

Introduction to Data Science 6

Overview

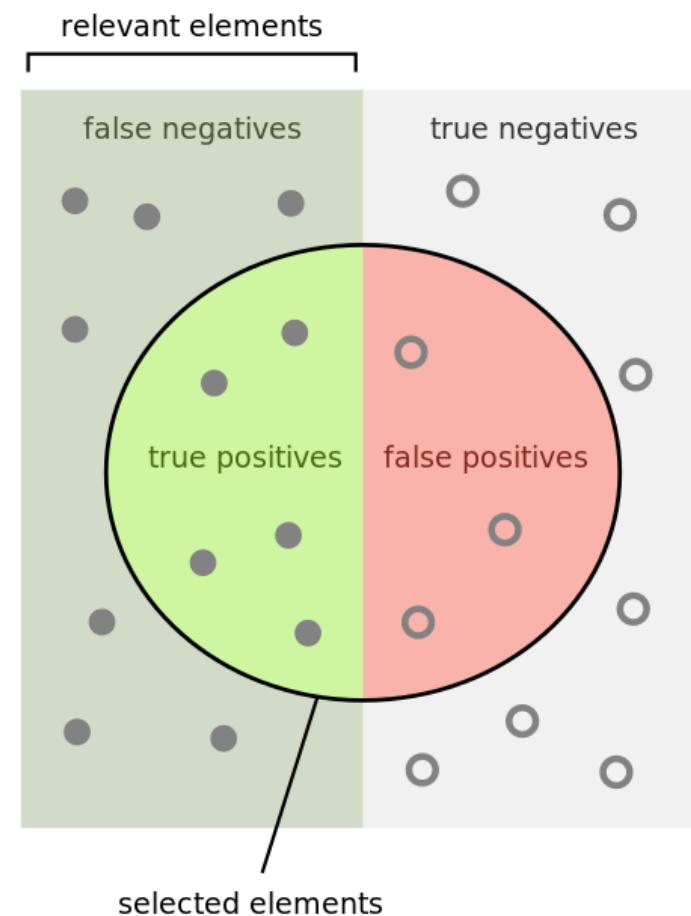
Precision & Recall (reprise)

PCA is color blind

3 new classifiers

- Random Decision Forests
- Naive Bayes
- Support Vector Machines

- ***precision*** is the fraction of retrieved instances that are relevant
- ***recall*** is the fraction of relevant instances that are retrieved



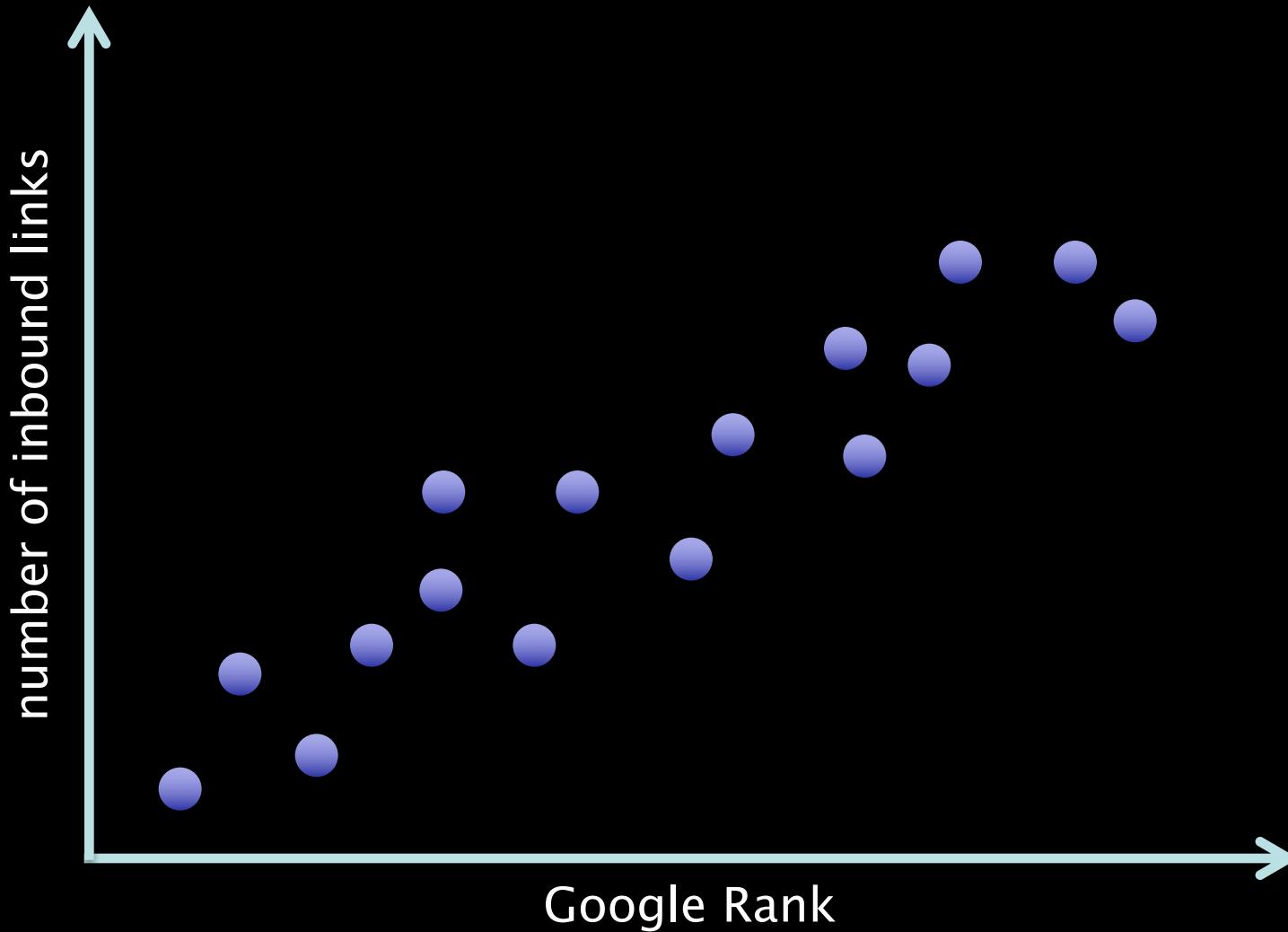
How many selected items are relevant?

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

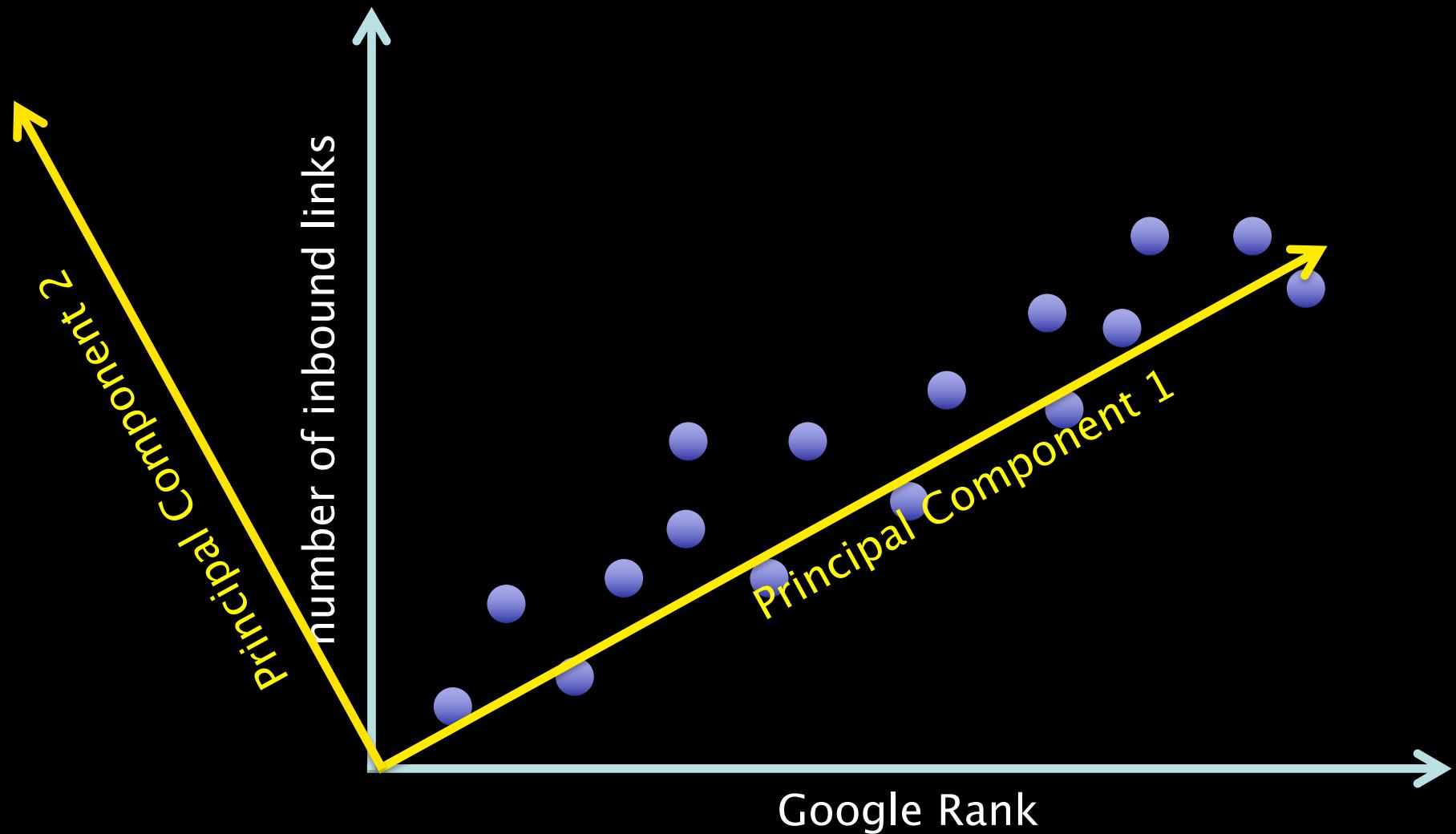
How many relevant items are selected?

