Data Science

Yogesh Kulkarni

SVM

Support Vector Machines (SVM)

- Support vector machines are supervised machine learning algorithms that construct separating planes for classification.
- They are conceptually fairly simple, but the underlying mathematics for learning them from data can be tricky.

Support Vector Machines (SVM) Inventors

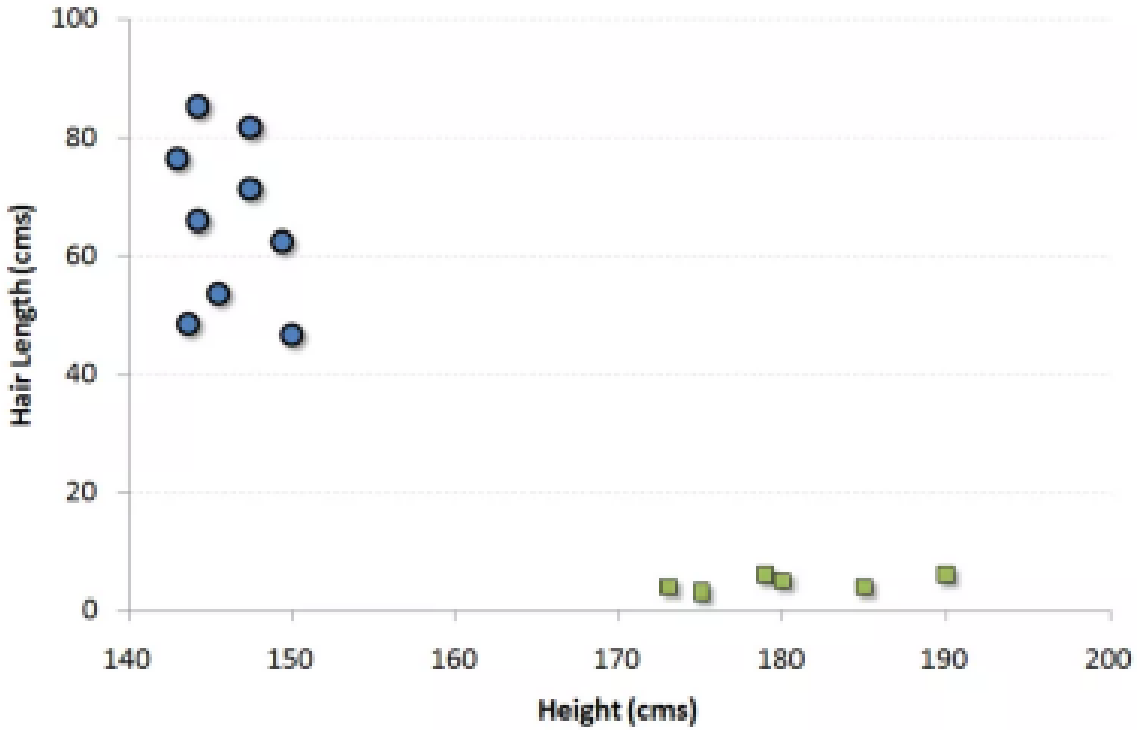
- See video "Vladimir Vapnik - 2012 Laureate of the Franklin Institute in Computer and Cognitive Science" at Youtube.
- Vladimir Vapnik discusses how his early years in Moscow helped shaped the origins of statistical learning theories, including the ideas behind the SVM.
- His collaborator, Isabelle Guyon, also discusses the Kernel Trick, which generalized the SVMs for non-linear decision boundaries (hence making them more popular).

Support Vector Machines (SVM) Inventors

"The invention of SVMs happened when Bernhard decided to implement Vladimir's algorithm in the three months we had left before we moved to Berkeley. After some initial success of the linear algorithm, Vladimir suggested introducing products of features. I proposed to rather use the kernel trick of the 'potential function' algorithm. Vladimir initially resisted the idea because the inventors of the 'potential functions' algorithm (Aizerman, Braverman, and Rozonoer) were from a competing team of his institute back in the 1960's in Russia!" — Isabelle Guyon

Support vector machines Example

- In SVM, we plot each data item as a point in n dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.
- For example, if we only had two features like Height and Hair length of an individual.
- We'd first plot these two variables in two dimensional space where each point has two-coordinates.



Now, we will find some line that splits the data between the two differently classified groups of data.

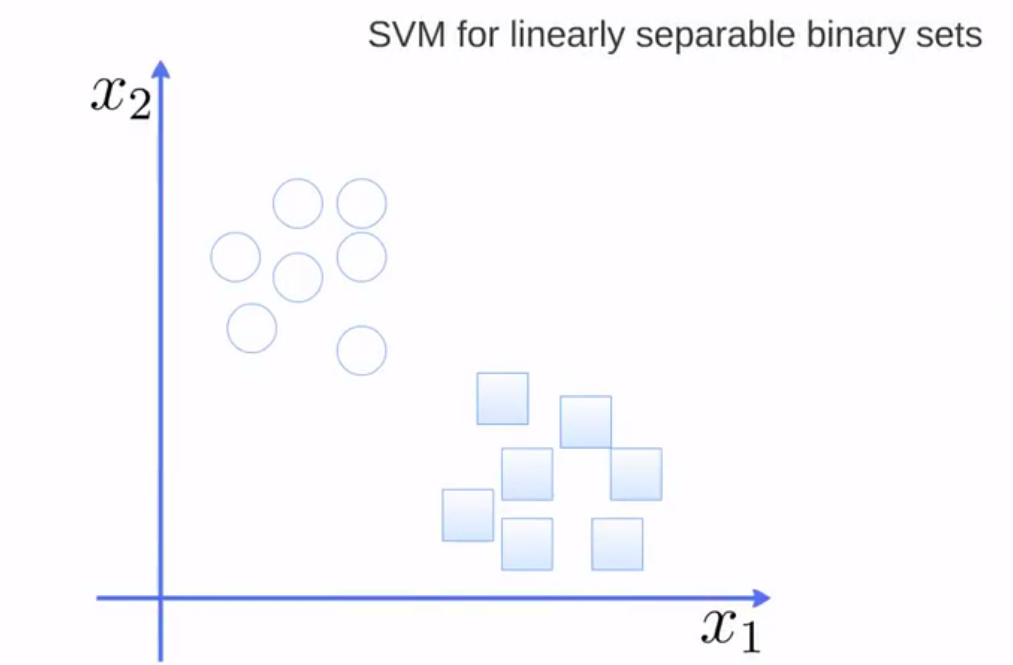
Support Vector Machines (SVM)

SVMs are the combination of two ideas

- Maximum margin classification
- The “kernel trick”

SVM Intuition

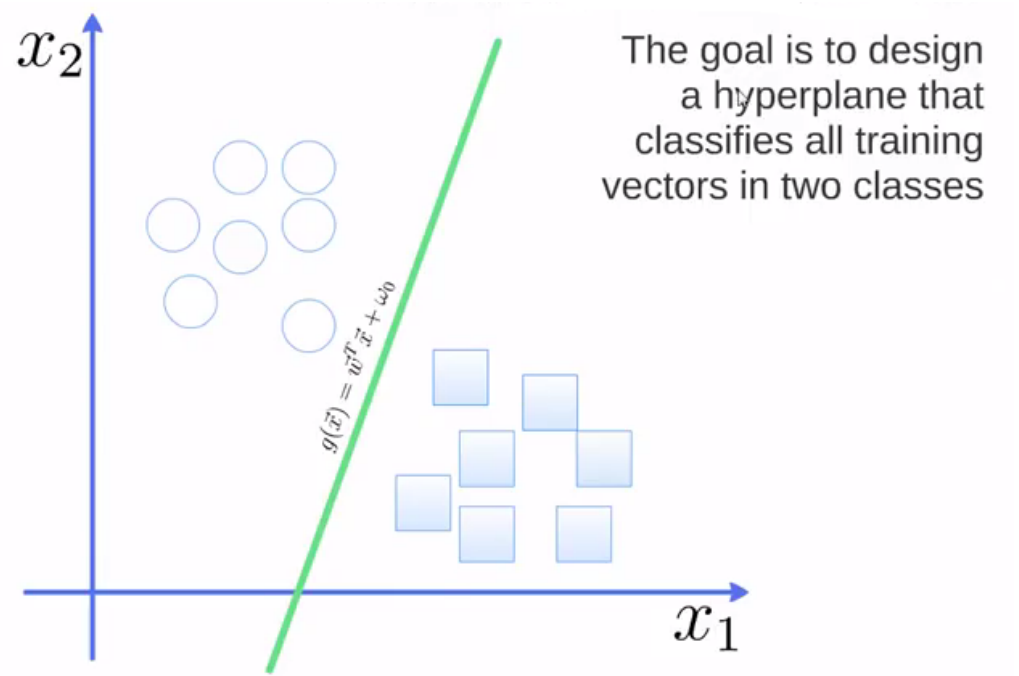
SVM Intuition



(Ref: How Classification Algorithms Work? - Thales Sehn Korting)

- Suppose there are points characterized by x_1, x_2.
- So these are features
- Points are labeled with either of two classes: circles and squares.

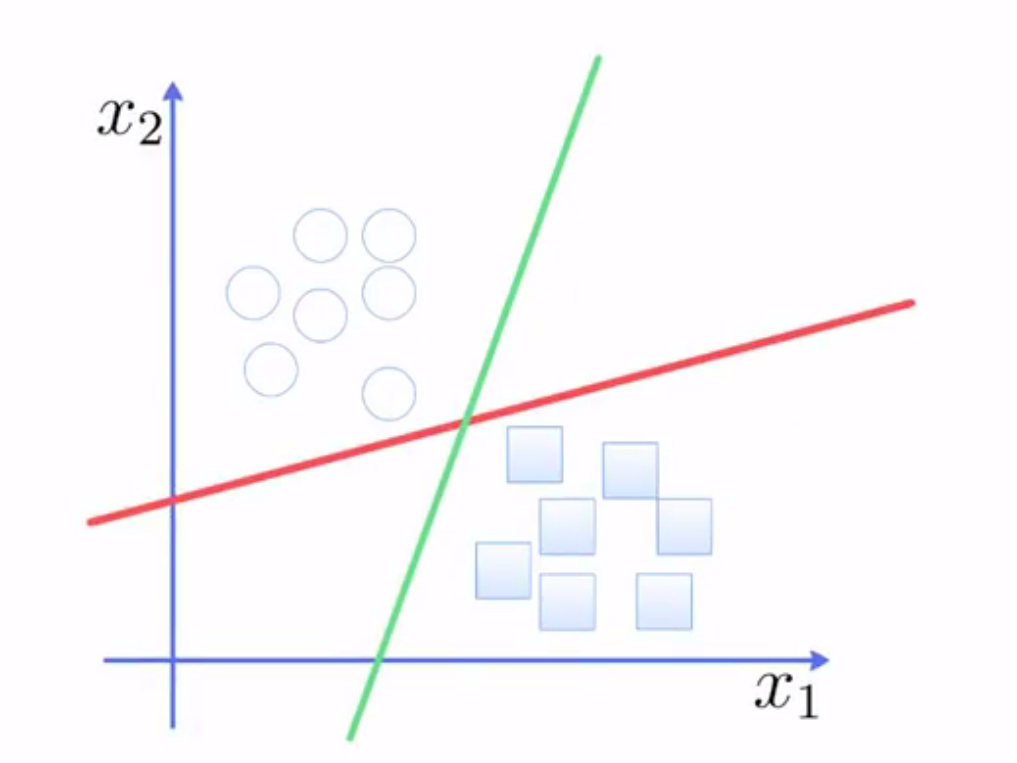
SVM Intuition



(Ref: How Classification Algorithms Work? - Thales Sehn Korting)

- Goal is to find a separating plane that separates the classes as per the training data
- Generic Hyper-plane's equation is taken as $g(\overrightarrow{x}) = \overrightarrow{w}^T\overrightarrow{x} + w_0$

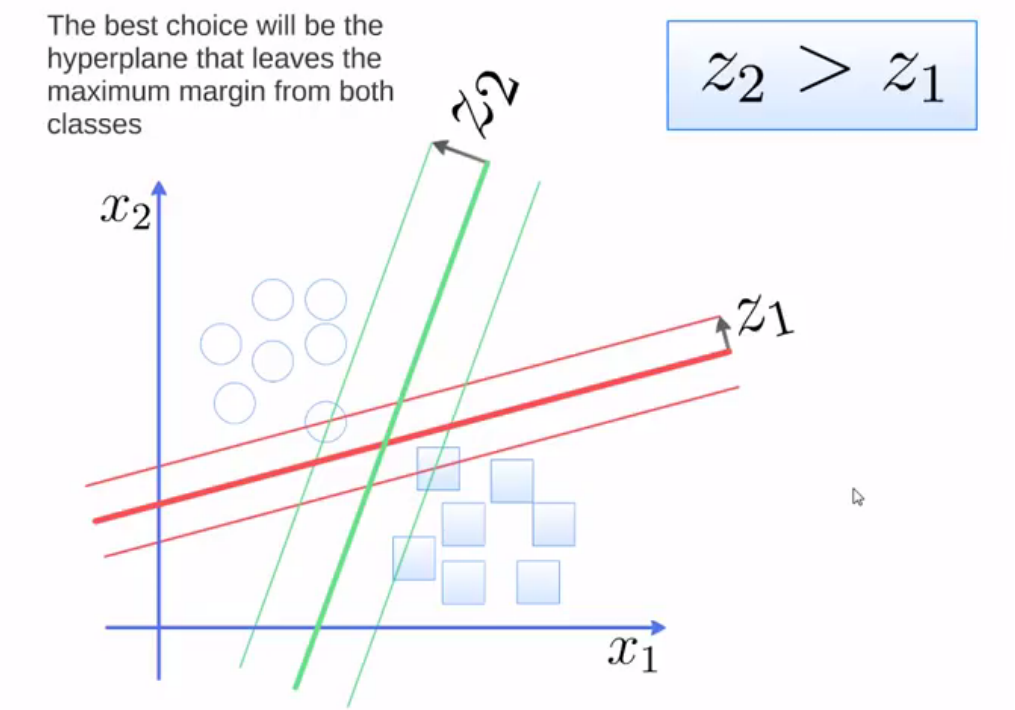
SVM Intuition



(Ref: How Classification Algorithms Work? - Thales Sehn Korting)

- Many Hyper-planes can classify the data correctly.
- But which one is the best?

