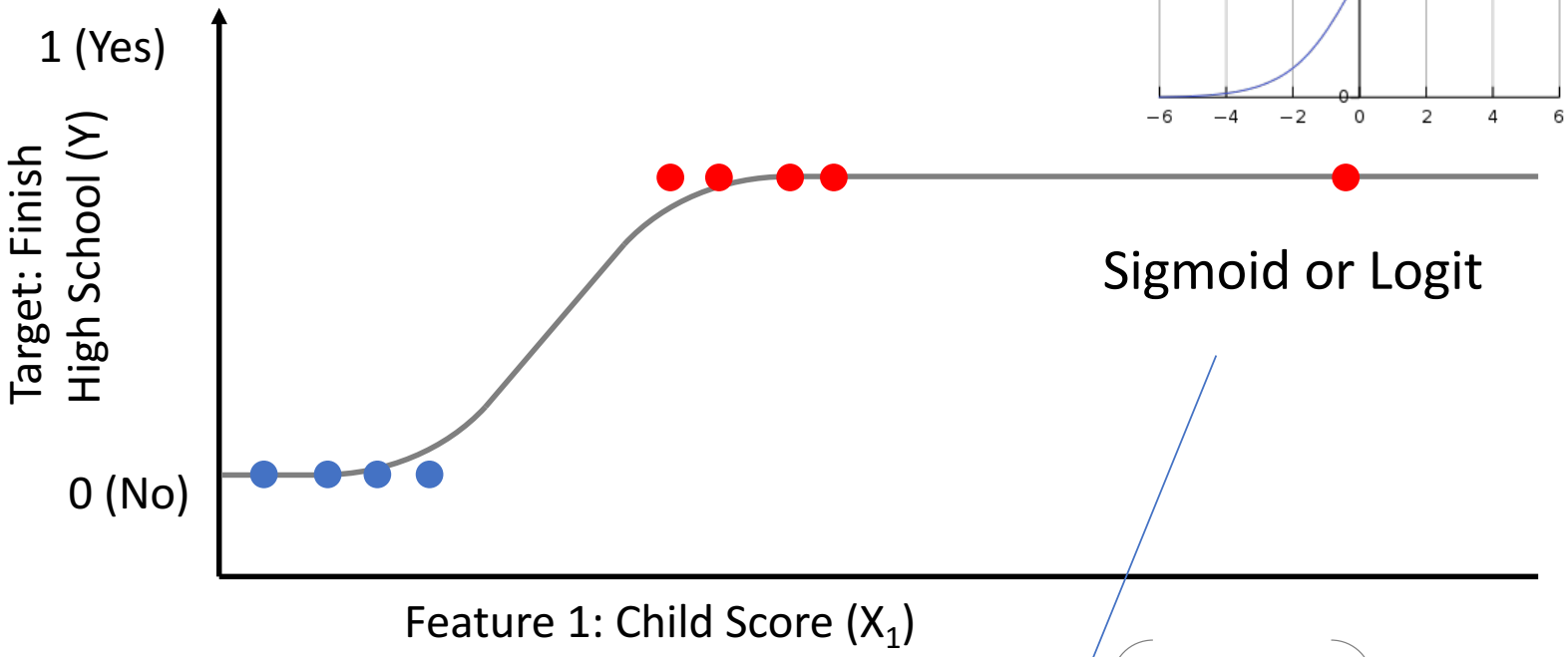
If f(x) > 0.5 predict $\hat{y} = 1$

If f(x) $\leq$ 0.5 predict $\hat{y} = 0$

Logistic Regression

Wants some f(X): $0 \leq \hat{Y} \leq 1$





$$Y = f(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1)}} \qquad g(f(X)) = \ln\left(\frac{f(X)}{1 - f(X)}\right) = \beta_0 + \beta_1 X_1$$

Logistic Regression

$$g(f(X)) = \ln\left(\frac{f(X)}{1 - f(X)}\right) = \beta_0 + \beta_1 X_1 = P(Y=1|X) = 1 - P(Y=0|X)$$

g(f(X)) = estimates the probability that Y = 1 on input X

Example: given child's IQ : P(Y=1IX) = 0.80 or P(Y=0|X) = 0.20)

→ *80% change that Mother completed High School*

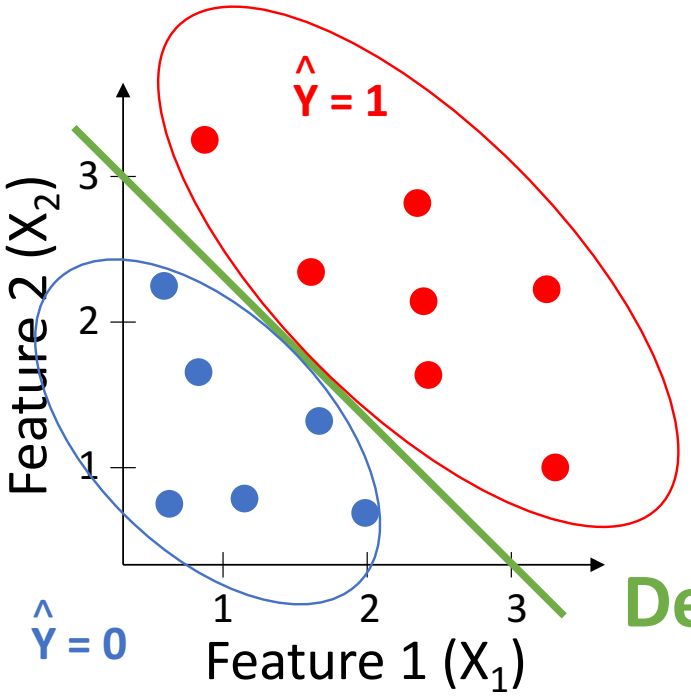
→ *20% change that Mother did not complete High School*

The β weights are often interpreted as odd ratio's:

Only valid in context of the model

$$P(\text{Mom high school} = 1 \mid IQ) = \frac{1}{1 + e^{-(21.2 + 0.197 * IQ)}} \qquad e^{0.197} = 1.02$$

Logistic Regression with two features



$$g(f(X)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

-3 1 1

Decision Boundary* $g(f(X)) = 0.5$

Predict "$\hat{Y} = 1$" if $-3 + X_1 + X_2 \geq 0$

$X_1 + X_2 \geq 3 \rightarrow X_1 + X_2 = 3$

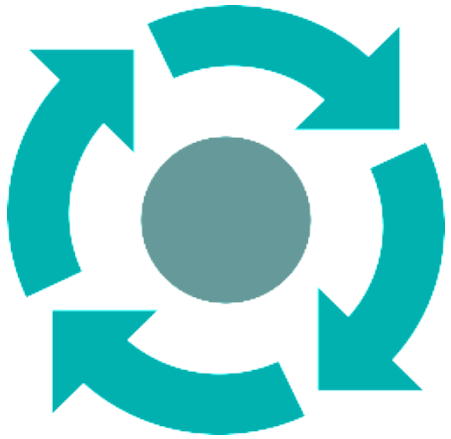
Logistic Regression: decision boundaries

The regression coefficients are usually estimated (using maximum likelihood estimation)

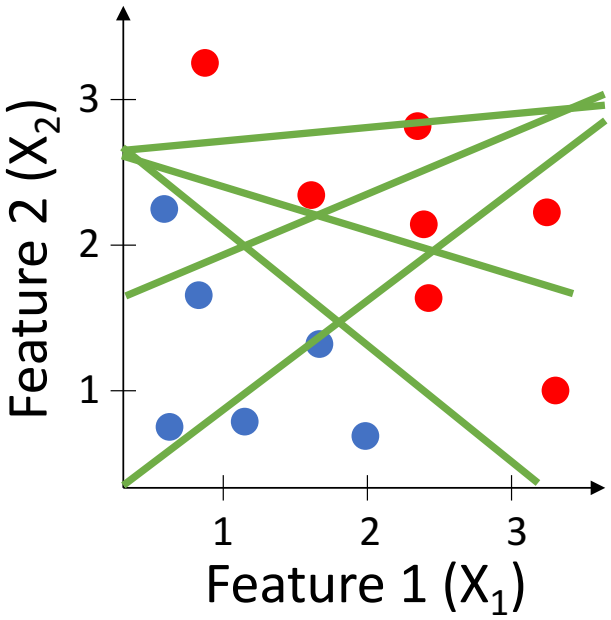
Like regression it is possible to use polynomial functions to fit a complex decision boundary ($y = a + bx + bx^2 + etc…$)

Unlike linear regression it is not possible to find a closed-form solution

Iterative process: begins with a tentative solution, revises it slightly to see if improves, and repeats this revision until no more improvement is made, at which point the process is said to have converged



Logistic Regression: fitting the boundary



Maximum Likelihood Estimation:
starts with arbitrary values of the regression coefficients and constructs an initial model for predicting the observed data.

Then evaluates errors in such prediction and changes the regression coefficients so as make the likelihood of the observed data greater under the new model.

Repeats until the model converges, meaning the differences between the newest model and the previous model are trivial.

