Supervised Machine Learning

Part Two: Classification

Agenda

- Classification Models (Algorithms)
  - K Nearest Neighbors
  - Decision Trees
  - Support Vector Machines
  - Naive Bayes
- Model Evaluation

Classification Models

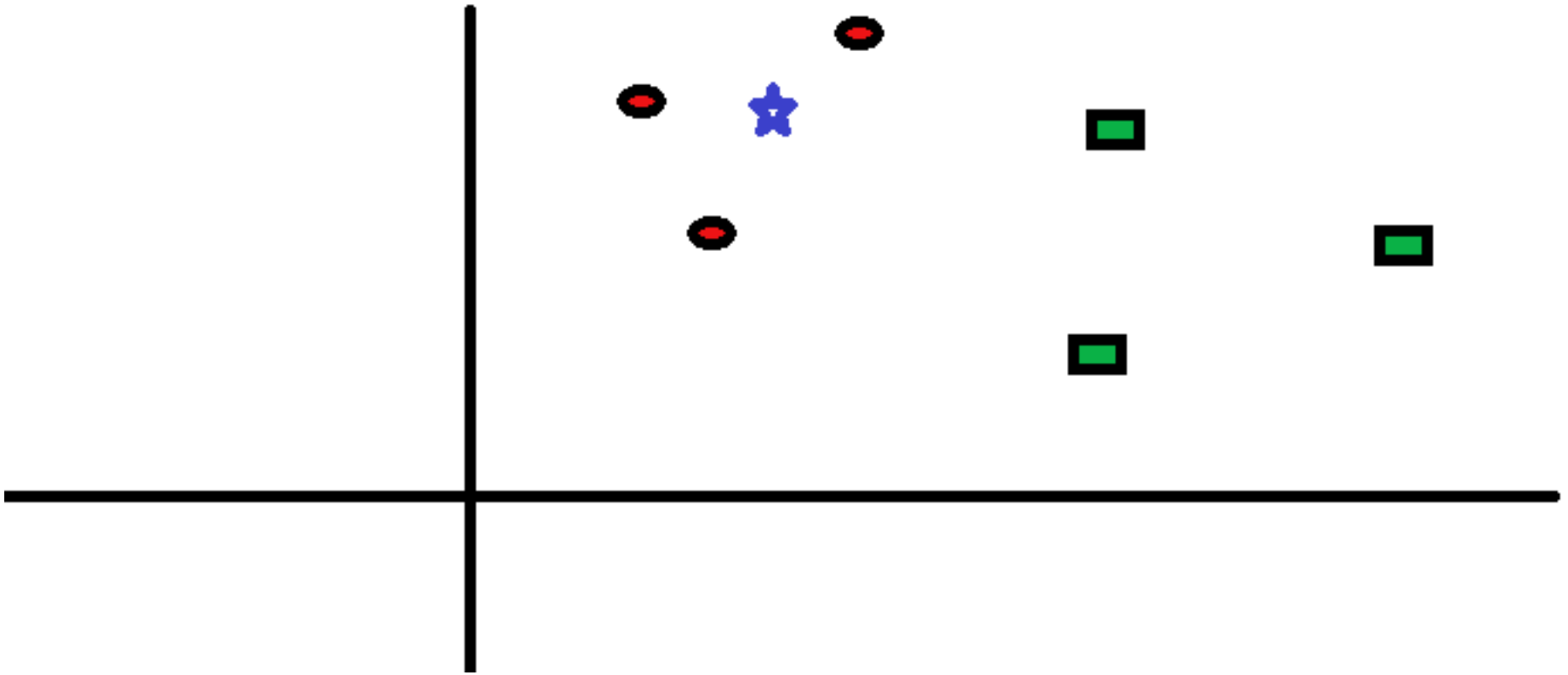
Classification Models

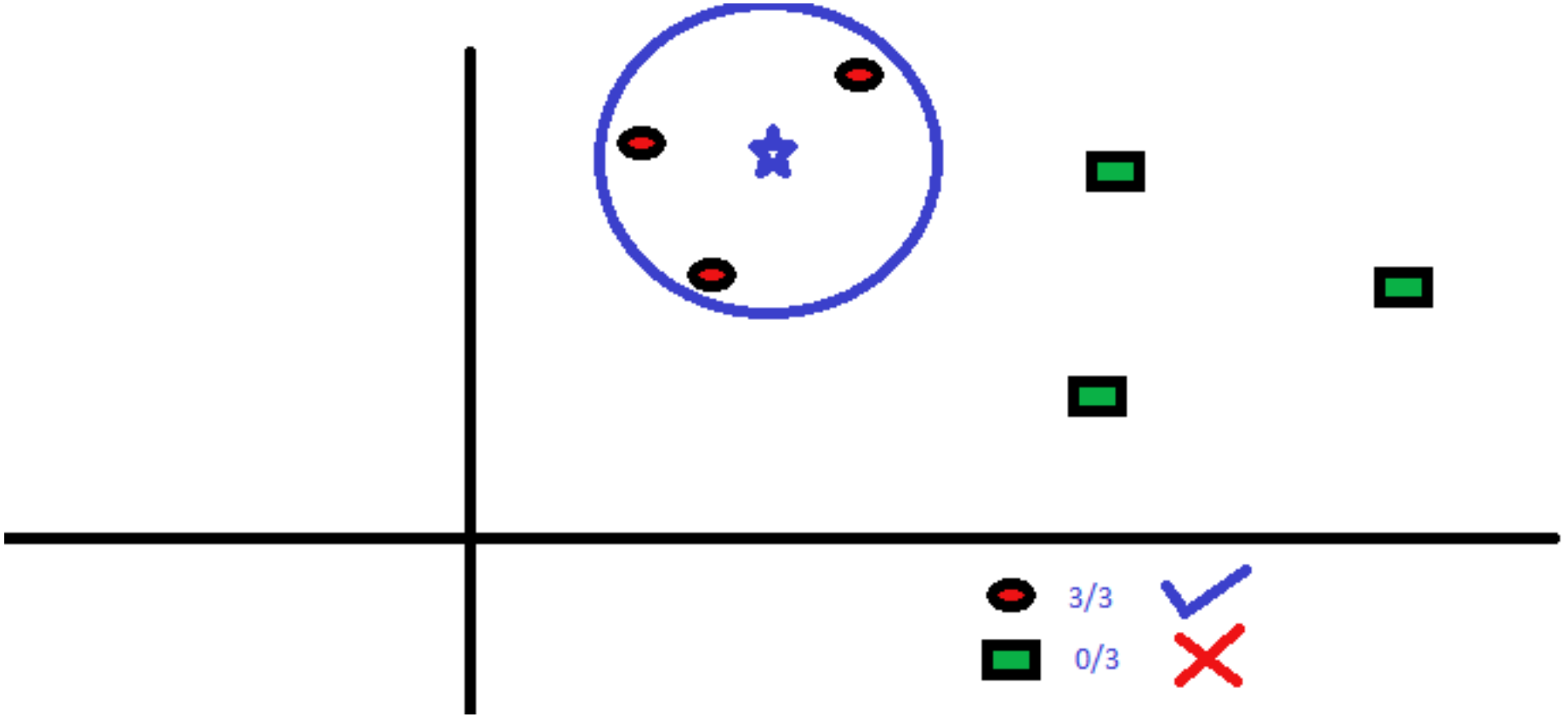
- Supervised learning algorithms that assign 2 or more discrete classes (categories) to an instance.
- Based on what is learned from the training data.
- An example of pattern recognition.
- Common algorithms
  - K Nearest Neighbors
  - Decision Trees
  - Support Vector Machines (SVM)
  - Naive Bayes

K Nearest Neighbors

K Nearest Neighbors (kNN)

- In Scikit-Learn the `neighbors` package provides both supervised and unsupervised nearest neighbors methods
  - these are simple and effective!
- Find a predefined (k) samples closest in distance to the input point and predict the label from those.
- Distance == similarity in this case, and can be any metric measure. Euclidean distance is most common.
- Non-generalizing: must "remember" all training data.