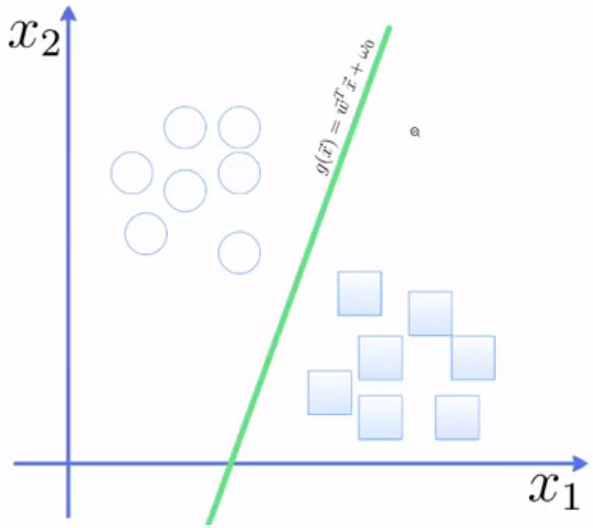
SVM Intuition



- The one with Maximum margin is the best.
- Z_2 is better than Z_1, so nice and distinct separation.

SVM Intuition

$$g(\vec{x}) \geqslant 1, \quad \forall \vec{x} \in \text{class 1}$$
$$g(\vec{x}) \leqslant -1, \quad \forall \vec{x} \in \text{class 2}$$

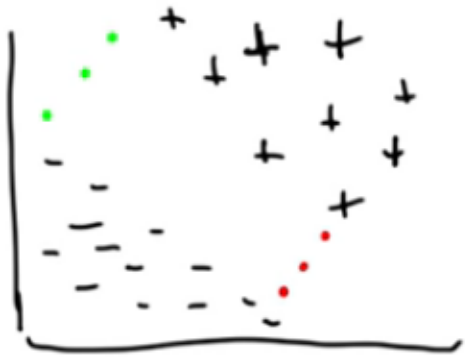


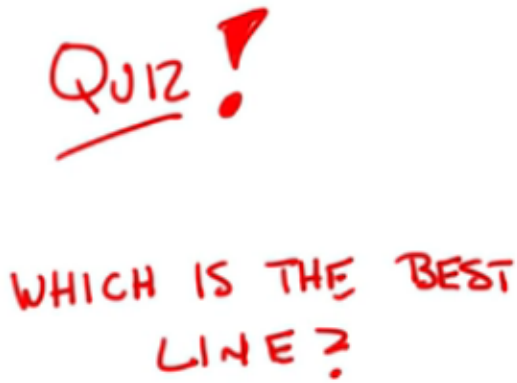
(Ref: How Classification Algorithms Work? - Thales Sehn Korting)

- For data points labeled as circles, when features are put into "g" equation, the result is more than equal to 1
- For data points labeled as squares, when features are put into "g" equation, the result is less than equal to -1

SVM Quick Derivation

Which is the best line?

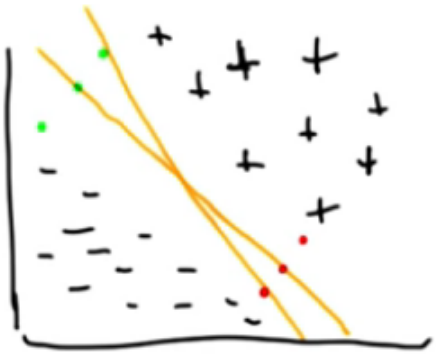


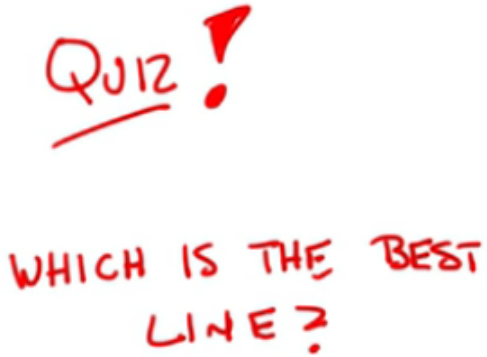


(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- Question: Which is the best line?
- Not any random line!!
- There are 3 green dots and 3 red dots in the gap (gutter/margin), which 2 dots form best line?

Better Margin for Unseen points

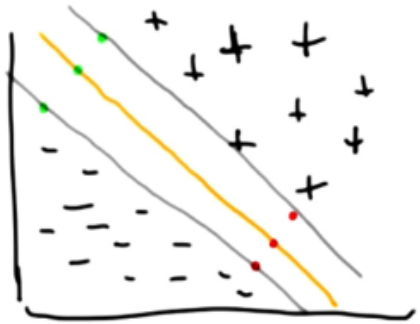




(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- With 3 green forming lines with 3 reds each, there are 9 possible lines.
- And the ALL separate the positive and negative points.
- Intuitively, we thought that the middle line looks GOOD.
- The second, more slanting line is not good as its closer to some of the points than the middle line. Like Over-fitting.
- We need clear and BIG separation to avoid mis-classifications as much as possible. Because we don't know what un-seen points will bring. Better the gutter, better chances of good classification.

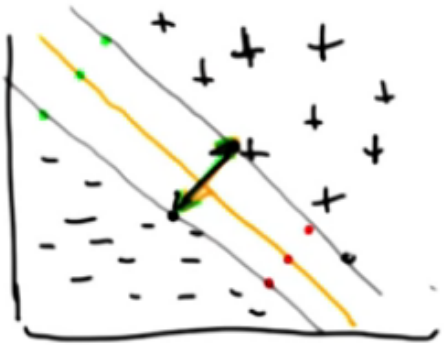
Maximum Margin



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- Even for parallel lines, the side lines are not good as they are CLOSE to either of the positive or negative lines.
- Top line is where the closest Positive line, bottom is to Negative line.
- Middle is the best as it has good space/margin around it.
- So, Goal is to find a separating plane that separates the classes as per the training data

Hperplane with Boolean output



$$y = mx + b$$

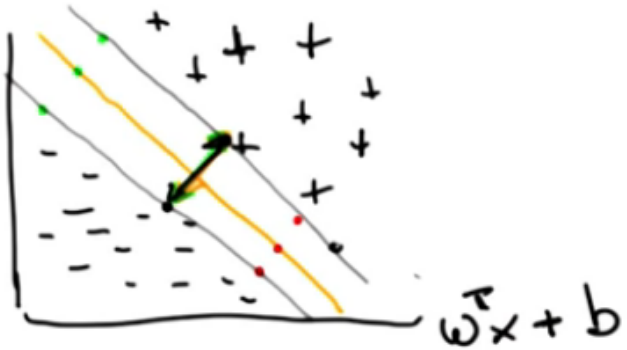
$$y = \omega^T x + b$$

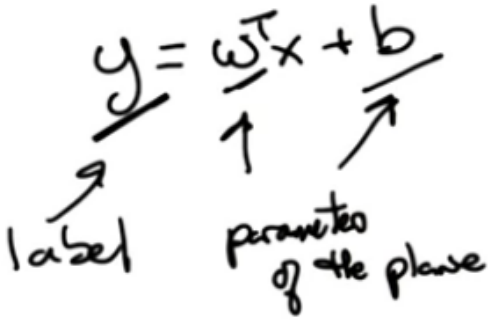
(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

Maths revision: Equations:

- Equation of line is $y = mx + b$. If x has more than 2 dimensions, then this is equation of a plane, a hyper plane and then mx changes to $w^T X$, where w is slopes vector.
- Here y is not the coordinate, but boolean values.

Middle Hyperplane



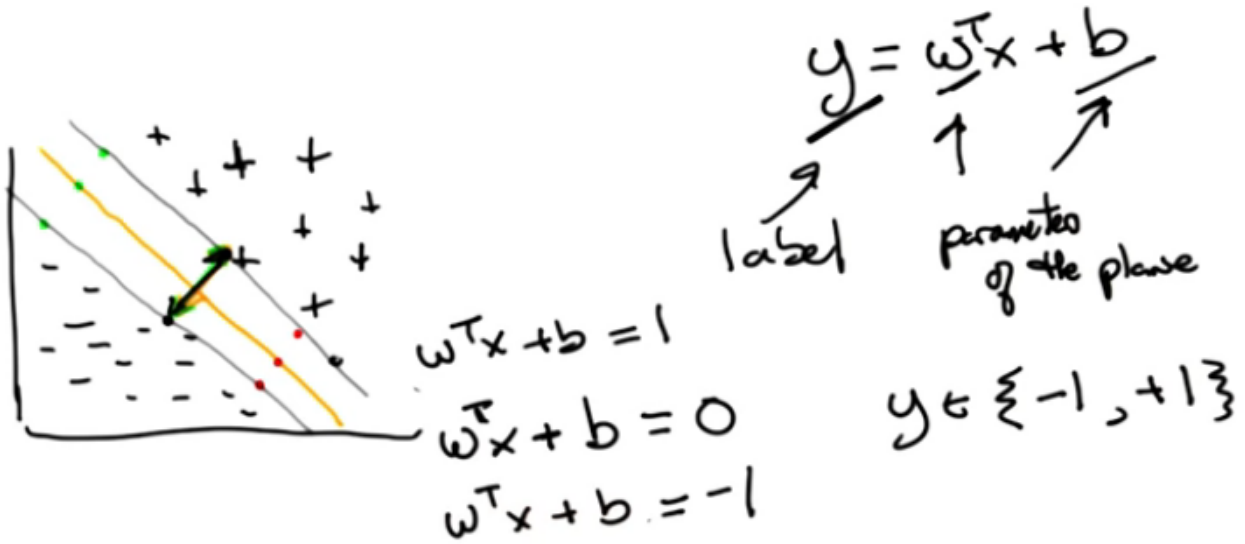


(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

Maths revision: Equations:

- y : classification label
- w^T : parameters, weights, coefficients
- b : bias
- Question: What would be output y for any point (x) on the middle line?

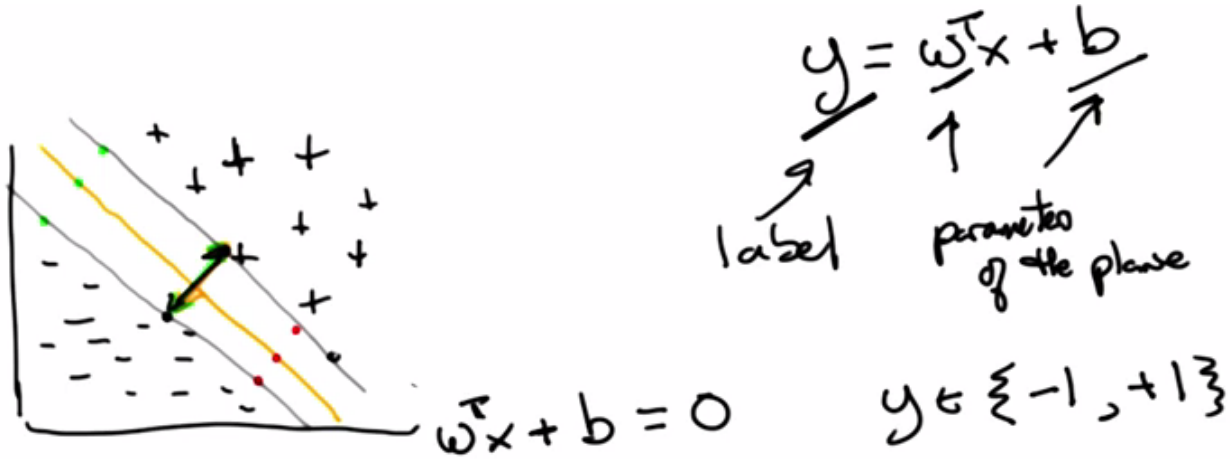
SVM Derivation



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- 0. As its neither positive nor negative.
- So, the equation of the middle line is $w^T X + b = 0$
- What are the equations of other boundary lines?
- So, lets make them -1 and $+1$, to denote negative and positive labels.

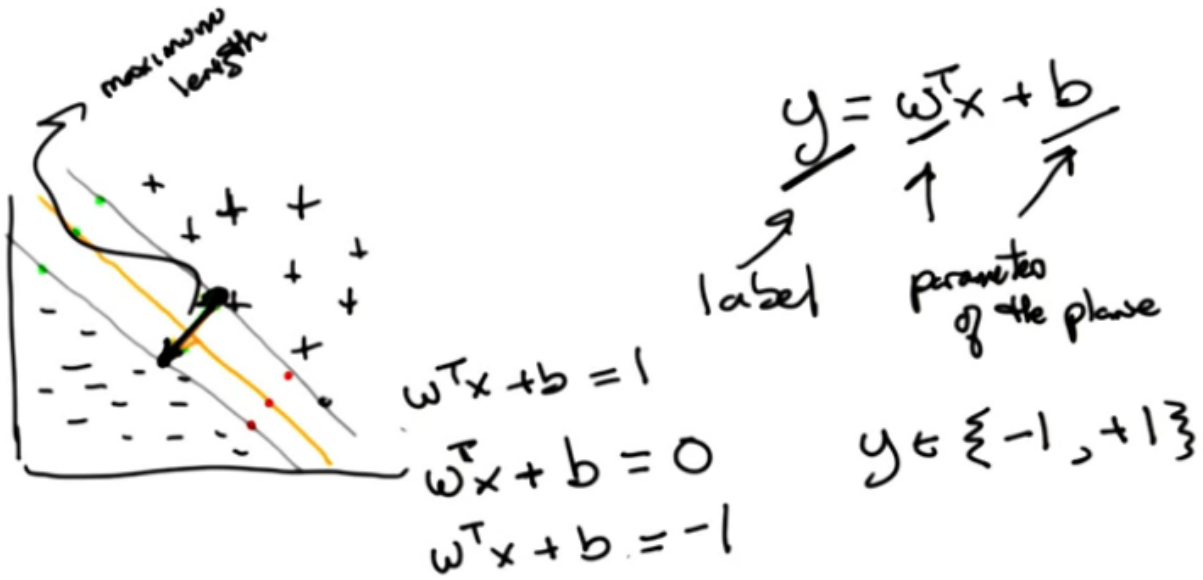
SVM Derivation



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- On the line $y = 0$ so eqn of the hyperplane is $\vec{w}^T \vec{x} + b = 0$
- For left support point, $y = -1$ so equation of left support boundary is $\vec{w}^T \vec{x} + b = -1$ And the full zone as ≤ -1
- For right support point, $y = +1$ so equation of right support boundary is $\vec{w}^T \vec{x} + b = +1$ And the full zone as $\geq +1$

