LINEAR MODELS FOR NLP

STARTER GUIDE

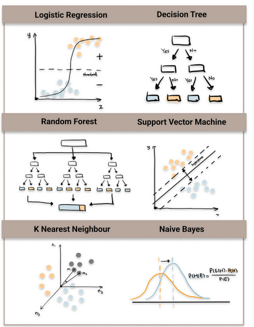


illustration from www.visual-design.net/

swipe right →

What are linear models?

These modules assume that the data is linearly seperable.

In simpler words, they work only if the data can be seperated by a hyper-plane (or a line).

These models capture the linear relationships between data points.

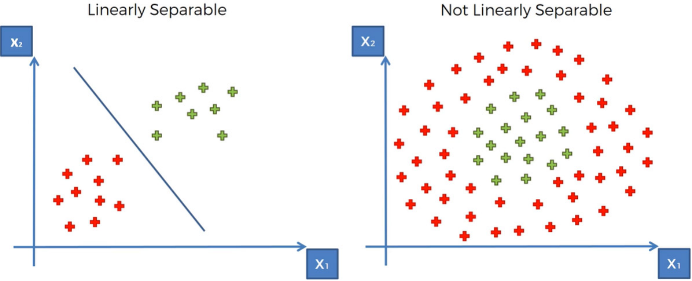
Linear operations are reversible. Hence, linear models are explainable.

They are interpretible, simple, and robust.

70% of corporate machine learning models are based on linear models. The edge cases are dealt with non-linear NN's.

swipe right →

Linear VS Non-linear


Linearly Separable

Not Linearly Separable

Figure by https://medium.com/@suvigya2001

swipe right →

Linear Regression

Predicts a continuous variable based on linear relationships between the features.

Formula : x1,x2,x3 are the independent variable and y is dependent variable that is predicted.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

β_0 is the y-interecept, $\beta_1, \beta_2, \ldots, \beta_n$ are directional gradients for the regression hyper-plane, and is ϵ he error term,

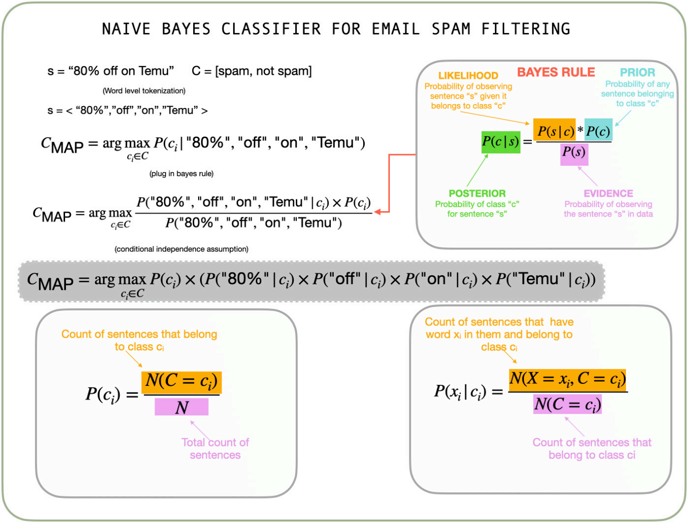
You know X, your goal is to calculate y, and you change the beta values such that the difference between y_true and y_pred is low.

Simple, interpretible and flexible.

swipe right →

Naive Bayes Classifier

Uses Bayes rule to calculate the class of a sentence based on it's likelihood and the class prior.



NAIVE BAYES CLASSIFIER FOR EMAIL SPAM FILTERING

s = “80% off on Temu”   C = [spam, not spam]

(Word level tokenization)

s = < “80%”,”off”,”on”,”Temu” >

$$C_{\text{MAP}} = \arg\max_{c_i \in C} P(c_i \,|\, \text{"80\%", "off", "on", "Temu"})$$

(plug in bayes rule)

$$C_{\text{MAP}} = \arg\max_{c_i \in C} \frac{P(\text{"80\%", "off", "on", "Temu"} \,|\, c_i) \times P(c_i)}{P(\text{"80\%", "off", "on", "Temu"})}$$

(conditional independence assumption)

$$C_{\text{MAP}} = \arg\max_{c_i \in C} P(c_i) \times (P(\text{"80\%"} \,|\, c_i) \times P(\text{"off"} \,|\, c_i) \times P(\text{"on"} \,|\, c_i) \times P(\text{"Temu"} \,|\, c_i))$$

BAYES RULE

LIKELIHOOD
Probability of observing sentence “s” given it belongs to class “c”