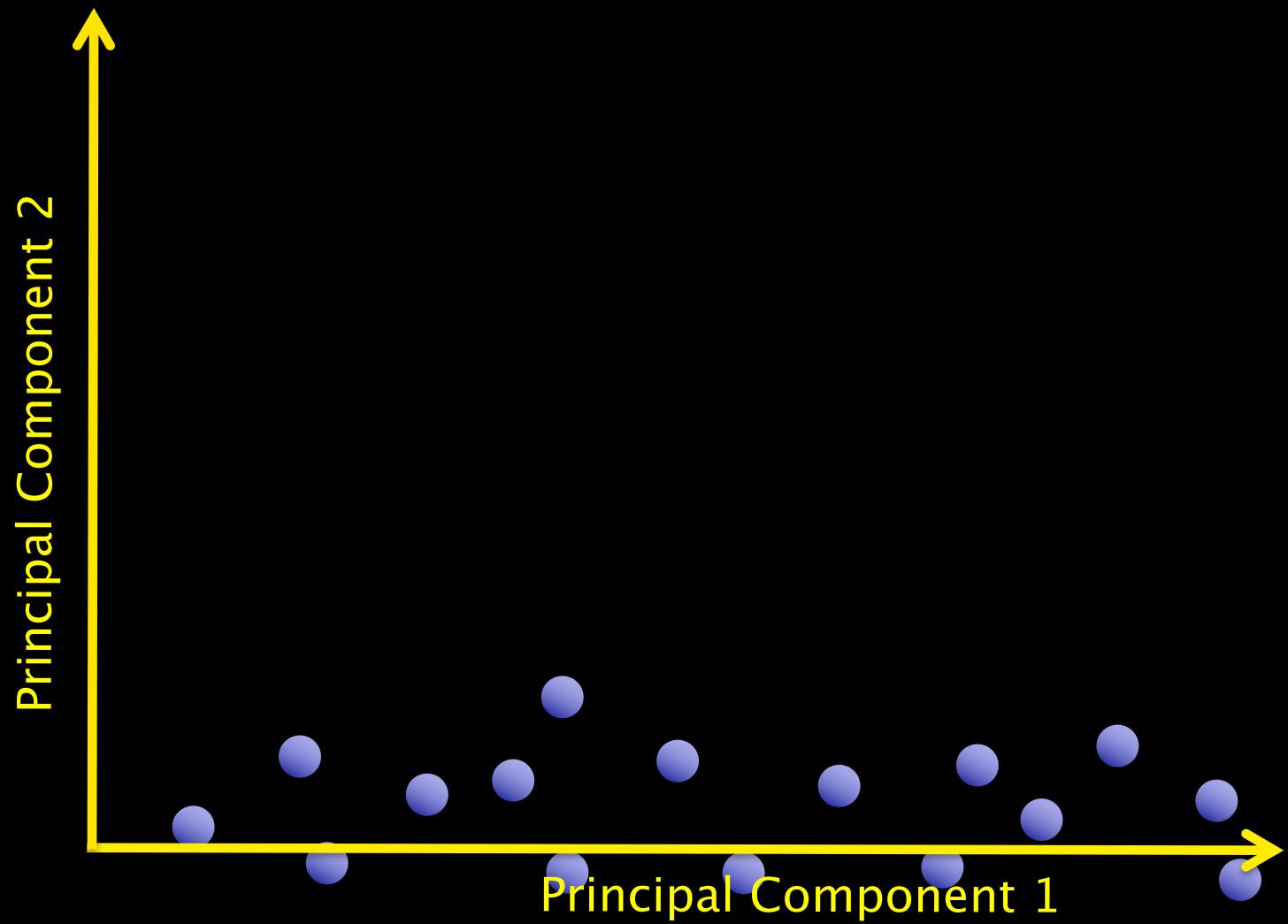
$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

1D structure in 2D feature space



PCA





PCA

Takes 2 features as input

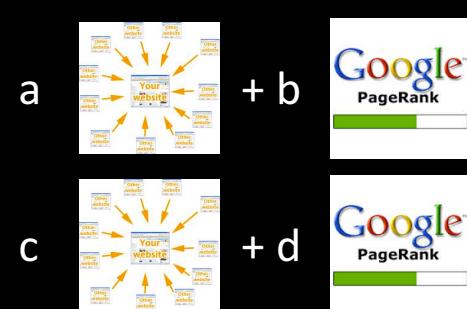
Feature 1: Number of inbound links

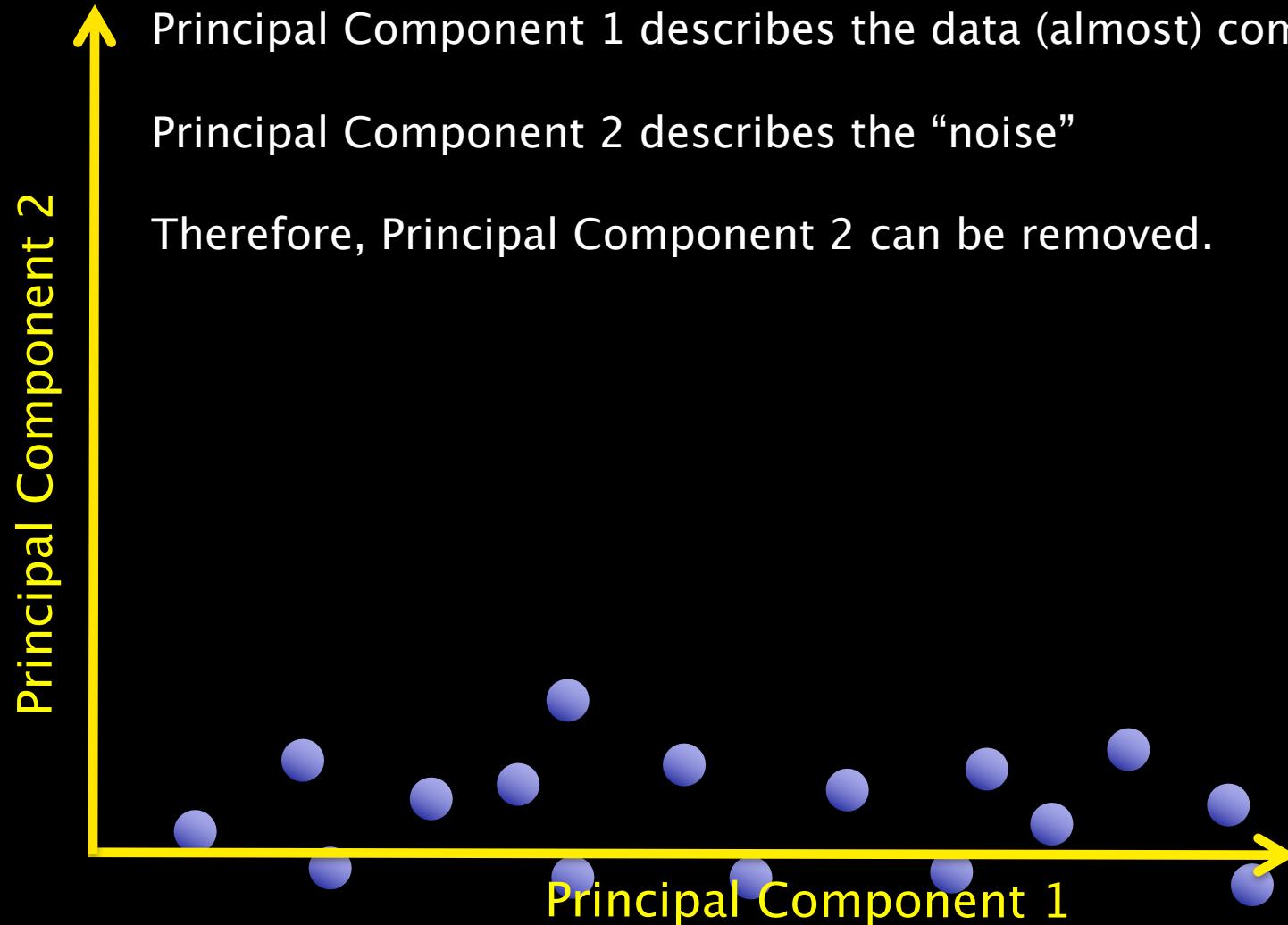
Feature 2: Google Rank

... and gives 2 new features as output

Principal Component 1

Principal Component 2





Principal Component 1 describes the data (almost) completely
Principal Component 2 describes the “noise”
Therefore, Principal Component 2 can be removed.

PCA + discarding highest PCs

Takes 2 features as input

Feature 1: Number of inbound links

Feature 2: Google Rank

... and gives 1 new feature as output

Principal Component 1



a

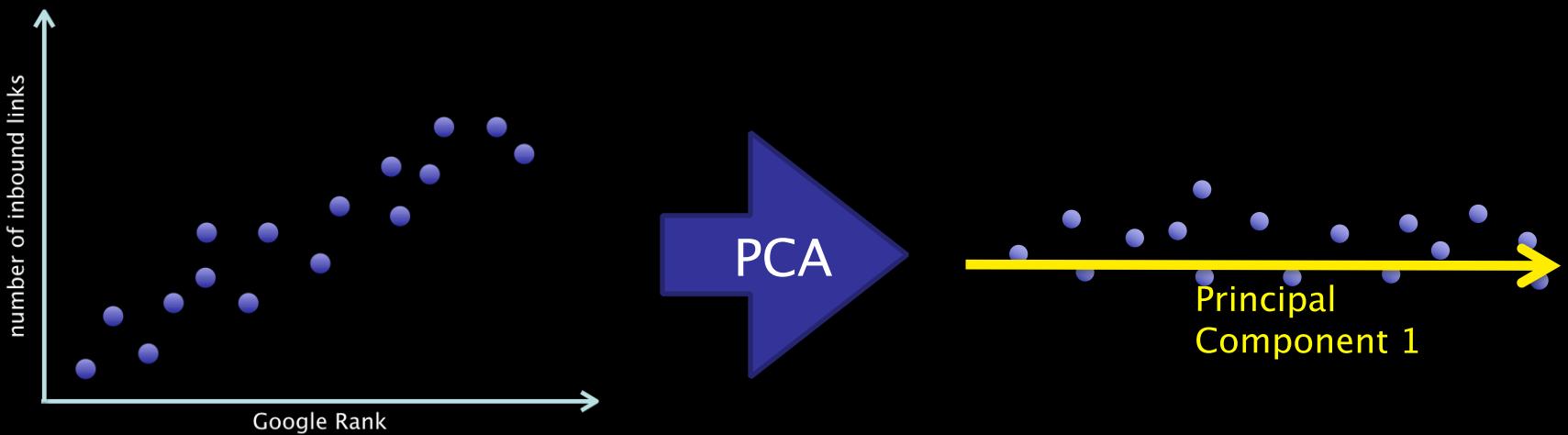


+ b



Dimensionality Reduction

- PCA performs **dimensionality reduction**
- It reduces 2 dimensions (features) to 1



PCA in higher dimensions

For datasets with N features ($N > 2$), PCA rotates the coordinate system in such a way that:

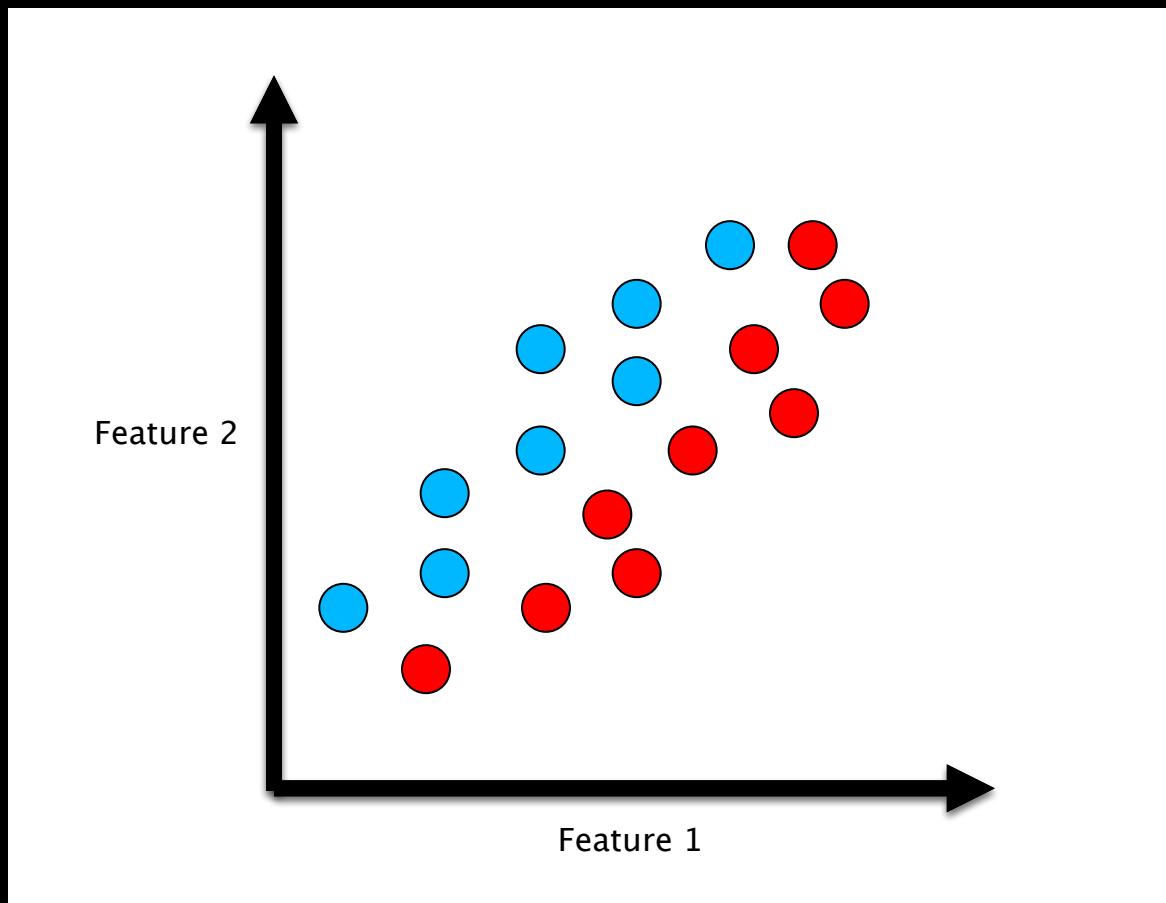
- the projection of the data on the first principal component (new axis) has the largest variance,
 - the projection of the data on the second principal component (new axis) has the one-but-largest variance,
- and so forth... (up to N principal components)

If the variation in the data is associated with relevance for classification (or regression), the most relevant features are captured by the first M principal components (and the rest captures noise)

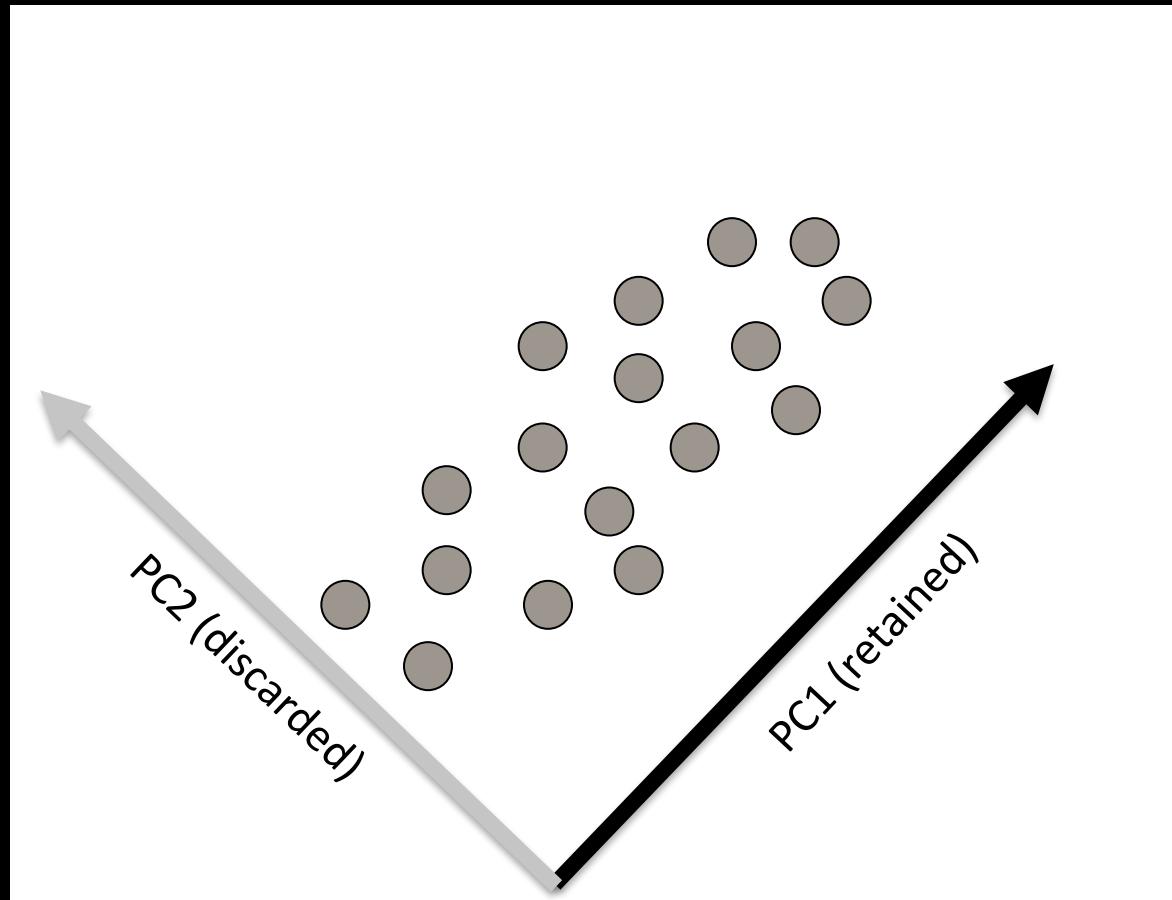
Retaining the first M principal components and throwing away the rest effectively reduces the dimensionality

M is typically much smaller than N , hence “dimensionality reduction”

PCA is color blind!

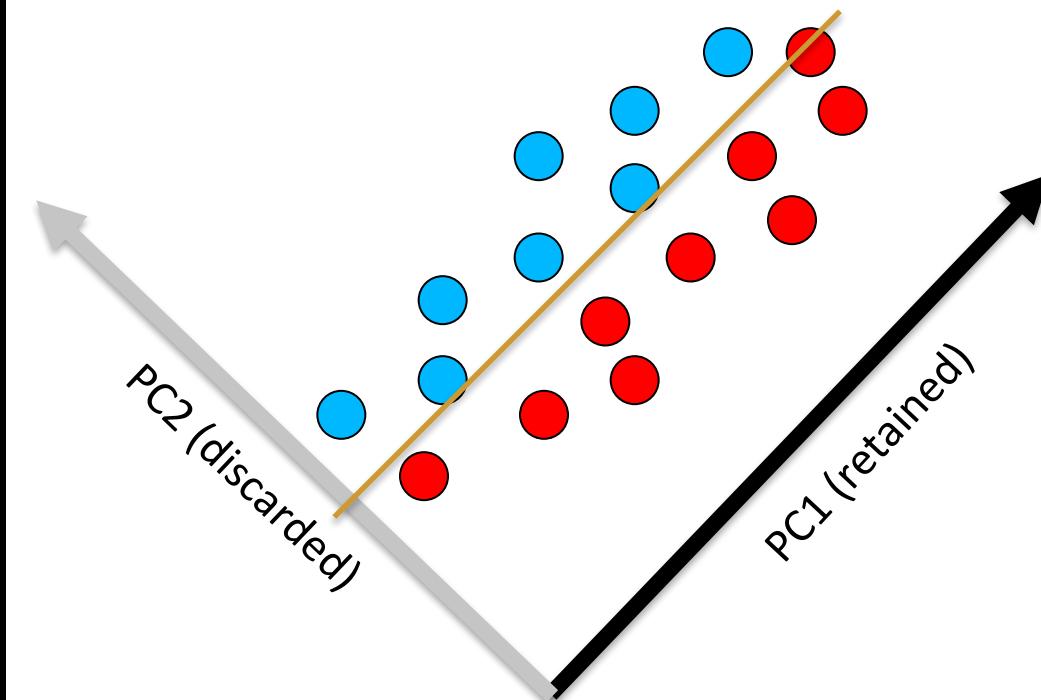


PCA is color blind!



PCA is color blind!

