Naïve Bayes & SVM

CLASSIFICATION | GAUSSIAN | BERNOULLI | KERNEL

Bayes’ Theorem

- How does computers learn about humans?
- An internet search for "movie automatic shoe laces" brings up "Back to the future“.
- Has the search engine watched the movie?
- No, but it knows from lots of other searches what people are **probably** looking for.
- And it calculates that probability using Bayes' Theorem.

Bayes' Theorem

- Bayes' Theorem is a way of finding a probability when we know certain other probabilities.
- The formula is:

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

- Which tells us: how often A happens given than B happens, P(A|B)
- When we know: how often B happens given that A happens, P(B|A)
- and how likely A is to happen independently, P(A)
- and how likely B is to happen independently, P(B)

Example

- Shall we go for a picnic?
- You are planning a picnic today, but the morning is cloudy
- Oh no! 50% of all rainy days start off cloudy!
- But cloudy mornings are common (about 40% of days start cloudy)
- And this is usually a dry month (only 3 of 30 days tend to be rainy, or 10%)
- **What is the chance of rain during the day?**

Example

- We will use Rain to mean rain during the day, and Cloud to mean cloudy morning.
- The chance of Rain given Cloud is written P(Rain|Cloud)
- So let's put that in the formula:

$$P(Rain|Cloud) = \frac{P(Rain)\ P(Cloud|Rain)}{P(Cloud)}$$

Example

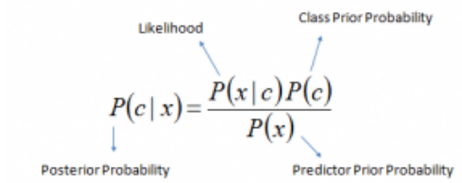
- P(Rain) is Probability of Rain = 10%
- P(Cloud|Rain) is Probability of Cloud, given that Rain happens = 50%
- P(Cloud) is Probability of Cloud = 40%

$$P(\text{Rain}|\text{Cloud}) = \frac{0.1 \times 0.5}{0.4} = .125$$

- Or a 12.5% chance of rain. Not too bad, let's have a picnic!

Summary

- Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c).



$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- $P(c|x)$ is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

Naïve Bayes

- Robust classifier
- Not affected by irrelevant features
- Assumption – Independent Variables are not correlated, hence the name
- Doesn’t care about distribution (except Gaussian NB), skewness, bias in data

Example

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14$ $P(\text{Play}=No) = 5/14$

Testing Phase

- Given a new instance,

  $\mathbf{x}'$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)

- Look up tables

P(Outlook=*Sunny*|Play=*Yes*) = 2/9	P(Outlook=*Sunny*|Play=*No*) = 3/5
P(Temperature=*Cool*|Play=*Yes*) = 3/9	P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*Yes*) = 3/9	P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=*Strong*|Play=*Yes*) = 3/9	P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*Yes*) = 9/14	P(Play=*No*) = 5/14

- MAP rule

P(*Yes*|$\mathbf{x}'$): [P(*Sunny*|Yes)P(*Cool*|*Yes*)P(*High*|*Yes*)P(*Strong*|*Yes*)]P(Play=*Yes*) = 0.0053

P(*No*|$\mathbf{x}'$): [P(*Sunny*|No) P(*Cool*|No)P(*High*|*No*)P(*Strong*|*No*)]P(Play=*No*) = 0.0206

Given the fact P(*Yes*|$\mathbf{x}'$) < P(*No*|$\mathbf{x}'$), we label $\mathbf{x}'$ to be "*No*".

MAP Rule

- MAP = Maximum A Posteriori
- Maximum a Posteriori or MAP for short is a Bayesian-based approach to estimating a distribution and model parameters that best explain an observed dataset.
- Maximum a Posteriori estimation is a probabilistic framework for solving the problem of density estimation.
- MAP involves calculating a conditional probability of observing the data given a model weighted by a prior probability or belief about the model.
- MAP provides an alternate probability framework to maximum likelihood estimation for machine learning.

The Zero Frequency Problem

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as Zero Frequency.
- To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.

Types of NB

- **Gaussian:** It is used in classification and it assumes that features follow a normal distribution.

- **Multinomial:** It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".

- **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

Advantages

- Naïve Bayes based on the independence assumption
  - Training is very easy and fast; just requiring considering each attribute in each class separately
  - Test is straightforward; just looking up tables or calculating conditional probabilities with normal distributions
  - Works well with less data!
- A popular generative model
  - Performance competitive to most of state-of-the-art classifiers even in presence of violating independence assumption
  - Many successful applications, e.g., spam mail filtering
  - A good candidate of a base learner in ensemble learning

Applications

- Text Classification
- Spam Filtering
- Hybrid Recommender System
  - Recommender Systems apply machine learning and data mining techniques for filtering unseen information and can predict whether a user would like a given resource
- Online Application
  - Simple Emotion Modeling

Text Classification

- Learning which articles are of interest
- Classify web pages by topic
- Information extraction
- Internet filters

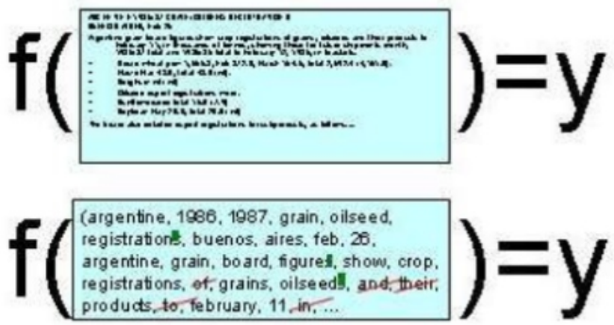
Examples of Text Classification

- CLASSES=BINARY
  - "spam" / "not spam"
- CLASSES =TOPICS
  - "finance" / "sports" / "politics"
- CLASSES =OPINION
  - "like" / "hate" / "neutral"
- CLASSES =TOPICS
  - "AI" / "Theory" / "Graphics"
- CLASSES =AUTHOR
  - "Shakespeare" / "Marlowe" / "Ben Jonson"

Naïve Bayes Approach

- Build the Vocabulary as the list of all distinct words that appear in all the documents of the training set.
- Remove stop words and markings
- The words in the vocabulary become the attributes, assuming that classification is independent of the positions of the words
- Each document in the training set becomes a record with frequencies for each word in the Vocabulary.
- Train the classifier based on the training data set, by computing the prior probabilities for each class and attributes.
- Evaluate the results on Test data

Representing Text: List of Words



- Common Refinements: Remove Stop Words, Symbols

Text Classification

$$\hat{P}(t|c) = \frac{T_{ct}+1}{\sum_{t' \in V}(T_{ct'}+1)} = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'}) + B'}$$

- Tct – Number of particular word in particular class
- Tct' – Number of total words in particular class
- B' – Number of distinct words in all class

Example

► **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$\begin{aligned} &\hat{P}(\text{Chinese}|c) \\ \hat{P}(\text{Tokyo}|c) &= \hat{P}(\text{Japan}|c) \\ &\hat{P}(\text{Chinese}|\overline{c}) \\ \hat{P}(\text{Tokyo}|\overline{c}) &= \hat{P}(\text{Japan}|\overline{c}) \end{aligned}$$

Example

$$P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c)$$

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003.$$
$$\hat{P}(\overline{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.$$

- Probability of Yes Class is more than that of No Class, Hence this document will go into Yes Class.

Naïve Bayes Results

- F1 score should be as close to 1 as possible
- Kappa Value (cohen_kappa_score) should be at least 0.9
- Precision and Recall
- Accuracy via confusion matrix

Tips to improve NB model

- If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.
- If test data set has zero frequency issue, apply smoothing techniques “Laplace Smoothing” to predict the class of test data set.
- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.
- Naive Bayes classifiers has limited options for parameter tuning like alpha=1 for smoothing, fit_prior=[True|False] to learn class prior probabilities or not and some other options. Focus on your pre-processing of data and the feature selection.
- You might think to apply some *classifier combination technique like* ensembling, bagging and boosting but these methods would not help. Actually, “ensembling, boosting, bagging” won’t help since their purpose is to reduce variance. Naive Bayes has no variance to minimize.

Hands on

- https://github.com/2796gaurav/Naive-bayes-explained/tree/master/Naive%20bayes
- https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn
- https://chrisalbon.com/machine_learning/naive_bayes/multinomial_naive_bayes_classifier/
- https://towardsdatascience.com/comparing-a-variety-of-naive-bayes-classification-algorithms-fc5fa298379e

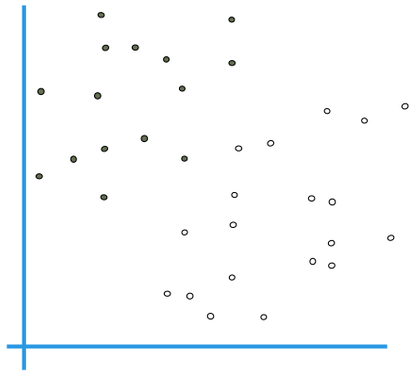
Good to know…

- Market Basket Analysis – Association Rule Mining
- Simple but very useful algorithm – Apriori.
- Key Concepts:
- Confidence - *It signifies the likelihood of item Y being purchased when item X is purchased.*
- *Lift - This signifies the likelihood of the item Y being purchased when item X is purchased while taking into account the popularity of Y.*

- https://medium.com/@notesharsha/market-basket-analysis-sneaky-psychology-of-supermarkets-simple-guide-using-python-eacfd33cc882
- https://intellipaat.com/blog/data-science-apriori-algorithm/
- https://pbpython.com/market-basket-analysis.html

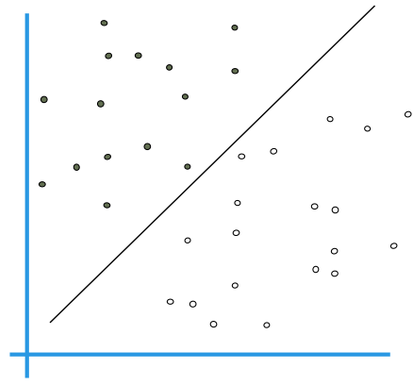
Support Vector Machines

- Say your dataset looks like this –



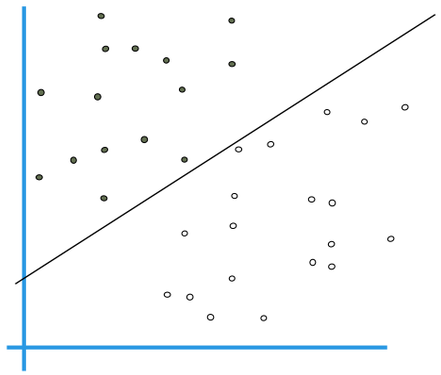
Support Vector Machines

- How can we separate this?