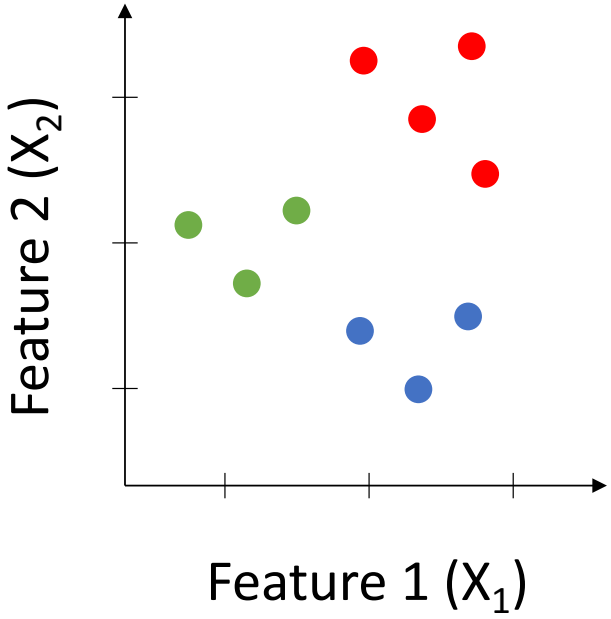
Multiclass Classification with Logistic Regression

Examples of classification tasks:

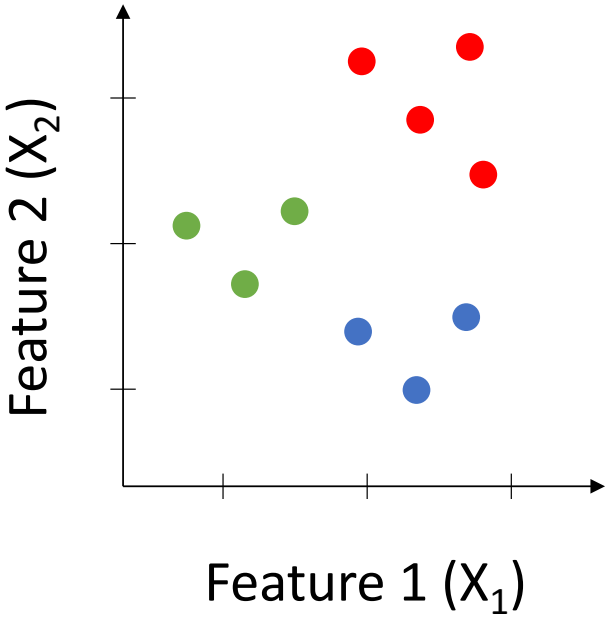
- Iris Species → {setosa, versicolor, virginica}
- Email folder tagging → {primary, social, promotions, updates}
- Weather → {sunny, cloudy, snow, rain}

Assignment to a class: $y \varepsilon \{1,2….\}$

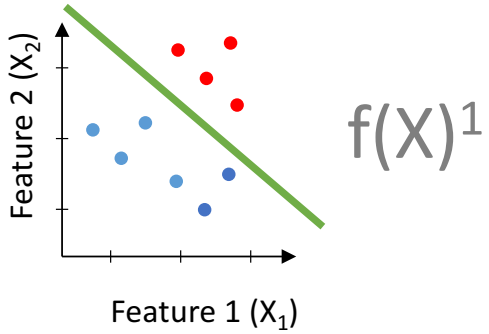
0 = can also start at 0

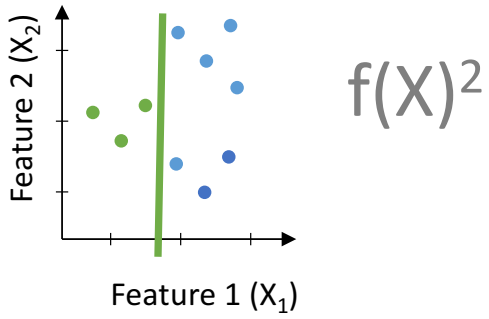


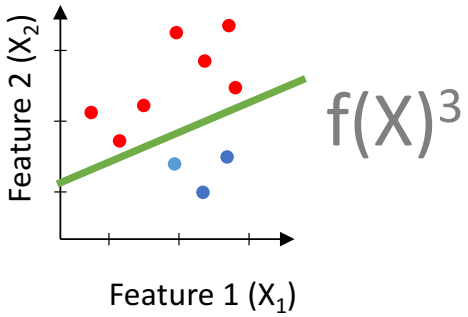
Multiclass Classification: one vs all



$f(X)^i = P(Y=i|X)$ $i = \{1,2,3\}$







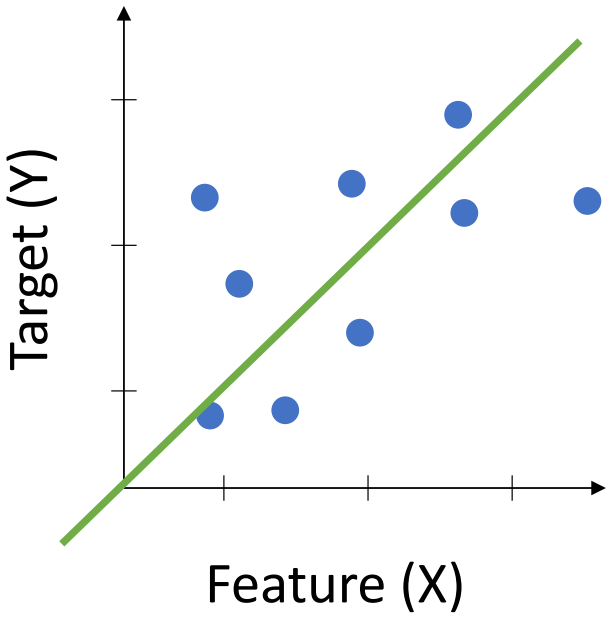
Multiclass Classification: one vs all

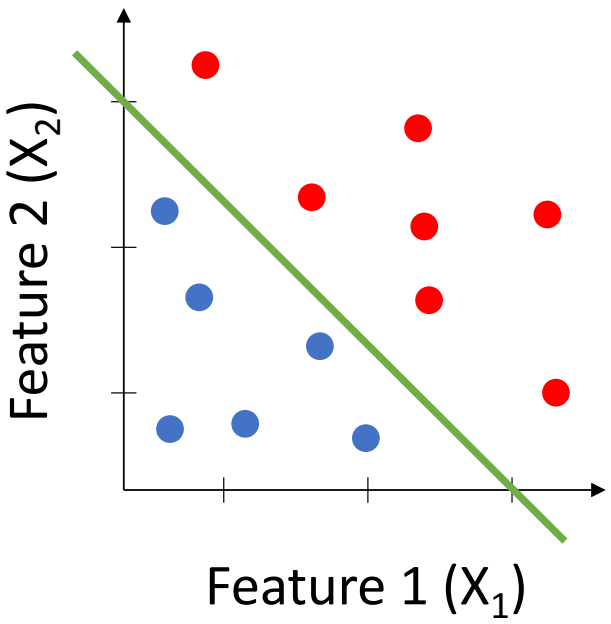
Train a logistic regression classifier $f(X)^i$ for each class *i* to predict the probability that $y = i$

On a new input X, to make a prediction, pick the class *i* that maximizes:

$$\max_i f(X)^i$$

Linear Regression vs Logistic Regression





Overview: Classification

- Classification
- Decision Boundaries
- Learning by Example
- Logistic Regression
- **K Nearest Neighbor (KNN)**
- Decision Tree

Simple idea: Similarity

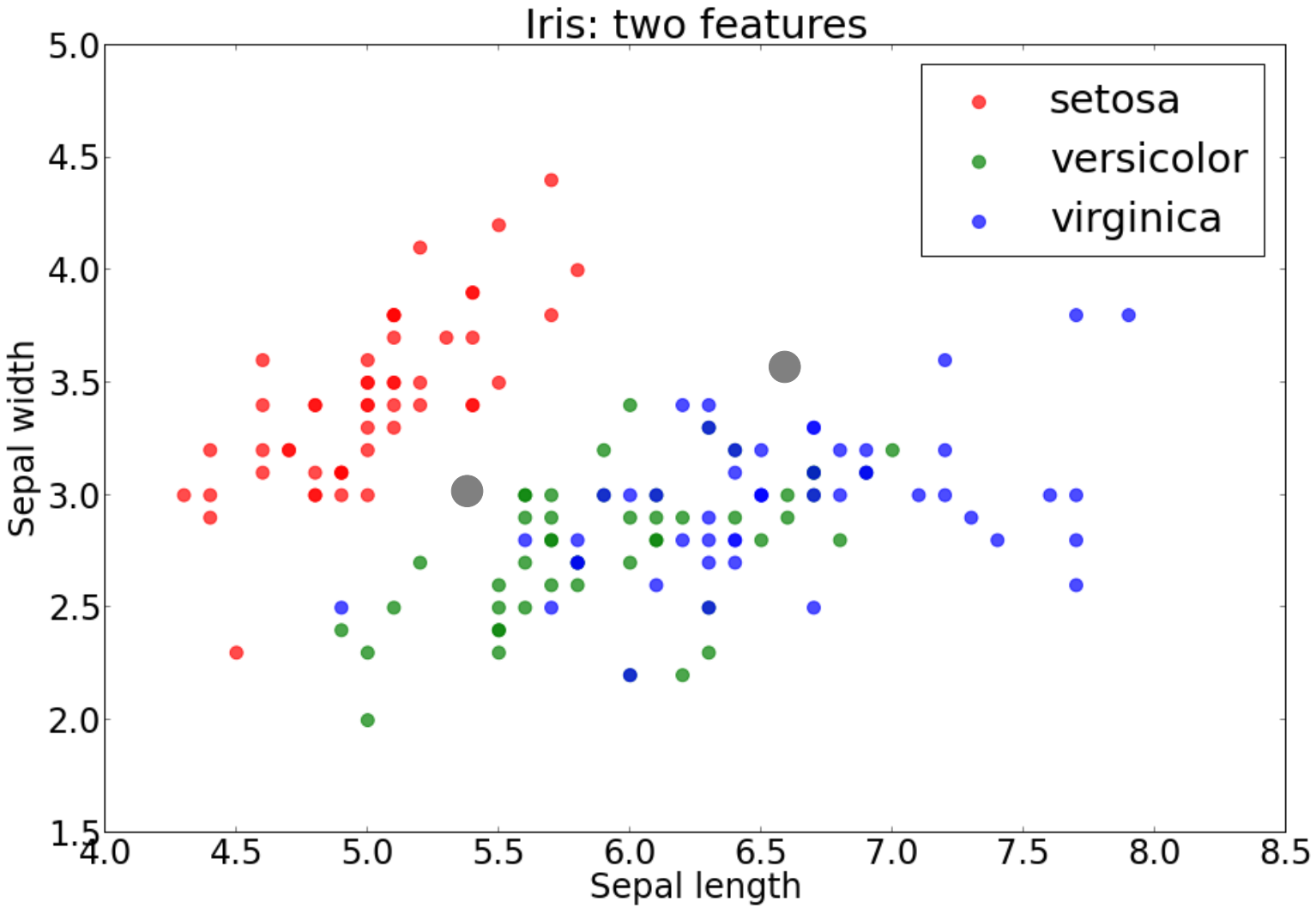
- Given a new example x_j
- We look for the most similar example in training set
- Predict the same target for x_j