PCA hinders rather than helps: proper classification requires PC2

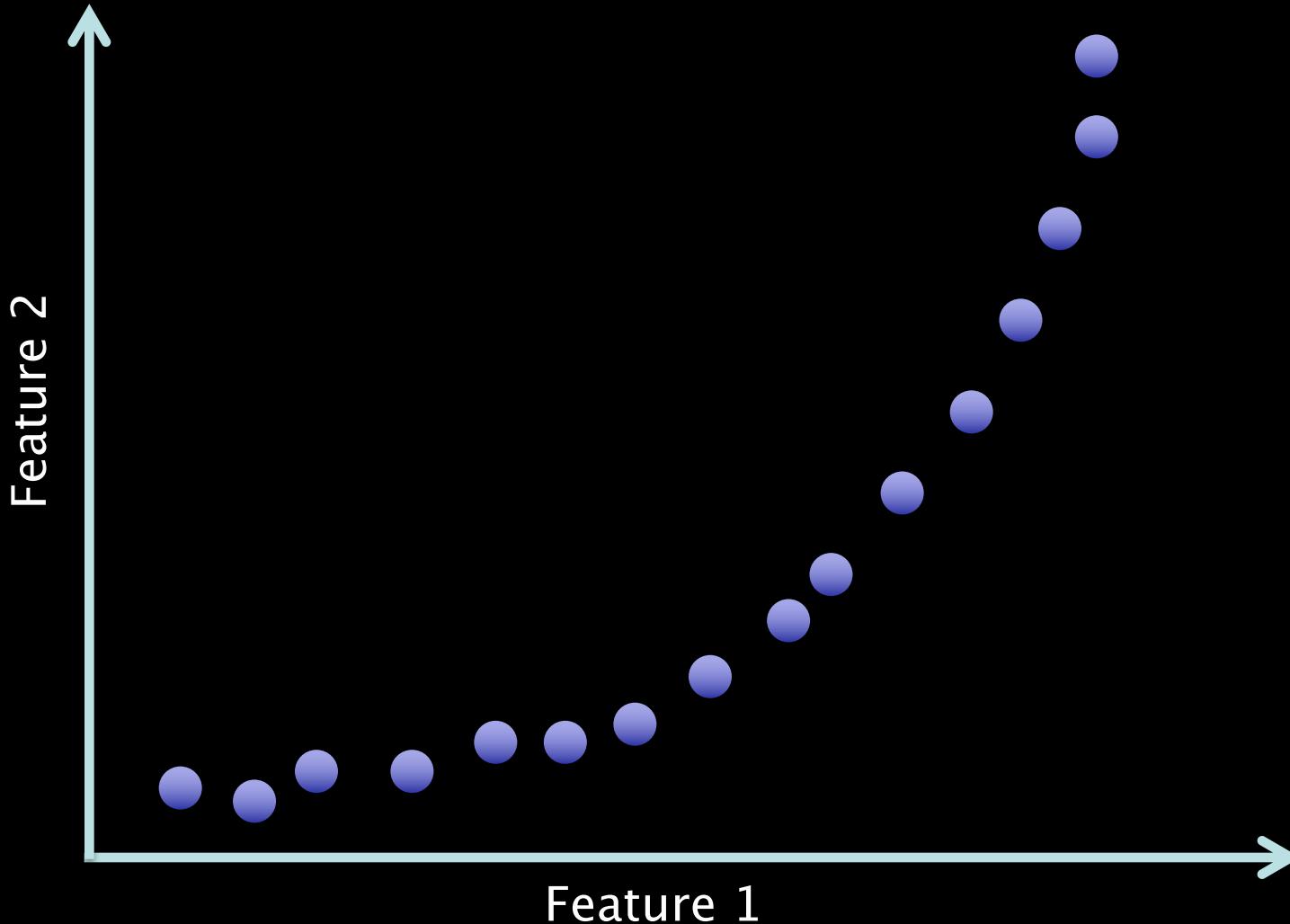


PCA is a “linear” method

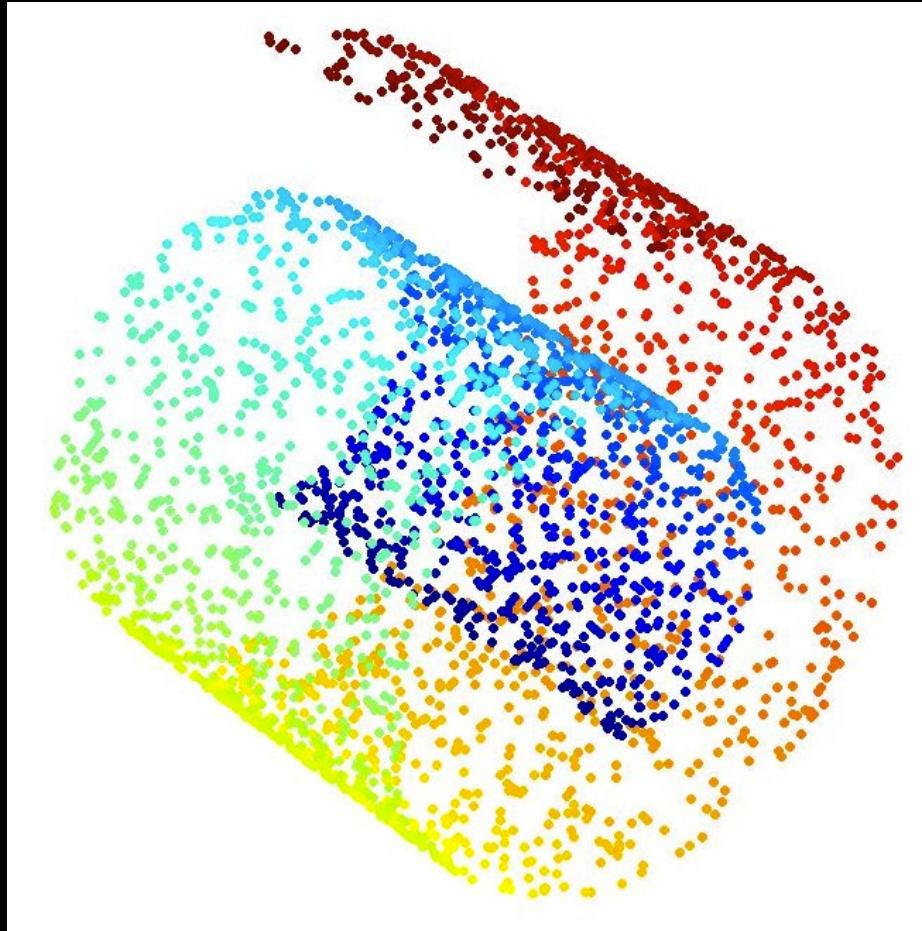
Another reason why PCA is not beneficial to the classification/regression is that it relies on “linear” (straight) data relations

It cannot deal with nonlinear (curved) data relations

1D curved structure in 2D feature space

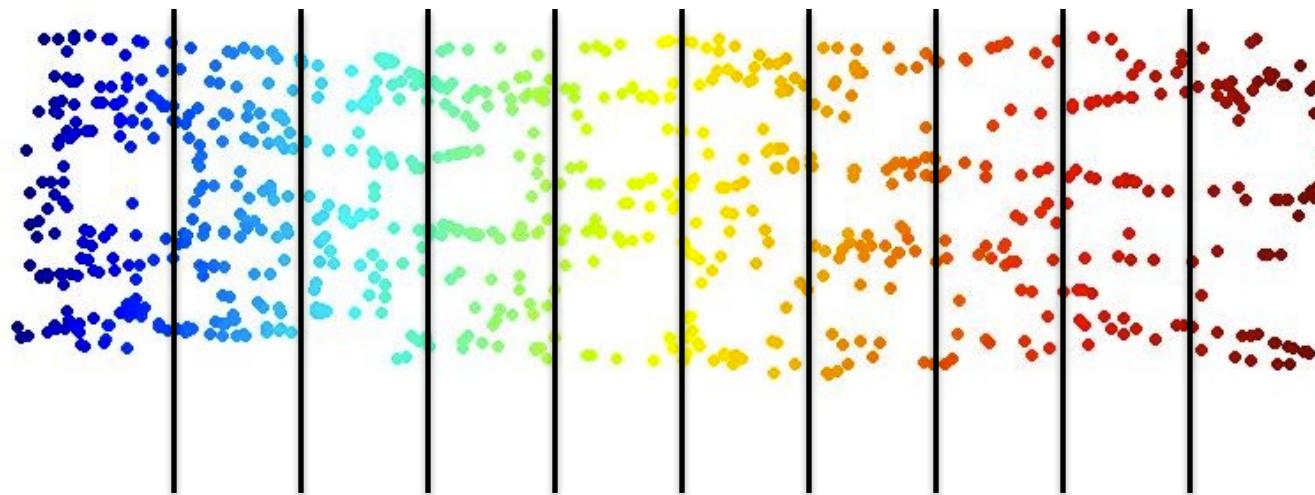


The Swiss Roll dataset



Data points in 3D (hard to classify)

A problem PCA cannot deal with

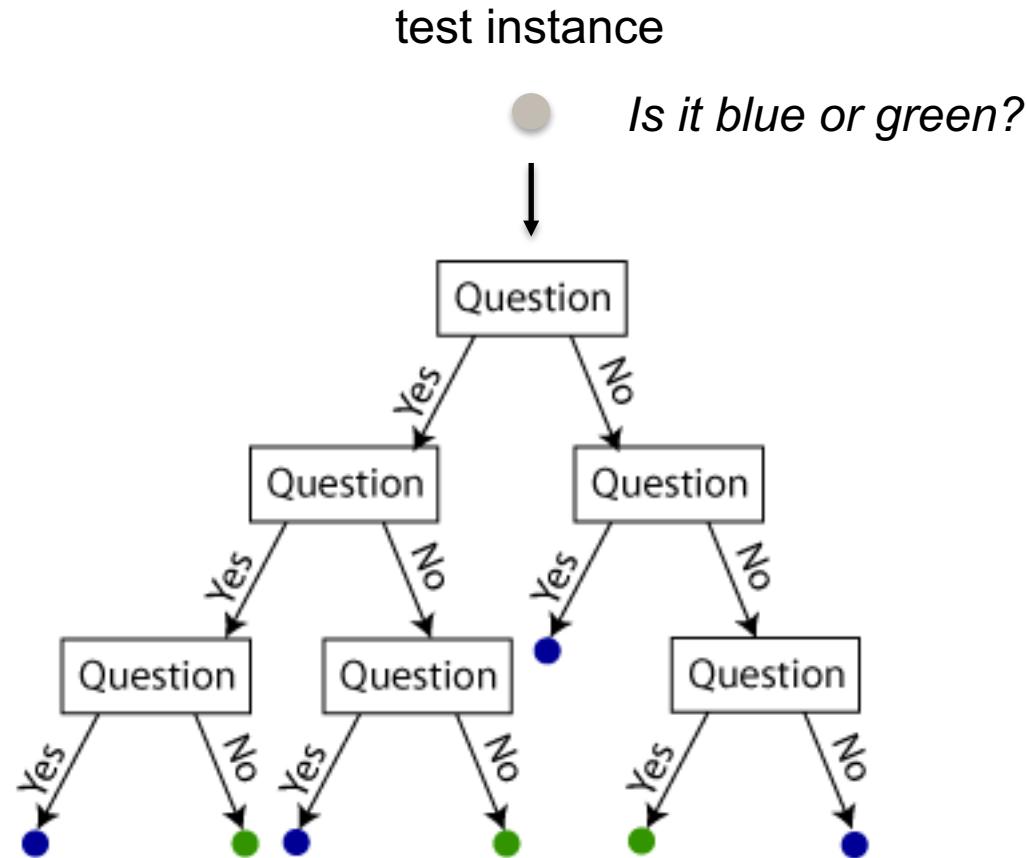


Unfolded data points in 2D (easy to classify)

3 new classifiers...