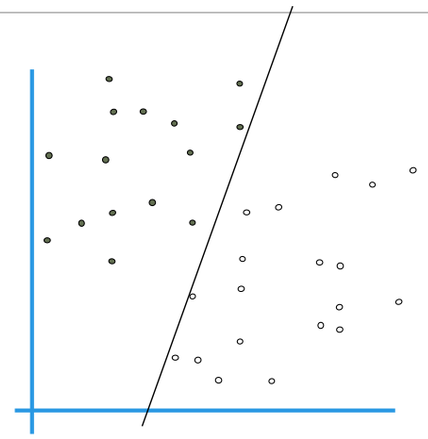
Support Vector Machines

- How about this?



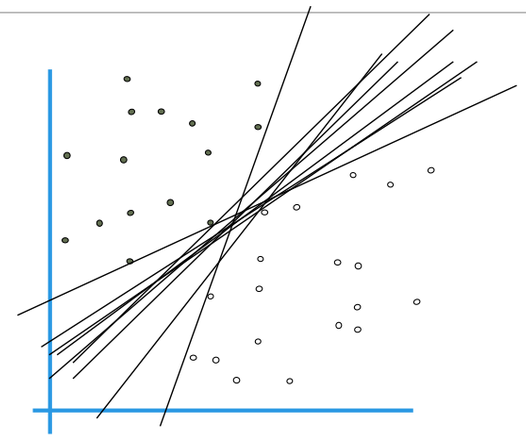
Support Vector Machines

- Or this?



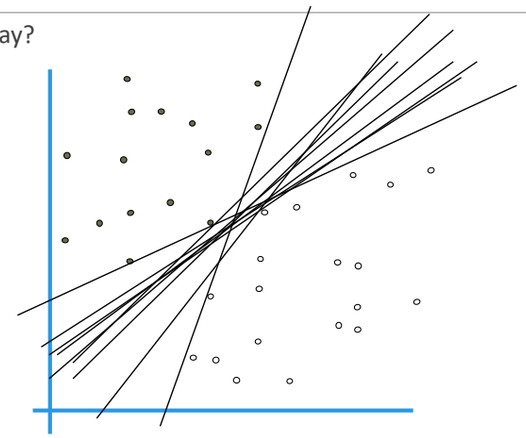
Support Vector Machines

- Infinite ways!



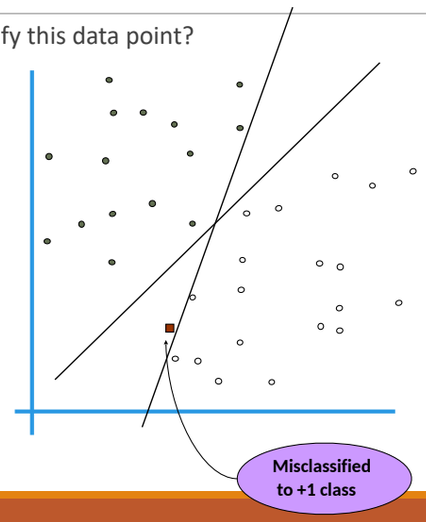
Support Vector Machines

- Which is the best way?



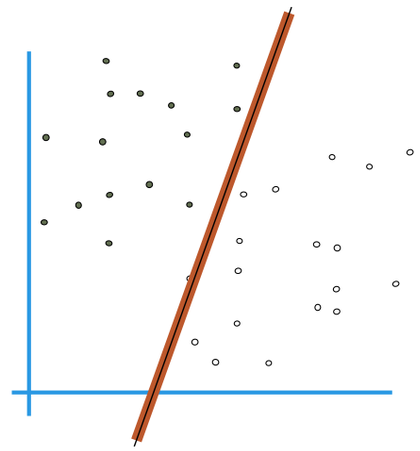
Support Vector Machines

- How would you classify this data point?



Support Vector Machines

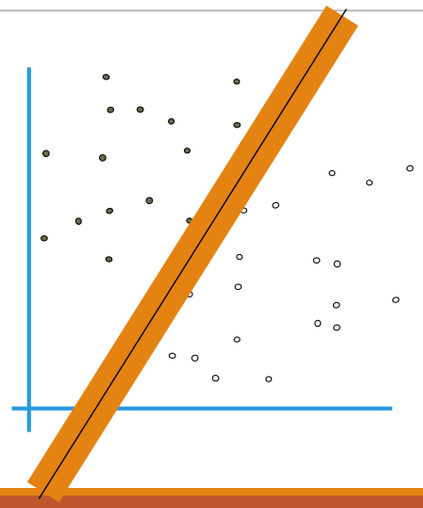
➢ Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a data point.



Support Vector Machines

➢Maximum Margin

1. Maximizing the margin is good
2. Implies that only support vectors are important; other training examples are ignorable.

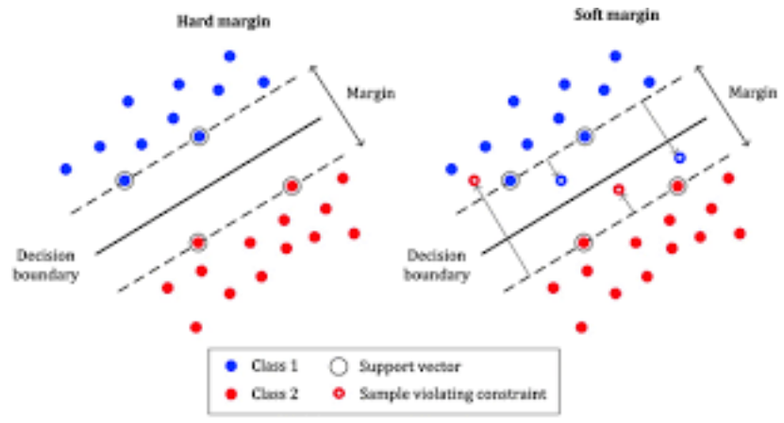


Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (called an LSVM)

Hard Margin Vs. Soft Margin

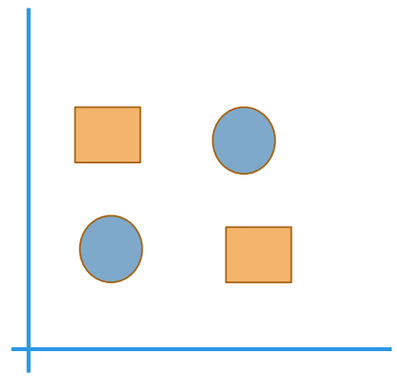


Hard Margin Vs. Soft Margin

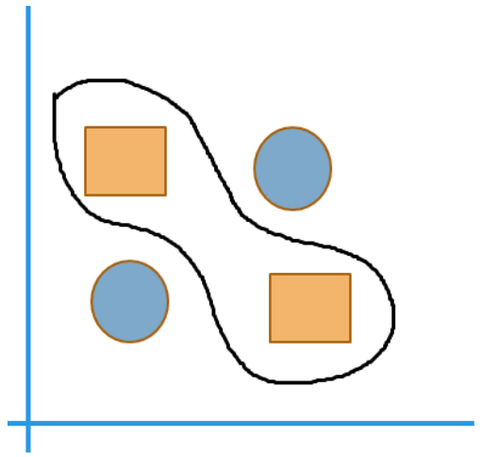
- **Hard margin SVM** can work only when data is completely linearly separable without any errors (noise or outliers).
- In case of errors either the **margin** is smaller or **hard margin SVM** fails.
- On the other hand **soft margin SVM** solves this problem by introducing slack variables.

Kernels

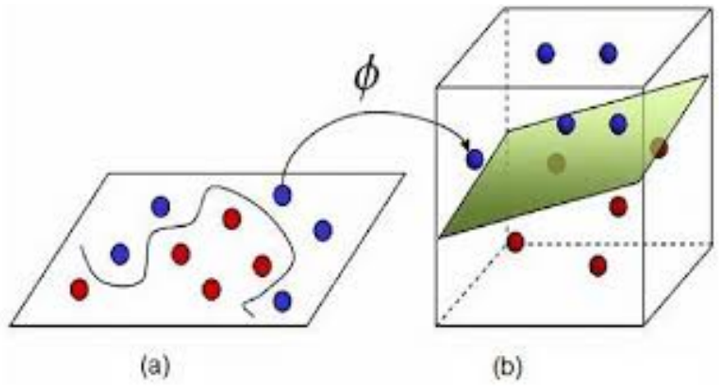
- Feature Projections to higher dimension
- What??
- Say your data is like this –
- How can you classify?