3/3
0/3

Pros and Cons of kNN

Pros

- Works well with distance-sensitive data
- Simple and effective for a wide array of tasks
- Non parametric - works well with irregular decision boundaries.
- Often a first approach
- No worse than 2x Bayes error rate as data → ∞

Cons

- kNN can be very arbitrary and requires hand-holding to model.
- How do you choose k?
- What is the definition of neighbor/similarity?
- Suffers from the curse of dimensionality.

Choosing K

K is an arbitrary number between 1 and the number of instances in the data set. Three options for choosing:

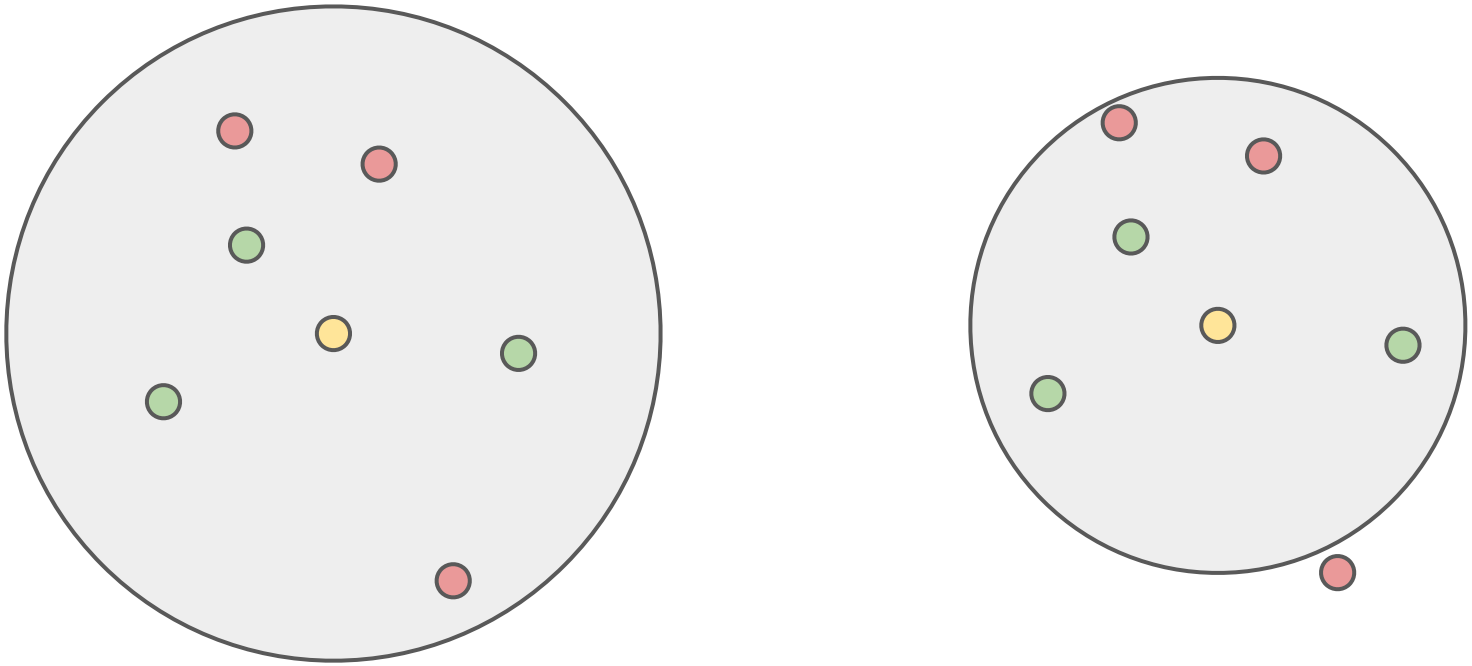
1. Guess
2. Heuristic
3. Optimization

Tips

- Avoid an even k with only 2 classes (tie break)
- Choose k >= # classes + 1
- Choose as low a k as possible

Use Coprime Class and k

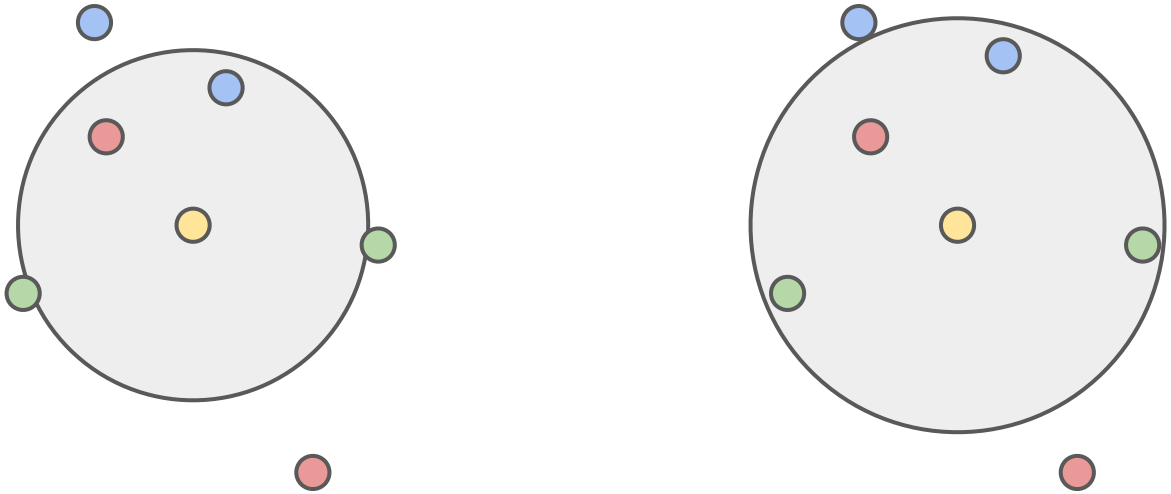
Coprime numbers don't share common divisors except for 1. So 4 and 9 are coprime but not 3 and 9.



Classes = 2 (3 red, 3 green)
k = 6 and k=5 respectively

Use Number of Classes + 1

With k < # of classes, there is no chance that all classes will be represented - but we still want to prevent ties.