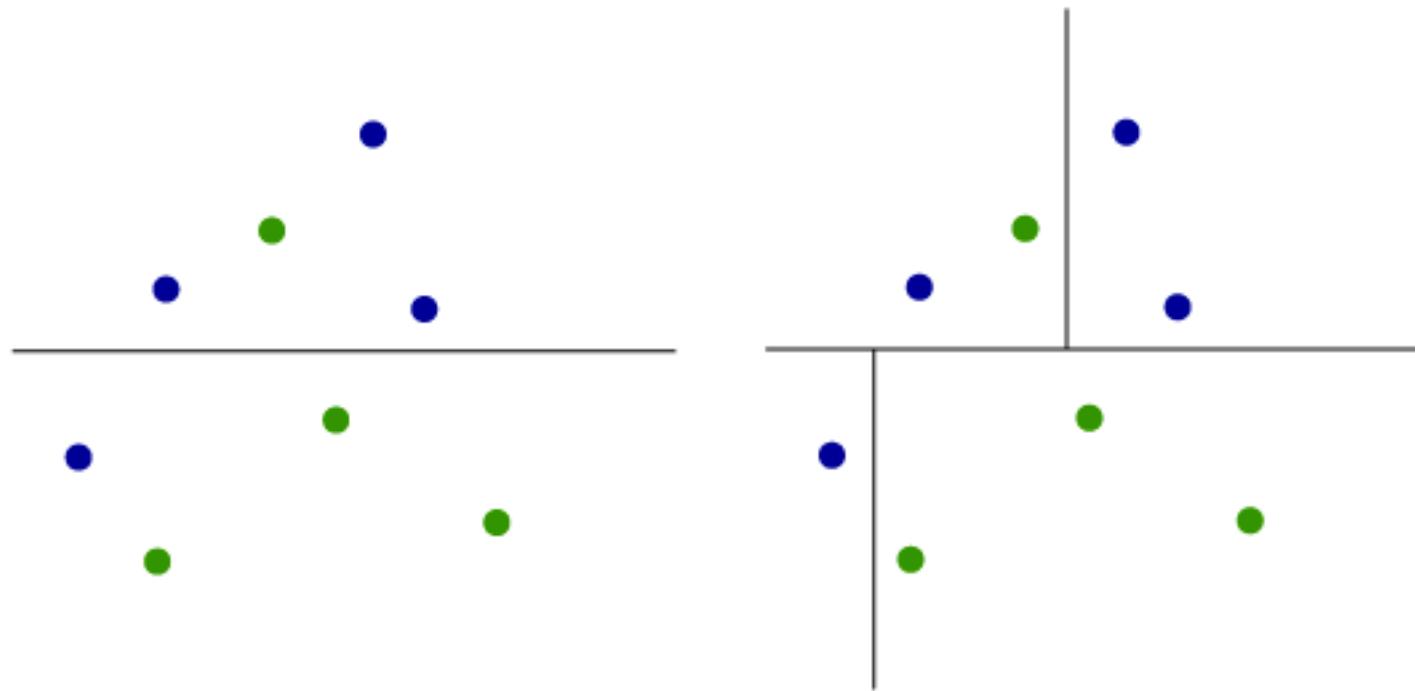
1. Random Decision Forests

Recall: decision tree



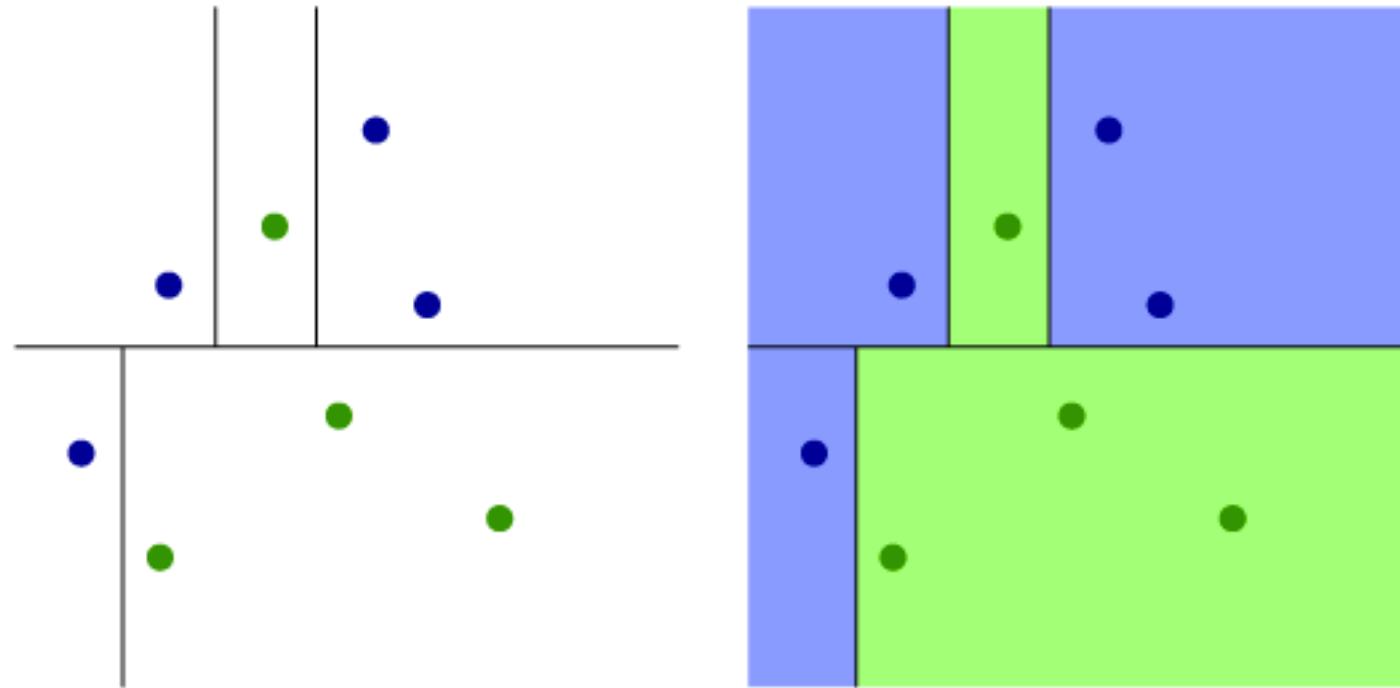
Reproduced from: <https://shapeofdata.wordpress.com/2013/07/02/decision-trees/>

Each test (node) adds a decision boundary



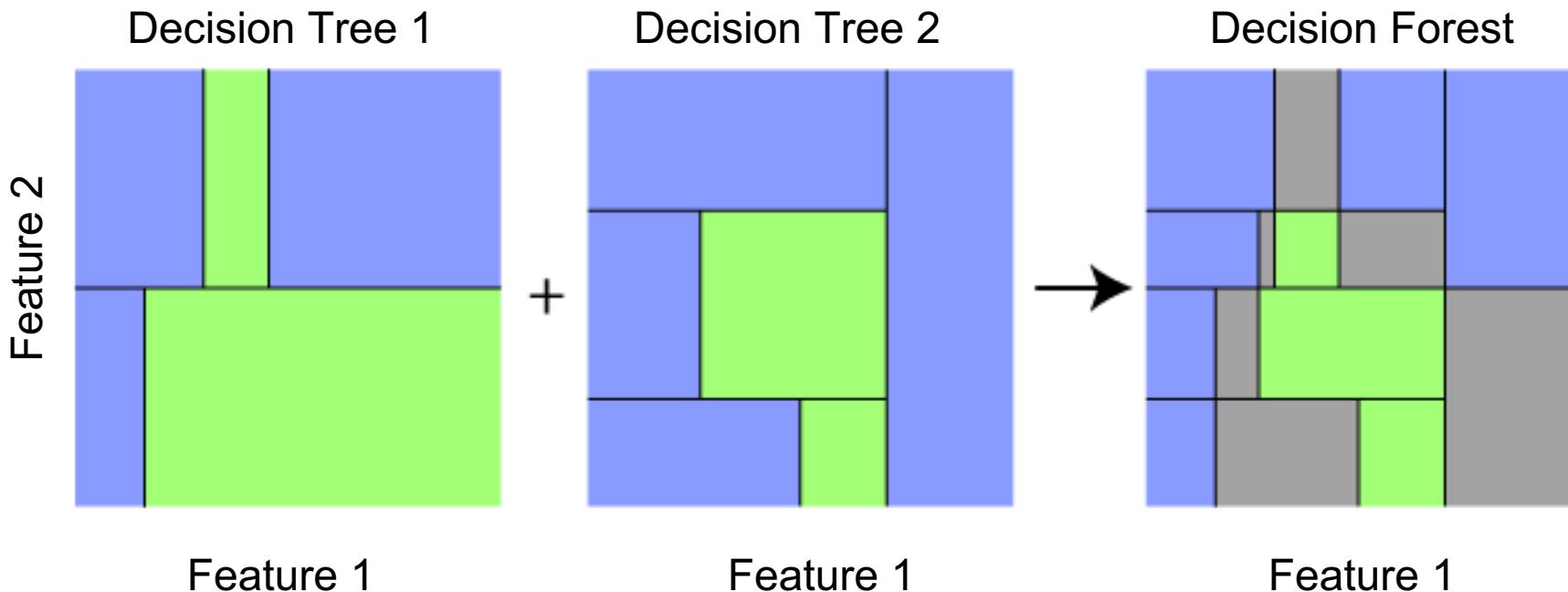
Reproduced from: <https://shapeofdata.wordpress.com/2013/07/02/decision-trees/>

Adding another node/decision boundary

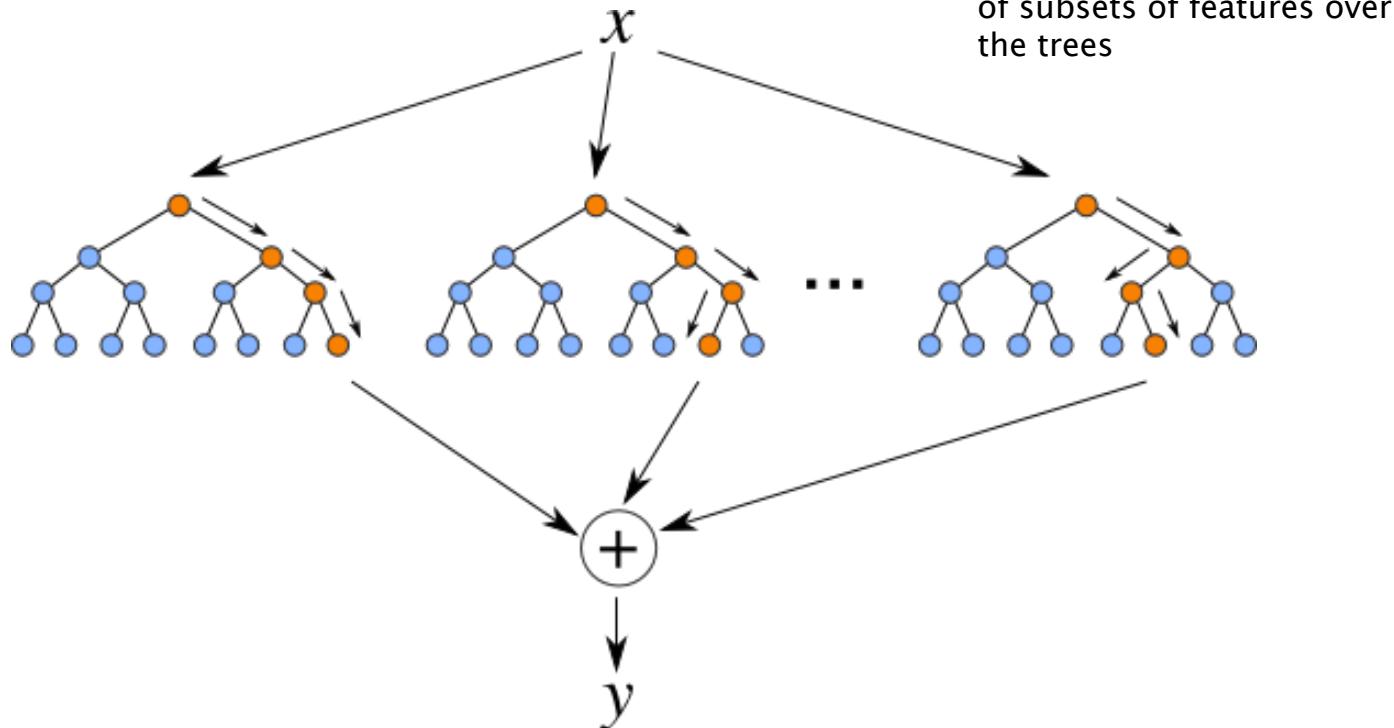


Reproduced from: <https://shapeofdata.wordpress.com/2013/07/02/decision-trees/>

Decision Forests: from one tree to many



Classification and Regression with Random Decision Forests



Classification: y = the mode of the classes outputted by the trees.
Regression: y = the mean of the values outputted by the trees.