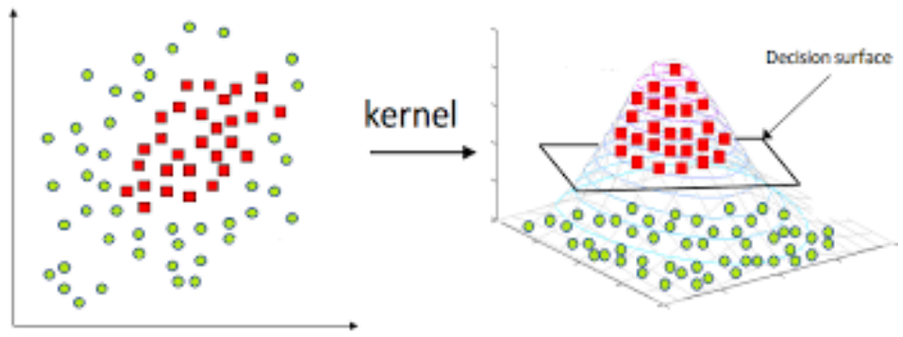
Kernels



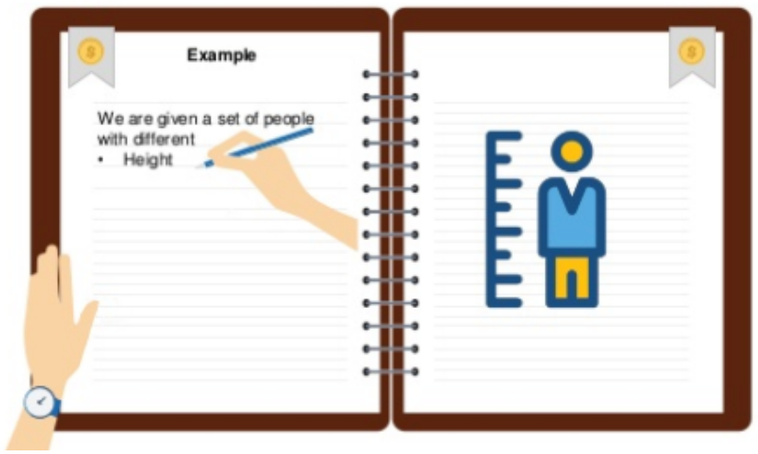
Kernels



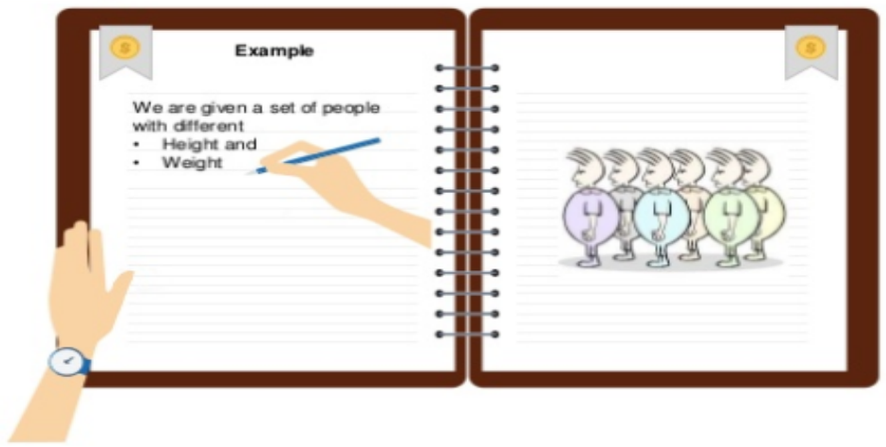
Kernels



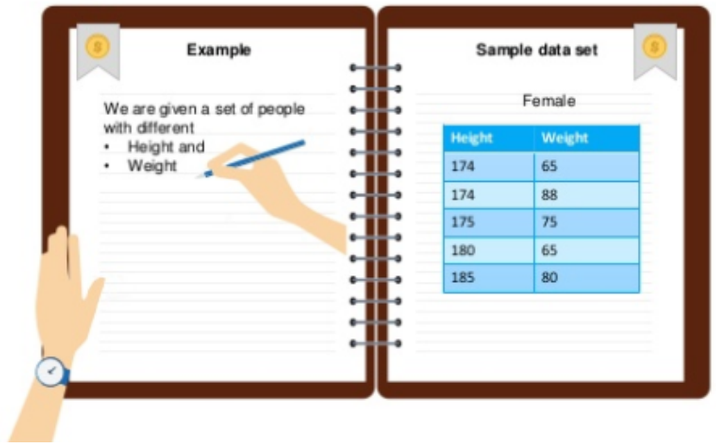
SVM – Another try



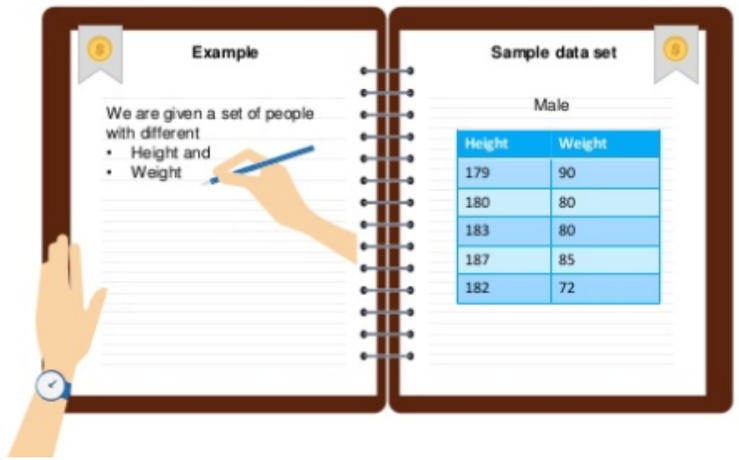
SVM – Another try



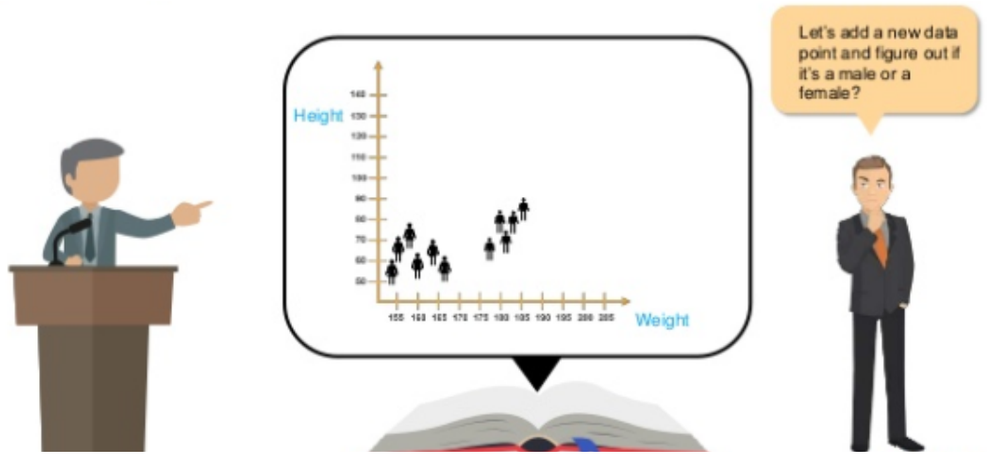
SVM – Another try



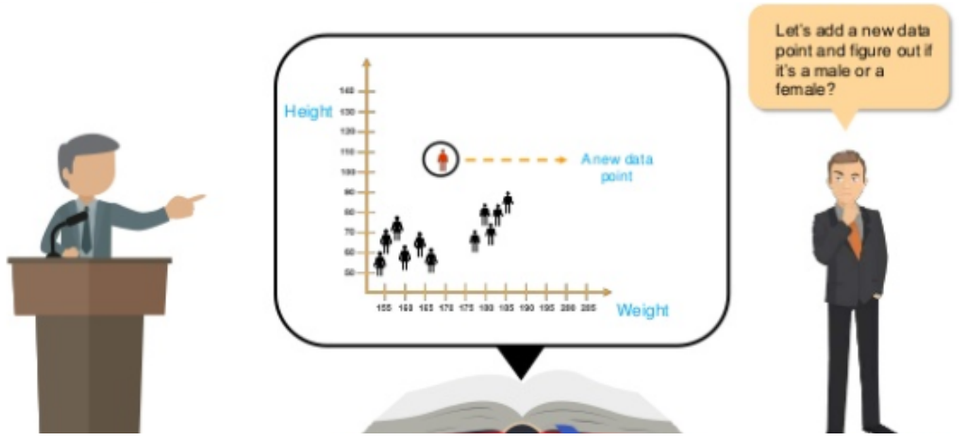
SVM – Another try



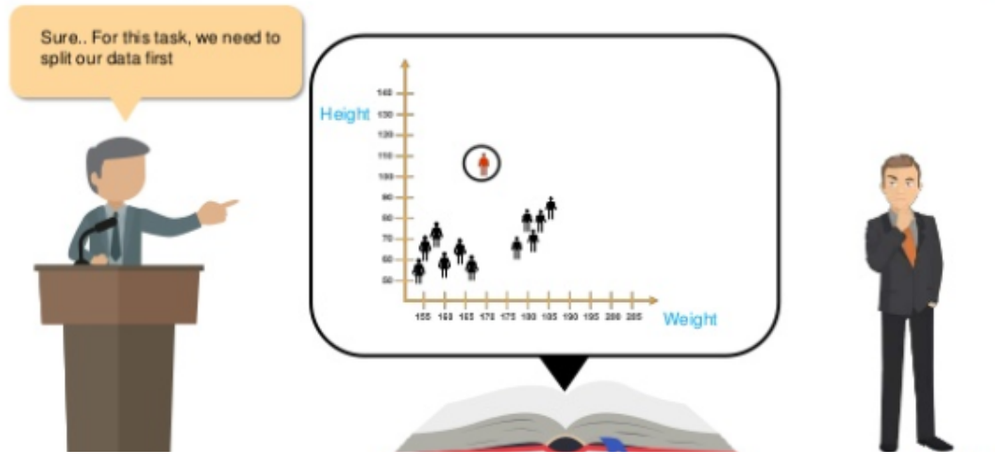
SVM – Another try



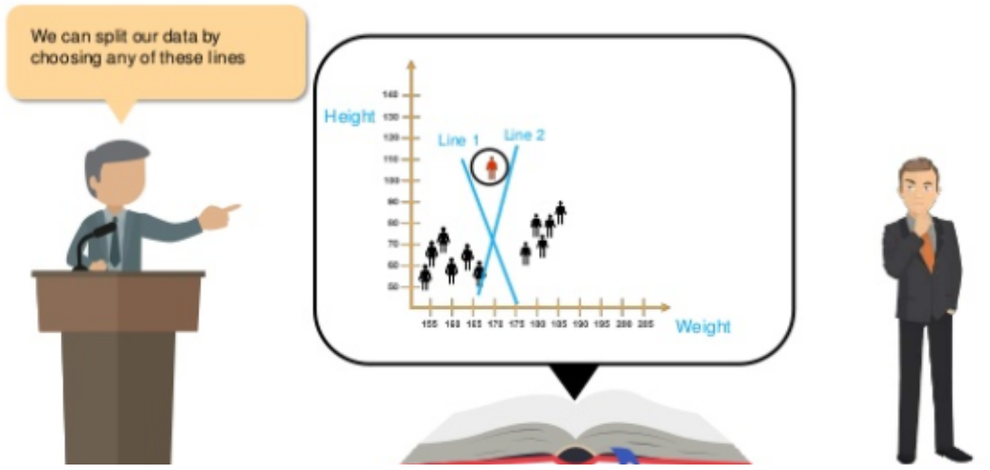
SVM – Another try



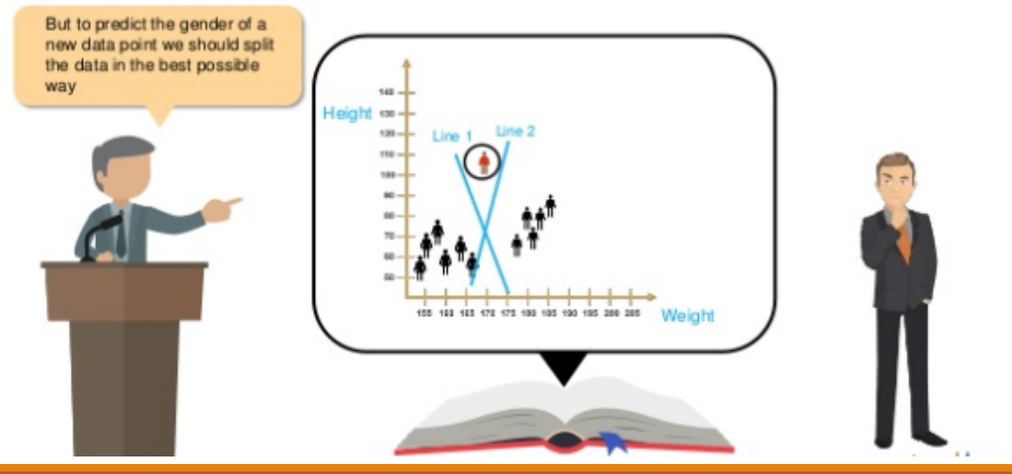
SVM – Another try



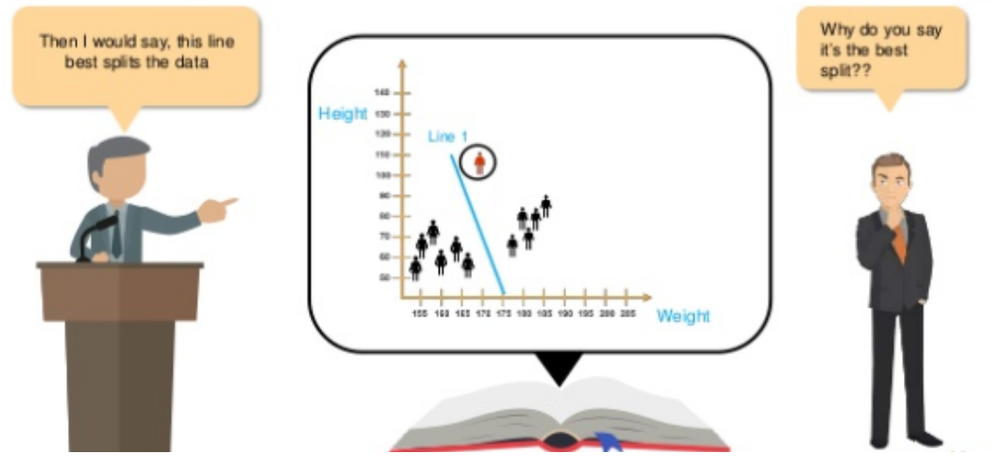
SVM – Another try



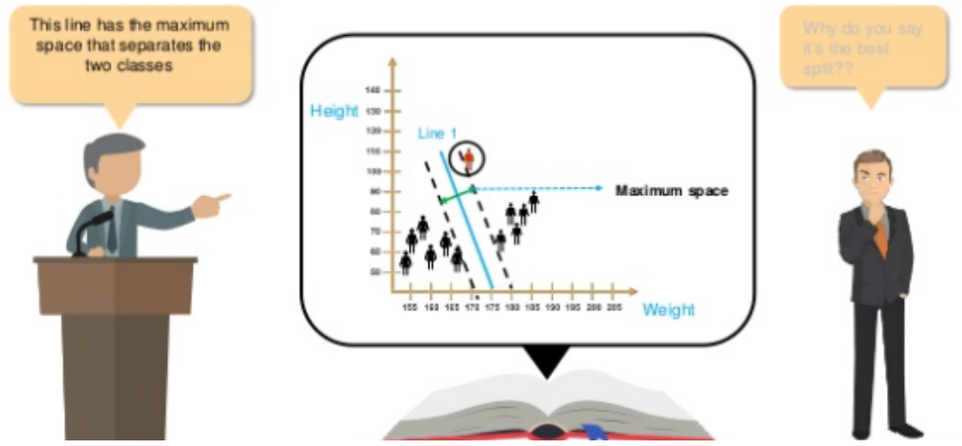
SVM – Another try



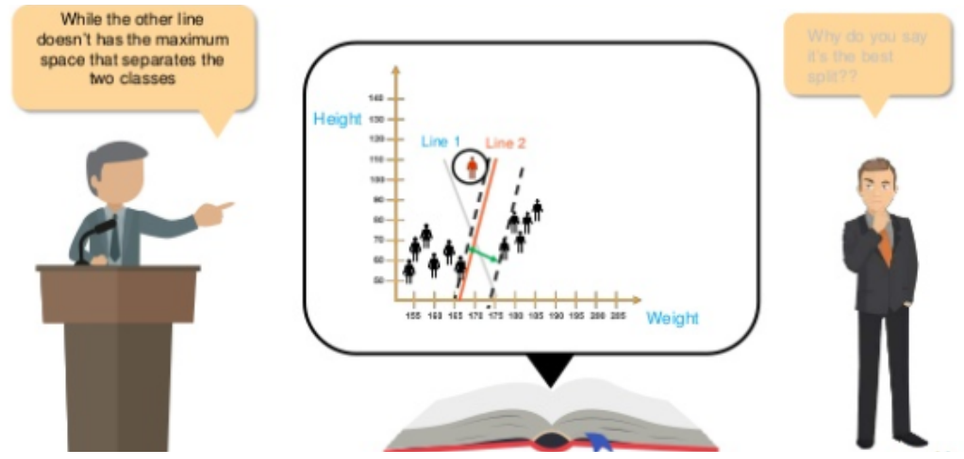
SVM – Another try



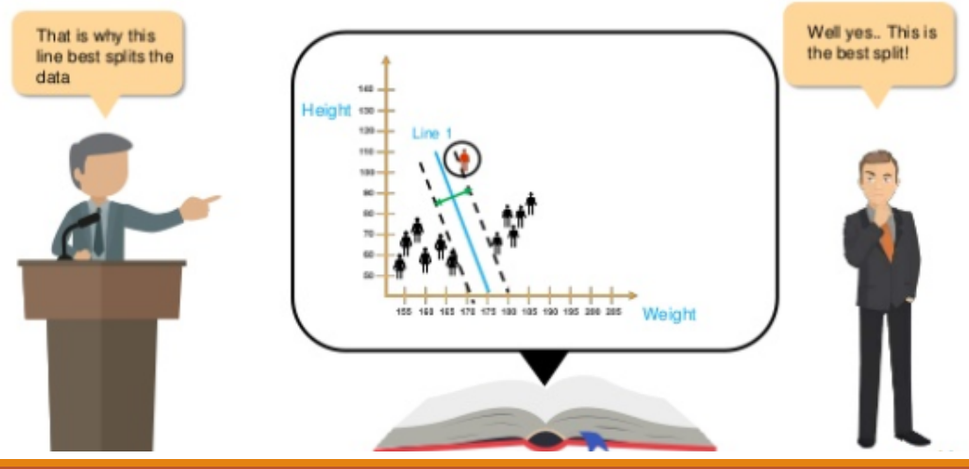
SVM – Another try



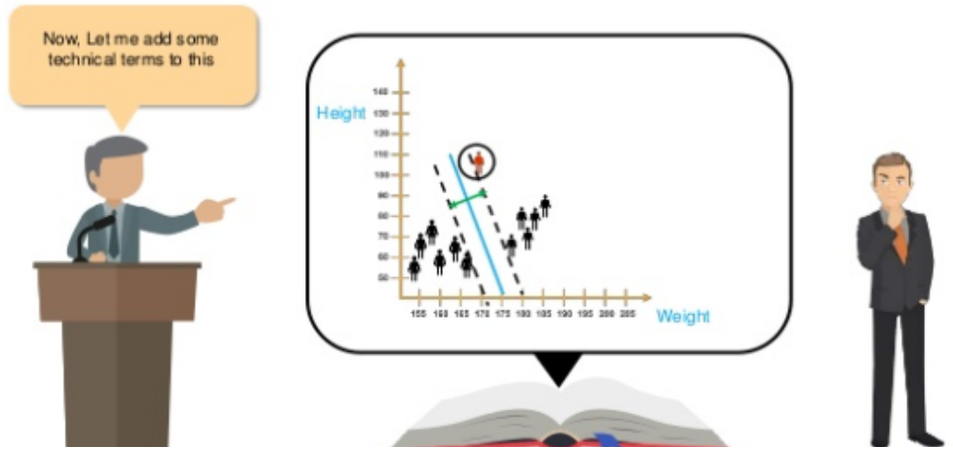
SVM – Another try



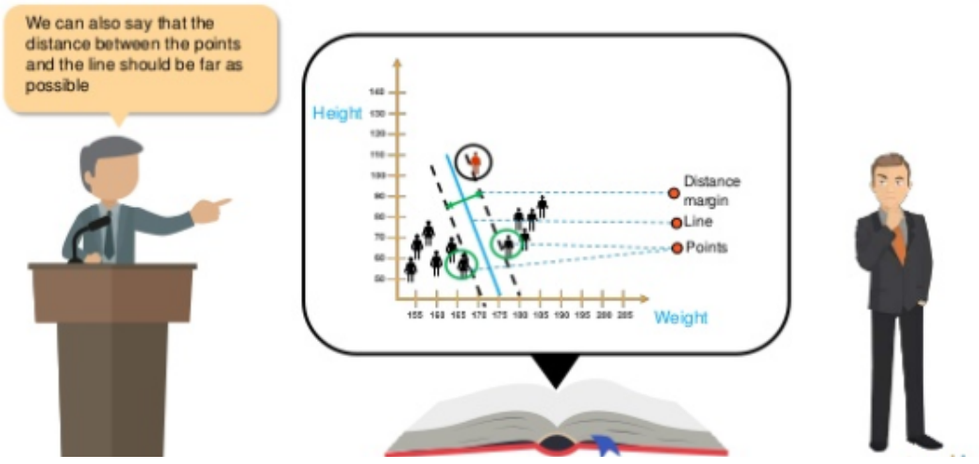
