FLATIRON SCHOOL

Representing Similarities with Instance-Based Algorithms.

hosted by KASH

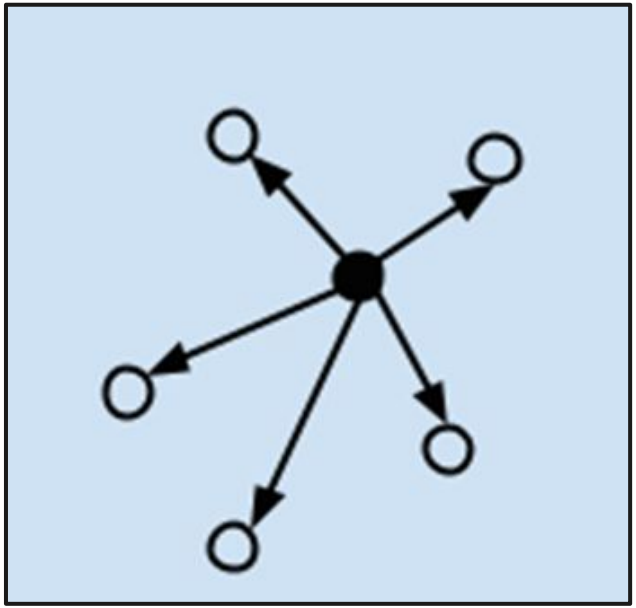
CRITICAL QUESTION

How can we construct and apply algorithms that can analyze single instances of data to determine the identity of unknown data points?

CRITICAL QUESTION

How can we utilize k-Nearest Neighbors and Support Vector Machine algorithms to classify new data?

AN INTRODUCTION TO INSTANCE-BASED ALGORITHMS



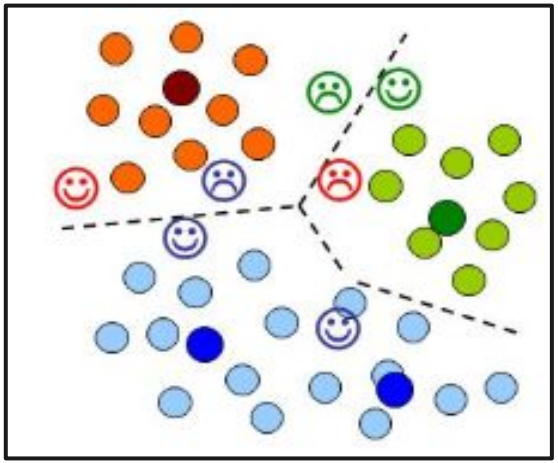
Instance-based algorithms are some of the simpler category of predictive models we have at our disposal as data scientists.

Rather than performing exhaustive explicit generalization across an entire dataset, they compare **new and unlabeled data points** with **instances** that they have analyzed in training, which have been **stored in memory**.

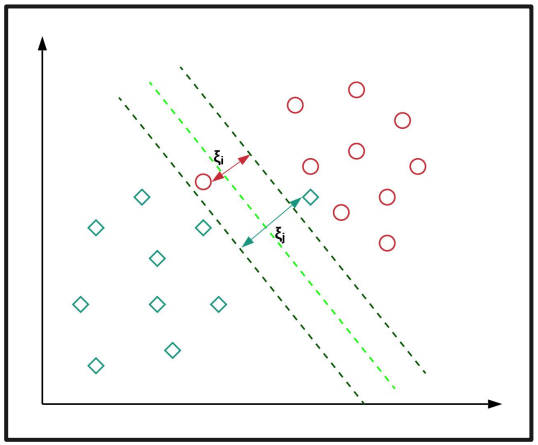
APPLYING INSTANCE-BASED ALGORITHMS TO TARGET DATA

The key is that instance-based algorithms are based on the assumption that **instances nearby to an unknown point are more likely to share feature similarity than distant points** – hence, by examining the labels of nearby points, **we can classify an unknown sample using the classes around it**.

This is reliant on plenty of **critical assumptions**, but is generally useful against **small, reliably structured datasets**.



UTILIZING SIMILARITY CALCULATIONS FOR TARGET PREDICTIONS



The “*secret sauce*” of these algorithms are twofold:

- How the algorithm **calculates distance and quantitative similarity** between two-or-more points.
- How the algorithm constructs the **boundary/barrier that separates classes** for future predictions.