Random Decision Forests

The complexity of RDFs is determined by the number of trees (and their depths)

In some decision forests trees are induced on the same complete set of features

In random decision forests, trees are induced on randomly selected subsets of features

2. Naive Bayes Classifier

Bayes's rule

- Data (\mathbf{d})
- Hypotheses (h_i)
- Evidence
- Bayes' rule:

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$$

“ $P(h|\mathbf{d})$ ” should be read as: the probability that h is true, given \mathbf{d}

Reading the Bayes equation

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$$

POSTERIOR ↑ LIKELIHOOD PRIOR

multiplication constant

New observations update your beliefs

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$$

You encounter a person with long hair and try to determine the probability of the person being female

PRIOR = 0.5 (both genders are equally likely)

LIKELIHOOD = $P(\text{long hair} | \text{female})$

POSTERIOR probability updates the future prior (e.g., flower power)

Example (adapted from Russell & Norvig)

Five types of candy bags

h_1 : 100% cherry

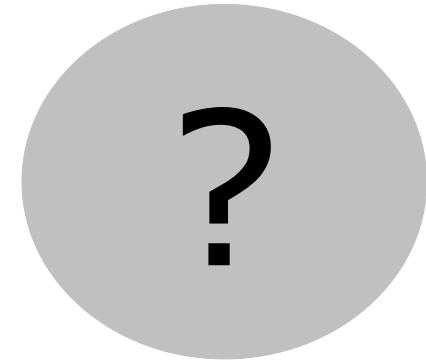
h_2 : 75% cherry + 25% lime

h_3 : 50% cherry + 50% lime

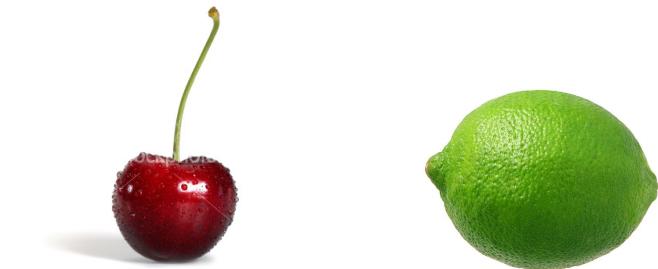
h_4 : 25% cherry + 75% lime

h_5 : 100% lime

Which one is it?



Candy in two flavors



Bayesian prediction

- Given a new opaque bag of candy
 - H denotes the type of bag (h_1, h_2, h_3, h_4, h_5)
 - \mathbf{d} denotes all observations of cherry and lime
 - TASK: predict the flavour of the next piece of candy
-
- In Bayesian learning: Calculate the probability of each hypothesis, given the observations (data)

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i)P(h_i)$$

Calculating Likelihoods

h_1 : 100% cherry

h_2 : 75% cherry + 25% lime

h_3 : 50% cherry + 50% lime

h_4 : 25% cherry + 75% lime

h_5 : 100% lime

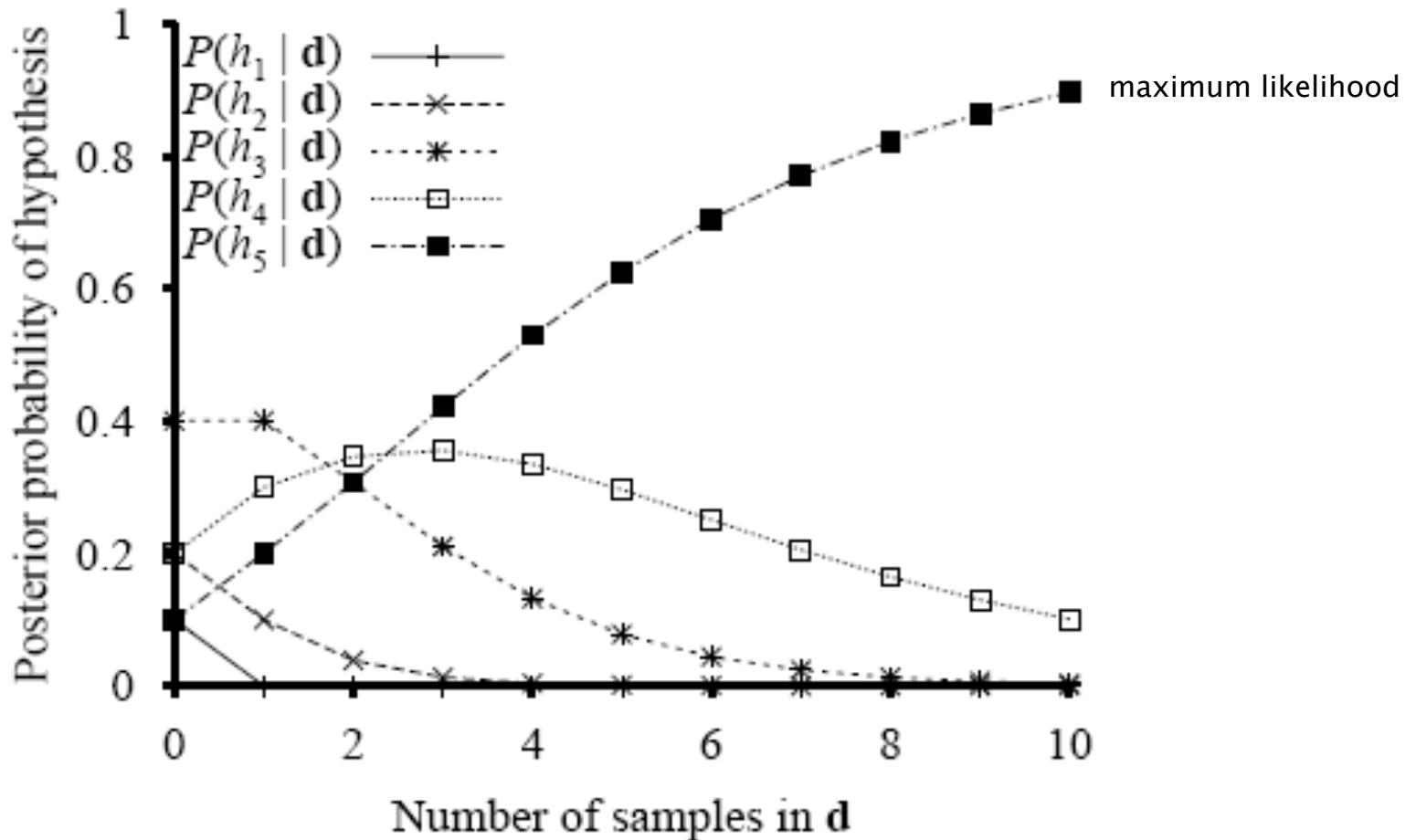
- With each observation, the likelihood is computed according to

$$P(d|h) = P(\text{cherry}|h) \times P(\text{lime}|h) \quad \leftarrow \text{assuming independence}$$

- In case the bag is all lime (h_5) and the first 10 observations are lime, than

$$P(d|h_3) = 0.5^{10} \text{ and } P(d|h_5) = 1^{10}$$

Development of the posterior probabilities (h_5 is the true hypothesis, all observations are 'lime')



Naive Bayes Classifier



- Given a dataset with two classes (Dem,Rep) and three features (A,B,C), compute

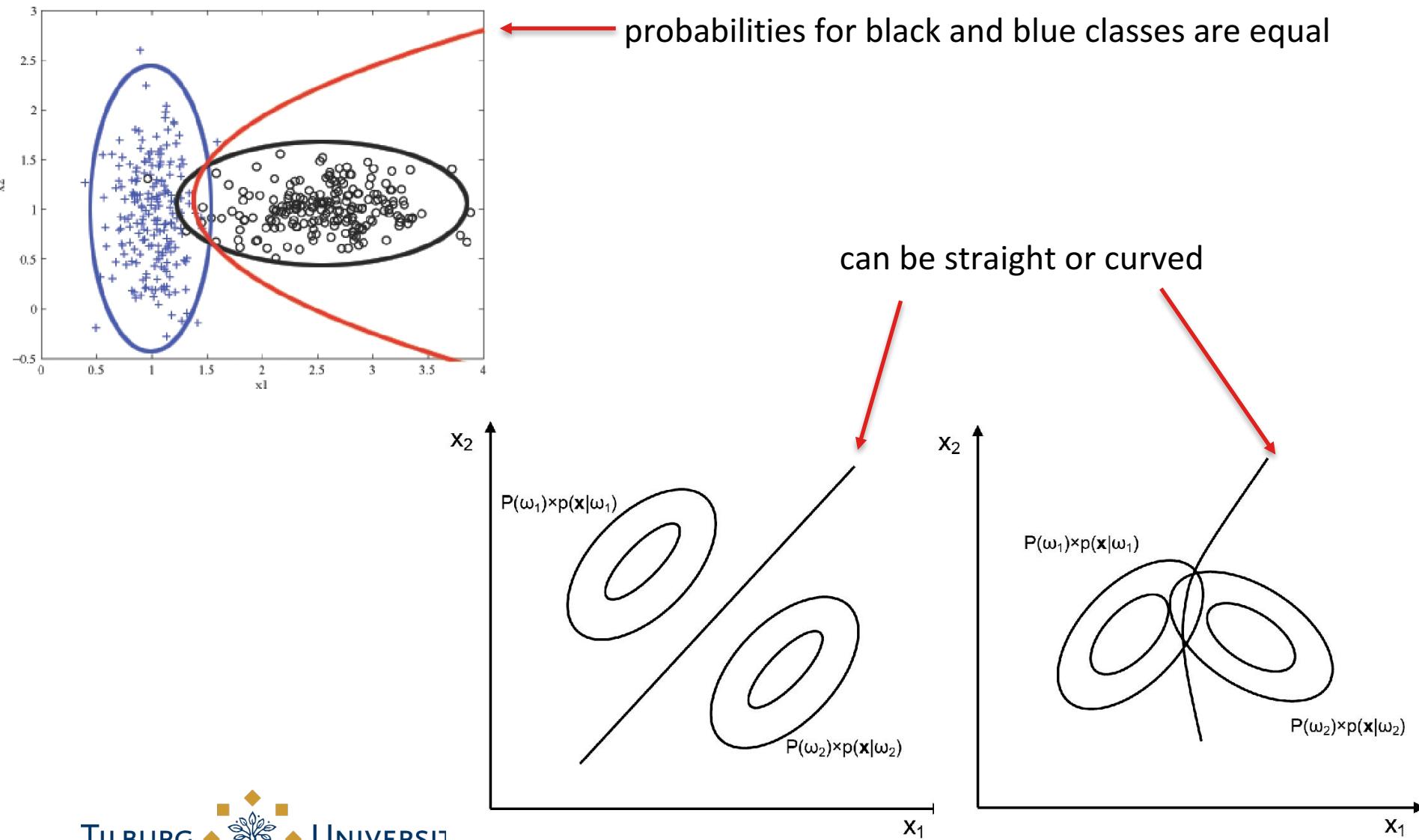
$$P(\text{Dem}|\text{data}) = P(\text{Dem}) \cdot P(A|\text{Dem}) \times P(B|\text{Dem}) \times P(C|\text{Dem})$$

$$P(\text{Rep}|\text{data}) = P(\text{Rep}) \cdot P(A|\text{Rep}) \times P(B|\text{Rep}) \times P(C|\text{Rep})$$