SVM – Another try



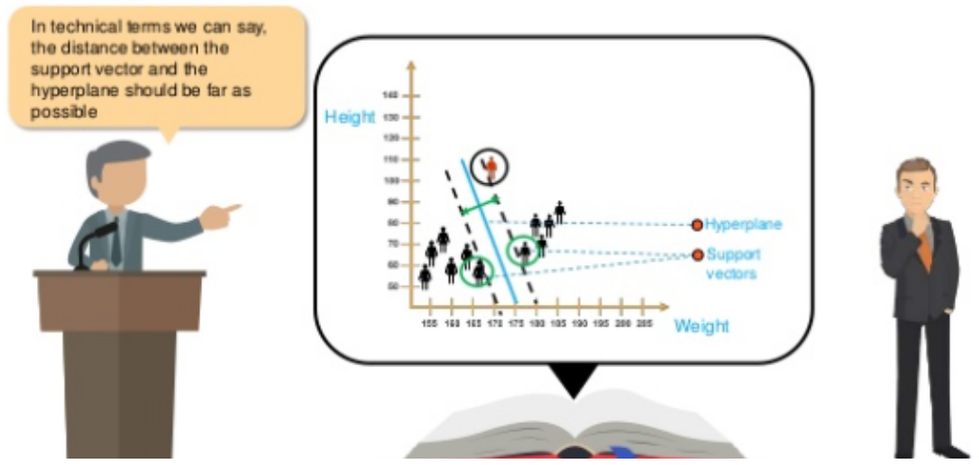
SVM – Another try



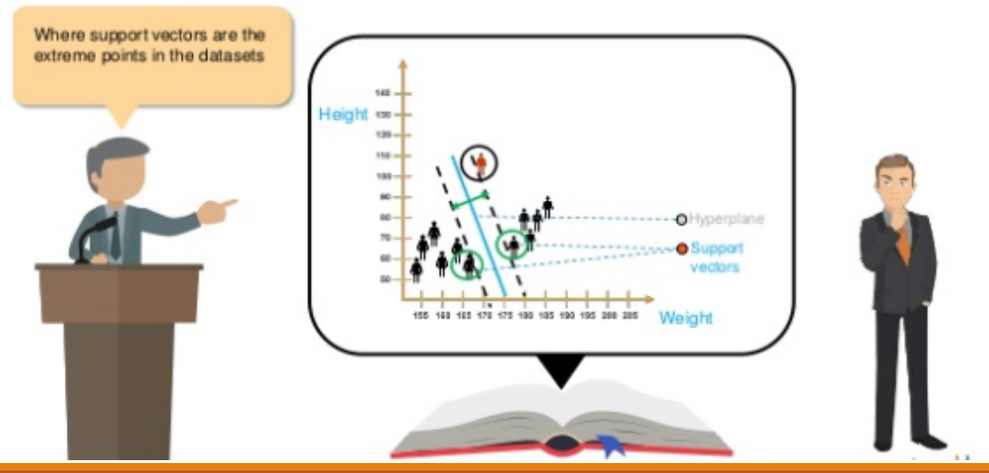
SVM – Another try



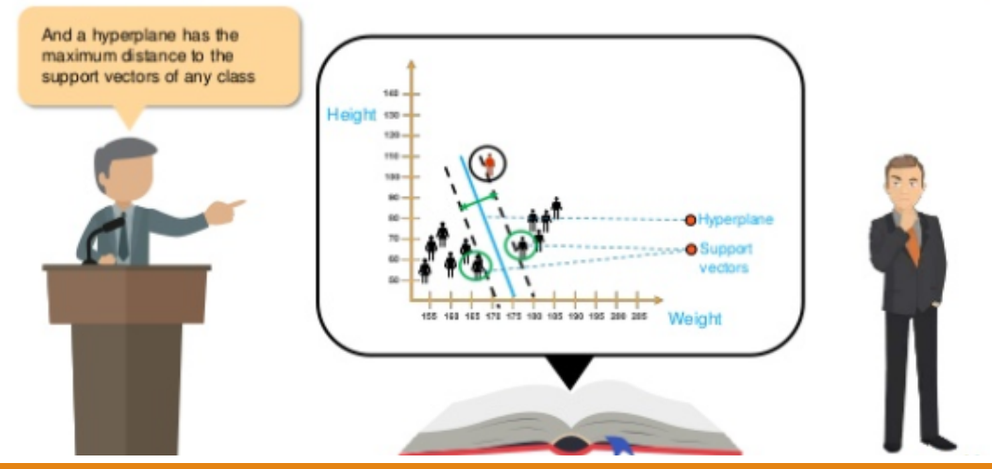
SVM – Another try



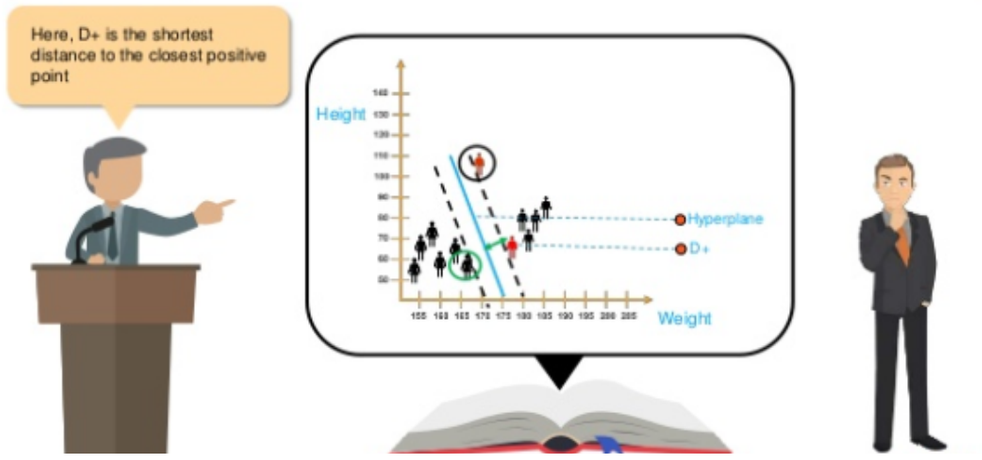
SVM – Another try



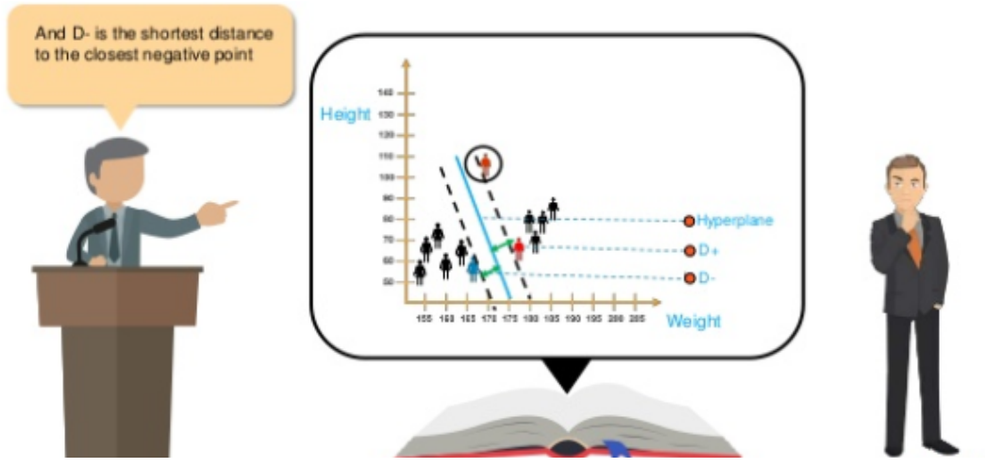
SVM – Another try



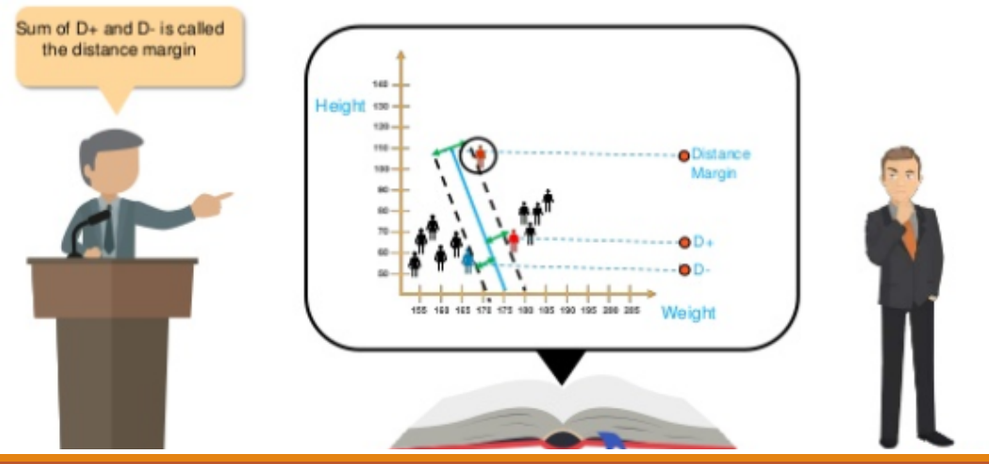
SVM – Another try



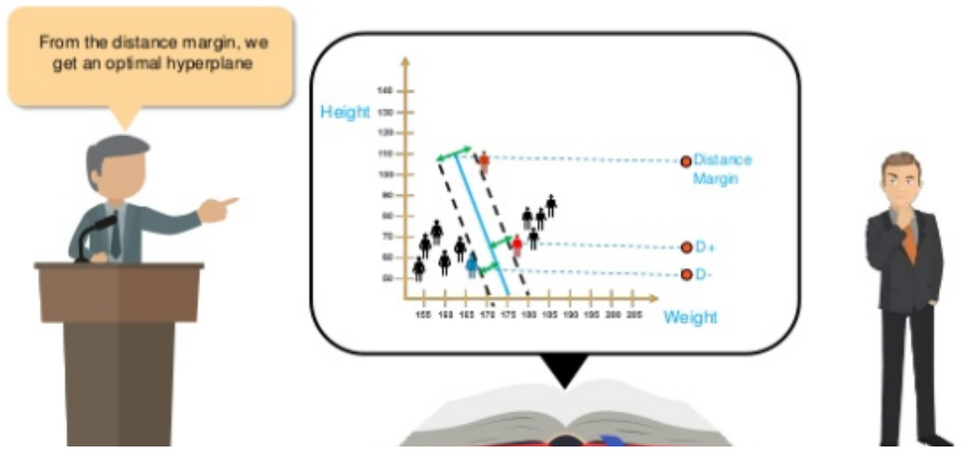
SVM – Another try



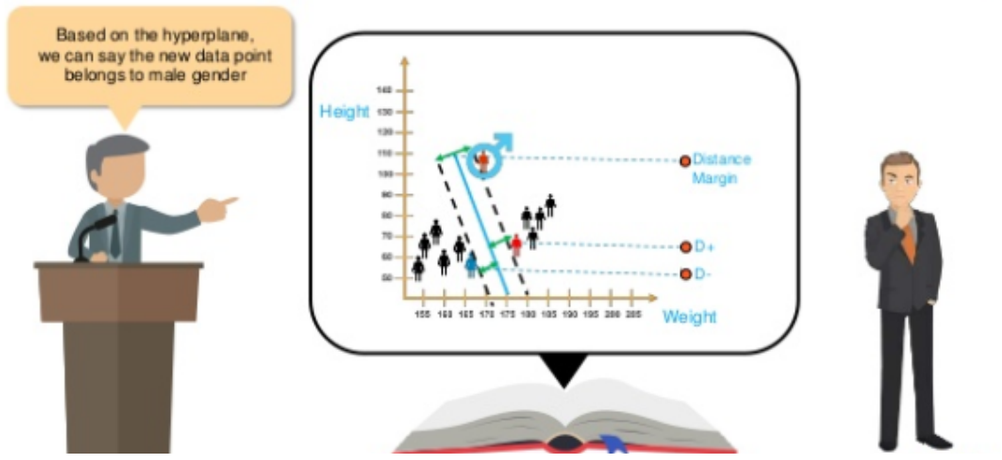
SVM – Another try



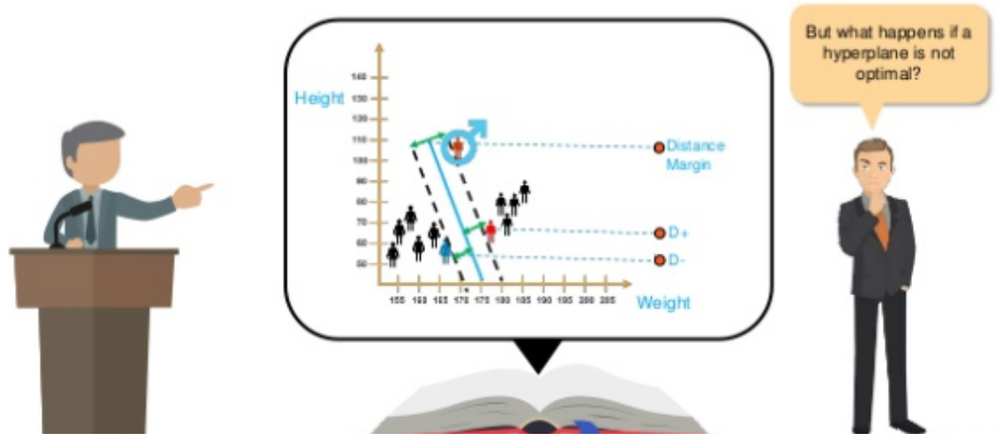
SVM – Another try



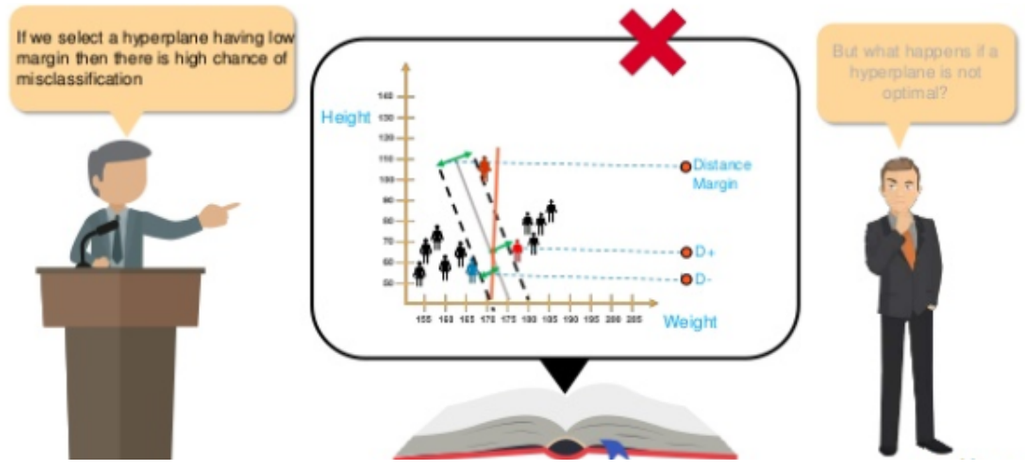
SVM – Another try



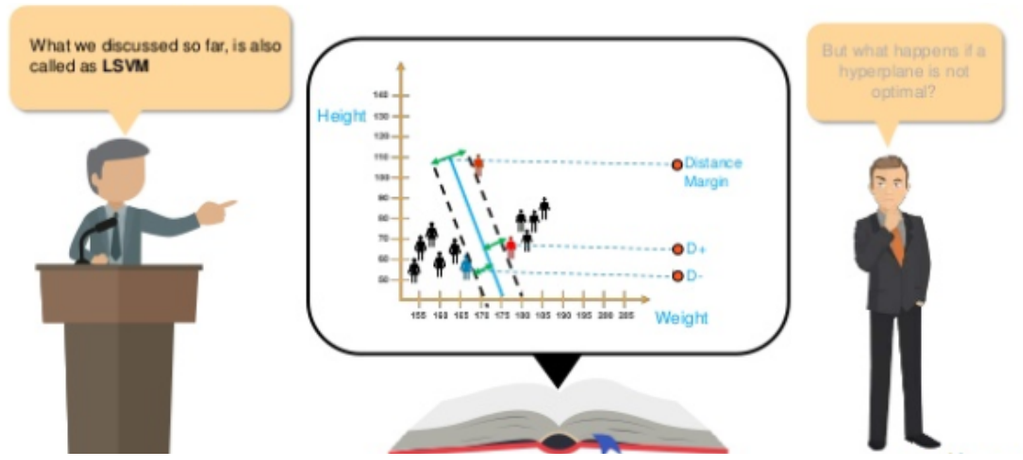
SVM – Another try



SVM – Another try



SVM – Another try

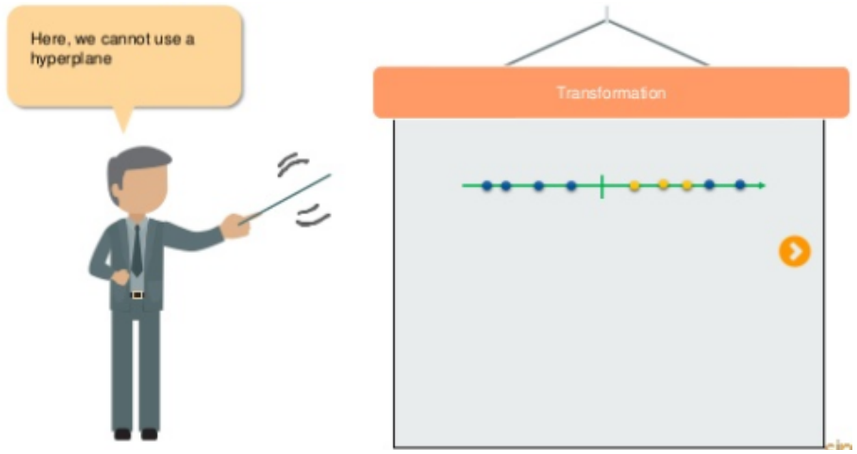


SVM – Another try

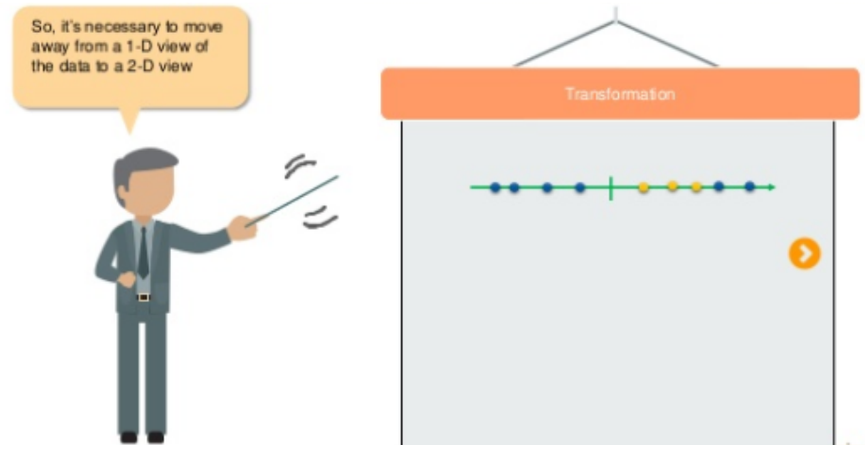
SVM – Another try

SVM – Another try

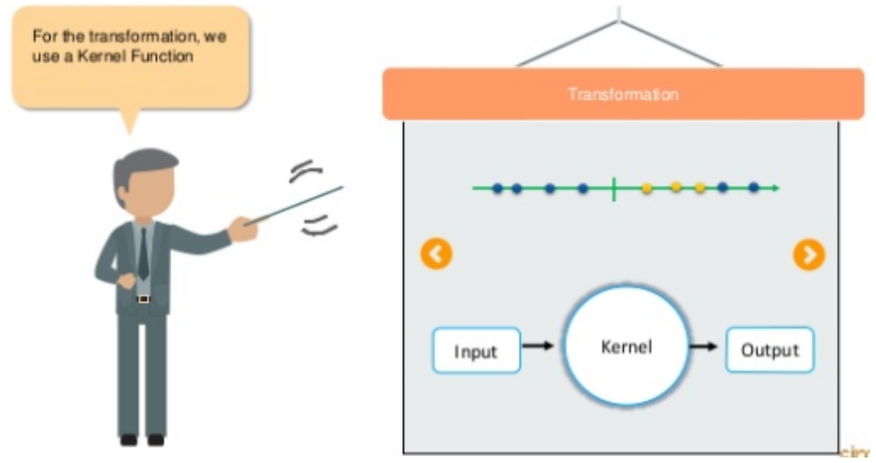