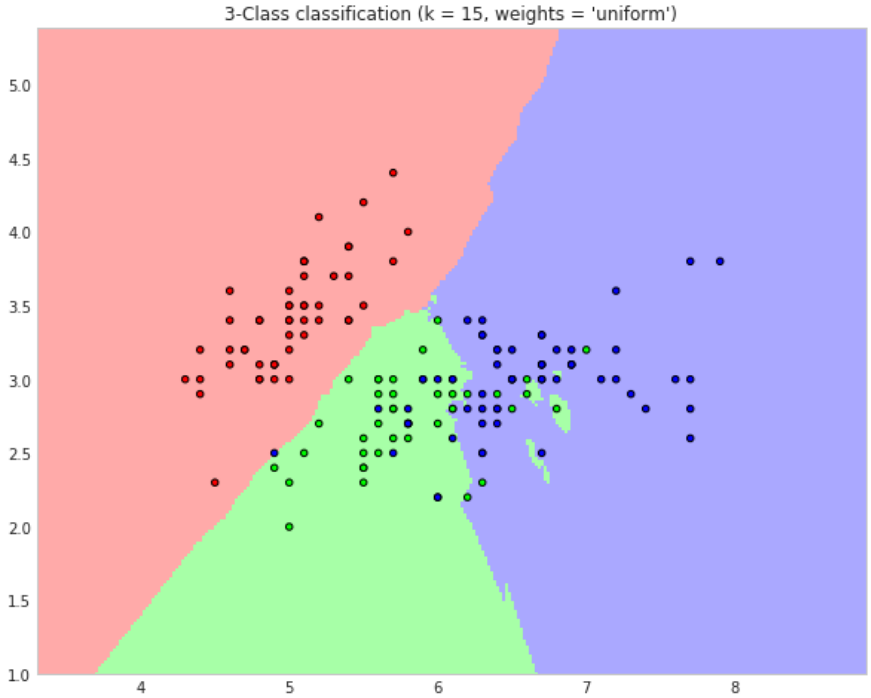
Classes = 3 (2 red, 2 green, 2 blue)
k = 2 and k=4 respectively

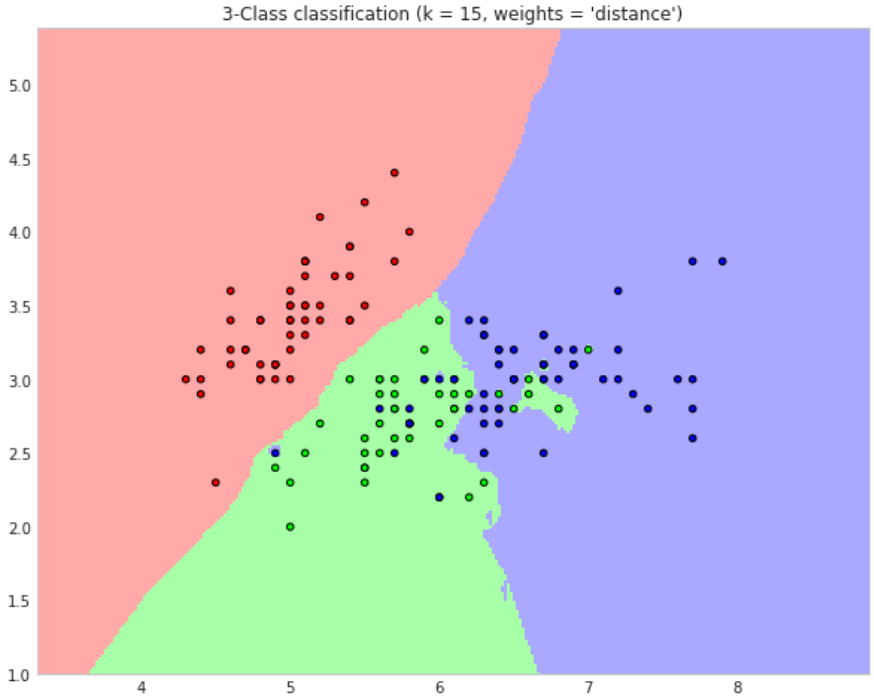
Optimization

- Many people assert k should be chosen via *domain knowledge*
- *A*s K increases, the *complexity* increases and slows down performance.
- Minimize error by trying different values of k
- If you have a large amount of data, iterate 2x through 1% of the data with a variety of K and compute the error.

Lab: K Nearest Neighbors

Sklearn


3-Class classification (k = 15, weights = 'uniform')
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8


3-Class classification (k = 15, weights = 'distance')
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8

Decision Trees

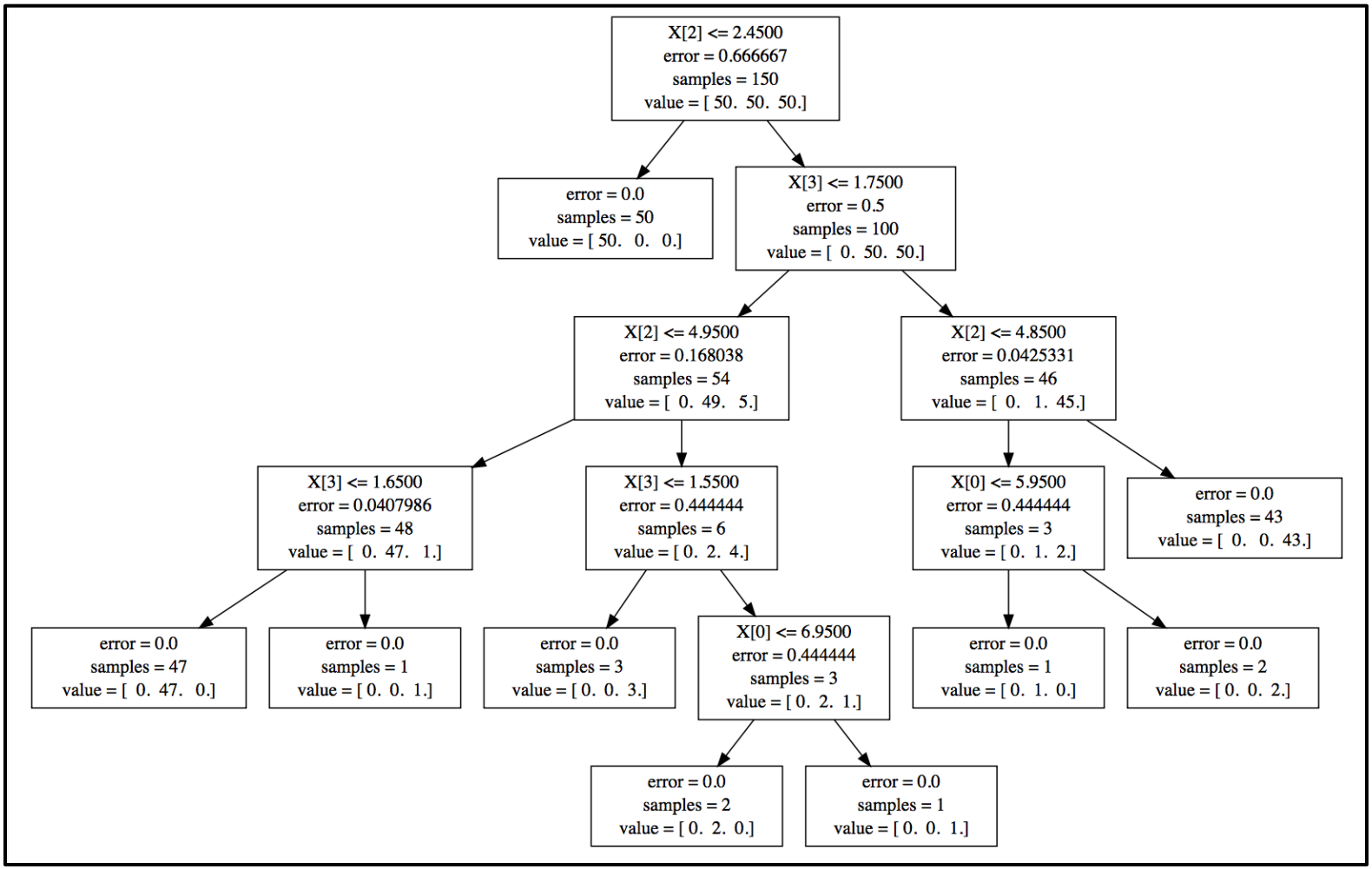
Decision Trees

- Non-parametric inductive learning that generalizes well.
- Creates a graphical model of rules that partitions the data until a decision is reached at one of the leaf nodes.
- Complexity is related to the amount of data and the partitioning method.