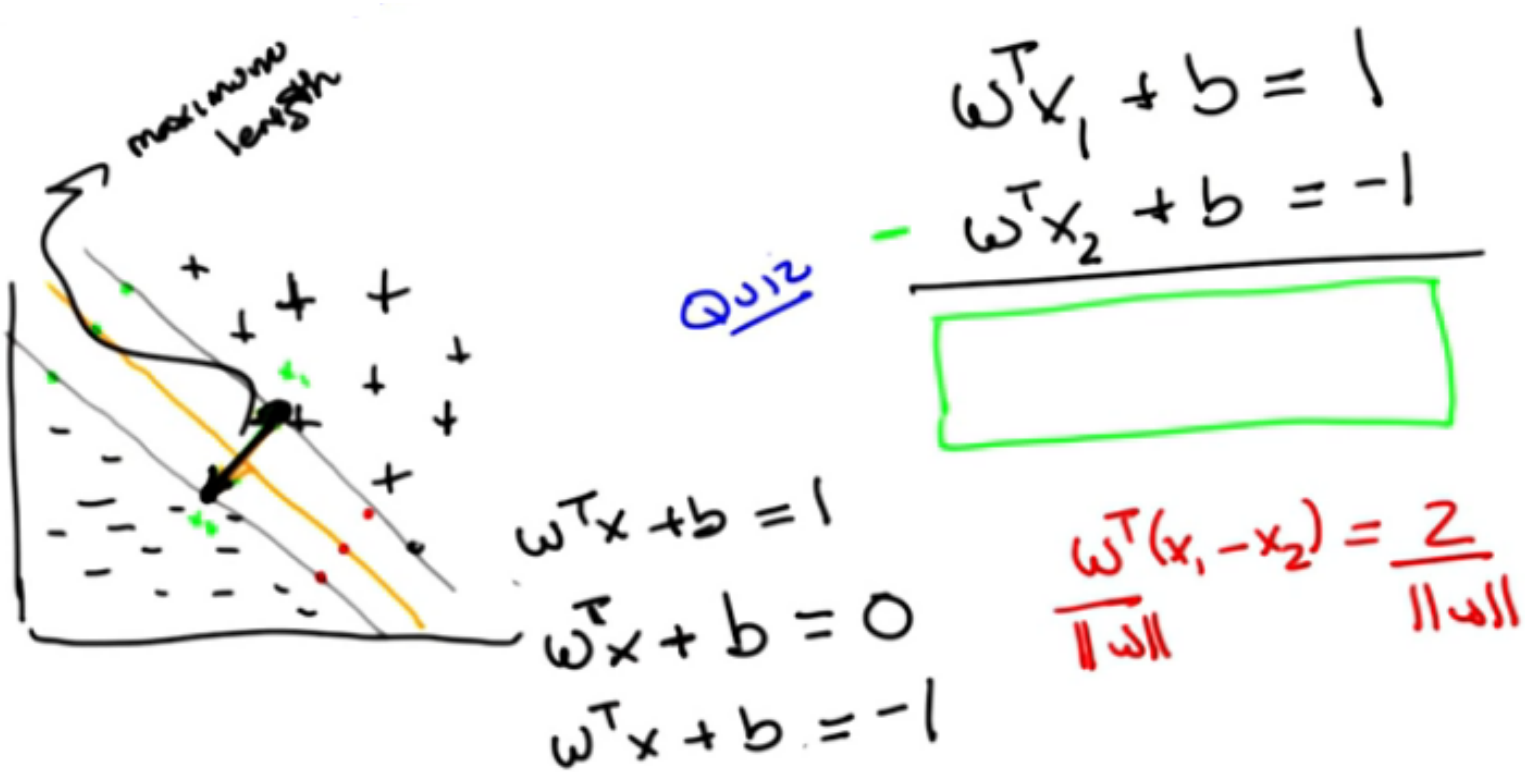
SVM Derivation



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- We want Margin/Width to be maximum. What is the equation of the Width?
- Can we find out, as the Width line is perpendicular to, to the boundary lines and its made of end points lying on the boundary lines?
- Lets call the points x_1 and x_2, so the vector diff between these is Width.

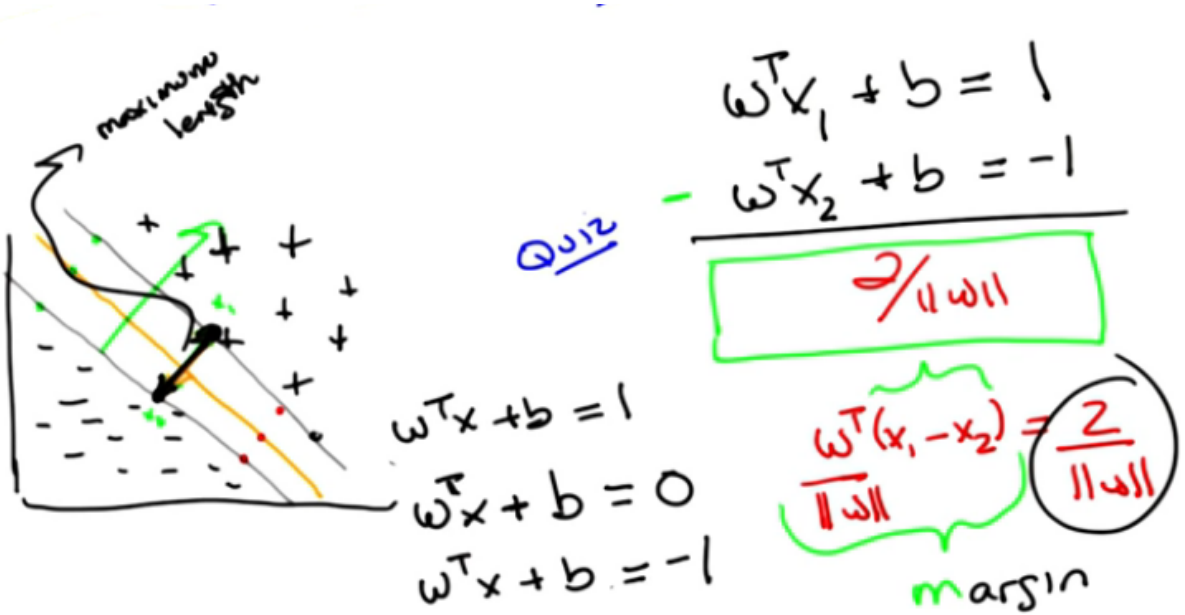
Equation of Width



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- Two equations, 2 unknowns. Subtract. Result is $w^T(x_1 - x_2) = 2$
- Assume w is a number , you would divide 2 by it. But it is a vector. Why not take divide each side by length of w. $w/||w||$ is unit vector.
- So, on left hand side: $x_1 - x_2$ vector has been projected on unit w vector. Thats dot product, which is 2.

Maximizing Width



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- w is the direction along perpendicular to boundary lines, the width direction.
- So, we maximize width m ("margin") while classifying everything correctly (thats the constraint)
- But how to put this constraint mathematically?

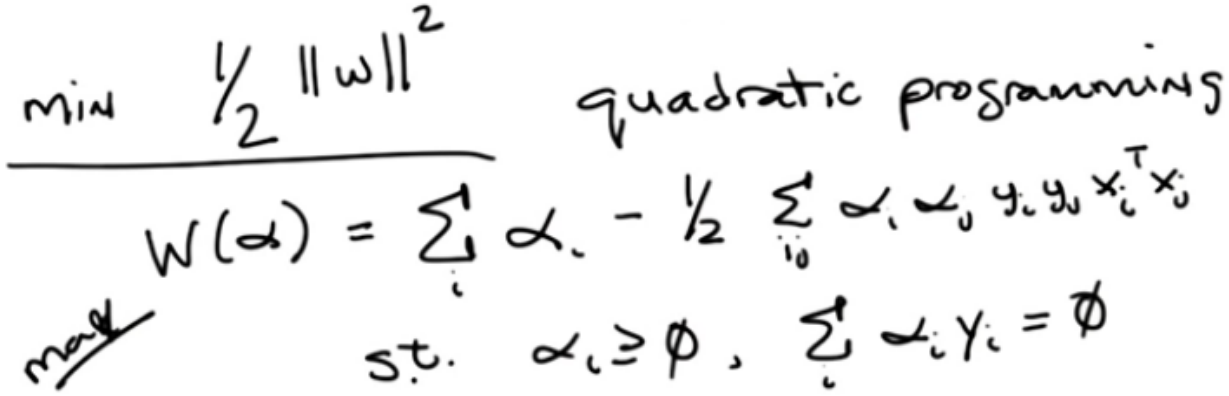
Combining 2 Equations

$$\max \frac{2}{\|w\|} \quad \text{while classifying everything correctly} \quad y_i(w^T x_i + b) \geq 1 \quad \forall_i$$

(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- Lets combine both boundary lines equations into one, by leveraging the label y_i itself.
- $y_i(w^T X_i + b) \geq 1$. The "plus minus 1" thing on the right side has changed to just "plus 1" because y_i becomes minus if right hand side is minus, and both cancel out!!! Wonderful trick.

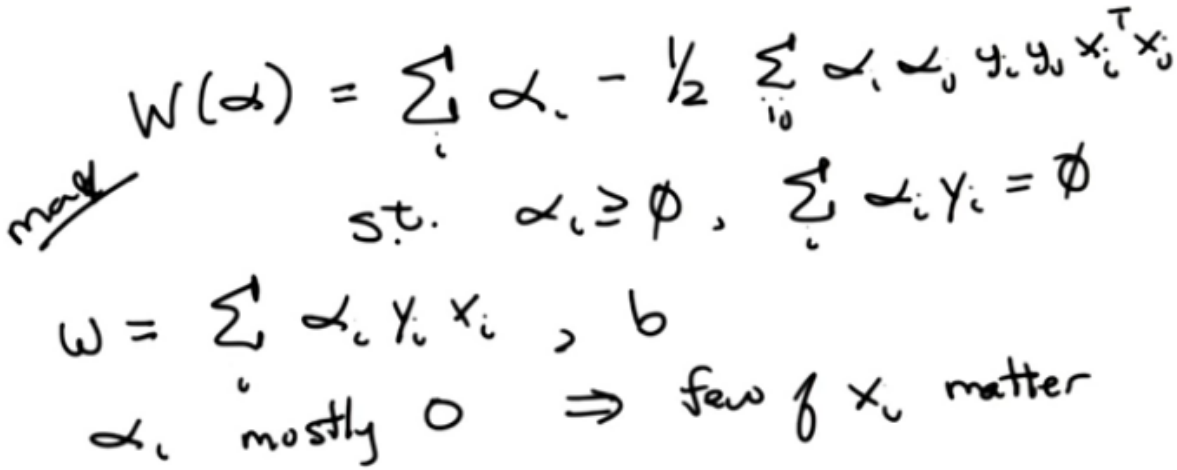
Hard Stuff, Leap of Faith



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- Instead of solving "max $\frac{2}{||w||}$ problem, is better to solve "min $\frac{1}{2}||w||^2$" problem. (Because lengths are positive and now we get good convex function to optimize, easier to compute by algorithm. Its a quadratic programming problem, and people know how to solve it. It has unique solution.)
- Another (not to be questioned!!) change is to convert the quadratic programming eqn to maximizing Lagrange Multiplier optimization form.
- Its sum of alphas (for all i data points) minus half time, every pair points, x and label(y) values, multiplied by own alphas; with constraints such as alphas are positives and the other constraint shown.

Just assume, This Works!!

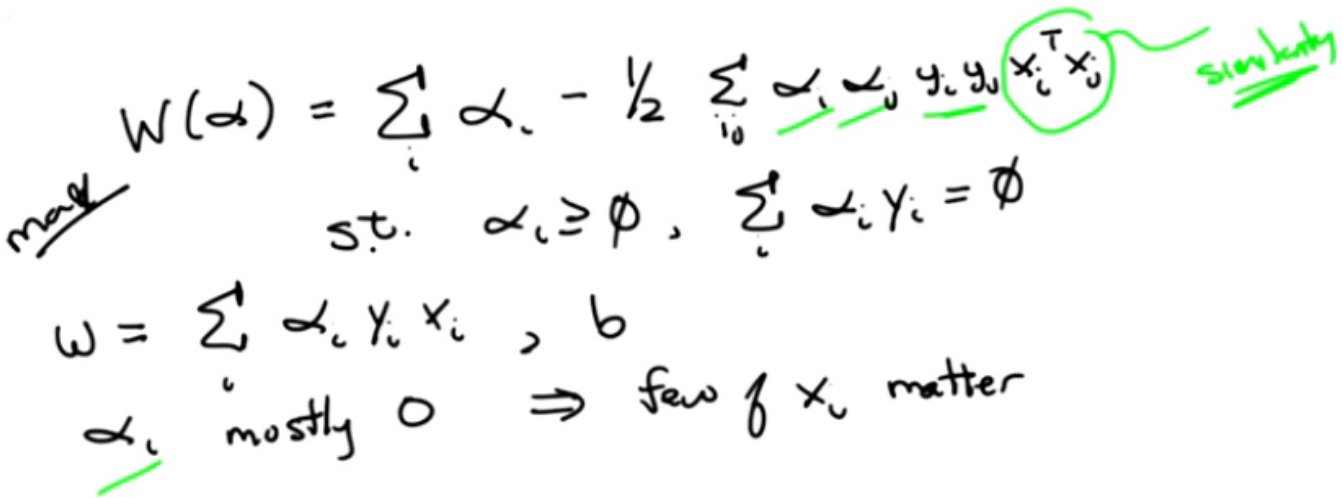


(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

Properties:

- Maximizing alphas, then W can be calculated the b can be calculated.
- Lets make small w is the 2nd term.
- Most of the alphas are 0. Means only few points are part of the summation. The points for which alphas are non-zero are called support vectors. They are the points on the boundary lines.
- So, this is the Machine that needs only Support Vectors.