The two most popular instance-based algorithms perform these steps in wildly different ways: **kNNs** and **SVMs**.

//

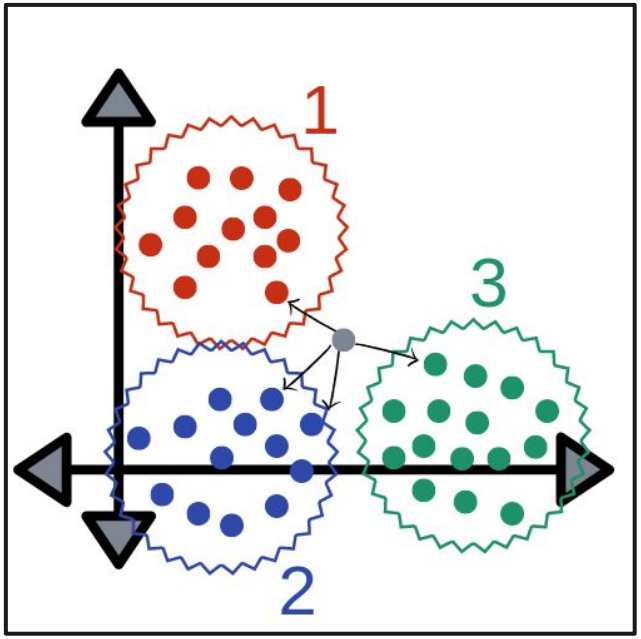
WRITTEN IDEATION

WHAT **DIFFERENT WAYS** ARE THERE TO **CALCULATE THE DISTANCE BETWEEN TWO POINTS**?

CODING WALKTHROUGH

Applying Popular Instance-Based Algorithms to the Wisconsin Breast Cancer Dataset

KNNs: THE K-NEAREST-NEIGHBORS ALGORITHM



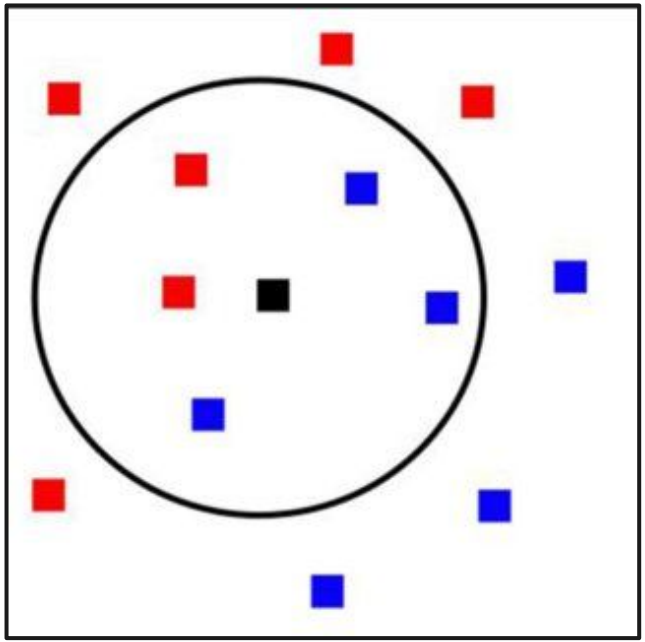
k-Nearest Neighbors is one of the simplest and most effective machine learning classifiers in the field, effective for classification and regression.

kNNs measure the **distance between test data points and each sample of training data** using various distance calculation methods and **sort the distance-datum pairs** in order to **select the top *k* data points** and **assign the most frequent class** to the unknown sample as a prediction.

AN ILLUSTRATIVE APPROACH TO UNDERSTANDING KNNs

Imagine you see a new person sitting at one of two lunch tables at a *superhero convention* – one of the tables is for members of the **Avengers** while the other is for members of the **Justice League**.

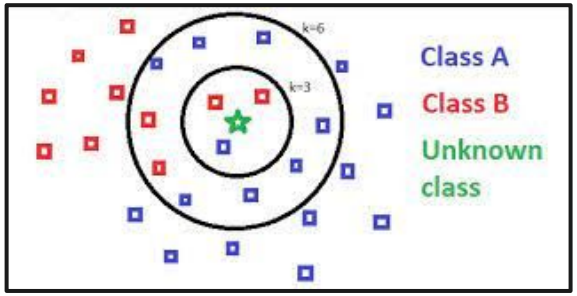
Using elementary logic, one could assume that **if the new person is sitting at the Avengers table**, it stands to reason that **they're an Avenger**; similarly, **if they're sitting with the Justice League**, it stands to reason that **they're a new member of the Justice League**.