SVM – Another try



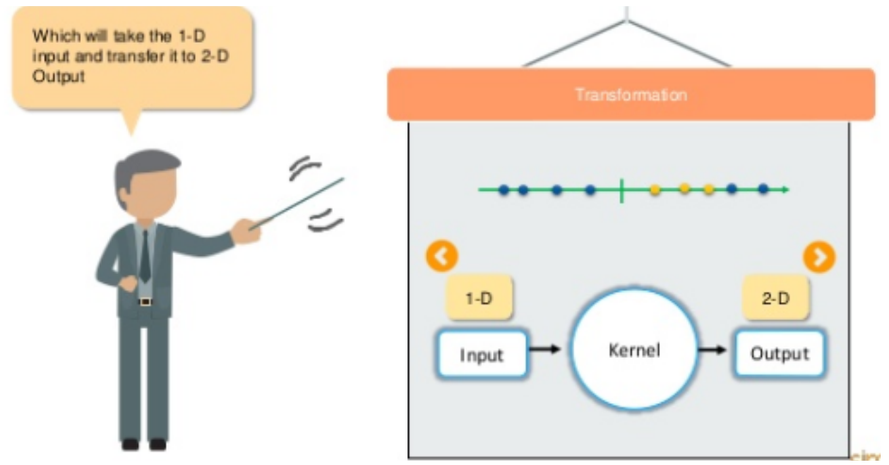
SVM – Another try



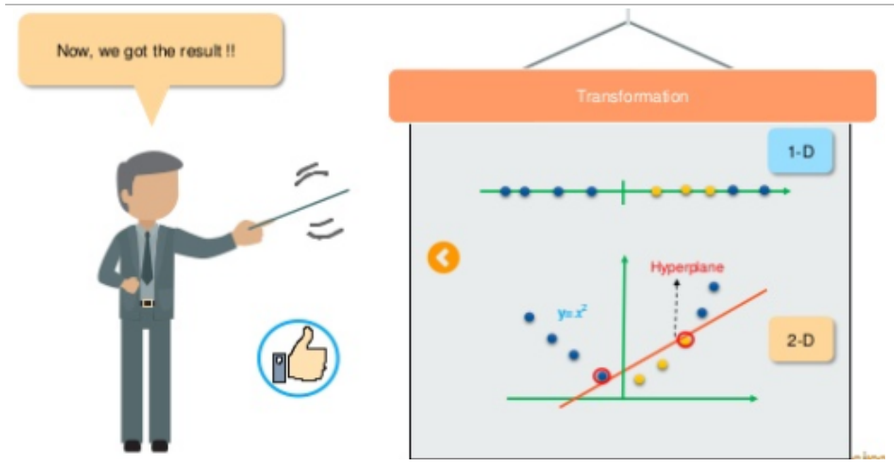
SVM – Another try



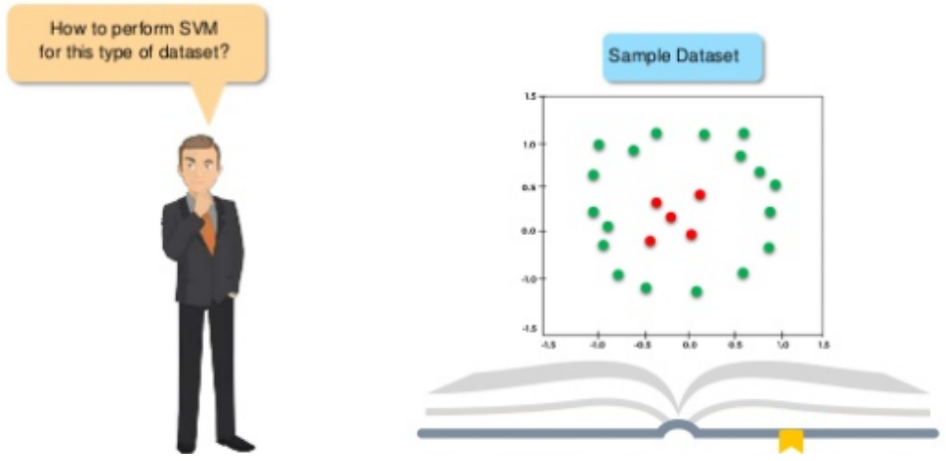
SVM – Another try



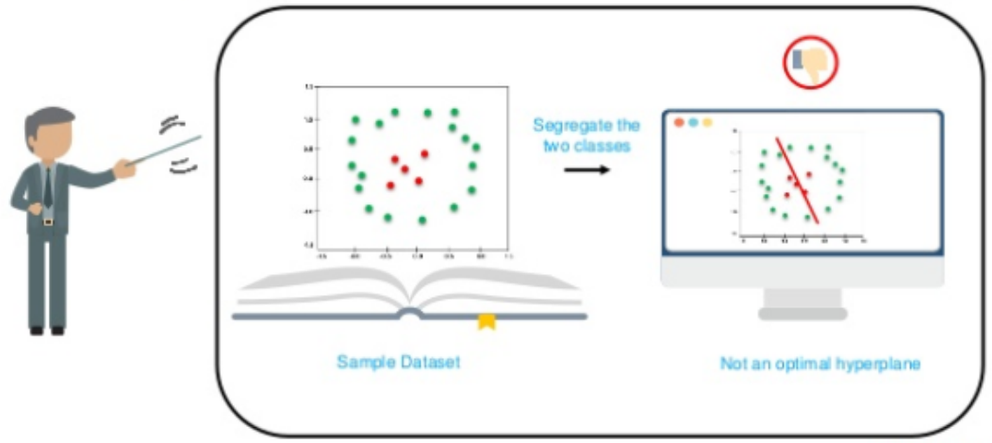
SVM – Another try



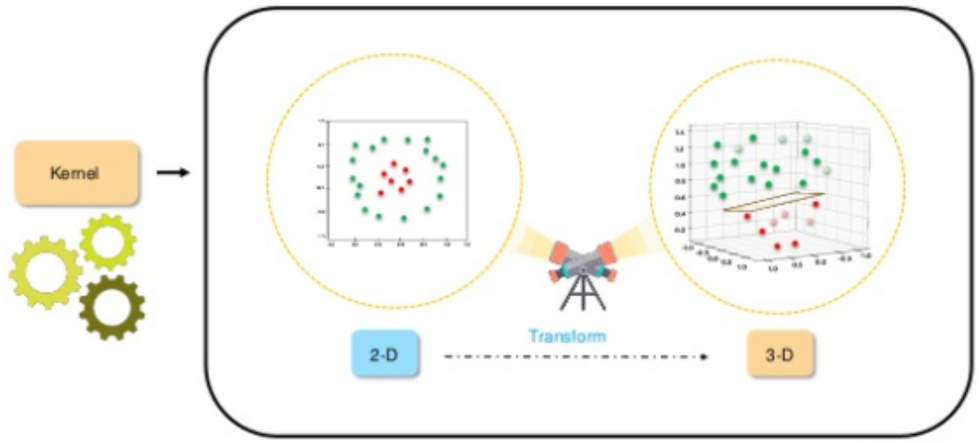
SVM – Another try



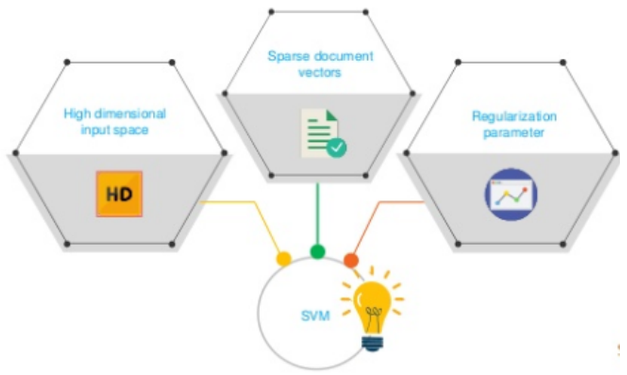
SVM – Another try



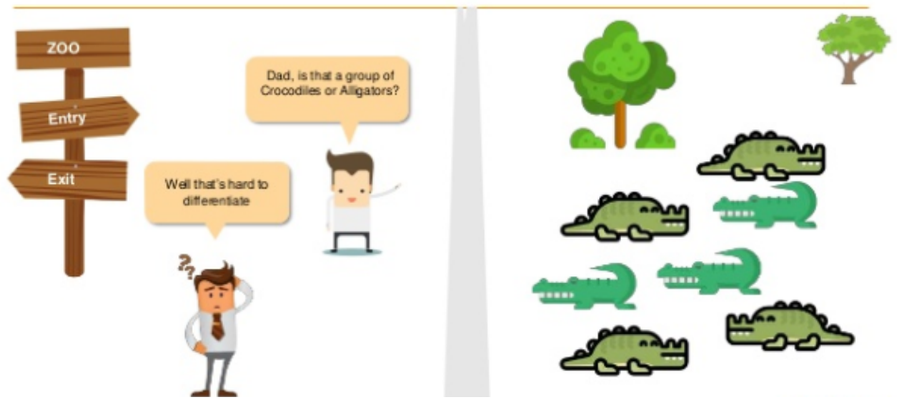
SVM – Another try



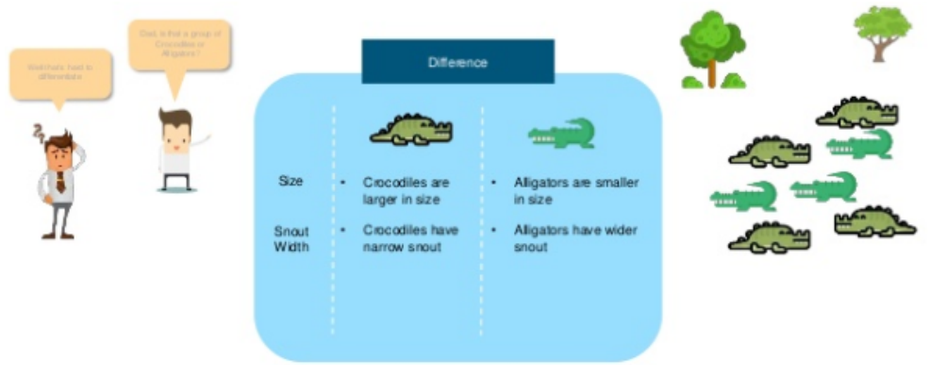
SVM – Another try



SVM – Another try



SVM – Another try

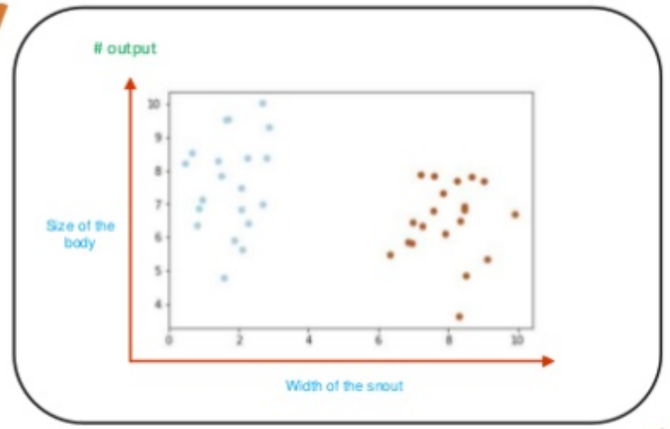


SVM – Another try

SVM – Another try

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets.samples_generator import make_blobs
# we create 40 separable points
X, y = make_blobs(n_samples=40, centers=2, random_state=20)

# fit the model, don't regularize for illustration purposes
clf = svm.SVC(kernel='linear', C=1000)