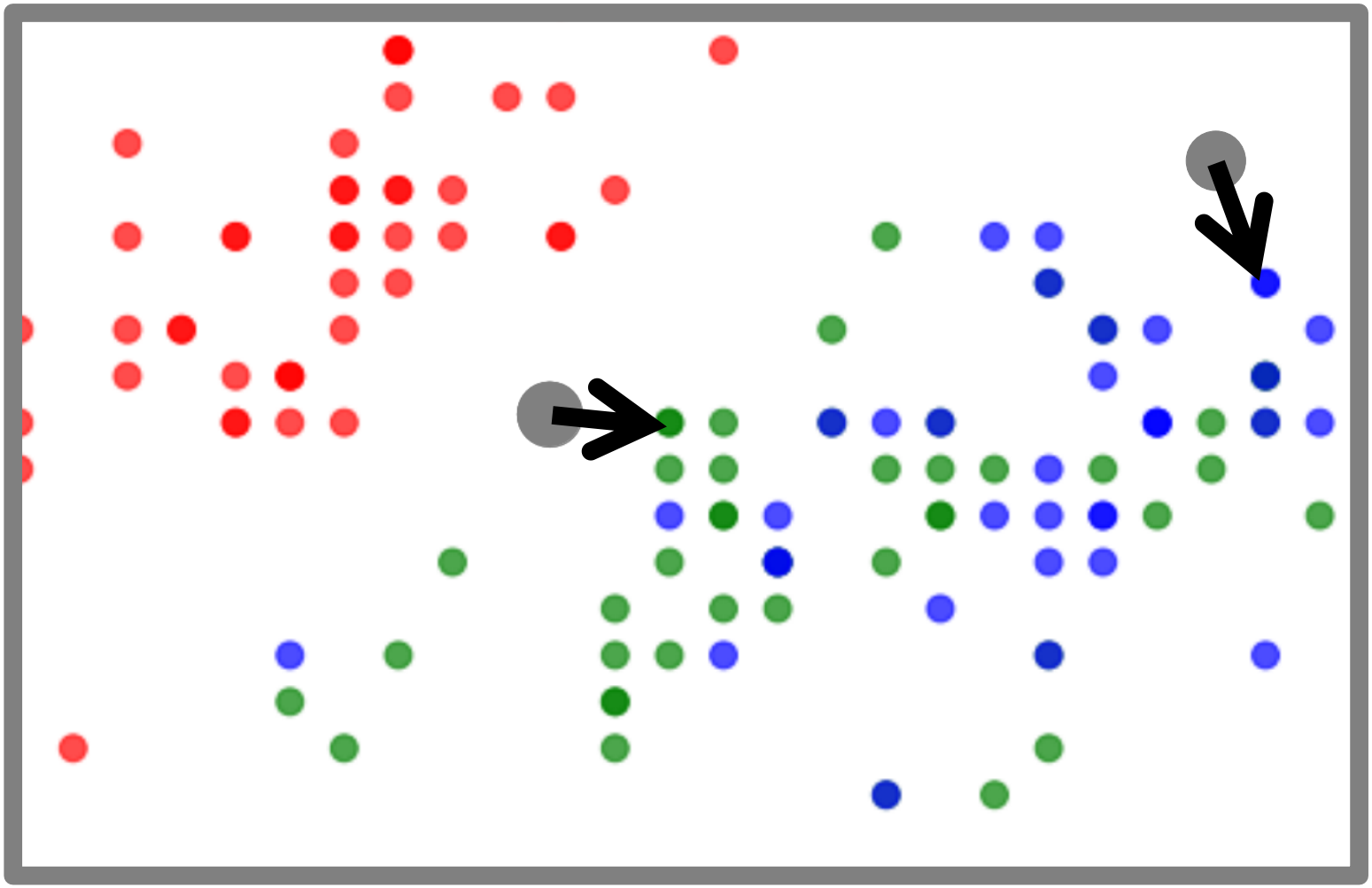
Iris setosa

Iris versicolor

Iris virginica

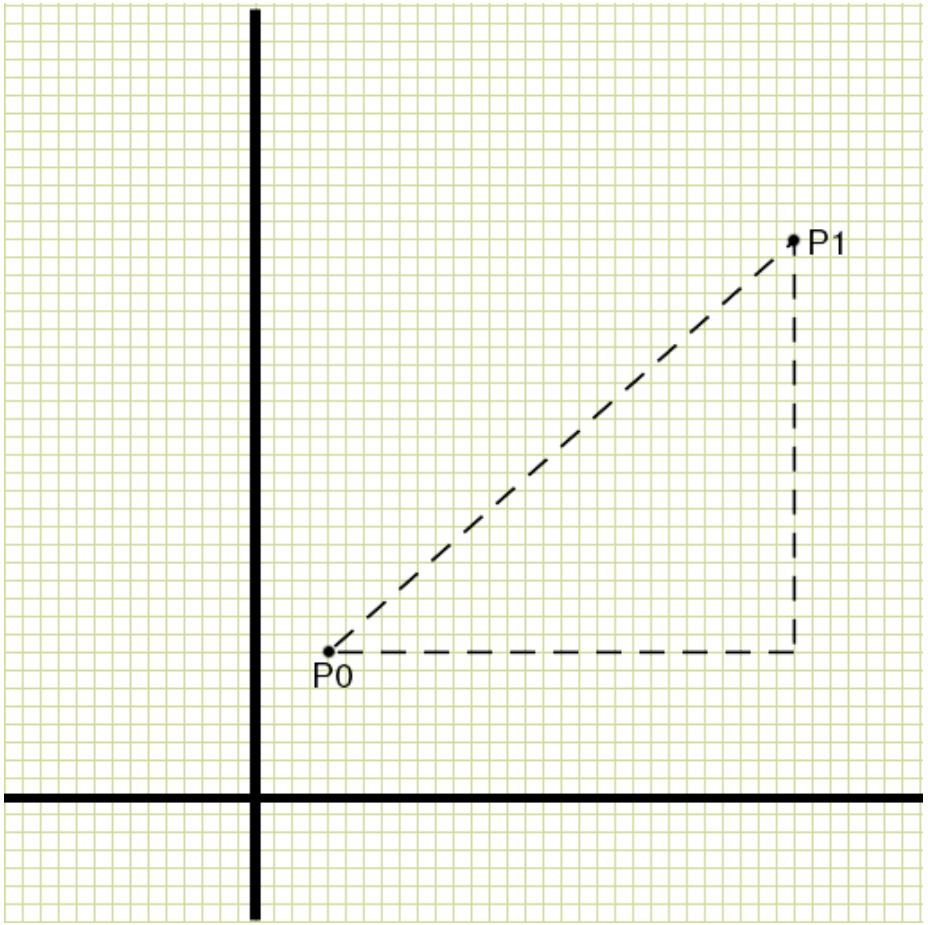
INPUT				OUTPUT
6.9	3.2	5.7	2.3	virginica
5.4	3.4	1.5	0.4	setosa
7.2	3.0	5.8	1.6	virginica
6.3	3.3	4.7	1.6	versicolor
5.8	2.7	3.9	1.2	versicolor
7.2	3.6	6.1	2.5	virginica
5.4	3.9	1.7	0.4	setosa


Iris: two features
setosa
versicolor
virginica
Sepal width
Sepal length
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5



How do we measure similarity?