A CONCEPTUAL APPROACH TO UNDERSTANDING KNNs

kNNs make that same "***distance-equals-similarity***" assumption to dictate how classes are evaluated and applied to new data – by **examining some *k* sorted training points** and **analyzing what the most frequent label is**, the algorithm can simply **apply that label to the unknown point**.

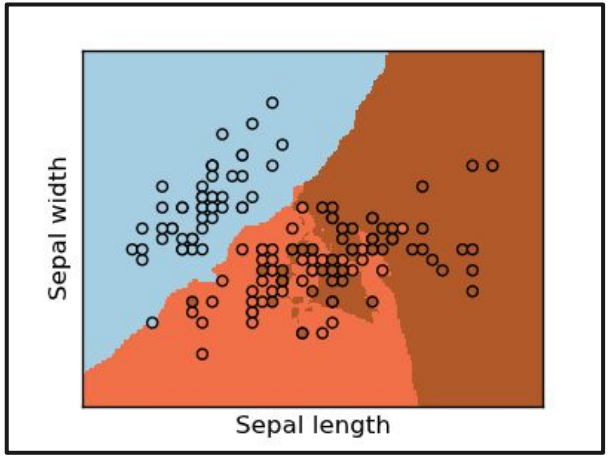
The precise distance calculation function, the specific value of ***k***, and other mutable factors fall into the category of what we call **"hyperparameters"**: aspects of the model we can alter to influence our overall **accuracy** and **reliability**.



A TECHNICAL APPROACH TO UNDERSTANDING KNNs

Programmatically, kNNs are **very easy and straightforward to design** – so much so that they're often considered barely machine learning algorithms as **they perform incredibly rudimentary "learning" across the data**.

Rather than extracting heuristic patterns during the training process, kNNs apply most their learning during testing by simply **sorting, ranking, and selecting the top *k* points**; this earns the model's moniker as a **"shallow learner"**.

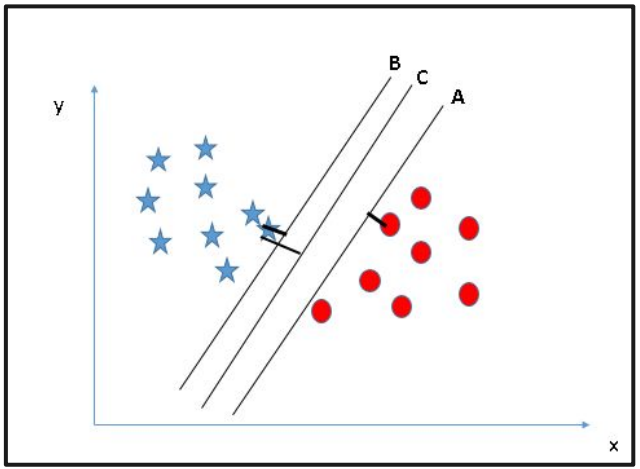


//

WRITTEN IDEATION

WHY MAY IT BE A PROBLEM THAT **KNNs INHERIT MOST ALGORITHMIC COMPLEXITY** DURING **TESTING** RATHER THAN **TRAINING**?

SVMs: THE SUPPORT VECTOR MACHINE ALGORITHM



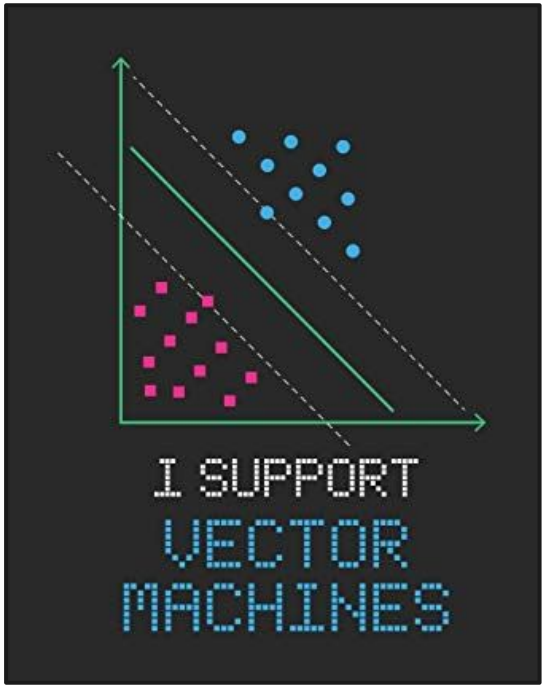
SVMs take the concept of instance-based learning to the next level by applying high-level linear algebra and calculus in order to optimize both **what specific points are used for distance calculation** as well as **how the boundaries are created between different predicted class regions of data**.