clf.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=plt.cm.Paired)
```

SVM – Another try



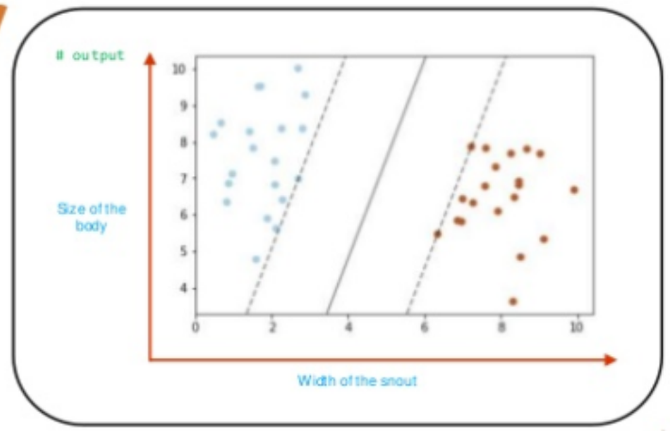
SVM – Another try

```
# fit the model, don't regularize for illustration purposes
clf = svm.SVC(kernel='linear', C=1000)
clf.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, s=30,cmap=plt.cm.Paired)
plt.show

# plot the decision function
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()

# create grid to evaluate model
xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
Z = clf.decision_function(xy).reshape(XX.shape)
```

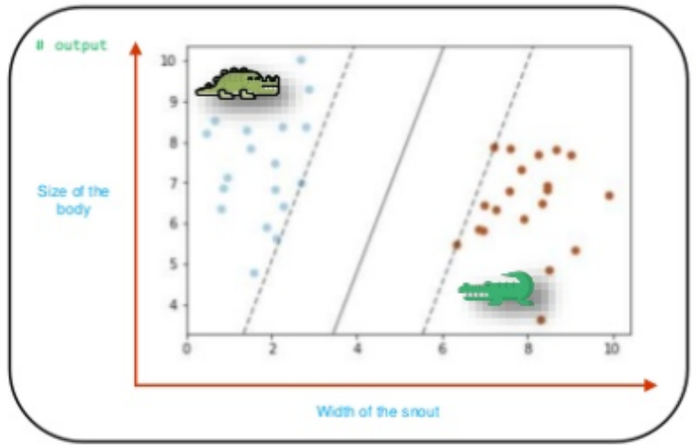
SVM – Another try

```
# plot decision boundary and margins
ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
           linestyles=['--', '-', '--'])
# plot support vectors
ax.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],
s=100,
           linewidth=1, facecolors='none')