- Distance – opposite of similarity?
- Find the nearest training example



Source: http://resumbrae.com/ub/dms423/05/triangleOnCartesian.png

Distance

Euclidean distance – like in physical space

In 2 dimensions:

$$D(\mathbf{u}, \mathbf{v}) = \sqrt{((u_1 - v_1)^2 + (u_2 - v_2)^2)}$$

In N dimensions:

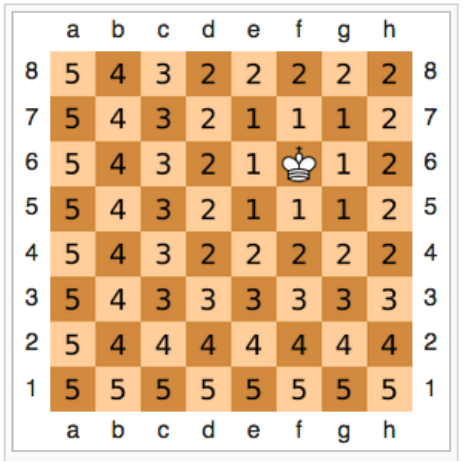
$$D(\mathbf{u}, \mathbf{v}) = \sqrt{\left(\sum_{i=1}^{N} (u_i - v_i)^2\right)}$$

Distance measure depends on the domain *(and thus feature vector)*

Hamming distance

- "karolin" and "kathrin" is 3.
- "karolin" and "kerstin" is 3.
- **1011101** and **1001001** is 2.
- **2173896** and **2233796** is 3.

Chebyshev distance



Finding the nearest neighbor

Check the distances to all the training points, and pick the point with the smallest distance

We need to remember the target of this point

Is KNN really Learning?

In what sense can K-NN be said be learning?

Illustrates most basic forms learning: **memorization**

No **abstraction**

K-Nearest Neighbors

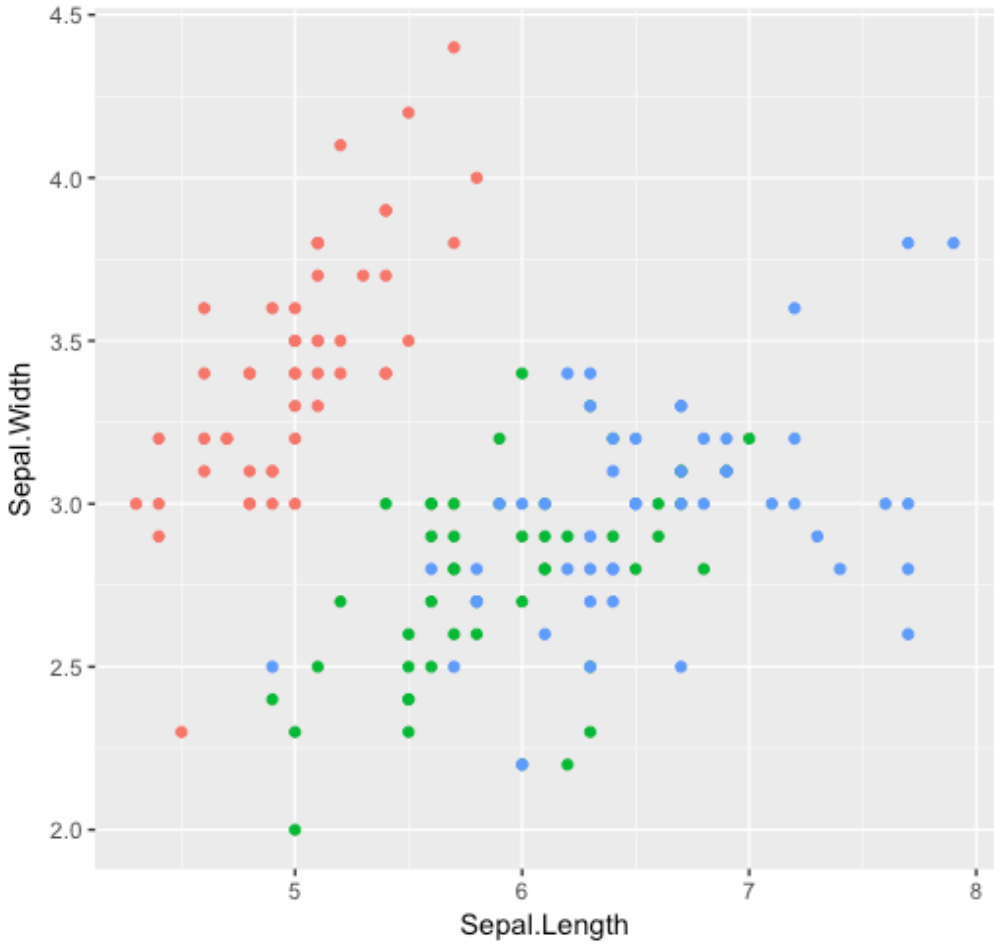
Instead of only the closest example, we can look at several