In this way, this algorithm is **much more robust and powerful** against complex and unruly data.

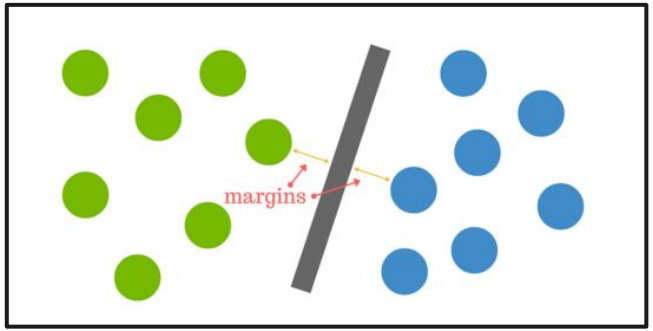
AN ILLUSTRATIVE APPROACH TO UNDERSTANDING SVMs

Imagine you're walking through a **dangerous mountain path with lava on one side and monsters on the other** – you're obviously not trying to get cooked or eaten, but you have to navigate through the mountain path to escape!

You don't have to know *how much lava is on one side* or *how many monsters are on the mountain*, you just have to know **where the closest lava/monsters are** and walk in a way that **creates the most distance between you and them**!



A CONCEPTUAL APPROACH TO UNDERSTANDING SVMs



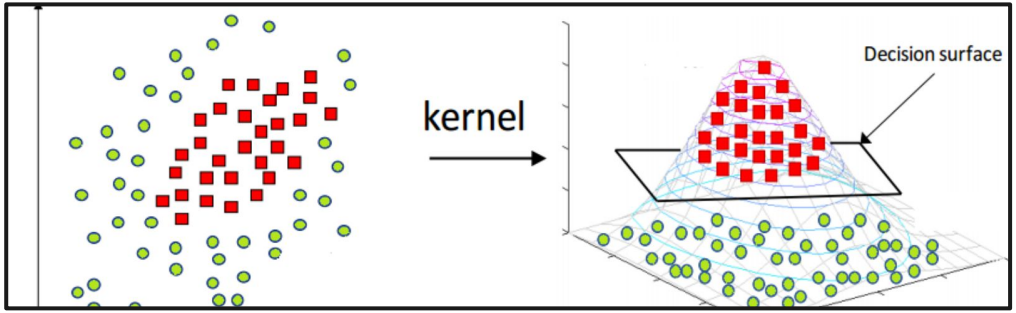
SVMs do this exact tactic by performing a complex *two-step calculation* across the training data; first, they analyze the data to find the **“support vectors”**, which are the ***n*-closest differently labeled points in our dataset**.

Next, they apply linear algebra and calculus to determine the **“best fit margin”**, which is the **optimal decision boundary that equally separates the dataset by fitting between the support vectors as best as possible**.

A TECHNICAL APPROACH TO UNDERSTANDING SVMs

The true power of SVMs is that it's able to mathematically determine strong margins and boundaries for **"linearly non-separable data"**, or data that doesn't appear to have a great line or curve to split classes.

It does so through some mathematical tricky called **"kerneling"**: the algorithm can **"look at data" through different dimensional perspectives called "hyperplanes"**, which allow for circular, exponential, and other nonlinear decision boundary formations.



RECOMMENDED RESOURCE

Jake van der Plas's In-Depth Guide to Working with Support Vector Machines

//

RECOMMENDED RESOURCE

A Guide to Constructing a Custom k-Nearest Neighbors Algorithm from Scratch in Python

THANK YOU FOR YOUR TIME!

ADVANCED LEARNING

Jason Brownlee's Guide on Implementing Learning Vector Quantization Algorithms

ADVANCED LEARNING

Xavier Bourret Sicotte's Blog on Mathematically Constructing Locally Weighted Learners for Linear Regression