SVM Derivation Closure



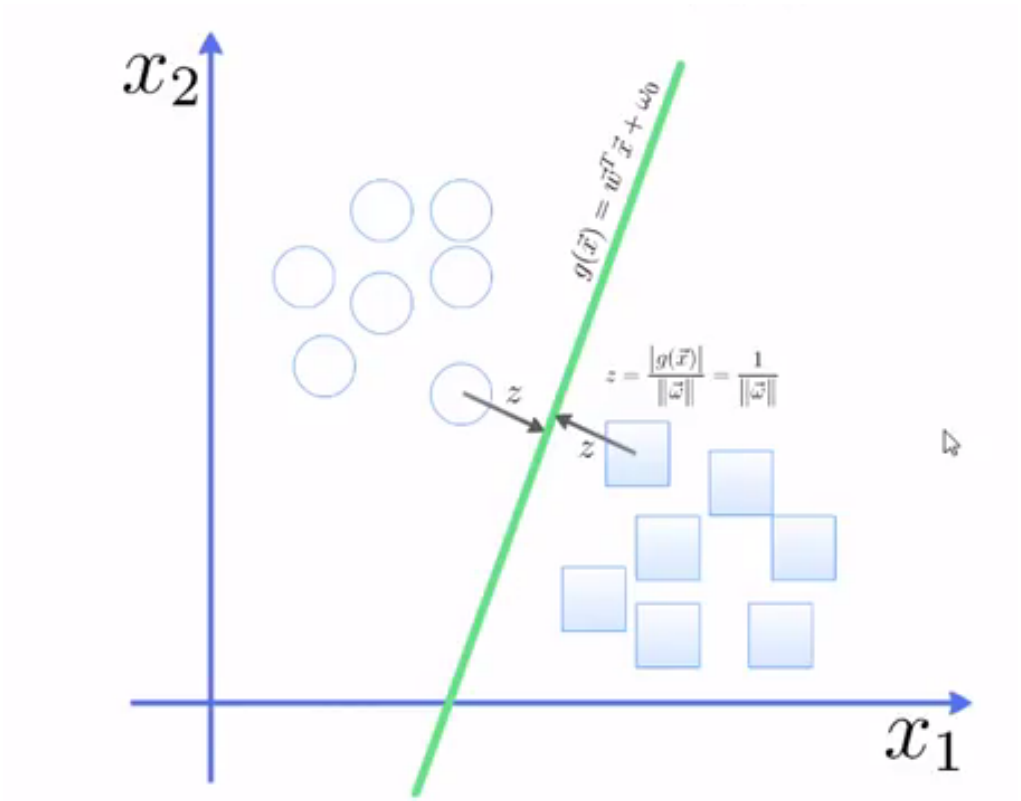
(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

Properties:

- Points away DO NOT MATTER. Its like Nearest Neighbor where only local points matter!! But here K is not provided, but we already know support points (and thus their count, ie K) by quadratic programing
- The x product terms, are the dot products. If both are in same line, its a large number, but if they are perpendicular its towards 0. Its similarity.
- So, find all support points, and from that keep only collinear like points.
- This maximizes W, and thus the width.

Back to SVM Intuition, old symbols

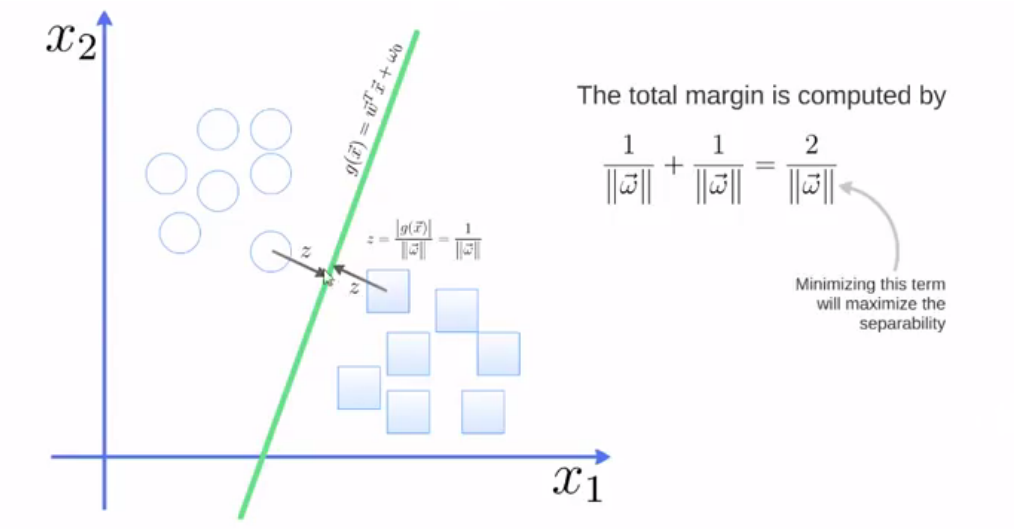
SVM Intuition



(Ref: How Classification Algorithms Work? - Thales Sehn Korting)

- Distance of a point from a hyper-plane is z
- Given by a formula $z = \frac{|g(x)|}{||w||}$, is also called 'margin'

SVM Intuition

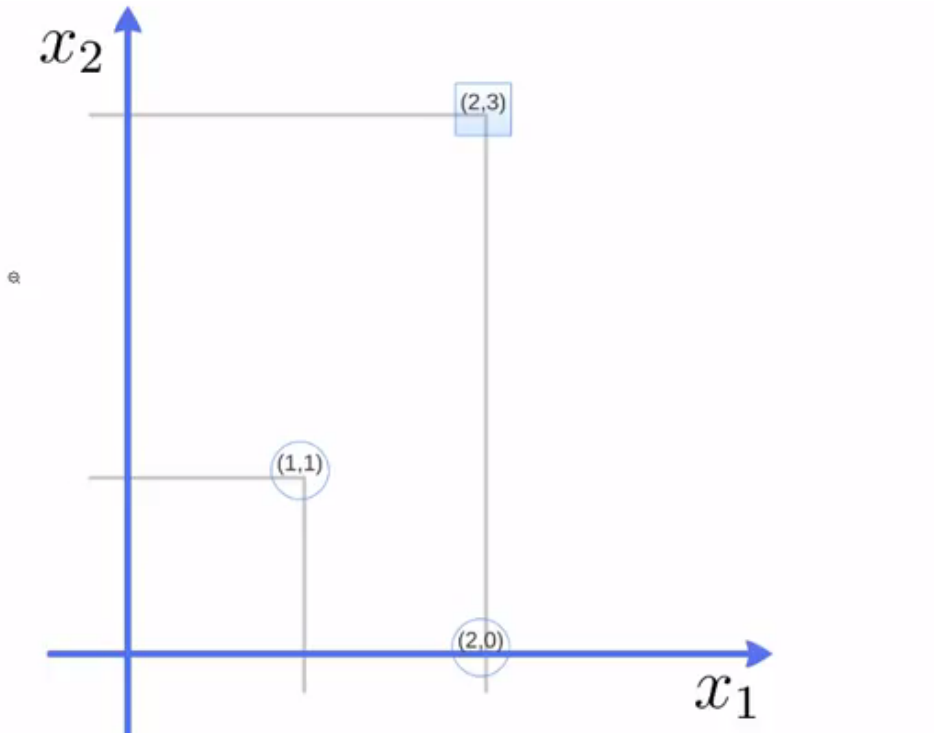


(Ref: How Classification Algorithms Work? - Thales Sehn Korting)

- For any point, $g(\vec{x})$ is atleast "1", on either side (this is the minimum-est, g(x) can get, from the formula)
- So min distance on either side is $\frac{1}{||\vec{w}||}$
- Total margin from both sides is $\frac{2}{||\vec{w}||}$
- So, if we minimize $\vec{w}$, then margin will be maximum.

SVM Example

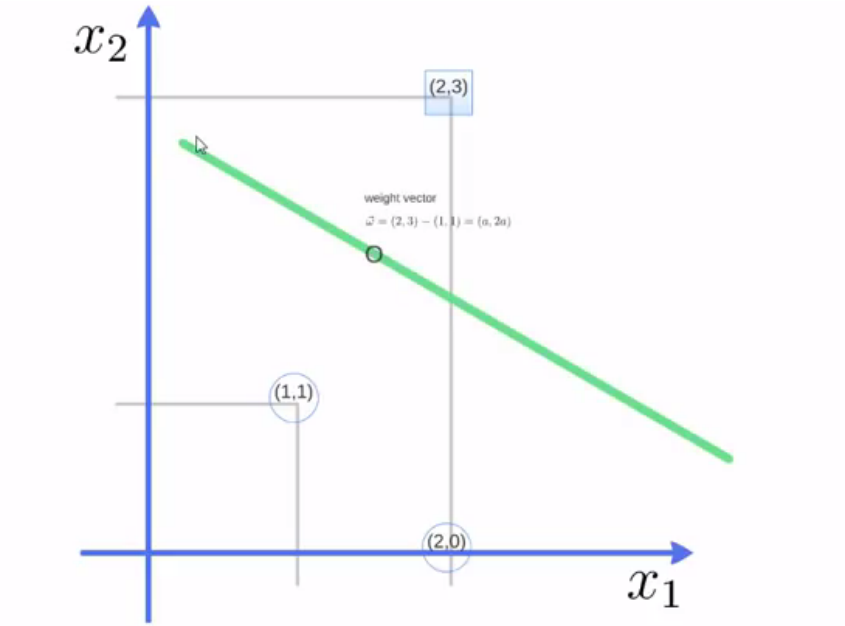
SVM Example



(Ref: How Classification Algorithms Work? - Thales Sehn Korting)

- Lets say we have just 3 points, 2 circles ([1,1] and [2,0]) and one square [2,3]
- We want to design SVM classifier, ie the Hyper-plane

SVM Example



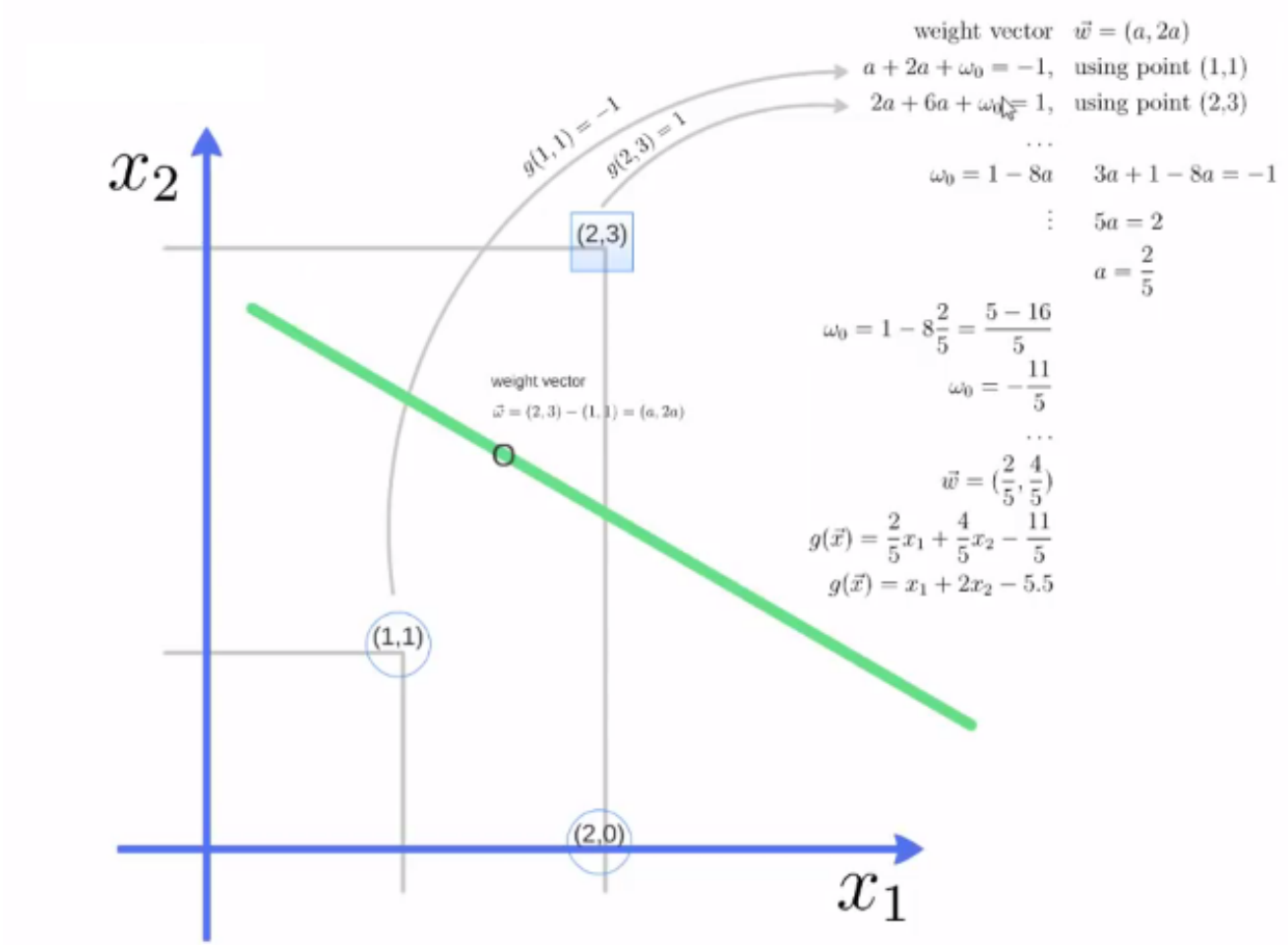
(Ref: How Classification Algorithms Work? - Thales Sehn Korting)

- We know that the hyper-plane line would be equidistant from two closest points (2,3) and (1,1)
- So, generically the weight vector would be of the form $\bar{w} = (2,3) - (1,1) = (a, 2a)$
- Need to find a and then $\bar{w}, w_0$

SVM Example

- weight vector $\bar{w} = (a, 2a)$
- Using Negative point (1,1), putting in equation $g(\bar{x}) = \bar{w}x + w_0 = y$ is $g(1,1) = (a, 2a).(1,1) + w_0 = -1$, is $a + 2a + w_0 = -1$
- Using Positive point (2,3), putting in equation $g(\bar{x}) = \bar{w}x + w_0 = y$ is $g(2,3) = (a, 2a).(2,3) + w_0 = 1$, is $2a + 6a + w_0 = 1$
- Solving these two equations simultaneously. From 1st, $w_0 = 1 - 8a$. Putting it in 2nd $3a + 1 - 8a = -1$, thus $5a = 2$ so $a = \frac{2}{5}$
- Putting a in the first equation, $w_0 = 1 - 8\frac{2}{5} = \frac{11}{5}$
- Thus , $\bar{x} = (\frac{2}{5}, \frac{4}{5})$
- Putting in original Hyperplane equation $g(\bar{x}) = \frac{2}{5}x_1 + \frac{4}{5}x_2 - \frac{11}{5}$
- Being equal to 0 for the middle line, it can be simplified to $g(\bar{x}) = x_1 + 2x_2 - 5.5$

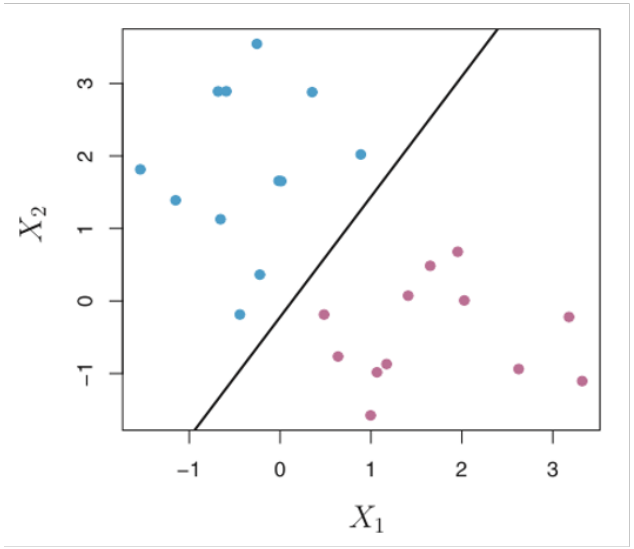
SVM Example



(Ref: How Classification Algorithms Work? - Thales Sehn Korting)

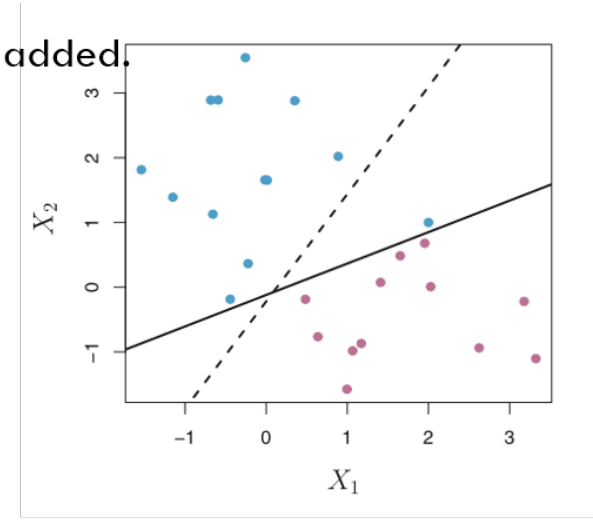
SVM Aspects

New point added



Maximum margin classifier.
Perfectly segments training data.