Predict the target which is most common among these

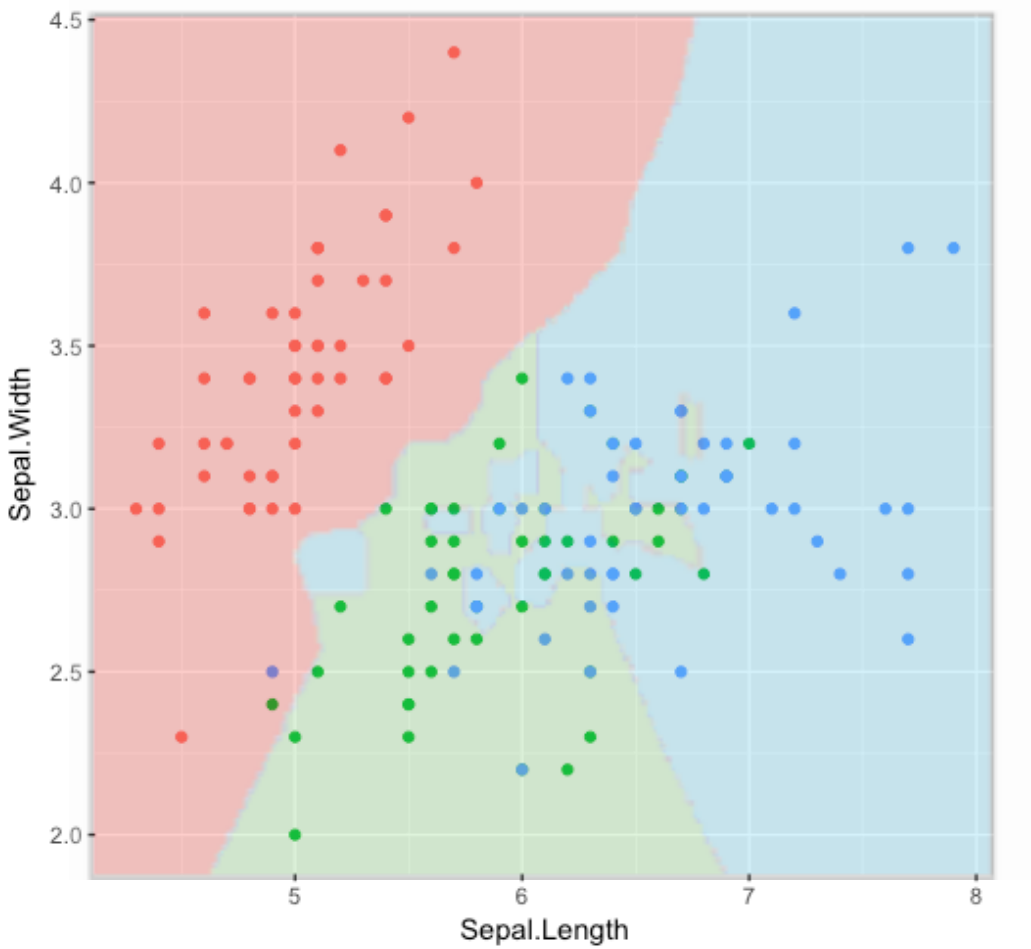
The role of K

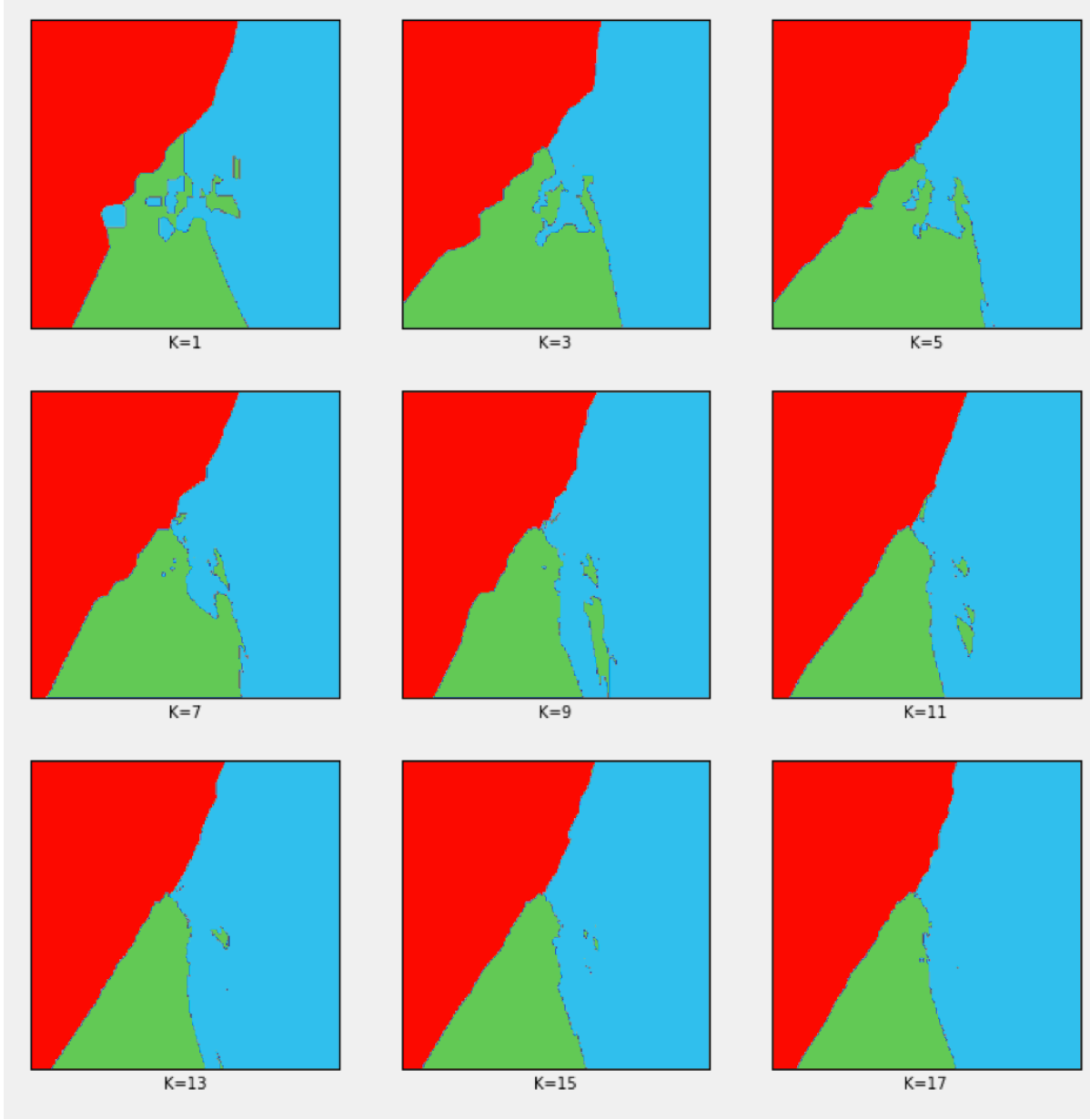
What effect does K have on the classification?

Iris: decision boundaries

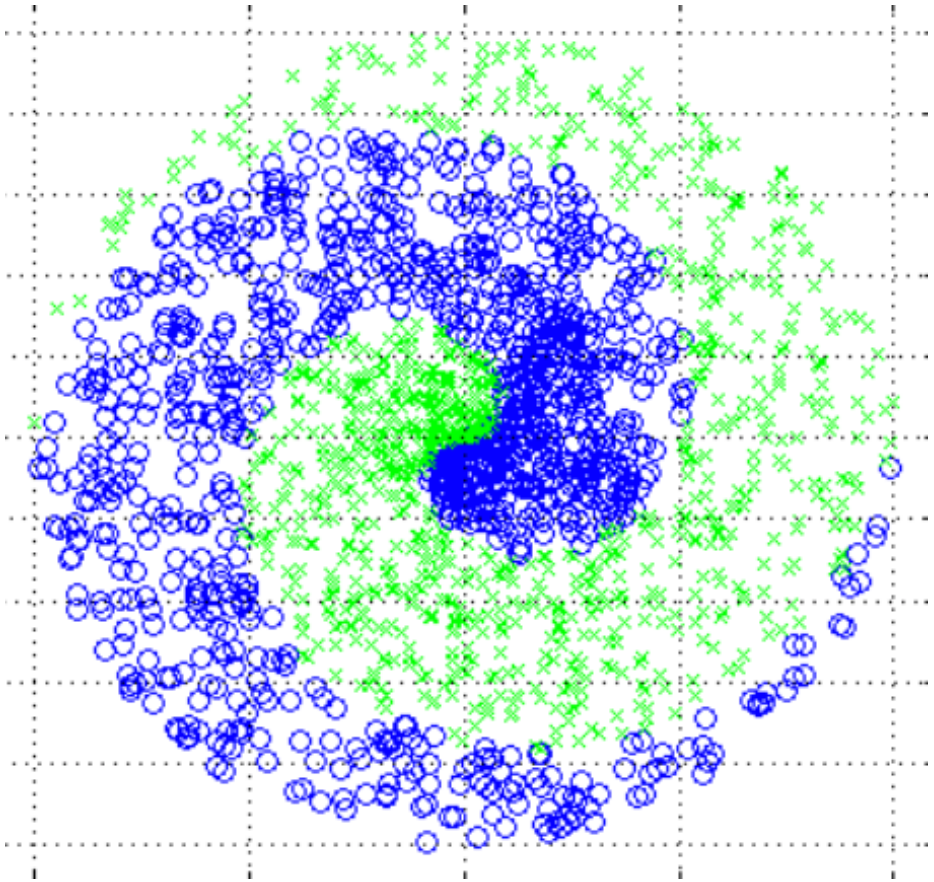


Iris: decision boundaries

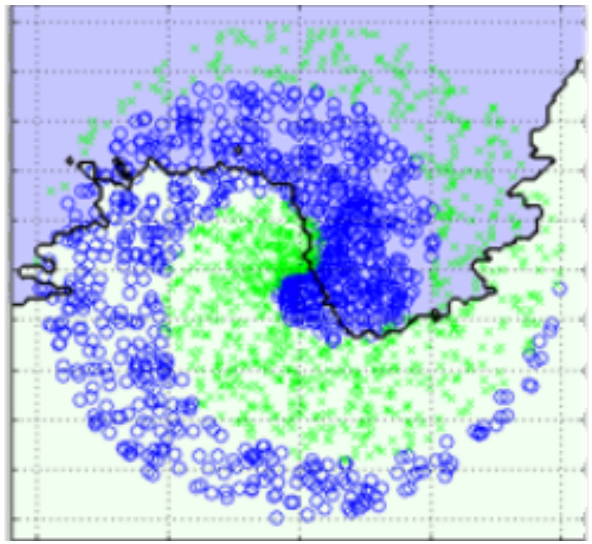



K=1
K=3
K=5
K=7
K=9
K=11
K=13
K=15
K=17

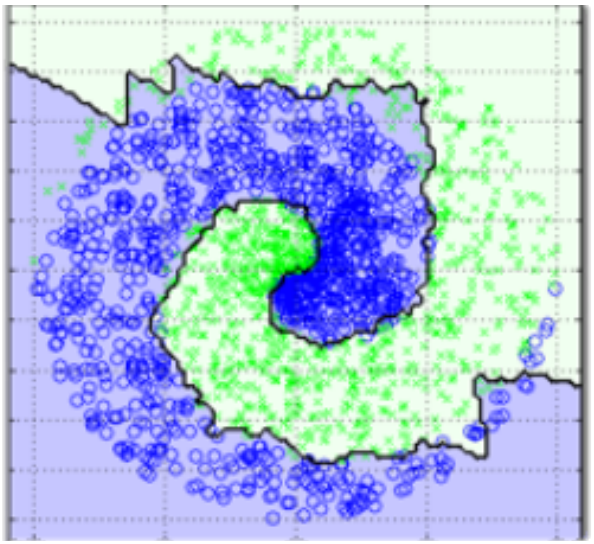
Does K matter?



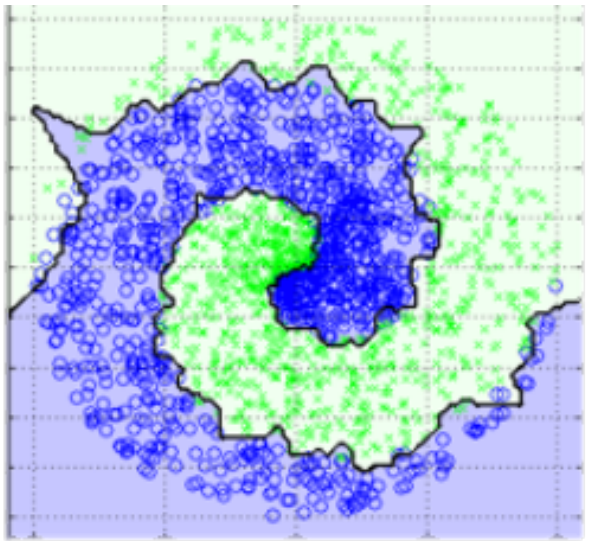
How to choose K?