Decision Trees

- In Scikit-Learn the `tree` package provides classifiers and regressors based on a decision tree model.
- In Scikit-Learn the ensemble package provides Random Forest classifiers.
- In Spark, the pyspark.mllib.tree package provides DecisionTree and DecisionTreeModel as well as the RandomForest models.

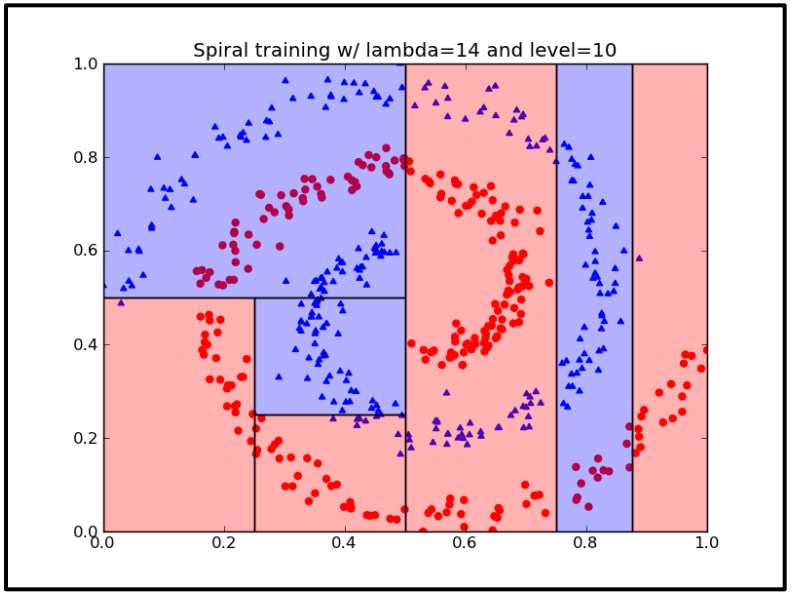
Decision Tree Example - Iris Dataset



Decision Tree Algorithm

Partition data as follows:

1. Start with whole training data
2. Select attribute along dimension that gives “best” split
3. Create child nodes based on split
4. Recurse on each child using child data until stop



What is the “best” split?

Split Metrics

Gini Impurity (used by CART)

Measures how often randomly chosen elements would be incorrectly labeled according to distribution in partition. Zero value means all cases fall into single target

Information Gain (ID3 and C5)

Uses entropy - the amount of information contained in every partition that represents some part of data.

Variance Reduction (CART)

Maximally select the split where the most variance would be reduced, used for continuous values of data.

Gini

For a set of items where i ∈ {1,2, …, m} and f_i is the proportion of items in the split labeled i, the Gini Impurity is computed as follows:

$$I_G(f) = 1 - \sum_{i=1}^{m} f_i^2$$

Essentially the sum of the probability of an item being chosen times the probability of a mistake being made.

Pros and Cons of Decision Trees

Pros

- Simple to understand; no representational mysticism. Trees can be drawn out.
- No need to normalize or standardize (need to deal with missing values).
- Low cost
- Handle multi-output
- Validate with Statistics!
- Handles wrong assumptions.

Cons

- Prone to overfit (deep trees don't generalize)
- Minor variations in data cause big changes in tree structure (unstable) - fix with ensemble methods.
- Create biases if some classes dominate
- Some functions are impossible to model

Tips for Using Decision Trees

Ration of samples to number of features is important, trees in high dimensional space is very likely to overfit.

Perform dimensionality reduction to give tree a better chance of finding features that are discriminative (PCA, ICA, Feature Selection)

Visualize your tree as you are training by using the export function. Use max_depth=3 as an initial tree depth to get a feel for how the tree is fitting to your data, and then increase the depth.

Tips for Using Decision Trees

Remember that the number of samples required to populate the tree doubles for each additional level the tree grows to. Use max_depth to control the size of the tree to prevent overfitting.

Balance your dataset before training to prevent the tree from creating a tree biased toward the classes that are dominant.

Lab: Decision Trees

Sklearn & MLlib

Sklearn Decision Tree Interpretation

1. Imagine that all data (all rows) start in a single bin at the top of the tree.
2. All features are considered to see how the data can be split in the most informative way– this uses the gini measure by default
3. At the top we see the most informative condition is PetalLength <= 2.4500. If this condition is true, take the left branch to get to the 50 samples of value = [50. 0. 0.]. This means there are 50 examples of class/target 0, in this case Iris-setosa. Unfortunately, the default scikit-learn export to graphviz/dot does not seem to be able to include this information (but see below). The other 100 samples, of the 150 total, go to the right bin.
4. This splitting continues until
    1. The split creates a bin with only one class– for example the bin with 50 Iris-setosa is not split again.Or,
    2. the resulting bin has less than 20 samples– this is because we set the min_samples_split=20when initializing the decision tree. If we had not set this value, the tree would keep splitting until all bins have a single class.

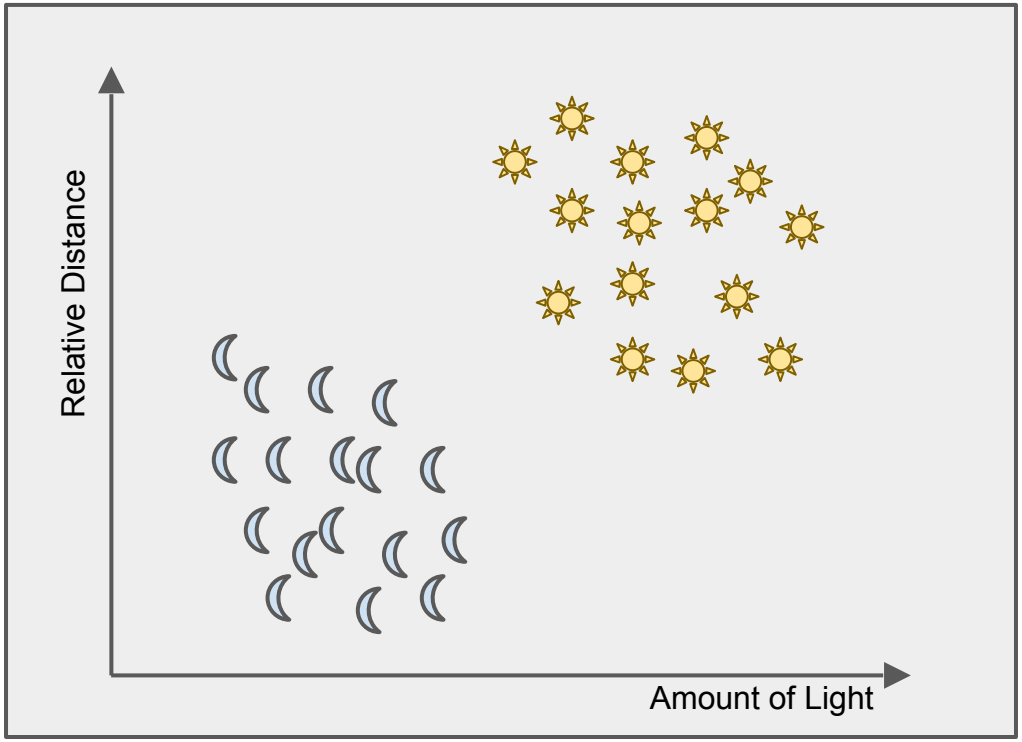
Support Vector Machines

Support Vector Machines

- Introduced by Vladimir Vapnik in the 1980s
- Designed to solve a two-group classification*
  - boolean (true, false)
  - ids (3,4)
  - sign (1, -1)
- Desirable
  - Avoids curse of dimensionality
  - Works in high dimensional space
  - FAST to compute