PRIOR
Probability of any sentence belonging to class “c”

$$P(c\,|\,s) = \frac{P(s\,|\,c) * P(c)}{P(s)}$$

POSTERIOR
Probability of class “c” for sentence “s”

EVIDENCE
Probability of observing the sentence “s” in data

Count of sentences that belong to class c_i

$$P(c_i) = \frac{N(C = c_i)}{N}$$

Total count of sentences

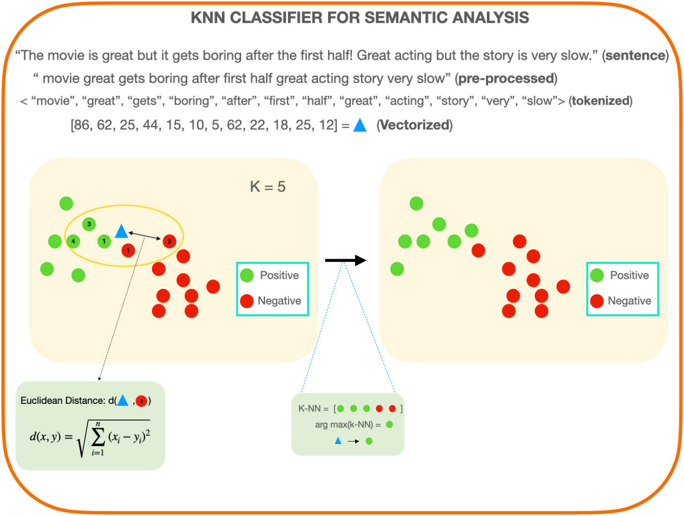
Count of sentences that have word x_i in them and belong to class c_i

$$P(x_i \,|\, c_i) = \frac{N(X = x_i, C = c_i)}{N(C = c_i)}$$

Count of sentences that belong to class ci

K-Nearest-Neighborhood Classifier

Assigns the class of "K" nearest neighbors to a test sample.



KNN CLASSIFIER FOR SEMANTIC ANALYSIS

“The movie is great but it gets boring after the first half! Great acting but the story is very slow.” (**sentence**)

“ movie great gets boring after first half great acting story very slow” (**pre-processed**)

< “movie”, “great”, “gets”, “boring”, “after”, “first”, “half”, “great”, “acting”, “story”, “very”, “slow”> (**tokenized**)

[86, 62, 25, 44, 15, 10, 5, 62, 22, 18, 25, 12] = ▲ (**Vectorized**)

Euclidean Distance: d(▲ , 🔴2)

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

K-NN = [🟢 🟢 🟢 🔴 🔴]

arg max(k-NN) = 🟢

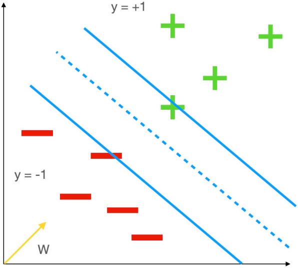
▲ → 🟢

Support Vector Machines (SVMs)

SVMs maximize the margin around the seperating hyper-plane.

All the models till now, find **a** seperating hyper-plane. SVMS finds the **optimal** hyper-plane.

The goal is to maximize the distance between the hyper-plane and the data points closest to it known as **support vectors**.



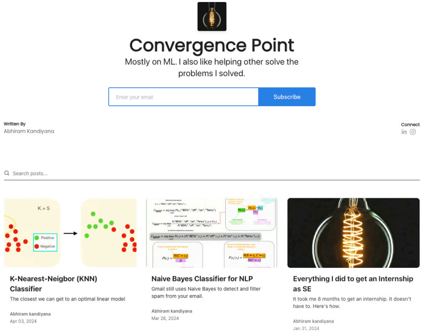
The red and green points touching the lines on either side are the support vectors.

Dive Deep

If you want to dive deep into these models, check out my blog.

I started the NLP series where I write blogs on different models starting from linear models to non-linear RNNs, LSTMs and **Transformers**.

LINK IN COMMENTS



Convergence Point

Mostly on ML. I also like helping other solve the problems I solved.

Written By
Abhiram Kandiyana

K-Nearest-Neigbor (KNN) Classifier

The closest we can get to an optimal linear model

Abhiram kandiyana
Apr 03, 2024

Naive Bayes Classifier for NLP

Gmail still uses Naive Bayes to detect and filter spam from your email.

Abhiram kandiyana
Mar 26, 2024

Everything I did to get an Internship as SE

It took me 8 months to get an internship. It doesn't have to. Here's how.

Abhiram kandiyana
Jan 31, 2024

THANK YOU FOR READING TILL THE END

SEE YOU IN THE NEXT ONE!

FOLLOW ME FOR MORE CONTENT LIKE THIS