Support Vector Machines

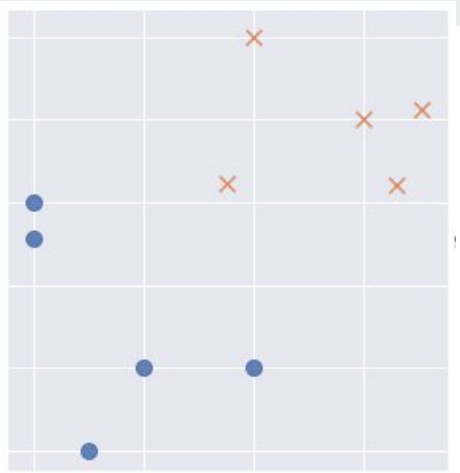
TJ Machine Learning

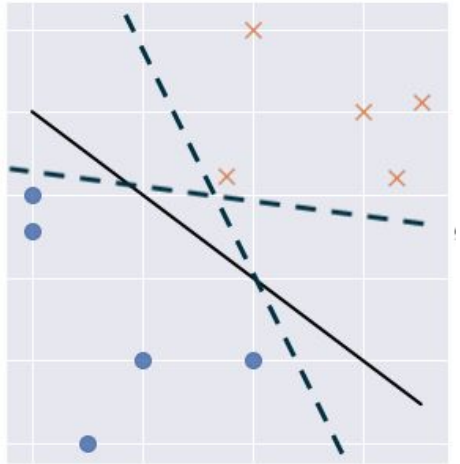
Overview

- Support vector machines (SVMs), in their most basic form, are supervised learning models that solve binary linear classification problems.
- They are commonly modified to separate multiple classes, classify non-linearly separable data, or perform regression analysis

Basics

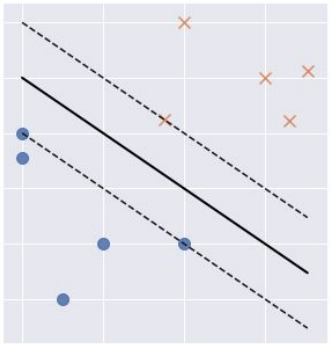
- A binary linear classification problem is one where, given that we have two classes of data, we can draw a line to separate the data into the proper classes.
- As shown below, there are multiple ways to separate the given data into two classes.
- How do we pick the best one?

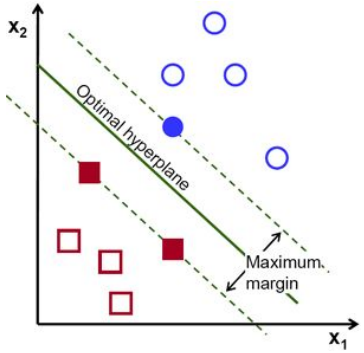




Maximizing the Margin

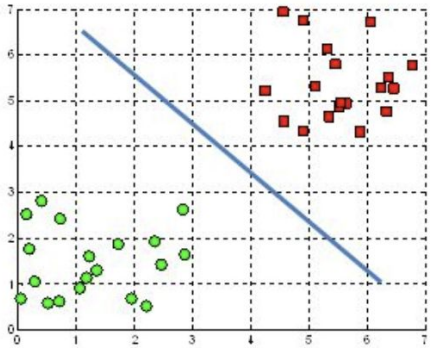
- We want to maximize the margin because our goal is to use the decision boundary to predict where unclassified data points will fall.
- If the decision boundary is too close to either group, our model will perform poorly when given new points.
- By maximizing the margin, we minimize the potential error from generalizing.

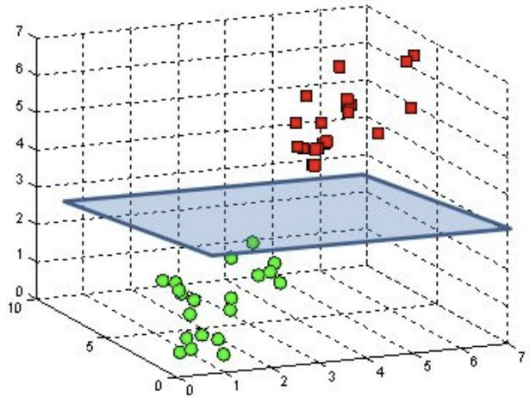




Multiple Dimensions

- When we add input features to an SVM, the decision boundary will necessarily have to increase in dimension with it.
- This is why the margins are referred to as positive and negative hyperplanes.
- Note that the dimension of a hyperplane is one less than the dimension of the feature space.