- IF $P(\text{Dem}|\text{data}) > P(\text{Rep}|\text{data})$
THEN Classification is Dem
ELSE Classification is Rep

maximum likelihood

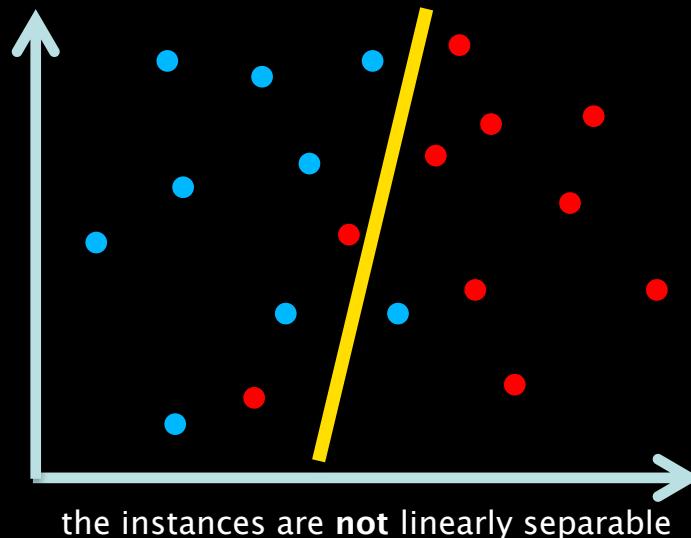
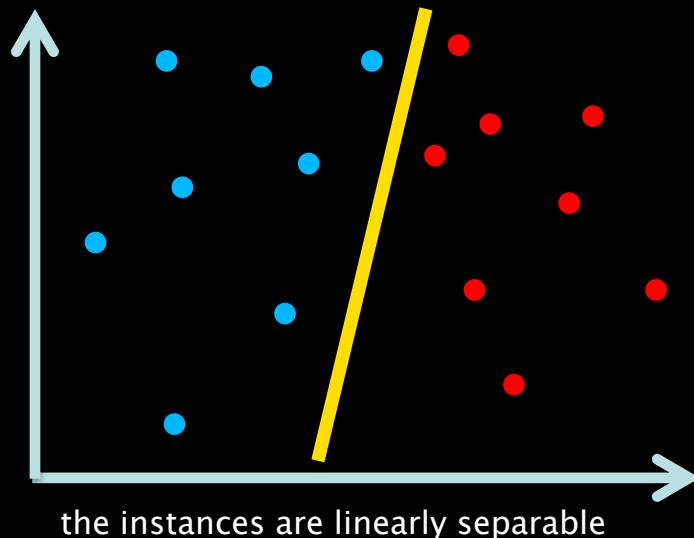
Naive Bayes decision boundaries



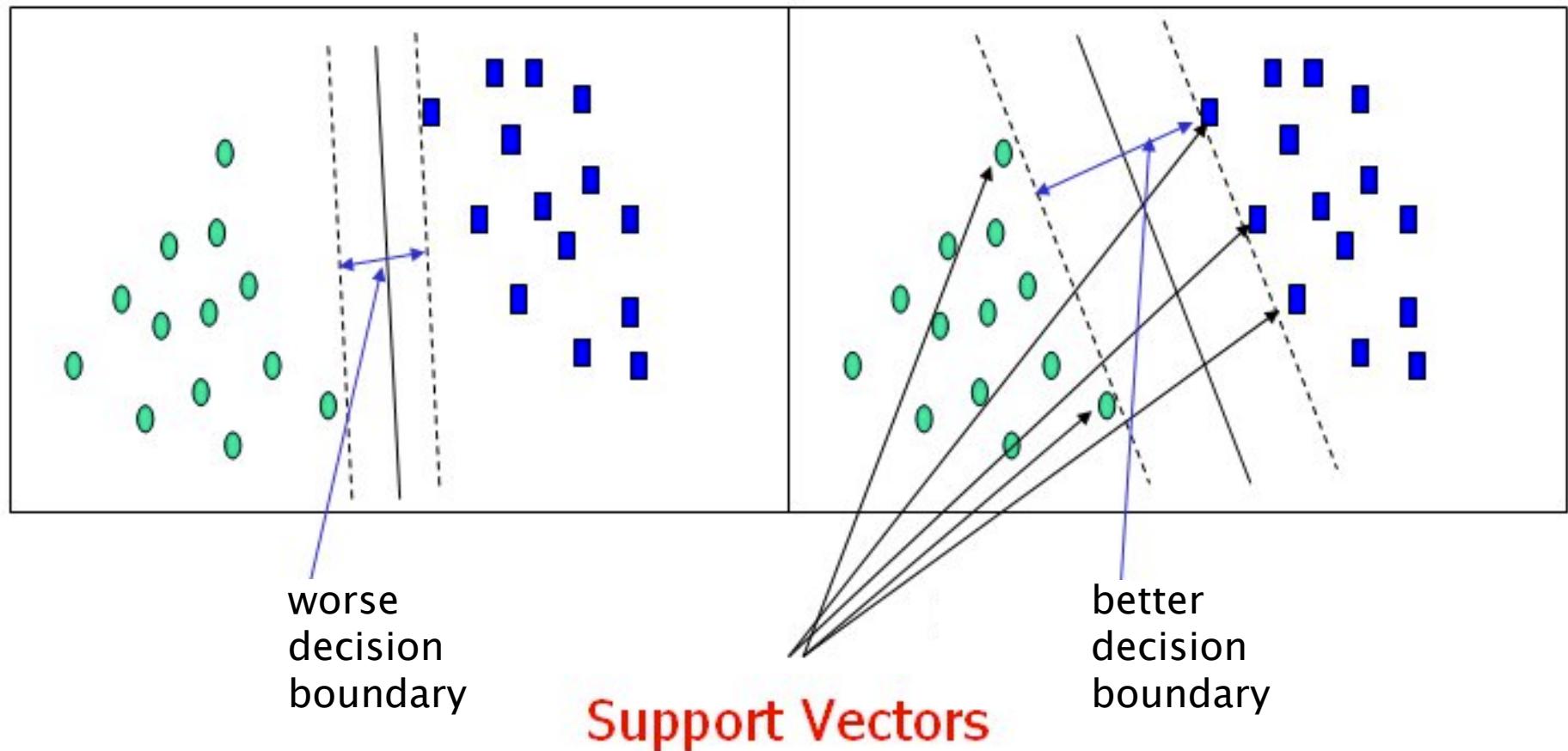
3. Support Vector Machines a.k.a. Kernel Machines

Simplest SVM

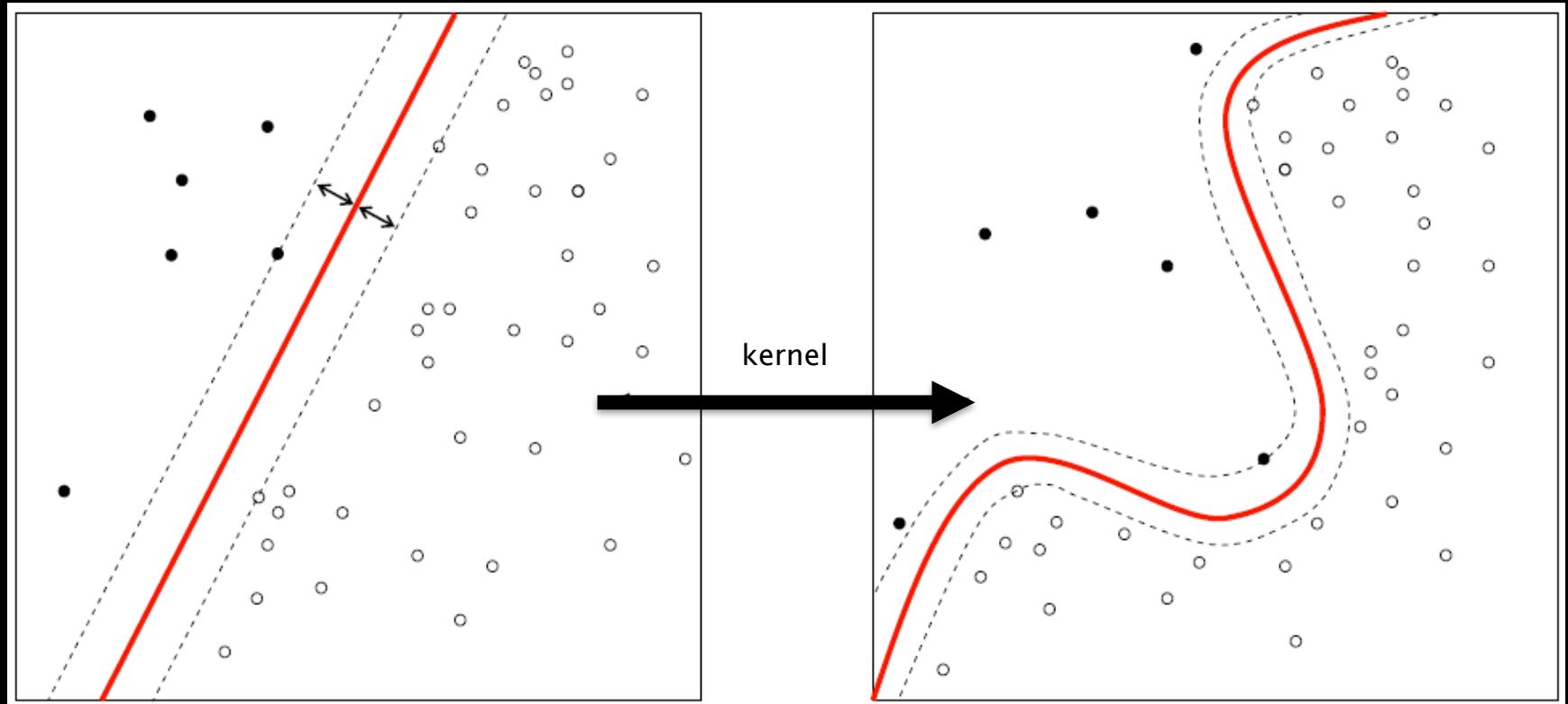
- Linear SVM
- Places a straight line between the classes (simplest model)



Why “Support Vector Machine”?