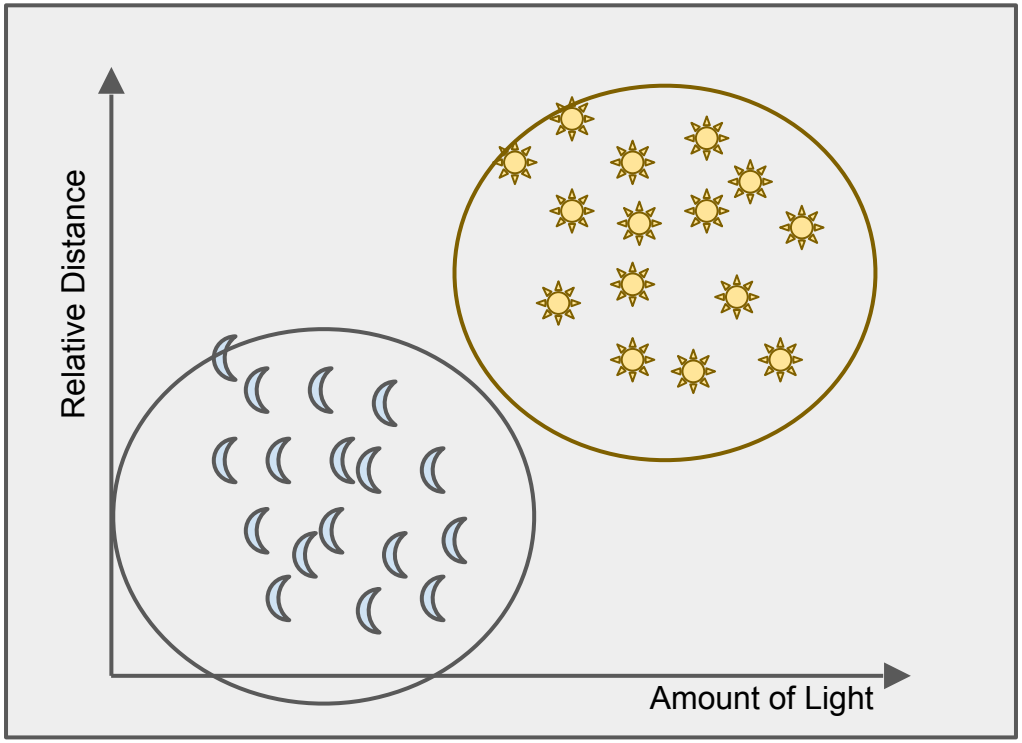
Stars or Moons?

Back to our classification problem, let's say we know we have two discernible groups (a strong hypothesis):



Stars or Moons?

Using kNN we're highly dependent on the instances, and we get centers:

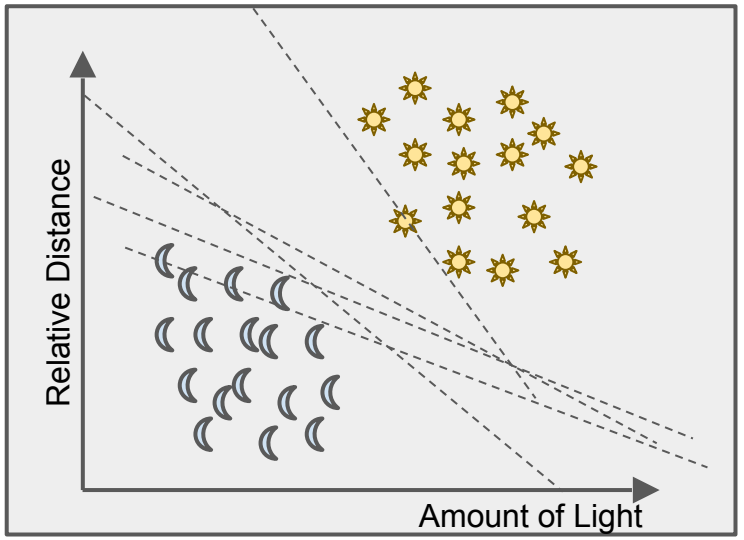


Decision Boundary

In order to *generalize* this model, we really want to find a *decision boundary* - a line between two classes of data.

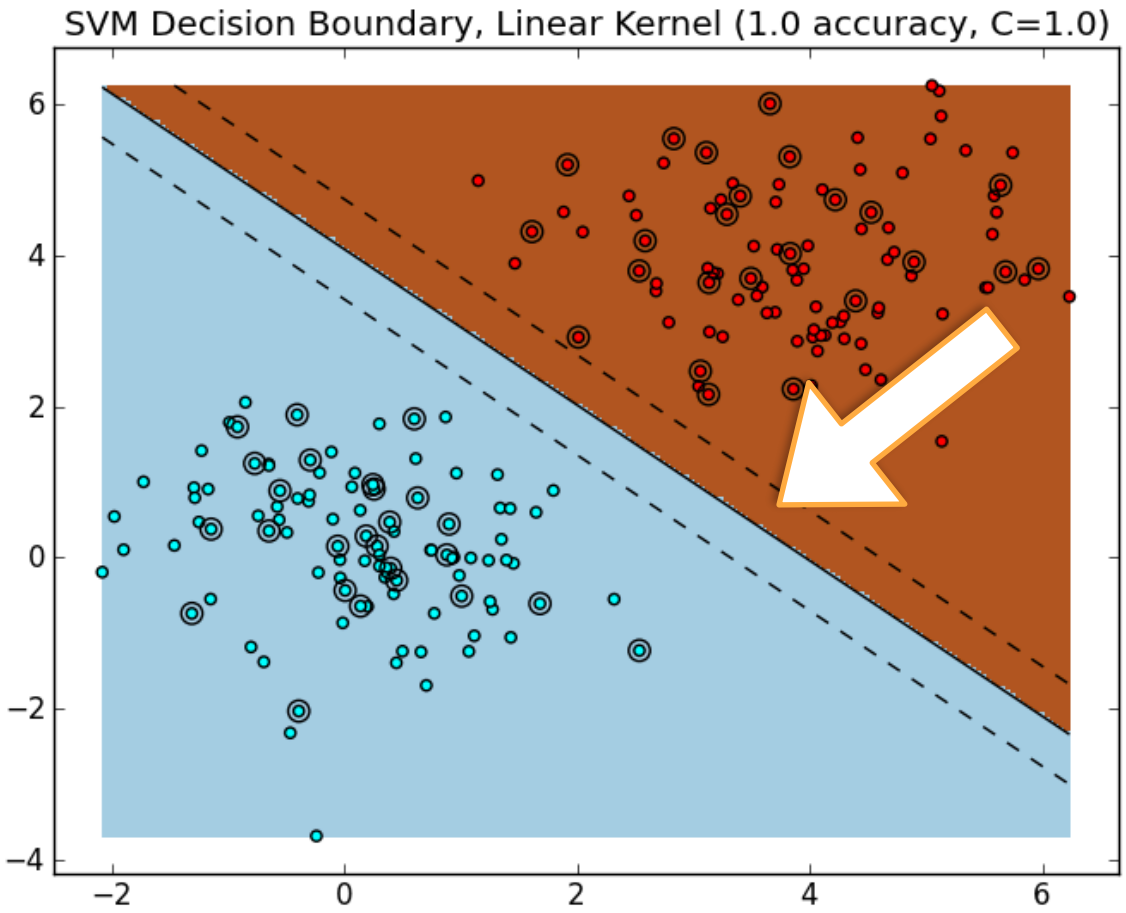
How many lines fit between moons and stars?