K = 100



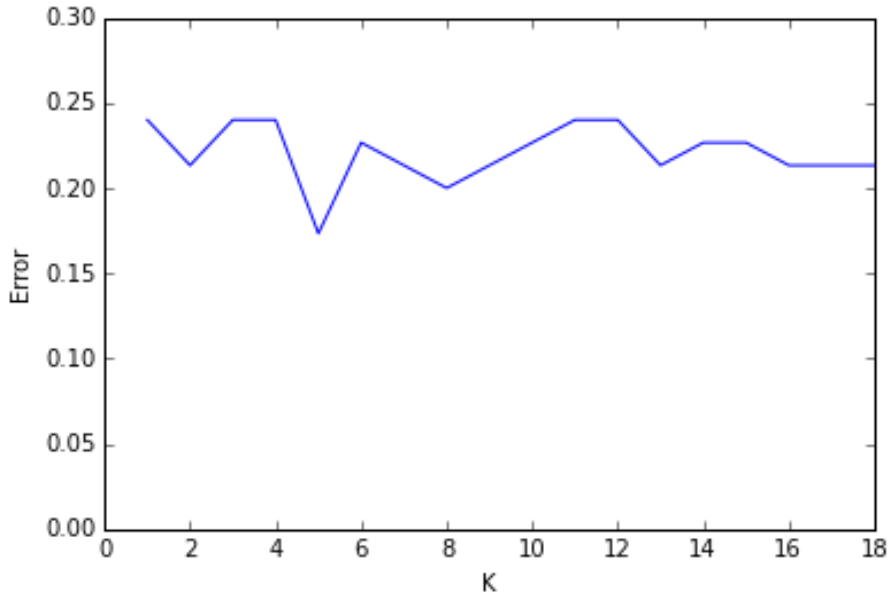
K = 10



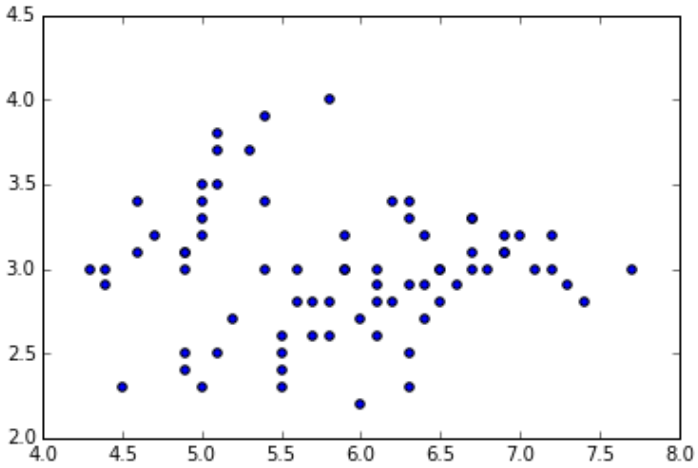
K = 1

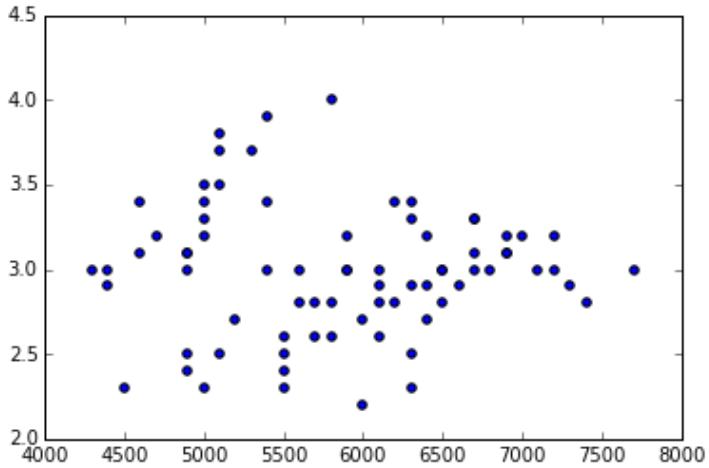
Decreasing Number of Neighbors →

Tuning on validation data

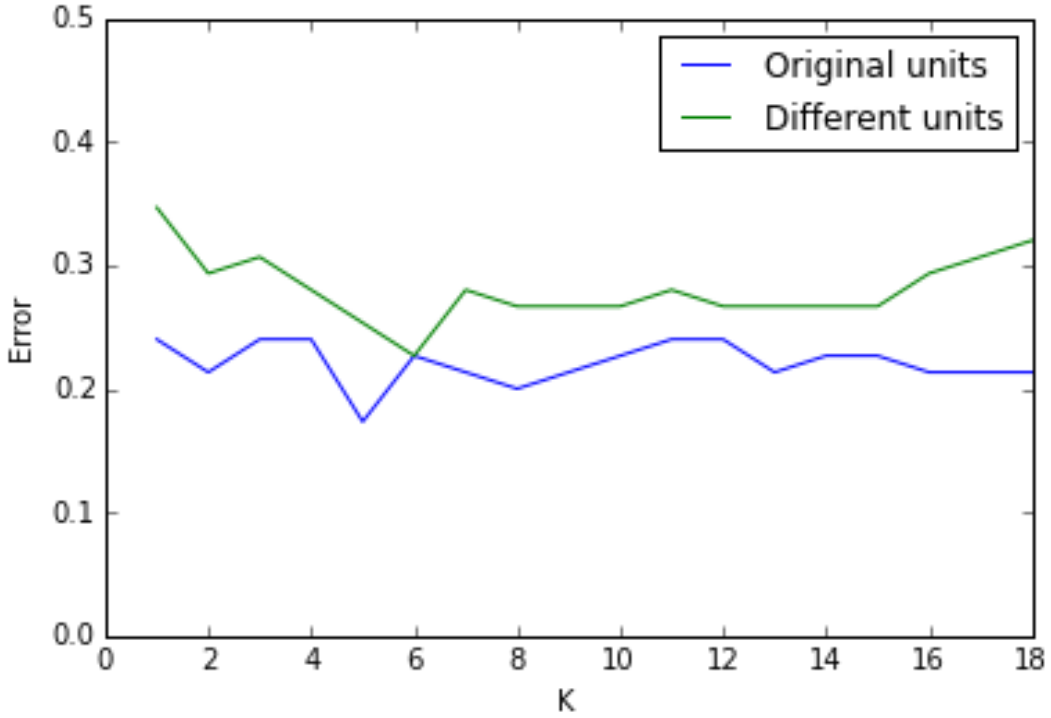


Units. Do they matter in KNN?



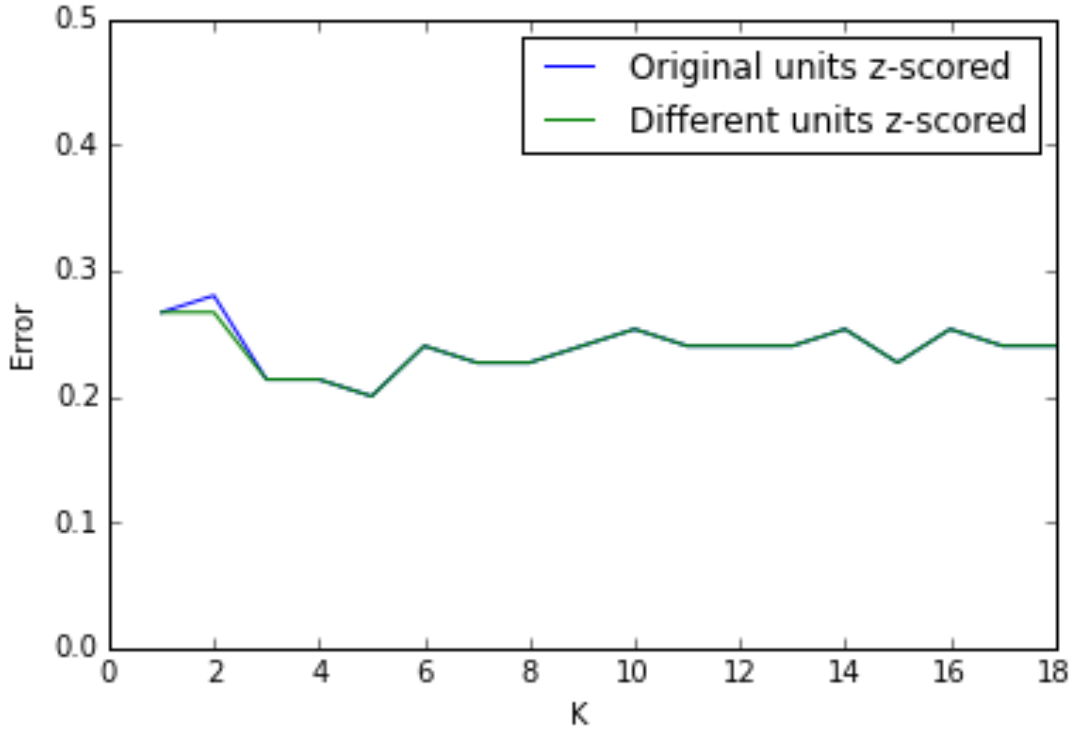


They do


Original units
Different units
Error
K
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
10
12
14
16
18

What's the solution?

Transform features: Standardize units → z-score dimensions



Problems

Some dimensions may be more informative about the class than others

Can we take this into account in the K-NN algorithm? How?