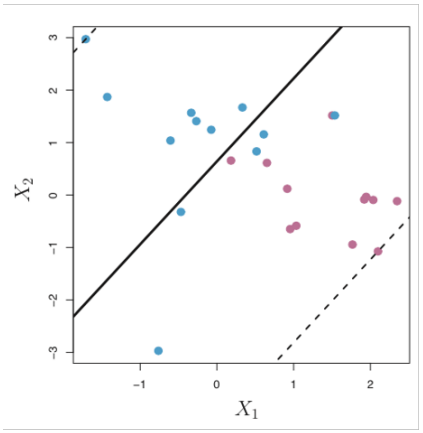
New data point added.



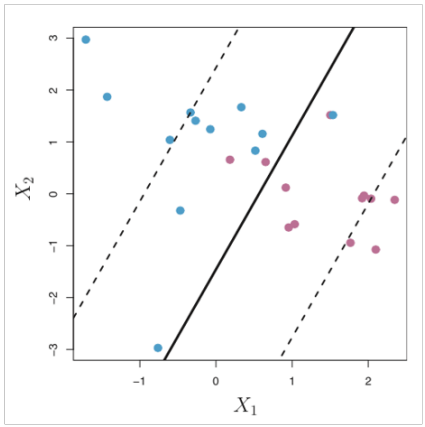
Dramatic shift in maximal margin hyperplane.
Model has *high variance* when trying to maintain perfect segmentation.

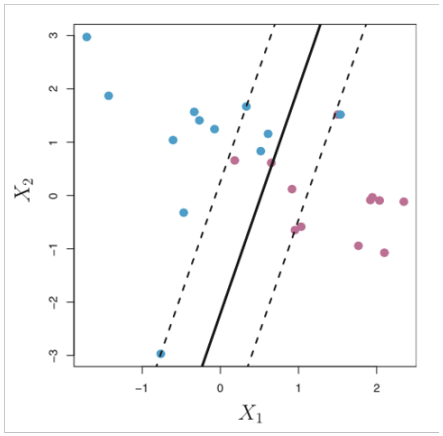
Constructing the Support Vector Classifier

- How much "softness" (mis-classifications) is ideal?
- Specification of non-negative tuning parameter C
  - Generally chosen by analyst following cross-validation
  - Large C: wider margin; more instances violate margin
  - Small C: narrower margin; less tolerance for instances that violate margin



Lower variance.

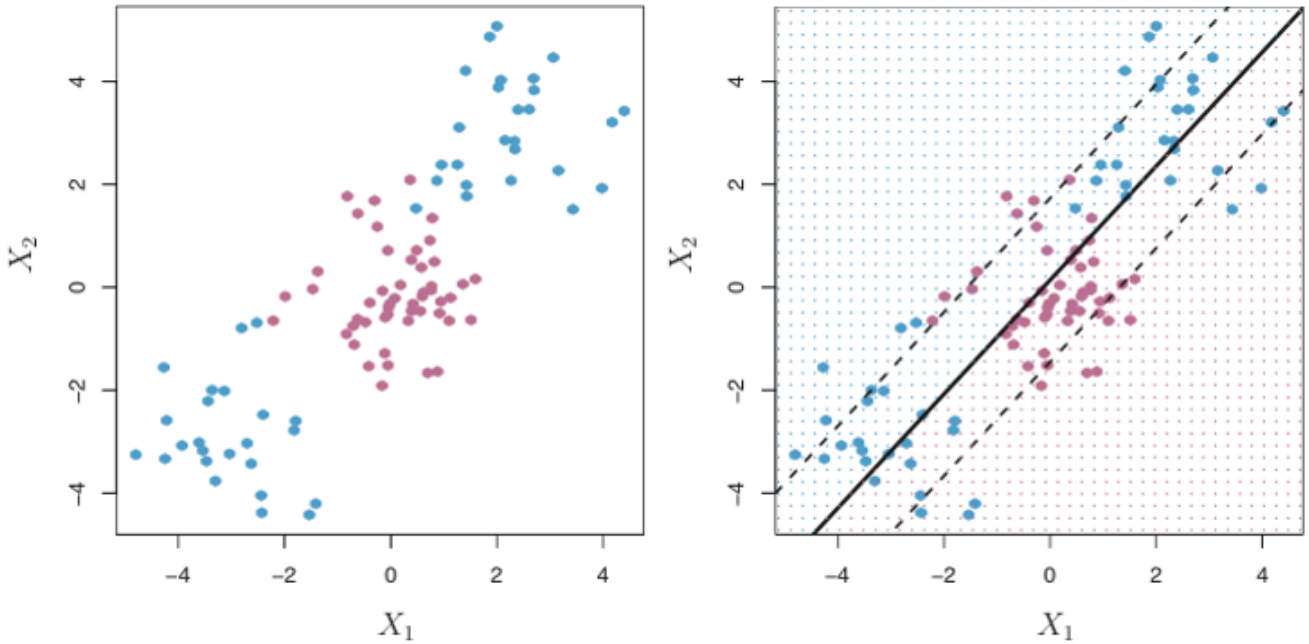




Higher variance.

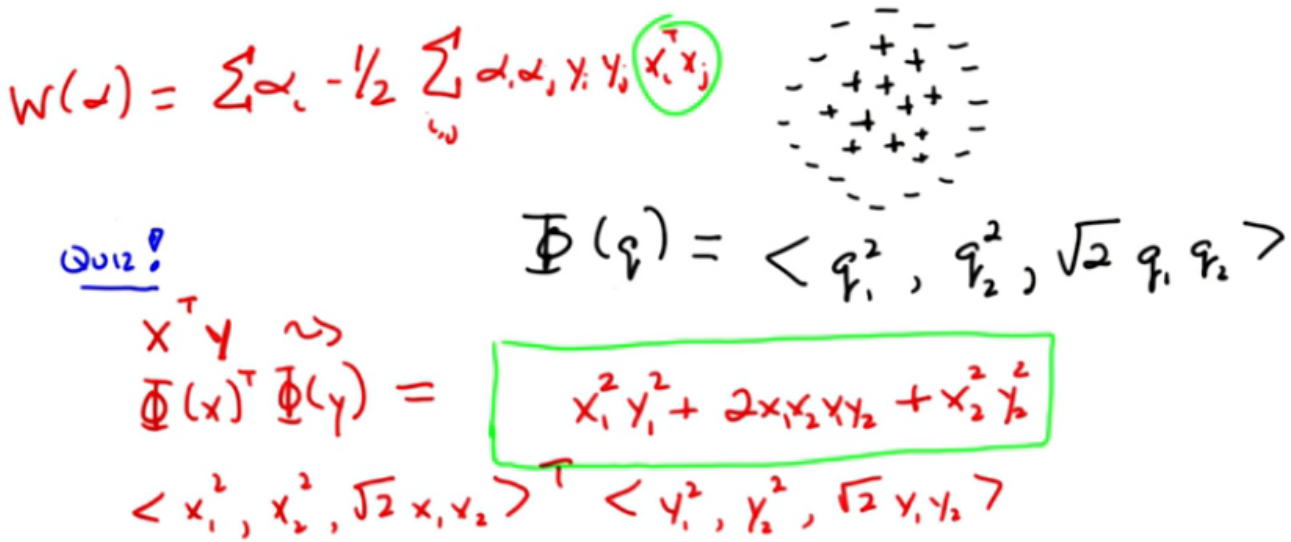
Non-linear decision boundary?

What if a non-linear decision boundary is needed?



Poor performance using this decision boundary. Can we twist the boundary? Or Can we twist the data?

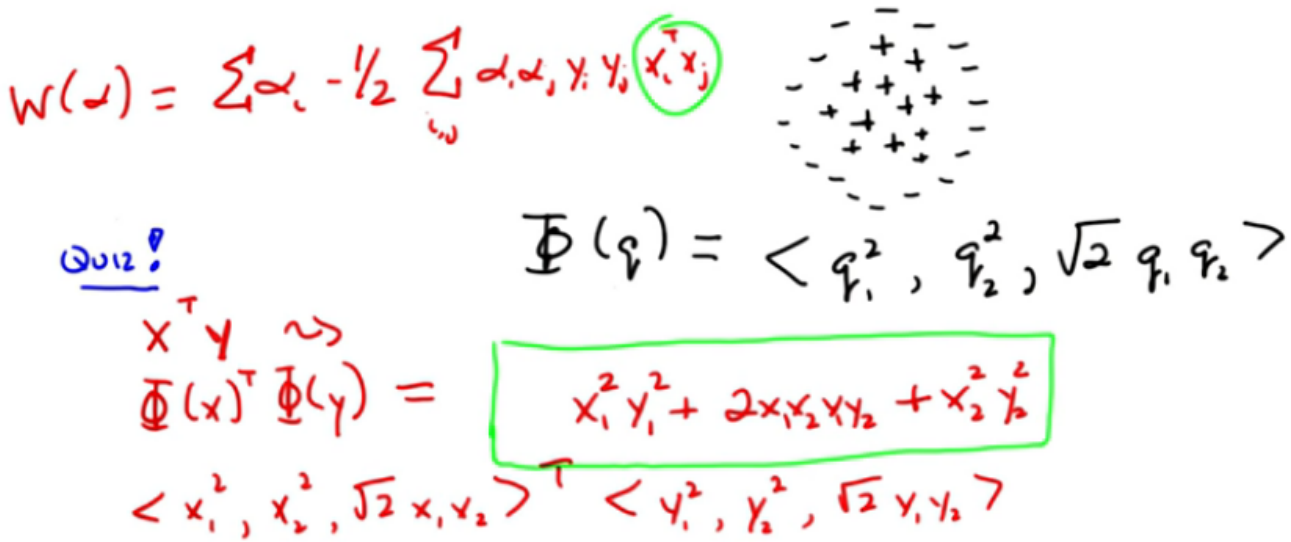
Transformations



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- Each data point is transformed into another space, with multiple value tuple
- Not changing original data, but creating new features (temporarily) for classification
- In Quadratic optimization problem, the $x_i^T x_j$ term, is turned into the tuple. So, for points like $(x_1, y_1), (x_2, y_2)$ the tuple becomes, for x vector $x_1^2, x_2^2, \sqrt{2}x_1x_2$ and for y vector $y_1^2, y_2^2, \sqrt{2}y_1y_2$

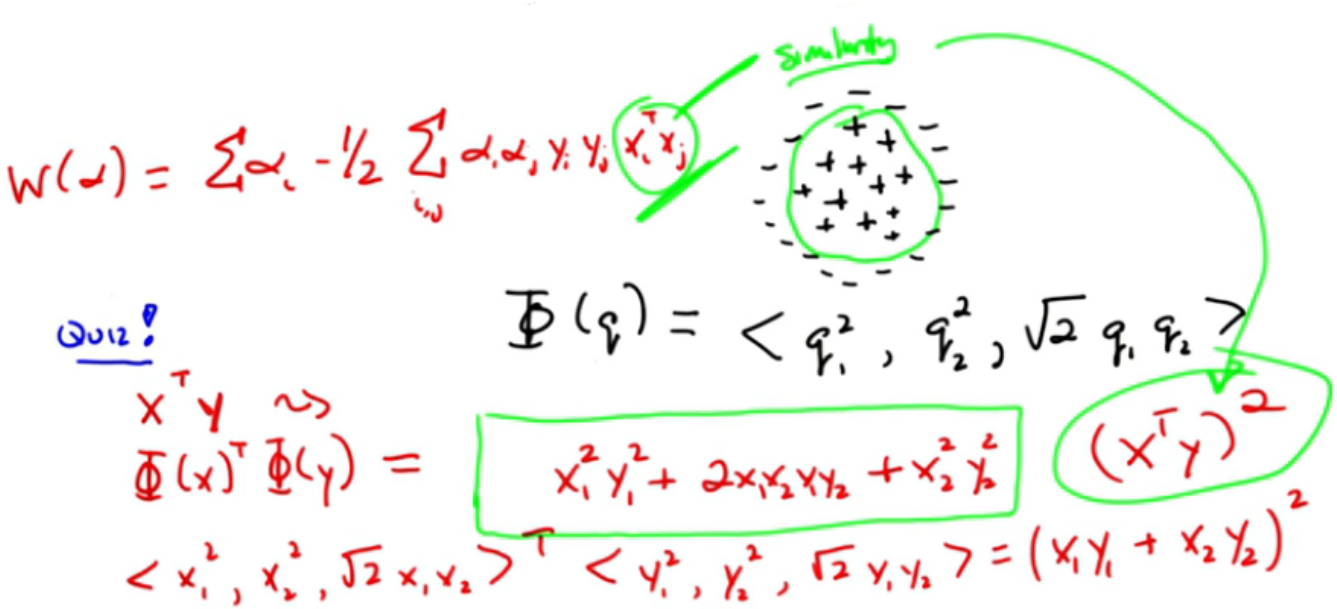
Transformations



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- Dot product of both vectors is: $x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2$, remember anything?
- Factorize $x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2$ to get $(x_1 y_1 + x_2 y_2)^2$, its actually $(X^T y)^2$
- Does that mean the dot product got transformed into dot product's square? Made a circle.

Transformations



(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- From 2D to 3D (tuple), the pluses have got above, the minuses have gone below, and thus we can have a separating Hyperplane
- Now we can have Circular Boundary of separation, and can be seen projection.
- Actually we don't need to form tuple, just square the dot product term, and then do quadratic programing, then its able to separate. That happens when you say "rbf" kernel in place of "linear" kernel.