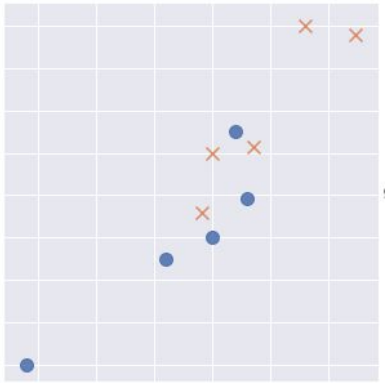
Loss Function

This is the loss function of a basic SVM (w is a vector normal to the hyperplanes), and can be minimized by using Lagrange multipliers. Minimizing this loss function is essentially equivalent to maximizing the margin.

$$\frac{1}{2}|\vec{w}|^2$$

Soft-Margin Classification

In the real world, most datasets won't be perfectly linearly separable like the data shown on the previous slides, and thus the margin can't be minimized by our previous equation. For example, there's no good way to draw a straight line to separate these points:

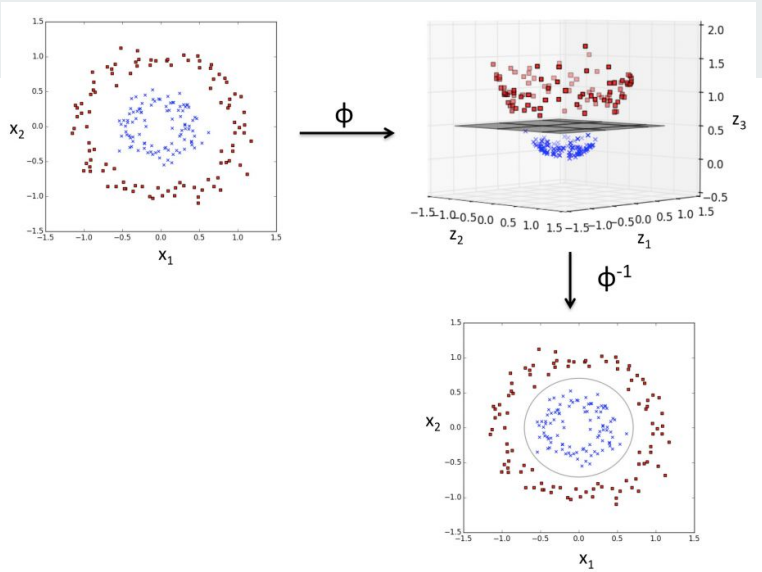


Soft-Margin Classification

To adjust for these cases when the data aren't linearly separable but can almost be separated through the use of a linear boundary, we can adjust our loss function. The new constraint contains a penalty of cost Cξi for any data point that falls within the margin on the correct side of the separating hyperplane (i.e., when $0 < \xi_i \leq 1$), or on the wrong side of the separating hyperplane (i.e., when $\xi_i > 1$).

$$\frac{1}{2}|\vec{w}|^2 + C\sum_i \xi_i$$

$$L = \frac{1}{2}\|w\|^2 + C(\#\ of\ mistakes)$$



Non-Linear Classification

- However, some real world datasets are neither linearly separable nor easily approximated by a linear decision boundary. In this case, even using a soft-margin SVM will fail to be predictive. We can work around this by using a mapping function: $\phi(\vec{x_1}, \vec{x_2}) => (\vec{z_1}, \vec{z_2}, \vec{z_3}) => (\vec{x_1}, \vec{x_2}, \vec{x_1}^2 + \vec{x_2}^2)$ to project our data to a higher dimension, finding a hyperplane, then returning to the original dimension.

Kernels

- Kernels are functions that allow us to calculate a dot product of higher-dimensional points, without actually doing any projection. In other words, functions of this form exist:

$$K(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$$

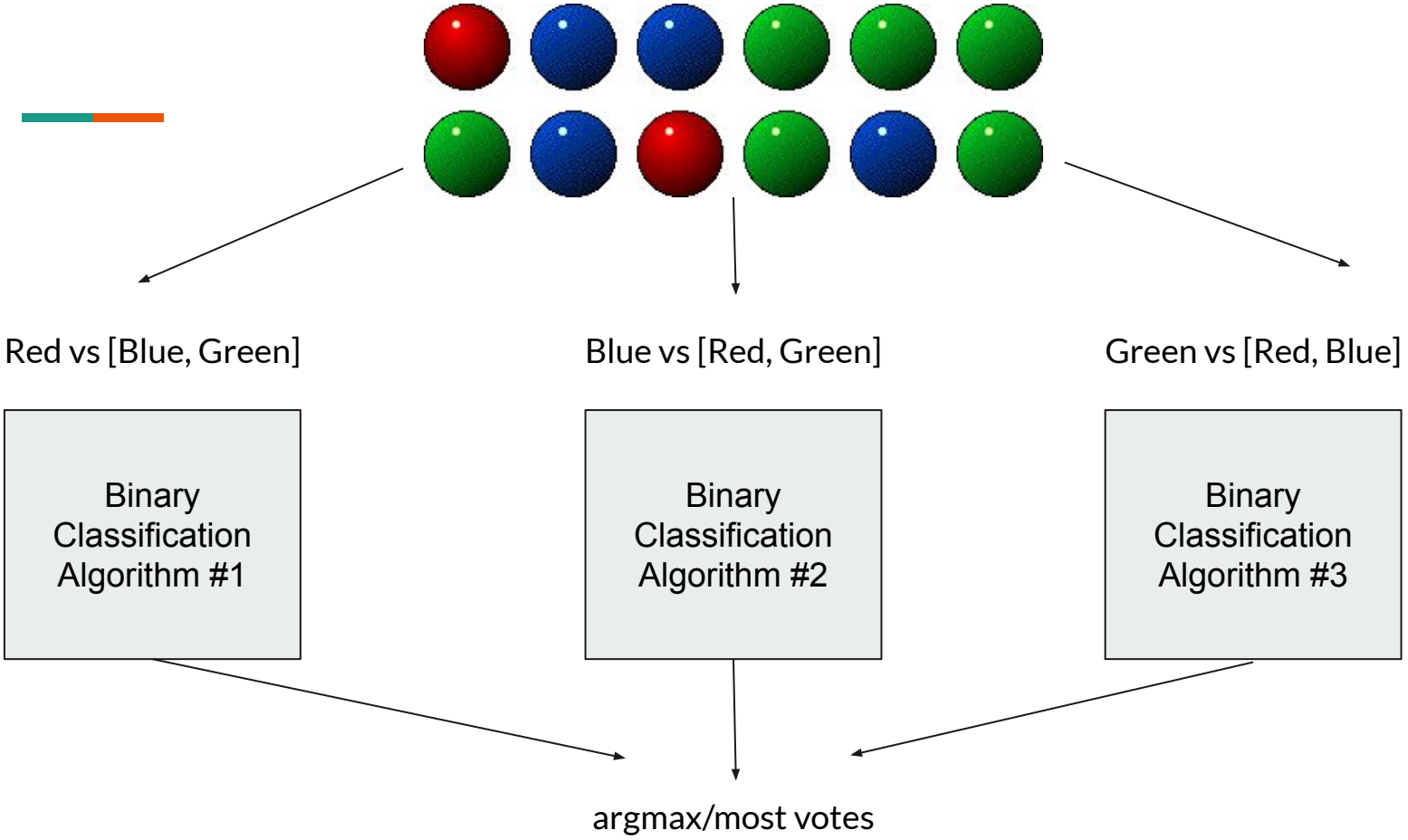
- Here we only need the inputs ~xi and ~xj to calculate the dot product φ(~xi)·

Non-Binary Classification

- Although SVMs are designed for binary data, they can be extended to multiple classes.
- There are two common ways of doing this:
- One-vs-rest (OvR):
    - OvR attempts to separate one class at a time from the rest of data, treating it as a series of binary classification problems, and so will create n models from n classes. This is computationally simple, but risks oversimplification.
- One-vs-one (OvO):
    - OvO makes a model to separate every class from every other class, creating n (n−1) / 2 models from n classes