Instead of picking an arbitrary line, we attempt to maximize the distance between classes.



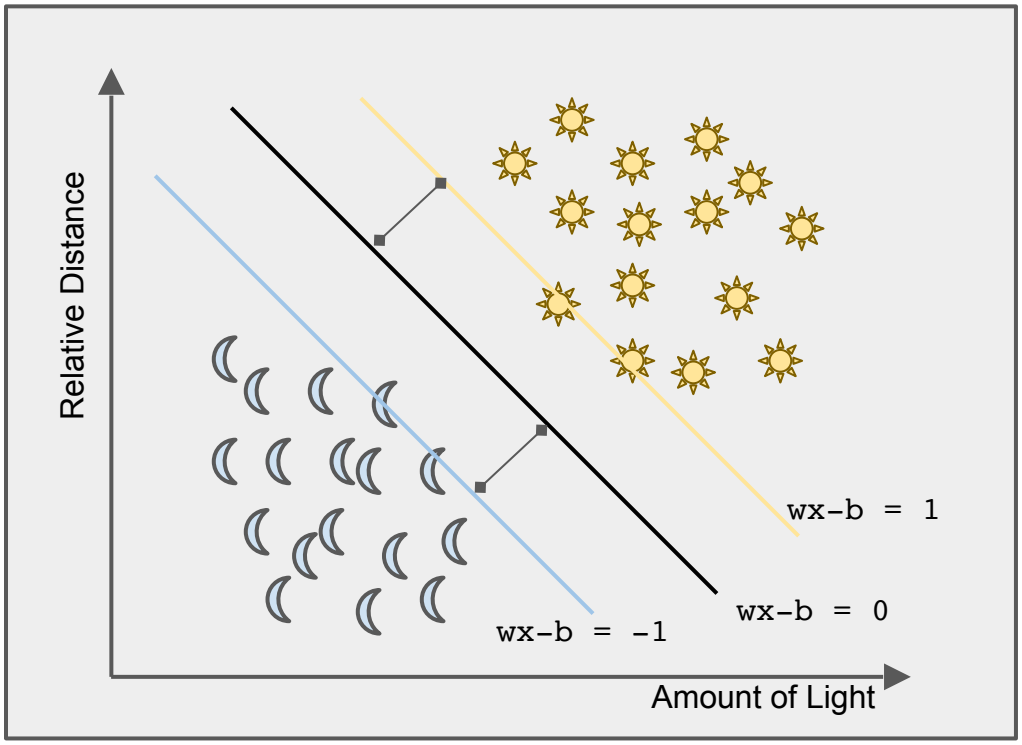
Definition
In geometry a **hyperplane** is a *subspace* of one dimension less than its ambient space. If a space is 3-dimensional then its **hyperplanes** are the 2-dimensional planes, while if the space is 2-dimensional, its **hyperplanes** are the 1-dimensional lines.

The Hyperplane



Maximizing Distance

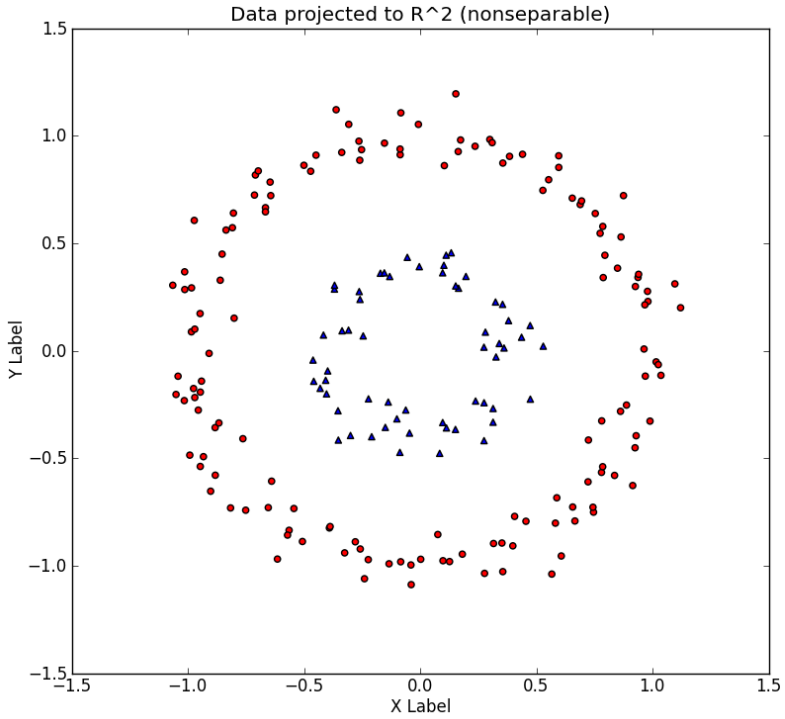
Solve for `w` for the function `wx-b=0` to determine the hyperplane maximizing the distance between two groups.

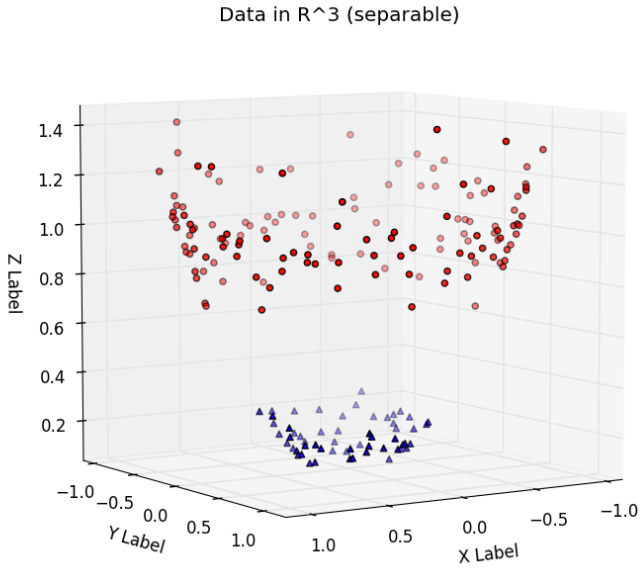


Solvable by finding the perpendicular hyperplane to all three parallel planes and dividing by 2.