Transformations

$$W(\alpha) = \sum \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

→ similarity

→ domain knowledge

$$K = (x^T y)^2$$

$$K = x^T y$$

$$K = (x^T y + c)^p$$

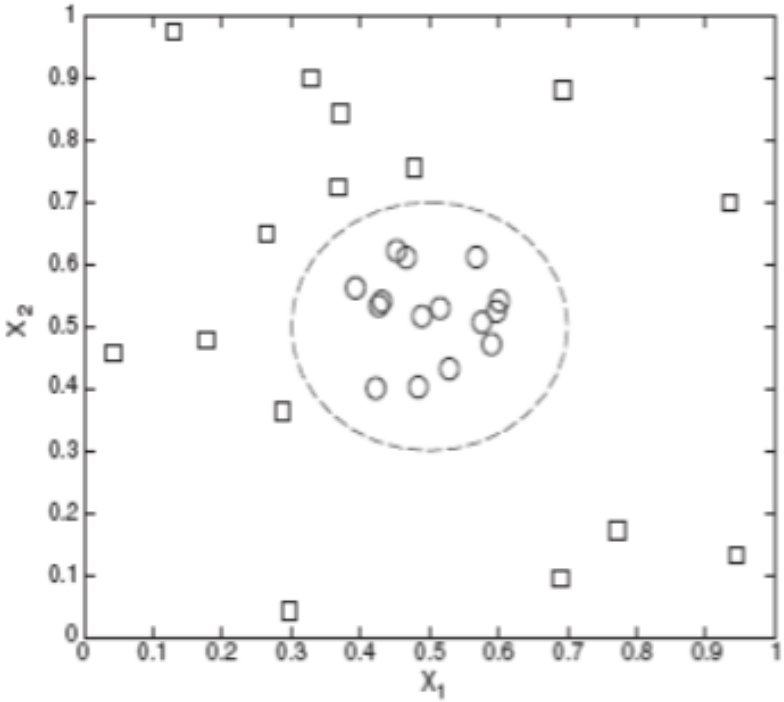
$$K = e^{-(\|x-y\|^2/2\sigma^2)}$$

$$K = \tanh(\alpha x^T y + \theta)$$

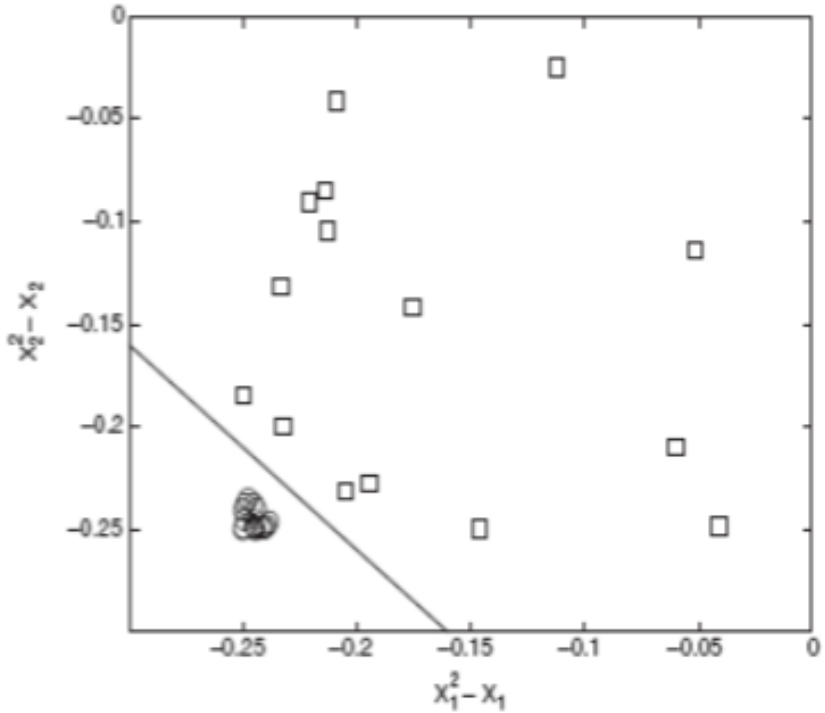
(Ref: Support Vector Machine - Georgia Tech - Machine Learning - Udacity)

- Lets call the kernel as function $K()$, its still a notion of similarity.
- Projecting in higher dimensional space then do linear separation.
- Many kernels are possible.
- (Mine: Please note that the y shown here is not a label, but the data points x and y.)
- The kernel functions have to follow the "Mercer Condition" (not to be discussed!!, its well behaved distance function.)

Kernel Tricks: Data Transformation



(a) Decision boundary in the original two-dimensional space.



(b) Decision boundary in the transformed space.

Even adding a third dimension and then separating 2D points in 3D is possible.

(Reference: Introduction to Machine Learning - Jeff Howbert)

SVM - Multiclass

Multiclass Problems

- Scenario: target class is more than 2 categories
- How to extend binary classifiers to handle multi-class problems?

#1 - Multiclass: One-Against-Rest

- Assume multi-class data-set with K target classes
- Decompose into K binary problems
- Idea: For each target class yi create a single binary problem, with classifier Ci: Class yi: positive ; All other classes: negative;
- Training: use all instances; each instance used in training each of the Ci classifiers;
- Testing: run testing instance through each classifier; record votes for each yi class (negative prediction is a vote for all other classes); class with most votes is the predicted class

#2 - Multiclass: One-Against-One

- Assume multi-class data-set with K target classes
- Train K(K-1)/2 binary classifiers (many more than One-Against-Rest)
- Idea: Each classifier distinguishes between pair of classes (yi, yj); Classifier ignores records that don't belong to y_i or y_j
- Training: use all instances; each instance only used in training "relevant classifiers" (K of them)
- Testing: run testing instance through each classifier; record votes for each y_i class; class with most votes is the predicted class;

SVM (Recap)

- Maximum Margin Classifier is natural way to perform classification if a separating hyper-plane exists.
- This generalization is called: support vector classifier
- In many cases, no separating hyper-plane will exist. So need to loosen a bit.
- Maximal Margin Classifier: no training errors allowed
- Support Margin Classifier: tolerate training errors. Approach: Soft margin
- Will allow construction of linear decision boundary even when classes are not linearly separable. Kernel Tricks.

Support Vector Machine with Scikit-Learn

Support Vector Machine

```
# Support Vector Machine
from sklearn import datasets
from sklearn import metrics
from sklearn.svm import SVC
# load the iris datasets
dataset = datasets.load_iris()
# fit a SVM model to the data
model = SVC()
model.fit(dataset.data, dataset.target)
print(model)
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

(Ref: Machine Learning Algorithm Recipes in scikit-learn - Jason Brownlee)

Naive Bayes

Naive Bayes

What is it ?

- Statistical method for classification.
- Supervised Learning Method.
- Assumes an underlying probabilistic model, the Bayes theorem.
- Can solve problems involving both categorical and continuous valued attributes.
- Named after Thomas Bayes, who proposed the Bayes Theorem.

Bayes Theorem

Probability (recap)

- Single Probability: "X has the value x" $P(X = x)$
- Joint probability: "X and Y": $P(X = x, Y = y)$
- The probability that variable X takes on the value x and variable Y has the value y: $P(X, Y)$
- Conditional probability: "Y" given observation of "X": $P(Y = y|X = x)$
- If "X" and "Y" are independent then $P(X, Y) = P(X) \times P(Y)$
- For dependent "X" and "Y" : $P(X, Y) = P(Y|X) \times P(X)$

Probability (recap)

Derivation

$$P(X,Y) = P(Y \mid X) \times P(X)$$

$$P(Y,X) = P(X \mid Y) \times P(Y)$$

$$P(X,Y) = P(Y,X)$$

$$P(X,Y) = P(Y \mid X) \times P(X) = P(X \mid Y) \times P(Y)$$

Bayes' Theorem: $$P(Y \mid X) = \frac{P(X \mid Y) \times P(Y)}{P(X)}$$

Example

- On a college campus, you meet a guy, say "Tom". He appears Shy.
- You are asked to guess, whether Tom is a Math PhD student or a Business MBA student. Assume that these are the only two types in that campus.
- Your guess is probably: Math PhD student, assuming that we have seen many Math PhD students, Shy.
- Let us check if this is true, using Bayes theorem.

(Ref: A visual guide to Bayesian thinking)

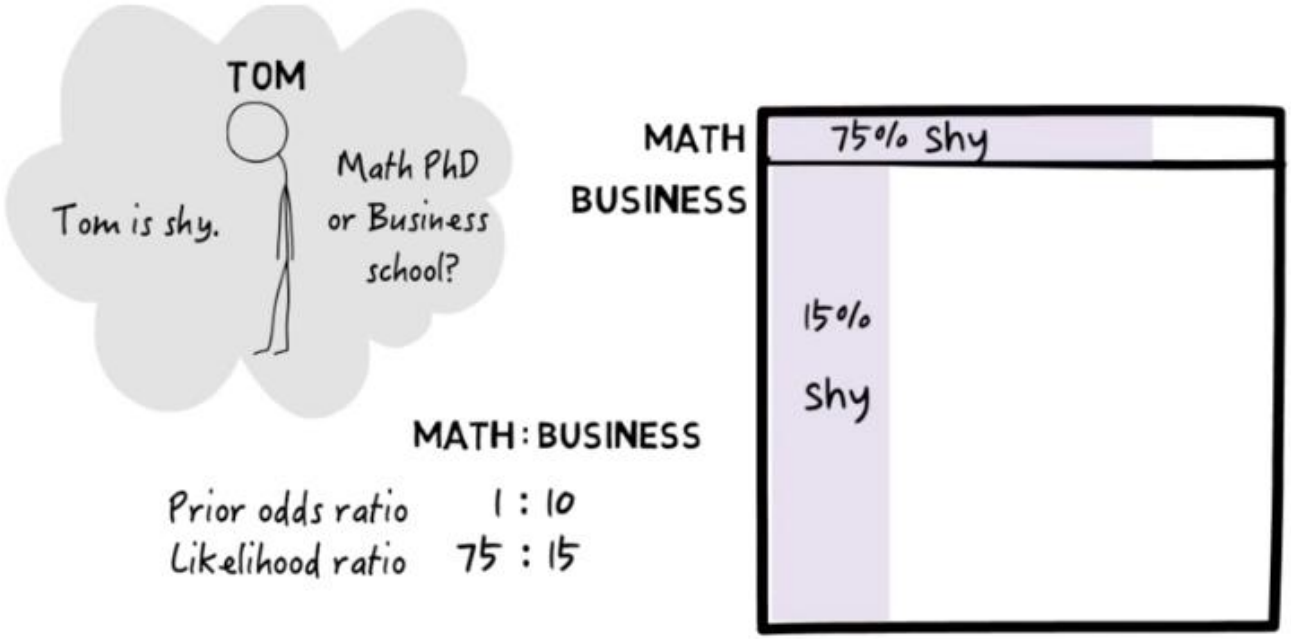
Example

- We need to see, how many Math PhD students are there? Compared to Business MBA students. Say 1: 10. This is called Prior ratio. It is the ratio if the domain or base itself in the population.
- Within Math PhD students, you find as many as 75% shy folks.
- Within Business MBA students, you find just about 15% shy folks.
- These above to form a likelihood ratio.

(Ref: A visual guide to Bayesian thinking)

Example

We need to find where Tom belongs. One thing for sure, being Shy, he must be in one of those shaded patches (shown below)



Just by looking at it the Business, shaded portion looks twice the Math shaded portion. Even though individual percentages show that Math are higher. But there are higher within smaller base!!

(Ref: A visual guide to Bayesian thinking)

Example

- If you multiply corresponding values, $1x75 : 10x15$, it is $75 : 150$, ie $1 : 2$
- So that's the shaded area ratio. Note that Math is in column 1, Business column 2, is 2.
- That's called Posterior Probability.
- So, Tom is more likely to be a Business MBA guy than Math PhD!!

(Ref: A visual guide to Bayesian thinking)

Bayes Learning

(Ref: Georgia tech, Udacity, Machine Learning)

Bayesian Learning

- Goal of machine learning is: We want best model/function for the given data.
- Can we say? Its equivalent to: We want most "probable" model, for the given data.
- That's represented by: $P(h|D)$ where h is hypothesis (think of it as a function) and D is data.
- We can try many hypotheses, and find max of them.

Bayes Rule (recall)

- $P(h|D) = \frac{P(D|h)P(h)}{P(D)}$
- Chain Rule: $P(a \cap b) = P(a|b)P(b) = P(b|a)(a)$ as its symmetric
- Better way to understand is $P(a|b)$ is probability of a given b, meaning within b, how much is a, ie the intersection of a and b.
- So $P(a|b)$ is $\frac{a \cap b}{b}$, right?

De-constructing the formula

- The bottom term, $P(D)$ is called as Prior on the Data, ie belief of seeing the data.
- As we are comparing many hypotheses, and this bottom term being there in all such calculations, one can ignore it. It's a normalization term.
- $P(D|h)$ is like learning backwards. Its easy to see it in the machine learning dataset. Its basically the LIKELIHOOD of seeing Data given the hypothesis.
- Note : $D = (x_i, y_i)$
- So it means, given x_i, and the hypothesis that we are testing, meaning the function, whats the likelihood of seeing y_i?

Example

- Hypothesis h = return true if $x > 10$
- Data $x = 7$
- Now the question is what is the probability of $y == True$.
- Its 0.
- What for False?
- Its 1.

Back to formula

- As we have labeled data, its easier to compute/adjust $P(D|h)$ (as h and D are known) than $P(h|D)$
- $P(h)$ is prior on h.
- Probability of this hypothesis h occurring amongst all hypotheses.
- Its actually domain knowledge, the features, the type of model we are trying.
- It's the probability of selected features or model.

Properties

- If we have left hand side, ie ,$P(h|D)$ to go up, what should be changed on the right hand side?
- $P(h)$ can go up. Probability of getting good hypothesis, ie better features or model, without seeing data, can go up
- $P(D|h)$: Once you see data, for that data, if you get that nicely predicting with h, then we are good. Good accuracy giving h.
- $P(D)$ cannot change, (ie cannot go down)

Example

- A man goes to a doctor. He gives him a test for a rare disease. The test generally returns correct positive results 98
- $TP = 0.98, TN = 0.97, FP = 0.02, FN = 0.03$
- The disease in general, in the population is to only .8% people.
- The result: the test comes positive.
- Question: Does he really have the disease?

Example

- We are looking for intersection set of $test = positive$ and $disease = true$
- Let's apply Bayes rule to see probability of disease being there given the test has come out positive.
- $P(disease = true|test = positive) = \frac{P(test=positive|disease=True)P(disease=True)}{P(test=Positive)}$
- Let's see the probability of not having the disease even of the test said true
- $P(disease = false|test = positive) = \frac{P(test=positive|disease=false)P(disease=False)}{P(test=Positive)}$
- Which is bigger?
- As you can see, for comparison, the denominators are same, so we need not calculate it.

Solution

- $P(disease = true | test == positive) = 0.98x0.008 = 0.00784$ (ignored denominator term)
- $P(disease = false | test == positive) = 0.03x0.992 = 0.02976$. This is bigger.
- SO, don't Panic!!
- Just within results ie 0.00784 and 0.02976, Probability of disease not being there, is $0.02976/(0.02976 + 0.00784) = 0.79$, ie about 79% probability that you won't have disease even if the test was Positive. Only remaining 21% chance that you may have the disease.

Algorithm