plt.show()
```

SVM – Another try



SVM – Another try



A Cautionary Example

Image classification of tanks. Autofire when an enemy tank is spotted.
Input data: Photos of own and enemy tanks.
Worked really good with the training set used.
In reality it failed completely.

Reason: All enemy tank photos taken in the morning. All own tanks in dawn.
The classifier couldn’t recognize dusk from dawn!!!!

Tips to tune SVM

- **Kernel**: The main function of the kernel is to transform the given dataset input data into the required form. There are various types of functions such as linear, polynomial, and radial basis function (RBF). Polynomial and RBF are useful for non-linear hyperplane.

- **Regularization**: Regularization parameter in python's Scikit-learn C parameter used to maintain regularization. Here C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimization how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term. A smaller value of C creates a small-margin hyperplane and a larger value of C creates a larger-margin hyperplane.

- **Gamma**: A lower value of Gamma will loosely fit the training dataset, whereas a higher value of gamma will exactly fit the training dataset, which causes over-fitting. In other words, you can say a low value of gamma considers only nearby points in calculating the separation line, while the a value of gamma considers all the data points in the calculation of the separation line.

SVM Results

- F1 score should be as close to 1 as possible
- Precision and Recall
- Accuracy via confusion matrix

Hands on

- https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
- https://www.kaggle.com/halien/simple-image-classifer-with-svm
- https://github.com/whimian/SVM-Image-Classification
- https://kapernikov.com/tutorial-image-classification-with-scikit-learn/

Questions?

Thank you!