Use Example and validation set

Training set:

Observe patterns, infer rules

Development set (or validation set):

Monitor performance, choose best learning options

Test set:

REAL EXAM

Not accessible in advance

Summary KNN

In order to learn from examples we decompose complex objects into features

Often we need to convert categorical features into indicator features

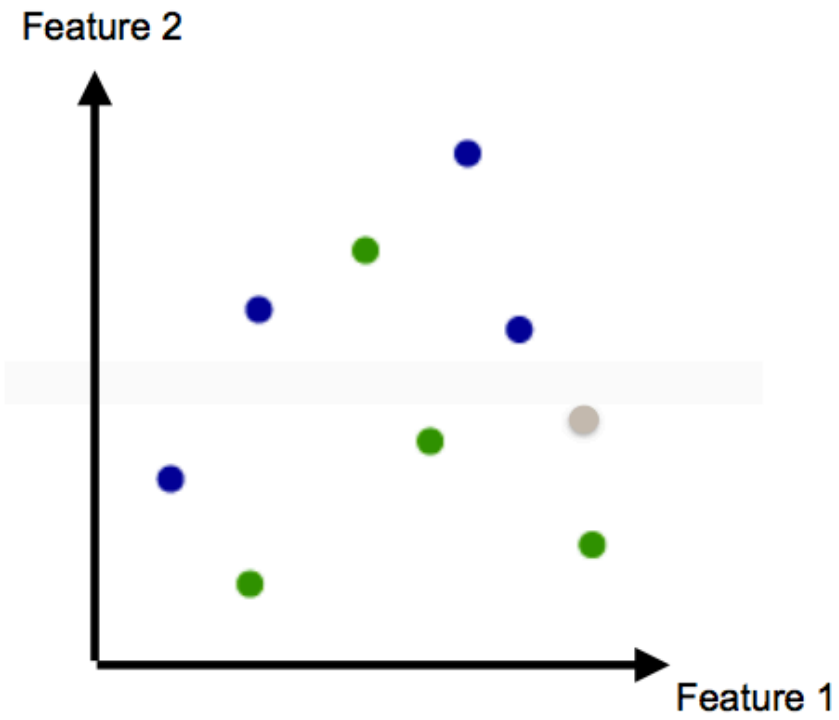
K-NN exemplified learning-as-memorization

Overview: Classification

- Classification
- Decision Boundaries
- Learning by Example
- Logistic Regression
- K Nearest Neighbor (KNN)
- **Decision Tree**

Decision tree: classification

A classification scheme which generates a tree and a set of rules from given data set



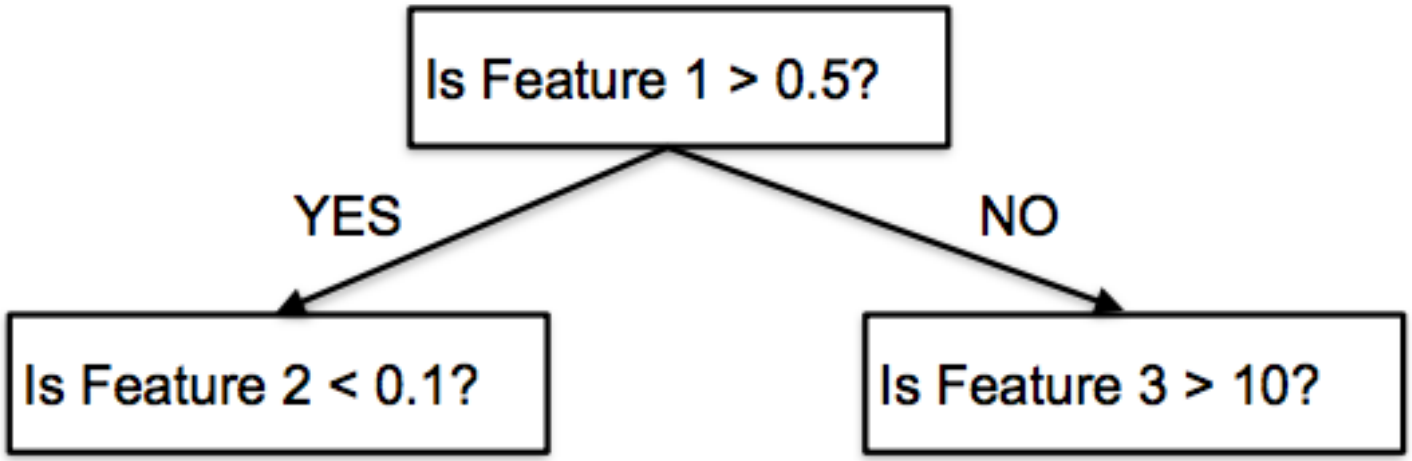
- Train instances
  - blue and green
- Test instance
  - gray
- Classifier induced from the data defines decision boundaries

Decision Trees

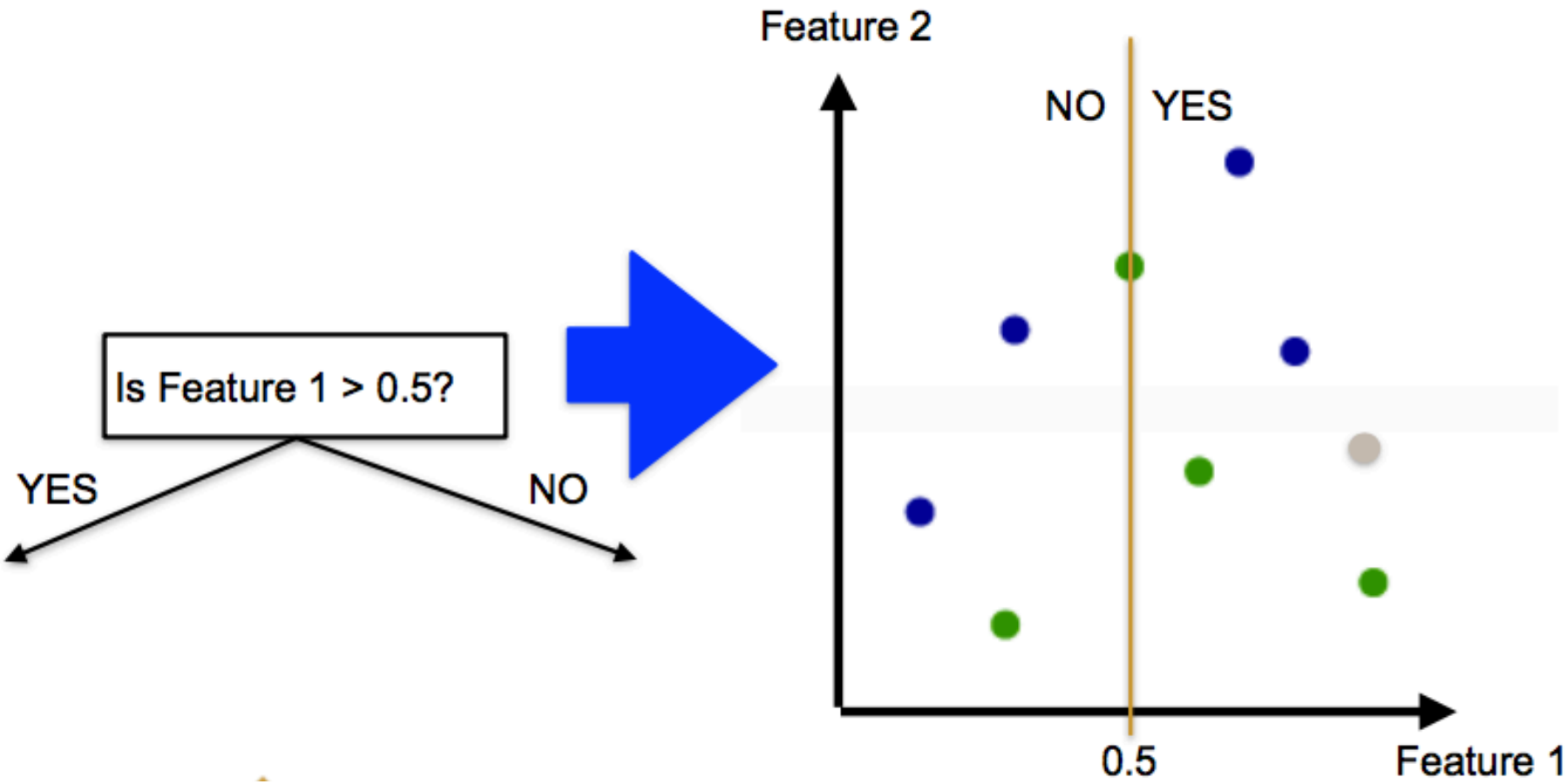
Decision Trees take one feature at a time and test a binary condition

For instance: is the feature (X_1) larger than 0.5?

- If the answer is YES, grow a node to the left
- If the answer is NOW grow a node to the right