```
For each h in H
    Calculate p(h|D) = P(D|h)P(h)/P(D)
Output:
    H = argmax_h P(h|D)
```

- Good part is, as we want argmax only, we don't need denominator P(D) , which is hard to calculate anyway.
- $P(h)$ is also hard to compute. So if we further drop that and find $argmax P(D|h)$ than its called Maximum Likelihood Hypothesis.
- That just makes an assumption, that all $P(h)$ are same.
- All model or feature scheme are equally likely

Bayesian Classifier

Bayes Classifier

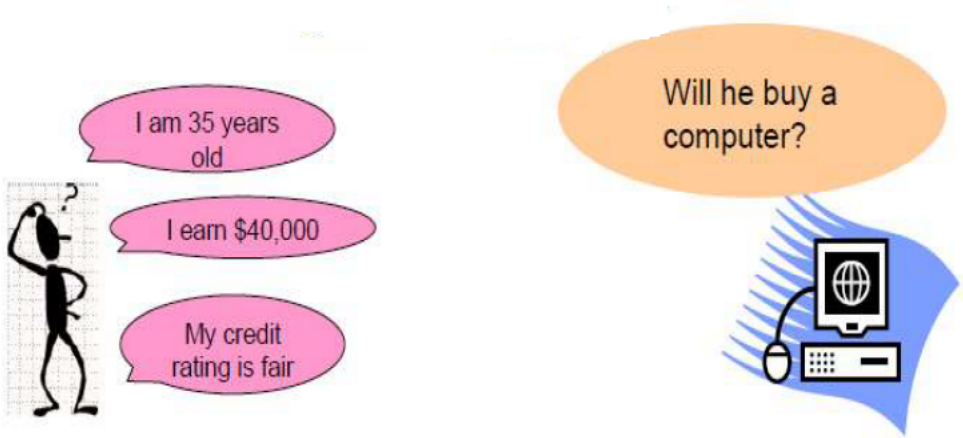
- A classifier is a function
- It takes features as inputs, so $x_1, x_2 \ldots$
- Gives y as output, which could be $y = 1, y = 2, \ldots$. These are the classes. For binary classifier y can take only $0, 1$.
- x can be real or numeric and it could also be categorical.
- With training data, the function is learnt. That's the classifier model.

Bayes Classifier

- Bayesian Classifier is probabilistic.
- Meaning, it does not produce y class values directly, but it gives probabilities of $y = 0$ and $y = 1$
- We pick max and say that, that's the output class
- $\hat{y} = argmax P(y|x)$

Bayesian classification

Example



- X: Independent Variables: Age, Income, Credit Rating
- Y: Dependent Variable: Will buy computer or not? Binary.

Bayesian classification

Bayesian probability of a class:

$$P(y|x) = \frac{\overbrace{P(x|y)}^{\text{class model}}\overbrace{P(y)}^{\text{prior}}}{\underbrace{\sum_{y'} P(x|y')P(y')}_{\text{normalizer } P(x)}}$$

- Where denominator term of Bayes Theorem: $P(Y|X) = \frac{P(X|Y)(P(Y)}{P(X)}$ is expanded
- Its summation of Probabilities that customer is 35 given that he buys computer, customer has fair credit rating given that he buys computer, and so on. And, also similarly for when he does not buy computer.

Bayesian classification Example

$$P(y|x) = \frac{P(x|y)P(y)}{\sum_{y'} P(x|y')P(y')}$$

Example:
y ... UK patient has Ebola
x ... observed symptoms

- y is binary, whether Ebola is present or not.
- x are many, whether s/he has temperature, looks pale, and similar symptoms.

Bayesian classification Example

$$P(y|x) = \frac{P(x|y)P(y)}{\sum_{y'} P(x|y')P(y')}$$

Example:
y ... UK patient has Ebola
x ... observed symptoms

- P(y): In general, without considering any features or symptoms, whats the probability of having Ebola.
- Say, in general, probability of having Ebola is very less. Its rare disease in the developed world.
- So, even if there are some symptoms, likelihood of they being for Ebola, is less. That's the role "prior" plays.

Bayesian classification Example

$$P(y|x) = \frac{P(x|y)P(y)}{\sum_{y'} P(x|y')P(y')}$$

Example:
y ... UK patient has Ebola
x ... observed symptoms

- $P(x|y)$: Given that he has Ebola, within this sub population, what are the chances of he having x symptom.
- Gen that he has Ebola how probable is he having (high) temperature.
- Similarly for other symptoms, or Xs

Bayesian classification Example

$$P(y|x) = \frac{P(x|y)P(y)}{\sum_{y'} P(x|y')P(y')}$$

Example:
y ... UK patient has Ebola
x ... observed symptoms

- P(x): It is summation of having certain symptom.
- Its sum of having this symptom with Ebola and not having Eblola
- $P(X) = P(X|y = Ebola)P(y = Ebola) + P(X|y = NoEbola)P(y = NoEbola)$
- Many a times, this term is left out from the formula, as it does not take part in the classification.
- Because, regardless of whether the guy has Ebola or not, this is the probability of having certain symptom. Its a constant as its there for both. The same.

Independence assumption

- Trick is to assume that all these x variables are independent, given y.
- That makes combined probability as just multiplication of individual conditional probabilities.
- So, it becomes sum of multiplications only

$$P(x_1 \dots x_d|y) = \underbrace{\prod_{i=1}^{d} P(x_i|x_1 \dots x_{i-1}, y)}_{\text{chain rule (exact)}} = \underbrace{\prod_{i=1}^{d} P(x_i|y)}_{\text{independence}}$$

- Individual probabilities are: whats probability of a particular pixel being black given the digit is 3.
- Multiply for all variables Xs.

Independence assumption

- Joint probability of Y and many features $X_1, X_2, X_3, \ldots, X_n$ can be expressed as the chain rule:
  $P(Y, X_1, X_2, X_3, \ldots, X_n)$
  $= P(X_1, X_2, X_3, \ldots, X_n|Y)P(Y)$
  $= P(Y)P(X_1, Y)P(X_2|Y, X_1)P(X_3|Y, X_1, X_2)\ldots$
- This is messy, but if we naively assume independence, then
  $P(X_2|Y, X_1) = P(X_2|Y)$
  $P(X_3|Y, X_1, X_2) = P(X_3|Y)$
  $P(X_n|Y, X_1, X_2, \ldots) = P(X_n|Y)$

Why is Bayes Classifier so popular?

- Look at what it eventually comes down to.
- Just some counting and multiplication.
- We can pre-compute all these terms, and so classifying becomes easy, quick and efficient.

Naive Bayes Example - Fruits Classification

Example

- So, let's say we have data on 1000 pieces of fruit.
- we know 3 features of each fruit, whether it's long or not, sweet or not and yellow or not
- Outcome is the type: a Banana, Orange or some Other fruit.

Example

Fruit	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Other	100	150	50	200
Total	**500**	**650**	**800**	**1000**

So from the table what do we already know?

- 50% of the fruits are bananas
- 30% are oranges
- 20% are other fruits

Example

More explicit table

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other Fruit	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

This is our 'training set.' We will use this to predict the type of any new fruit we encounter.

Example

We can pre-compute a lot of things about our fruit collection.

- The so-called "Prior" probabilities.
- If we didn't know any of the fruit attributes, this would be our guess.
- These are our base rates. P(Y).
- P(Banana) = 0.5 (500/1000)
- P(Orange) = 0.3
- P(Other Fruit) = 0.2

Example

Probability of "Evidence". P(X)

- P(Long) = 0.5
- P(Sweet) = 0.65
- P(Yellow) = 0.8

Example

Based on our training set we can also say the following:

- From 500 bananas 400 (0.8) are Long, So $P(Long|Banana) = 0.8$
- 350 (0.7) are Sweet. So, $P(Sweet|Banana) = 0.7$
- 450 (0.9) are Yellow, So, $P(Yellow|Banana) = 0.9$

Example

- Out of 300 oranges 0 are Long. So, $P(Long|Orange) = 0$. [Oranges are never long in all the fruit we have seen.]
- 150 (0.5) are Sweet . So, $P(Sweet|Orange) = 0.5$
- 300 (1) are Yellow. So, $P(Yellow|Orange) = 1$

Example

- From the remaining 200 fruits, 100 (0.5) are Long, So $P(Long|Other) = 0.5$
- 150 (0.75) are Sweet. So $P(Sweet|Other) = 0.75$
- 50 (0.25) are Yellow. So $P(Yellow|Other) = 0.25$
- Similarly, $P(NotYellow|Other) = 0.75$ and so on

Example

Testing

- Given the features of a piece of fruit and we need to predict the class