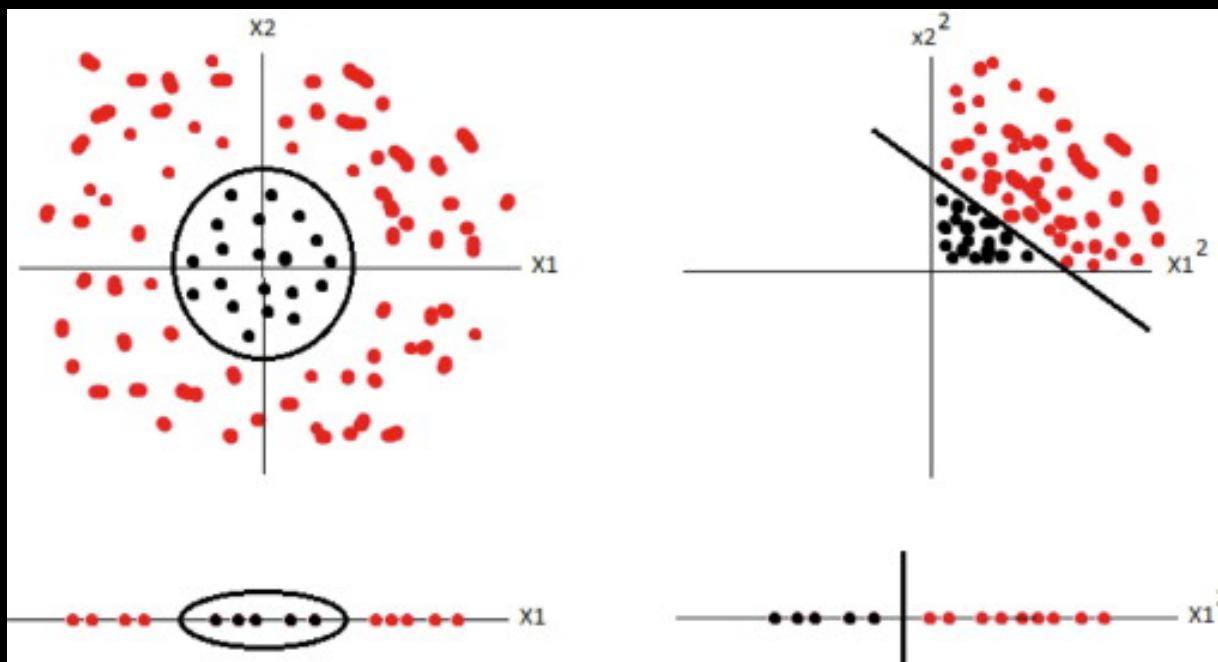


SVMs can make more complex decision boundaries by using kernels



Kernel

- A kernel performs a mathematical operation on instances in two classes that are **not** linearly separable, so that they do become linearly separable



WEKA

Random Decision Forests in WEKA

weka.gui.GenericObjectEditor

weka.classifiers.trees.RandomForest

About

Class for constructing a forest of random trees.

More Capabilities

debug False

maxDepth 0

numExecutionSlots 1

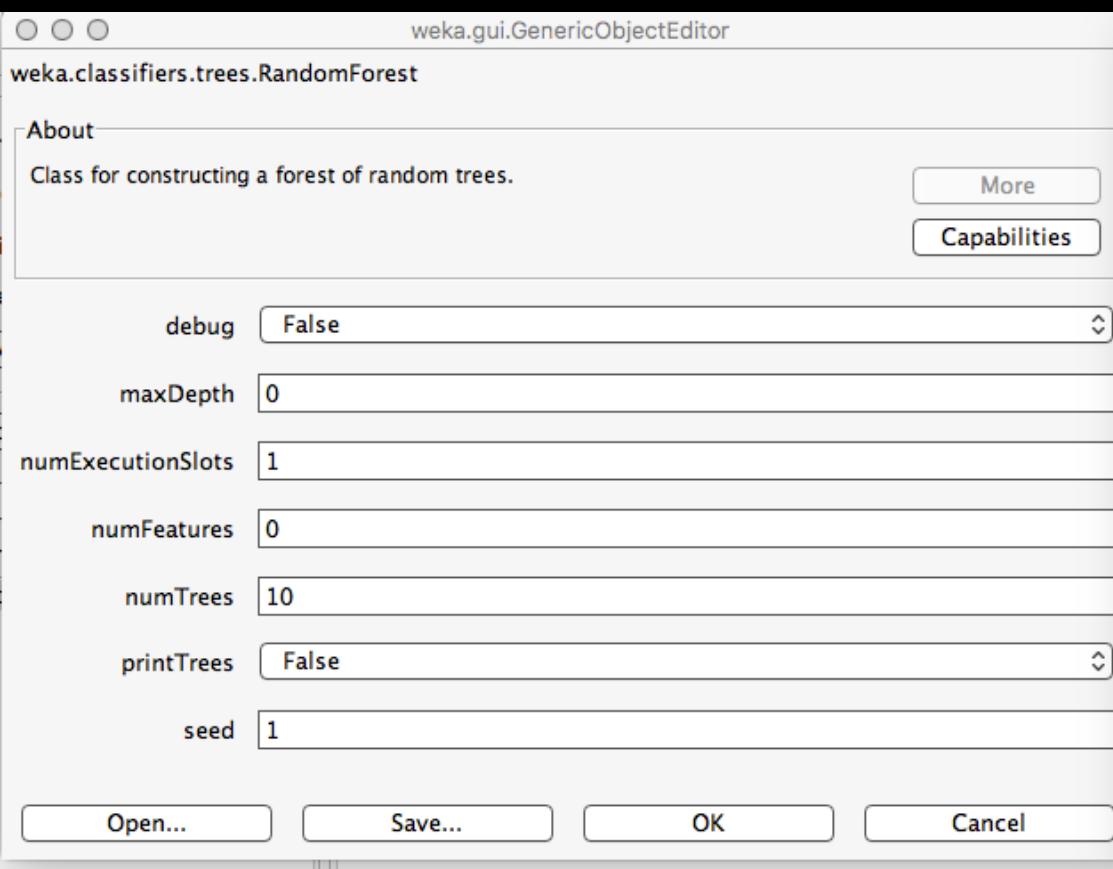
numFeatures 0

numTrees 10

printTrees False

seed 1

Open... Save... OK Cancel



Information

NAME
weka.classifiers.trees.RandomForest

SYNOPSIS
Class for constructing a forest of random trees.

For more information see:
Leo Breiman (2001). Random Forests. Machine Learning. 45(1):5-32.

OPTIONS
printTrees -- Print the individual trees in the output

debug -- If set to true, classifier may output additional info to the console.

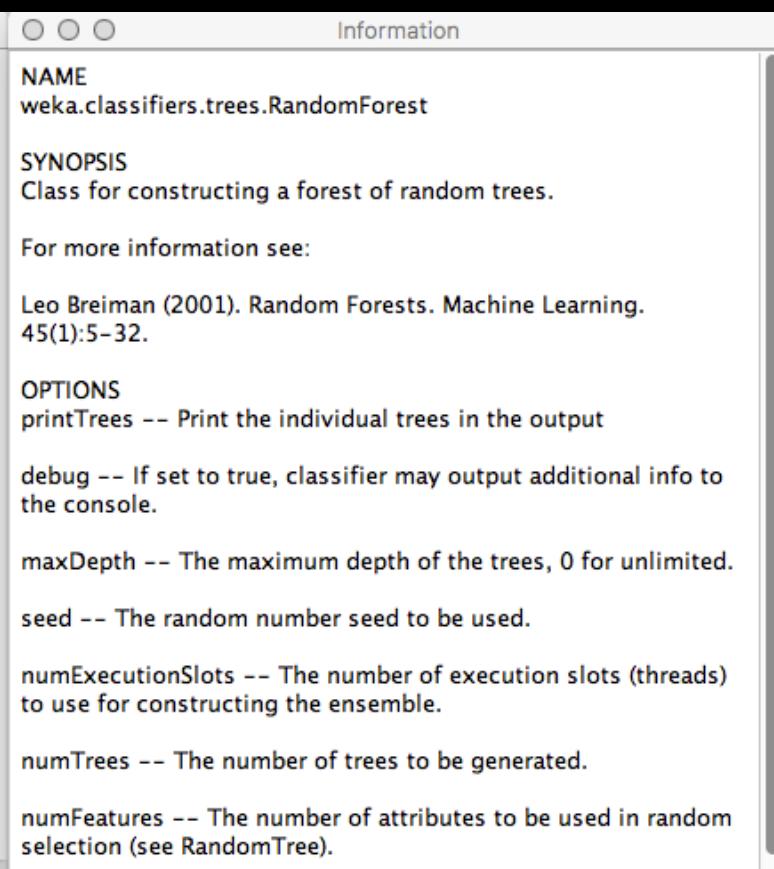
maxDepth -- The maximum depth of the trees, 0 for unlimited.

seed -- The random number seed to be used.

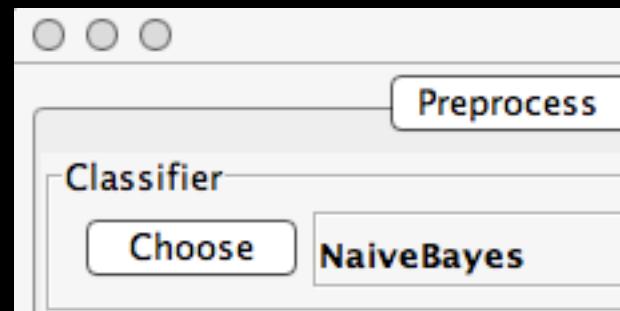
numExecutionSlots -- The number of execution slots (threads) to use for constructing the ensemble.

numTrees -- The number of trees to be generated.

numFeatures -- The number of attributes to be used in random selection (see RandomTree).



Naive Bayes in WEKA



SVM (SMO) in WEKA

weka.gui.GenericObjectEditor
weka.classifiers.functions.SMO

About
Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier.

buildLogisticModels: False
c: 1.0
checksTurnedOff: False
debug: False
epsilon: 1.0E-12
filterType: No normalization/standardization
kernel: Choose PolyKernel -C 250007 -E 1.0
numFolds: -1
randomSeed: 1
toleranceParameter: 0.0010

Open... Save... OK Cancel

Information

kernel -- The kernel to use.
debug -- If set to true, classifier may output additional info to the console.
toleranceParameter -- The tolerance parameter (shouldn't be changed).
c -- The complexity parameter C.
numFolds -- The number of folds for cross-validation used to generate training data for logistic models (-1 means use training data).
epsilon -- The epsilon for round-off error (shouldn't be changed).
filterType -- Determines how/if the data will be transformed.
checksTurnedOff -- Turns time-consuming checks off – use with caution.
buildLogisticModels -- Whether to fit logistic models to the outputs (for proper probability estimates).
randomSeed -- Random number seed for the cross-validation.

(There is also a LibSVM version.)

Suggested Reading

WEKA book

Kernel methods: Section 6.4, pp. 223–229

Naive Bayes: Section 4.2, pp. 90–99

Please note that these sections contain many equations.