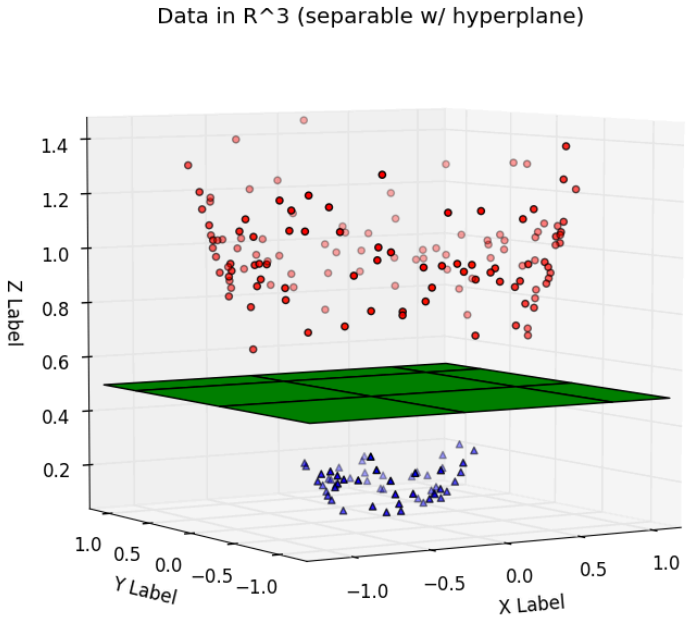
The Kernel Trick

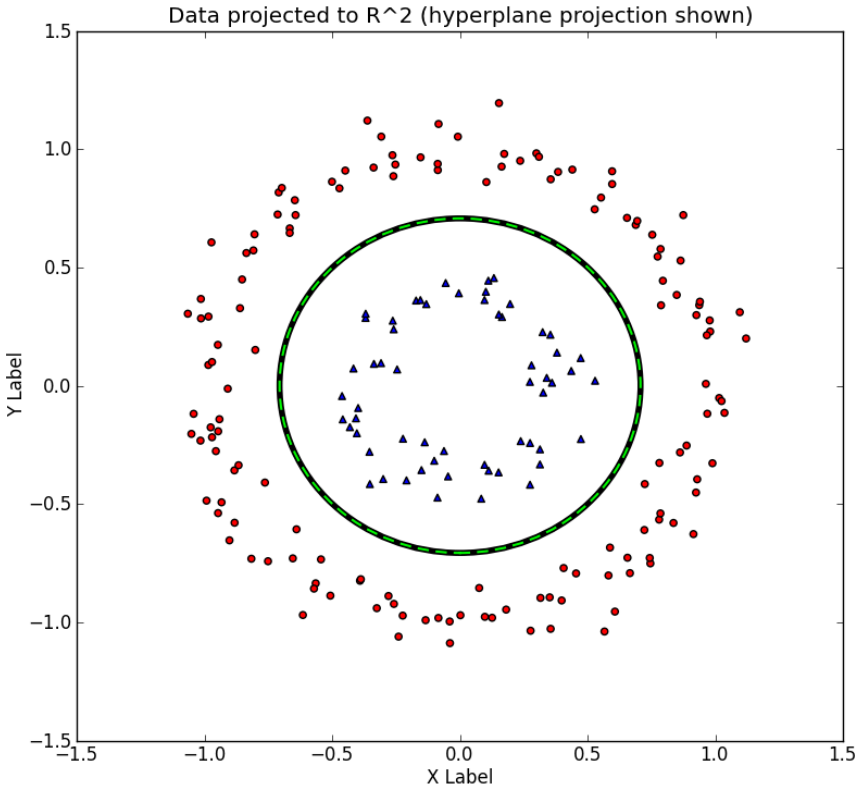
- Transpose data into a higher dimensional space.





The Kernel Trick





Kernel Functions

Homogeneous Polynomial
- d is the degree, any number > 0
- works best in most cases.

Heterogeneous Polynomial
- Add a non-negative, non-zero constant, c
- Increases relevance of higher order features

Radial Basis Function (RBF)
- Can create infinite new dimensions
- Often used more: high-D performance

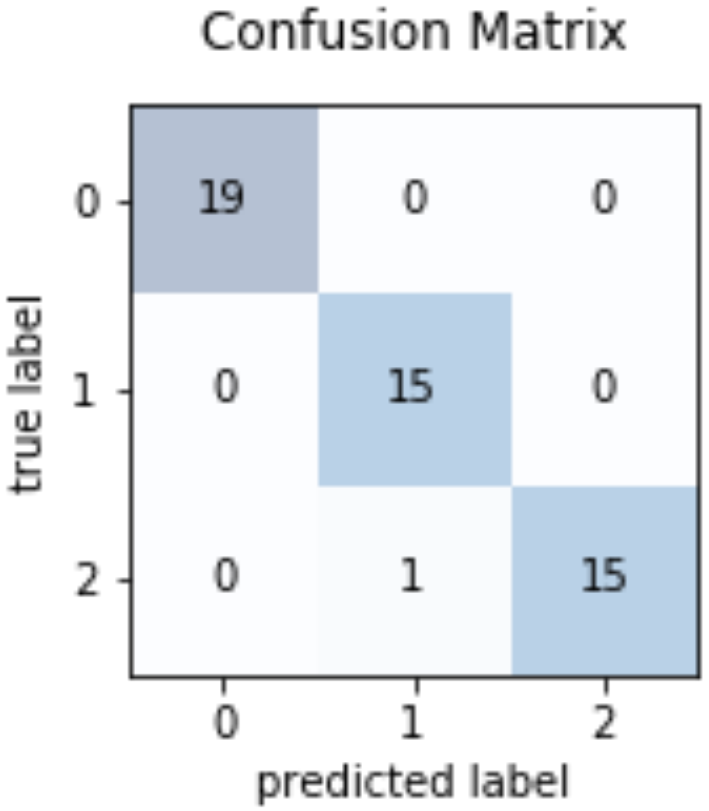
Kernel Functions in Sklearn

- Kernel functions
  - linear
  - poly
  - rbf
  - sigmoid
  - precomputed
  - callable
- If none is given, ‘rbf’ will be used

Lab: Support Vector Machines

Sklearn & MLlib

Confusion Matrices



Model Output

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Standard Confusion Matrix

Naive Bayes

Naive Bayes

- Based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
- Bayes' Theorem: describes the probability of an event, based on prior knowledge of conditions that might be related to the event
  - If cancer is related to age, then you can use Bayes' theorem to more accurately assess the probability that they have cancer, compared to the assessment of the probability of cancer made without knowledge of the person's age.

The Maths of a Conditional Probability

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)},$$

$P(A \mid B)$	A conditional probability: the likelihood of event A occurring given that B is true.
$P(B \mid A)$	A conditional probability: the likelihood of event B occurring given that A is true.
$P(A)$ and $P(B)$	Probabilities of observing A and B independently of each other.

Example: Drug Testing

Drug test is 99% sensitive and 99% specific.
Suppose 0.5% of people are users of the drug.
What's the probability that a randomly selected individual with a positive test is a user?

$$
\begin{aligned}
P(\text{User} \mid +) &= \frac{P(+ \mid \text{User})P(\text{User})}{P(+)} \\
&= \frac{P(+ \mid \text{User})P(\text{User})}{P(+ \mid \text{User})P(\text{User}) + P(+ \mid \text{Non-user})P(\text{Non-user})} \\
&= \frac{0.99 \times 0.005}{0.99 \times 0.005 + 0.01 \times 0.995} \\
&\approx 33.2\%
\end{aligned}
$$

Drug Testing: Interpretation

Even if an individual tests positive, it is more likely that they do not use the drug than that they do. Why? Even though the test appears to be highly accurate, the number of non-users is large compared to the number of users. The number of false positives outweighs the number of true positives.

If 1000 individuals are tested, there are expected to be 995 non-users and 5 users.