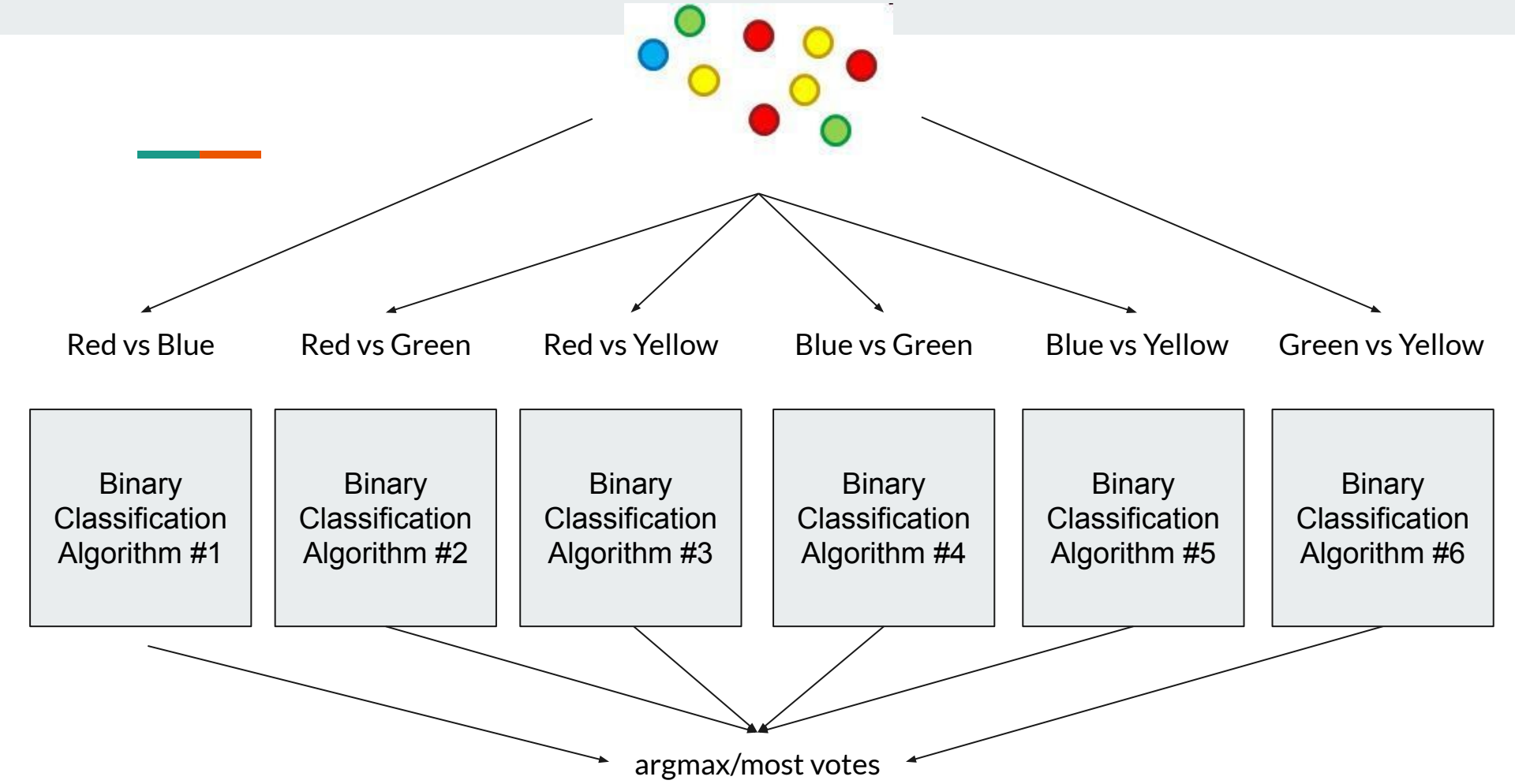
One vs Rest (OvR)

- OvR involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.


Red vs [Blue, Green]
Blue vs [Red, Green]
Green vs [Red, Blue]
Binary Classification Algorithm #1
Binary Classification Algorithm #2
Binary Classification Algorithm #3
argmax/most votes

One vs One (OvO)

- Like one-vs-rest, one-vs-one splits a multi-class classification dataset into binary classification problems. Unlike one-vs-rest that splits it into one binary dataset for each class, the one-vs-one approach splits the dataset into one dataset for each class versus every other class.
- The formula for calculating the number of binary datasets, and in turn, models, is as follows:
    - (NumClasses * (NumClasses – 1)) / 2
- A downside of this approach is that it can require more models than classes. This could be an issue for large datasets (e.g. millions of rows) or very large numbers of classes.


Red vs Blue
Red vs Green
Red vs Yellow
Blue vs Green
Blue vs Yellow
Green vs Yellow
Binary Classification Algorithm #1
Binary Classification Algorithm #2
Binary Classification Algorithm #3
Binary Classification Algorithm #4
Binary Classification Algorithm #5
Binary Classification Algorithm #6
argmax/most votes

SVMs Pros and Cons

- Pros:
  - SVMs can have high accuracy
  - Work well on smaller cleaner datasets
  - Can be more efficient because they uses a subset of training points
- Cons
  - SVMs are not suited to larger datasets as the training time with SVMs can be high
  - Less effective on noisier datasets with overlapping classes

SVM Uses

SVMs are used for text classification tasks such as category assignment, detecting spam and sentiment analysis. They are also commonly used for image recognition challenges, performing particularly well in aspect-based recognition and color-based classification.