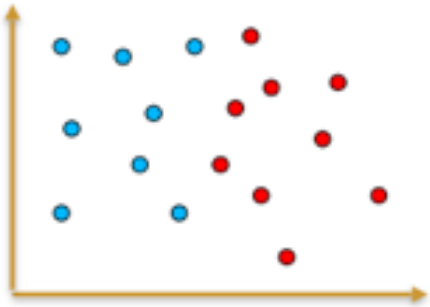


This results as a decision boundary

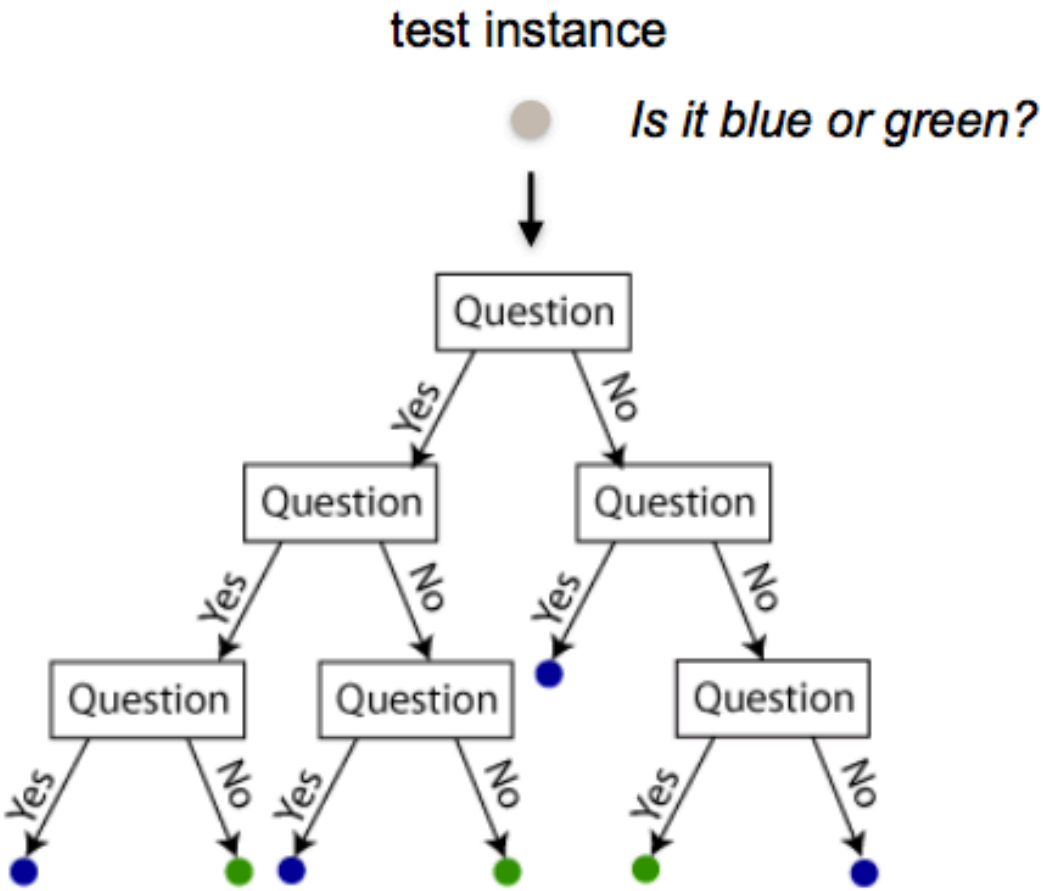


Decision Tree grows with each level of questions

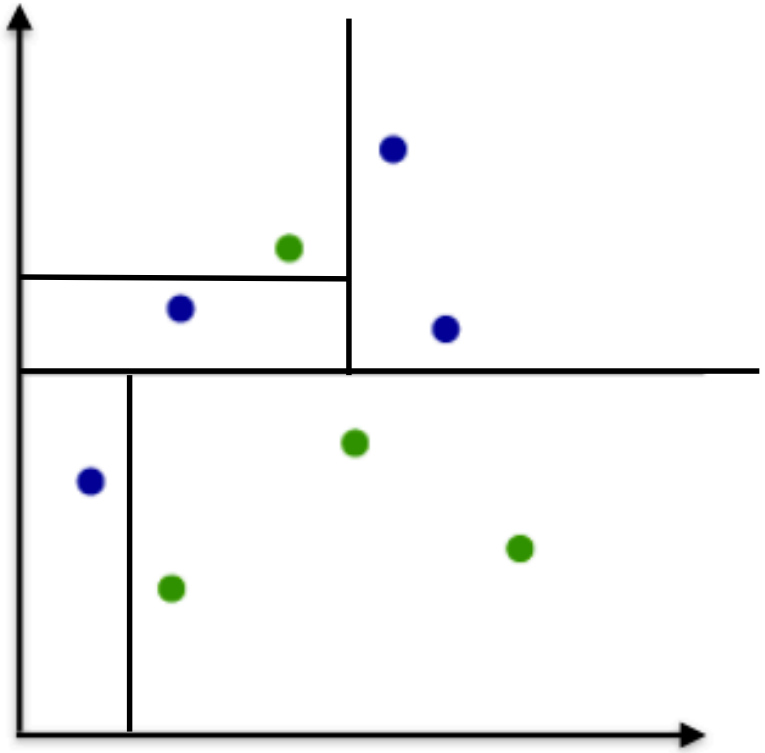
- Each node of the decision tree tests a condition on a feature (if else statement)
- The order of the nodes is important
- It is like playing “20 questions” or “Guess the person”: it is better to start with the question “Is he male?”, rather than with “Is it Chris?”
- The reason is that the answer to the first question maximizes the information (“entropy”) gained from the answer
- The order of features to be tested is determined by means of information theory



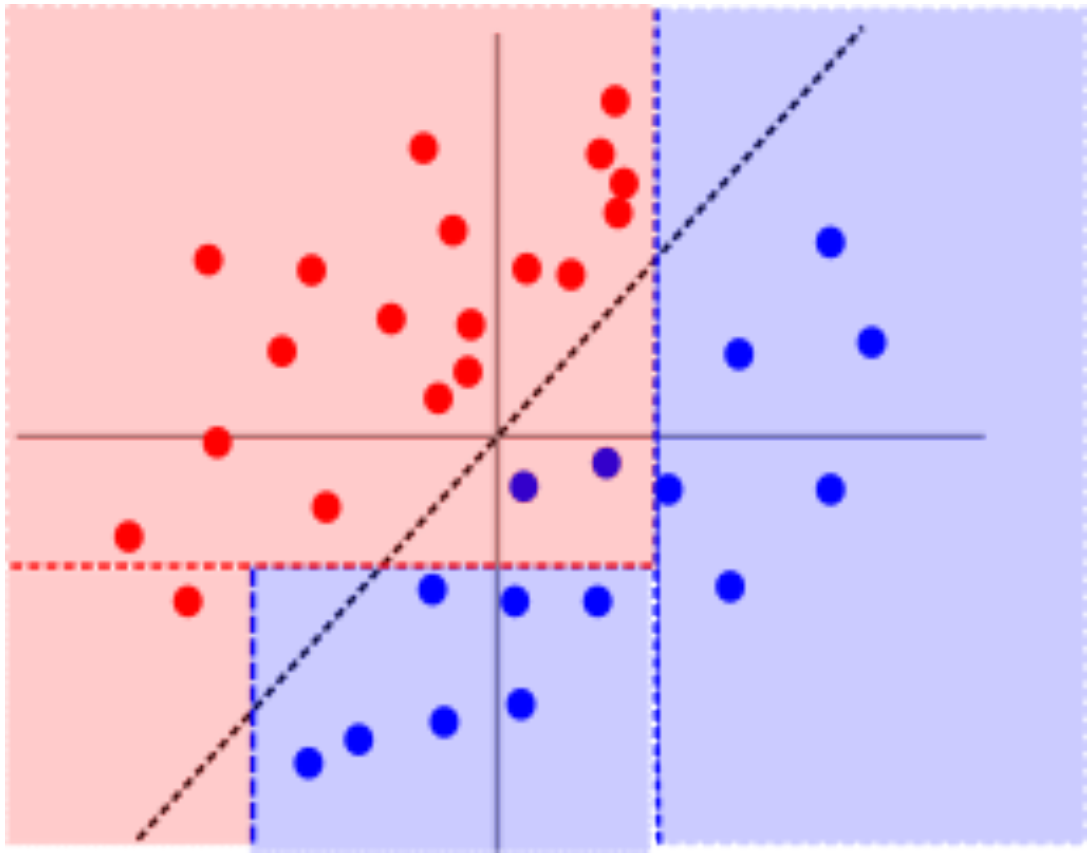
Decision Tree



Each test (box) adds a decision boundary

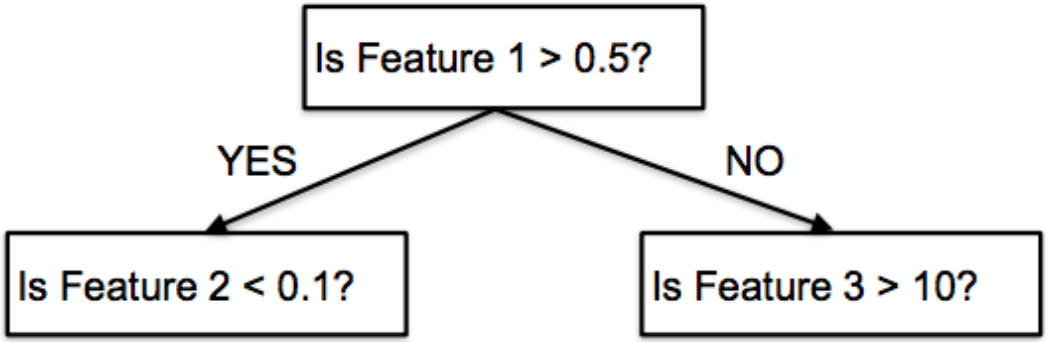


Decision boundaries: Logistic Regression vs Decision Tree



Complexity of the induced model

- The complexity of the model induced by a decision tree is determined by the depth of the tree
- Increasing the depth of the tree increases the number of decision boundaries
- All decision boundaries are perpendicular to the feature axes, because at each node a decision is made about a single feature



Some advantages of decision trees

- Are simple to understand and interpret
- Can work with relatively little data
- Help find which feature is most important for classification

  description (understanding) vs. prediction

Summary

- Classification → {0,1,2..}
- Decision Boundaries → represents a model
- Learning by Example → features/generalization
- Logistic Regression → sigmoid, find boundaries
- K Nearest Neighbor (KNN) → find examples
- Decision Tree → rule base