- From the 995 non-users, $0.01 \times 995 \simeq 10$ false positives are expected.
- From the 5 users, $0.99 \times 5 \approx 5$ true positives are expected.
- Out of 15 positive results, only 5 (~33%) are genuine.

Classification

Building our model

- Compute the probabilities of all features from a training set
- Compute the probabilities of all classes from training set
- Compute the parameter Z
- Build a truth table

Using our model

- Compute “probabilities” of input vector
- Use truth table, trained probabilities and parameters to compute likelihood of class

Smoothing

What happens if we receive new information?

If the information has a probability = 0, then the independence assumption will cause skew.

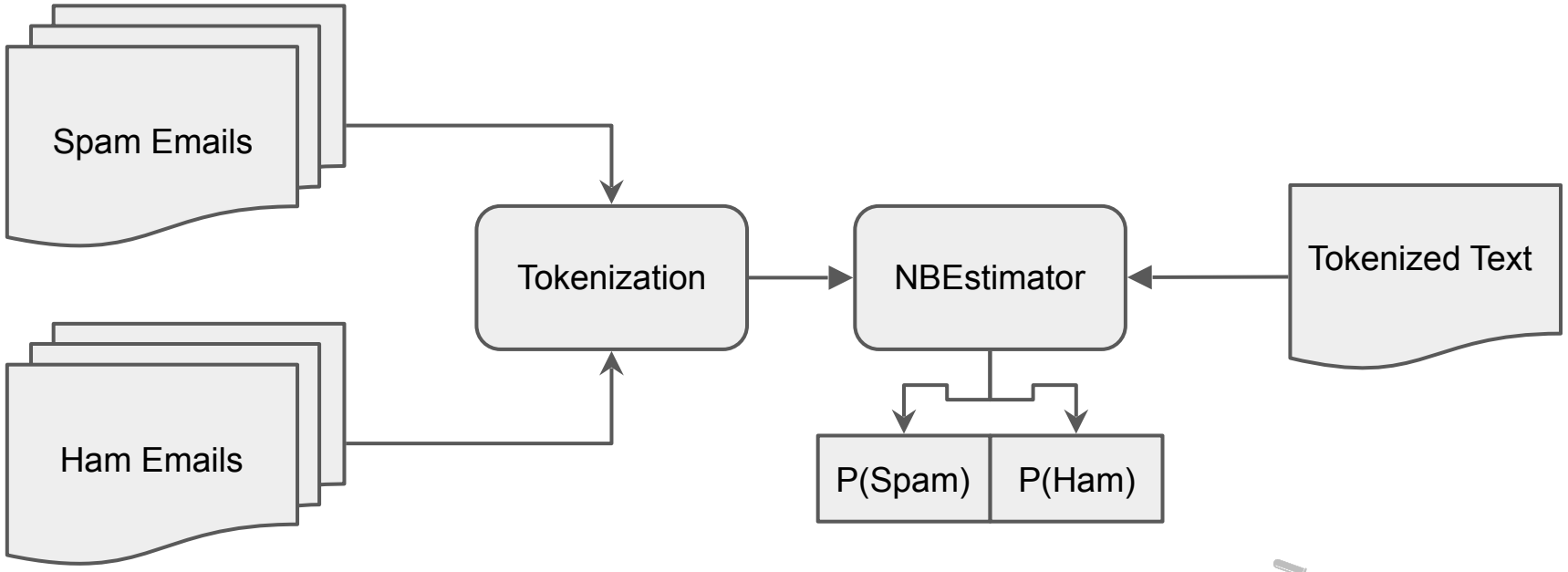
Smoothing: Donating some of the probability density to unknown information.

Easiest smoothing technique: *laplacian (additive)*
Simply give all unknown items a frequency of 1 - then recompute probabilities accordingly.

Text Classification with NB

Spam/Ham is the canonical Naive Bayesian classifier for text and is popular because of it's wide use.

Compute using "bag of words" - each word is a feature.



Lab: Naive Bayes

Sklearn & MLlib

Model Evaluation

Methods of Model Evaluation

- Training and testing on the same data
- Train/test split
- K-fold cross validation
- F1
- Precision
- Recall
- Accuracy
- Confusion Matrix

Training and Testing on the Same Data

- Rewards overly complex models that "overfit" the training data and won't necessarily generalize.

DON'T DO THIS
(REALLY, DON'T)

Train/Test Split

- Split the dataset into two pieces, so that the model can be trained and tested on different data
  - Typically 80/20.
- Better estimate of out-of-sample performance, but still a "high variance" estimate
- Useful due to its speed, simplicity, and flexibility

K-Fold Cross Validation

Assess how model will generalize to independent data set (e.g. data not in the training set).

1. Divide data into training and test splits
2. Fit model on training, predict on test
3. Determine *accuracy*, *precision* and *recall*
4. Repeat *k* times with different splits then average as *F1*

F1 Score

- A measure of a test's accuracy
- Considers both precision and recall of the test to compute the score.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Precision

How many selected items are relevant?

$$precision = \frac{true\ \ positives}{true\ \ positives + false\ \ positives}$$

Recall

How many relevant items are selected?

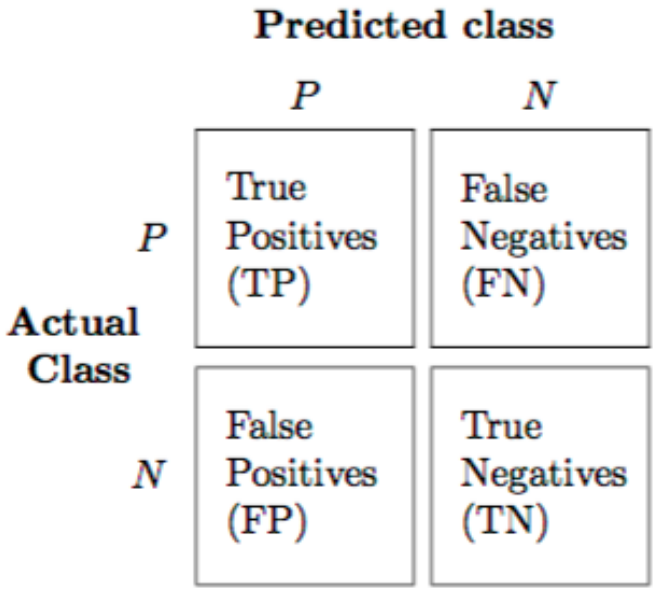
$$recall = \frac{true\ \ positives}{false\ negatives + true\ \ positives}$$

Accuracy

How accurate is the model?

$$accuracy = \frac{true\ \ positives + true\ negatives}{total}$$

Confusion Matrix



Confusion Matrix

```
[[118   12]
 [ 47   15]]
```

n=192	Predicted: 0	Predicted: 1
Actual: 0	118	12
Actual: 1	47	15

0: negative class
1: positive class

- True Positives: correctly predicted a positive (15)
- True Negatives: correctly predicted a negative (118)
- False Positives: incorrectly predicted a positive (12) - Type I Error
- False Negative: incorrectly predicted a negative (47) - Type II Error

What a Confusion Matrix Gives Us

- Accuracy: overall how often is the classifier correct?
- Error: overall, how often is the classifier incorrect?
- False Positive Rate: when the actual value is negative, how often is the prediction incorrect?
- Precision: when a positive value is predicted, how often is the prediction correct?
- Sensitivity: when the actual value is positive, how often is the prediction correct?
- Specificity: when the actual value is negative, how often is the prediction correct?