- Test fruit is Long, Sweet and Yellow
- The one (fruit) with the highest probability (score) being the winner.

Example

$$P(\text{Banana}|\text{Long, Sweet and Yellow}) = \frac{P(\text{Long}|\text{Banana}) * P(\text{Sweet}|\text{Banana}) * P(\text{Yellow}|\text{Banana}) * P(\text{banana})}{P(\text{Long}) * P(\text{Sweet}) * P(\text{Yellow})}$$

$$= 0.8 * 0.7 * 0.9 * 0.5 / P(\text{evidence})$$

$$= 0.252 / P(\text{evidence})$$

$$P(\text{Orange}|\text{Long, Sweet and Yellow}) = 0$$

$$P(\text{Other Fruit}|\text{Long, Sweet and Yellow}) = \frac{P(\text{Long}|\text{Other fruit}) * P(\text{Sweet}|\text{Other fruit}) * P(\text{Yellow}|\text{Other fruit}) * P(\text{Other Fruit})}{P(\text{evidence})}$$

$$= (100/200 * 150/200 * 50/200 * 200/1000) / P(\text{evidence})$$

$$= 0.01875 / P(\text{evidence})$$

By an overwhelming margin ($0.252 >> 0.01875$), we classify this Sweet/Long/Yellow fruit as likely to be a Banana.

Naive Bayes Example - Loan Default

Example

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Target class: Evade
- Predictor variables: Refund, Status, Income
- What is probability of Evade given the values of Refund, Status, Income?
- $P(E|R, S, I)$

Example

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Test instance is:
  - Refund=Yes
  - Status=Married
  - Income=60K
- Issue: we don't have any training example that these same three attributes values.
- How to compute? $P(E|R, S, I)$

Estimating Prior Probabilities

Class target $P(Y)$ (Count these from the table below)

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- $P(Evade=yes)$
- $= 3/10$
- $P(Evade=no)$
- $=7/10$

Estimating Prior Probabilities

Categorical Attributes $P(X_1|Y)$ (Count these from the table below)

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- $P(Refund=yes|Evade=no)$
- $= 3/7$
- $P(Status=married|Evade=yes)$
- $=0/3$
  - *Yikes!*
  - *Will handle the 0% probability later*

Estimating Prior Probabilities

Continuous Attributes $P(X_1|Y)$ (Count these from the table below)

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- For continuous attributes:
  1. Discretize into bins
  2. Two-way split:
     - $(A <= v)$ or $(A > v)$

Estimating Prior Probabilities

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

P(NO) = 7/10
P(YES) = 3/10

P(Refund=YES | NO) = 3/7
P(Refund=NO | NO) = 4/7
P(Refund=YES | YES) = 0/3
P(Refund=NO | YES) = 3/3

P(Status=SINGLE | NO) = 2/7
P(Status=DIVORSED | NO) = 1/7
P(Status=MARRIED | NO) = 4/7
P(Status=SINGLE | YES) = 2/3
P(Status=DIVORSED | YES) = 1/3
P(Status=MARRIED | YES) = 0/3

For taxable income:
P(Income=above 101K | NO) = 3/7
P(Income=below101K | NO) = 4/7
P(Income=above 101K | YES) = 0/3
P(Income=below 101K | YES) = 3/3

Estimating Prior Probabilities

Test instance $X = (Refund = No, Married, Income = 75k)$

P(NO) = 7/10
P(YES) = 3/10

P(Refund=YES | NO) = 3/7
P(Refund=NO | NO) = 4/7
P(Refund=YES | YES) = 0/3
P(Refund=NO | YES) = 3/3

P(Status=SINGLE | NO) = 2/7
P(Status=DIVORSED | NO) = 1/7
P(Status=MARRIED | NO) = 4/7
P(Status=SINGLE | YES) = 2/3
P(Status=DIVORSED | YES) = 1/3
P(Status=MARRIED | YES) = 0/3

For taxable income:
P(Income=above 101K | NO) = 3/7
P(Income=below101K | NO) = 4/7
P(Income=above 101K | YES) = 0/3
P(Income=below 101K | YES) = 3/3

P(X | Class=No) = P(Refund=No | Class=No)
× P(Married | Class=No)
× P(Income=below 101K | Class=No)
= 4/7 × 4/7 × 4/7 = 0.1866

P(X | Class=Yes) = P(Refund=No | Class=Yes)
× P(Married | Class=Yes)
× P(Income=below 101K | Class=Yes)
= 1 × 0 × 1 = 0

Since P(X | No)P(No) > P(X | Yes)P(Yes)

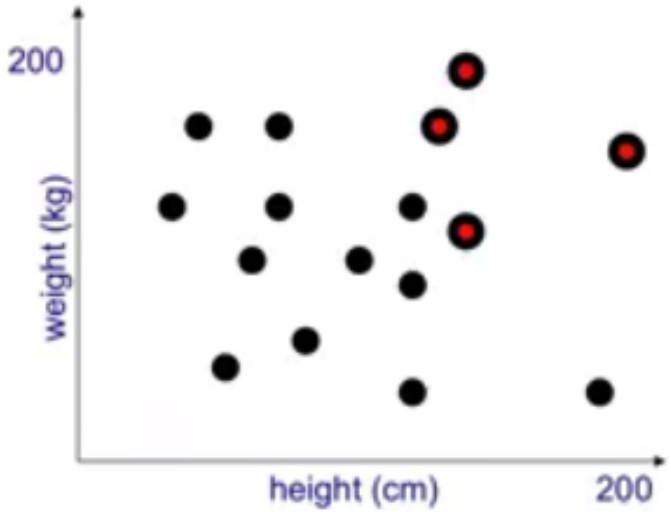
Therefore P(No | X) > P(Yes | X)

=> Class = No

Naive Bayes Classification with Continuous Input Variables

Continuous Variable Example

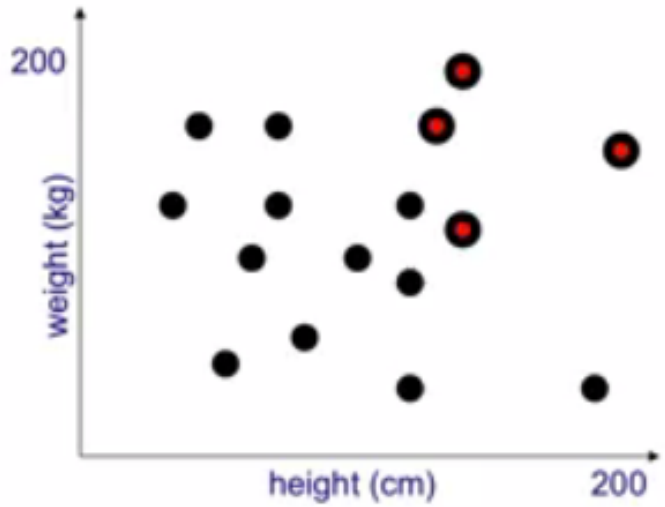
- Features: Height and Weight
- Outcome: class children, adults



Continuous Variable Example

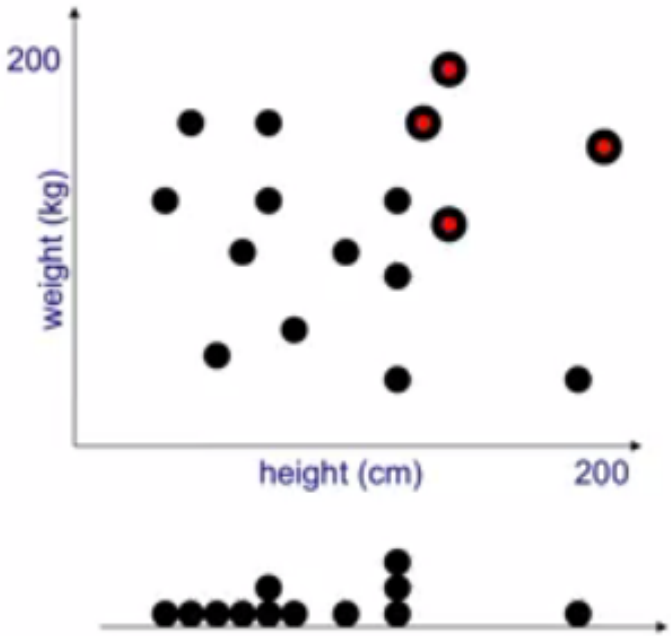
Compute priors, for children and Adults

$$P(a)=\frac{4}{4+12}=0.25\,;P(c)=0.75$$



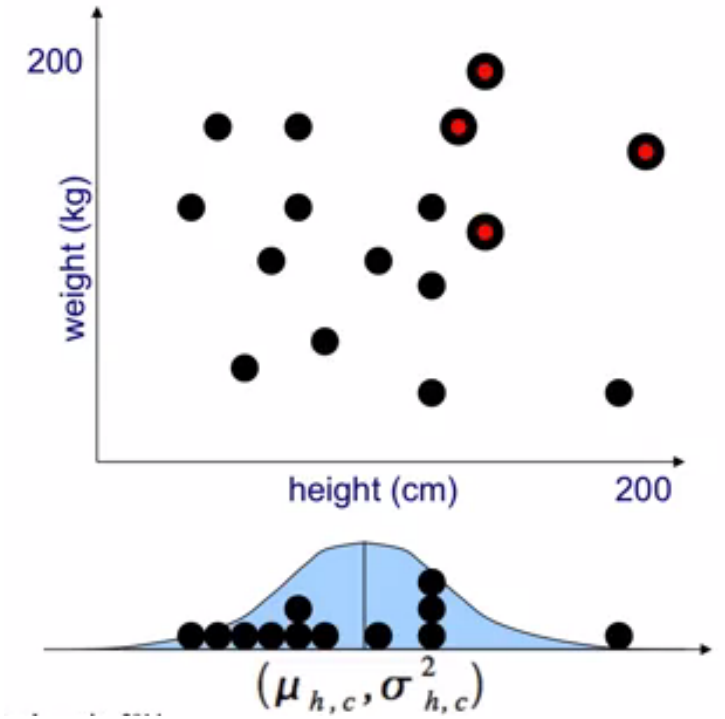
Continuous Variable Example

- Now, within children (black dots), look at only Height for now.
- That would be projection of black dots on X axis.



- One child is really tall.
- Rest are sort of together on smaller values.

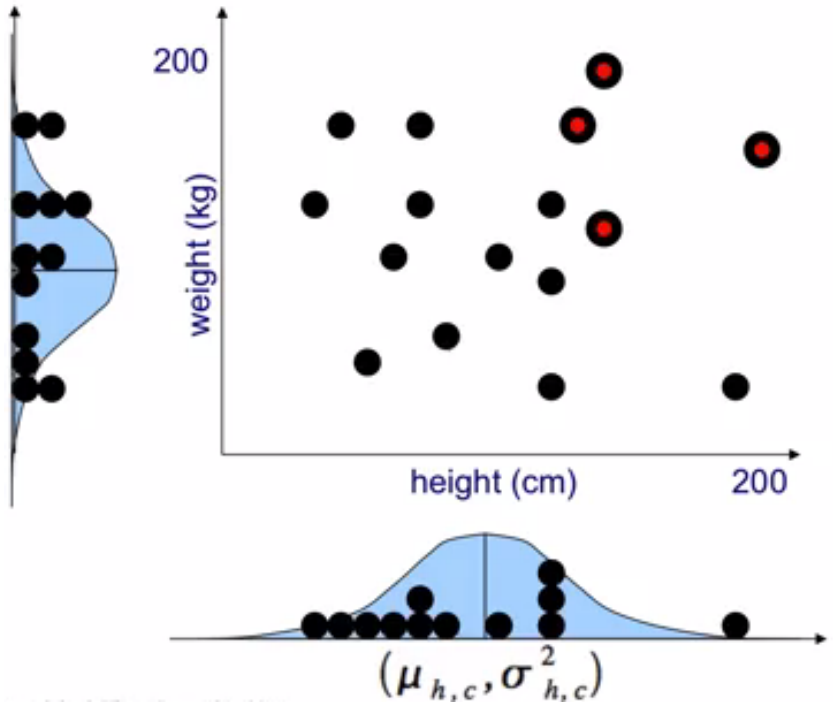
Continuous Variable Example



- Modeling the Histogram with Gaussian (as this is continuous data!!)
- Its not a good fit, but Gaussian is easy to handle.

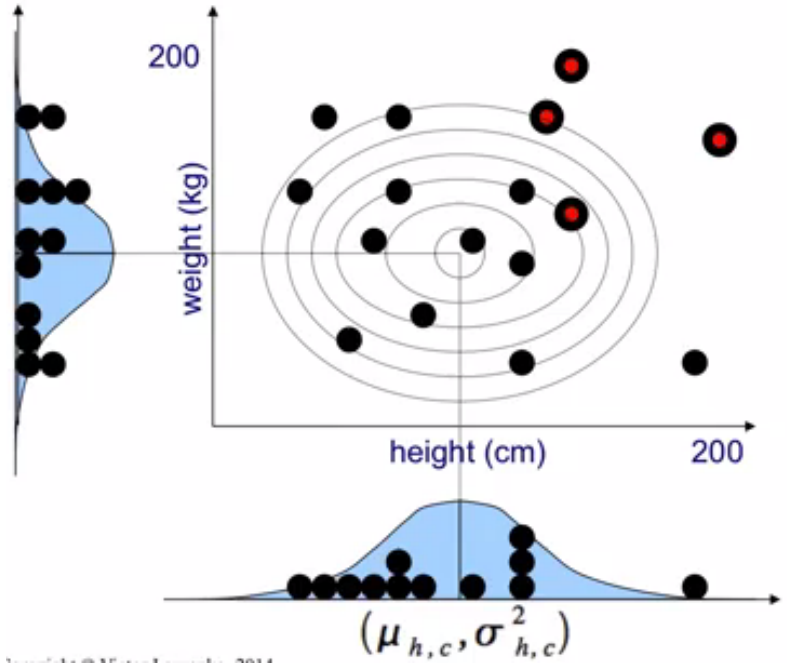
Continuous Variable Example

- Same thing for Weights.
- These, both, Gaussians are independent.
- Their respective values can be multiplied



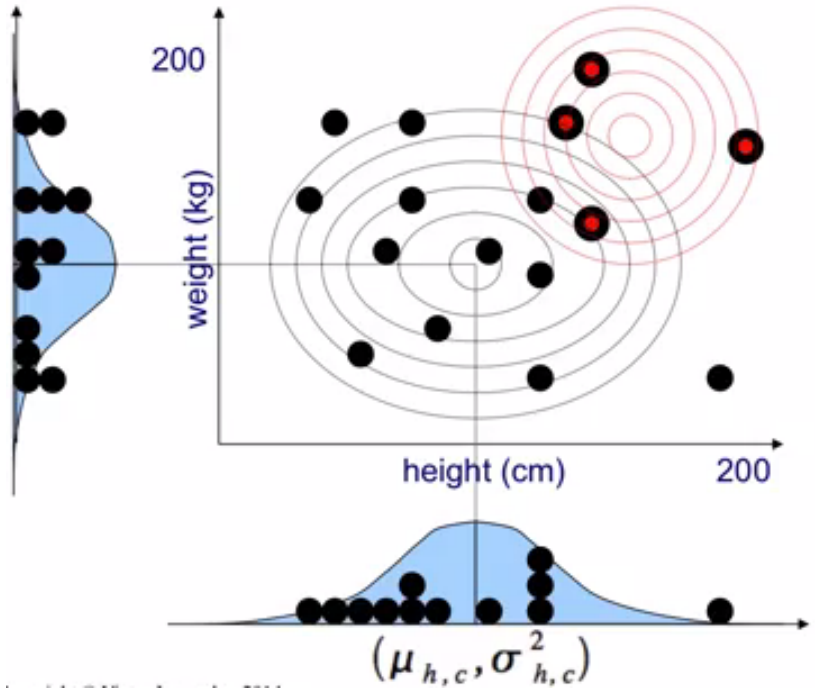
Continuous Variable Example

- Cross plot with contours (iso values)
- Peak in the middle,where Height and Weights are max.
- This is for Children



Continuous Variable Example

- Adding Gaussian plot for Adults as well
- You can start seeing separate regions, ie classification.
- Thats the Gaussian Naive Bayes model



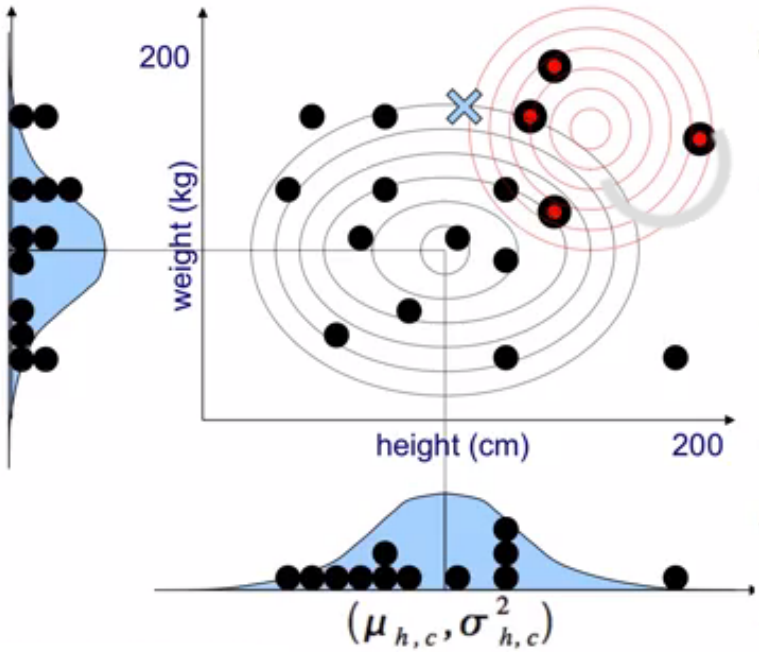
Continuous Variable Example

- A test point needs to see if its a child or an adult
- He has height as well as weight
- Plug those in for Gaussian of Height and Weight and for both Adults and Children
- Combine them using Bayes rule

Continuous Variable Example

Continuous example

$$P(a)=\frac{4}{4+12}=0.25\,;\,P(c)=0.75$$



$$p(h_x|c)=\frac{1}{\sqrt{2\pi\,\sigma^2_{h,c}}}\exp-\frac{1}{2}\left(\frac{(h_x-\mu_{h,c})^2}{\sigma^2_{h,c}}\right)$$

$$p(w_x|c)=\frac{1}{\sqrt{2\pi\,\sigma^2_{w,c}}}\exp-\frac{1}{2}\left(\frac{(w_x-\mu_{w,c})^2}{\sigma^2_{w,c}}\right)$$

$$p(h_x|a)=\frac{1}{\sqrt{2\pi\,\sigma^2_{h,a}}}\exp-\frac{1}{2}\left(\frac{(h_x-\mu_{h,a})^2}{\sigma^2_{h,a}}\right)$$

$$p(w_x|a)=\frac{1}{\sqrt{2\pi\,\sigma^2_{w,a}}}\exp-\frac{1}{2}\left(\frac{(w_x-\mu_{w,a})^2}{\sigma^2_{w,a}}\right)$$

$$P(x|a)=p(h_x|a)\,p(w_x|a)$$

$$P(x|c)=p(h_x|c)\,p(w_x|c)$$

$$P(a|x)=\frac{P(x|a)P(a)}{P(x|a)P(a)+P(x|c)P(c)}$$

Pros and cons

Advantages

- It's relatively simple to understand and build
- It's easily trained, even with a small data-set
- It's fast!
- It's not sensitive to irrelevant features

Disadvantages

- It assumes every feature is independent, which isn't always the case

Naive Bayes (Summary)

- Classification based on Bayes theorem
- Assumption of independence between predictors.
- That is a too 'naive' assumption to make.
- Naive Bayesian model is easy to build and particularly useful for very large data sets.
- Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Naive Bayes (Summary)

- In many datasets, relationship between attributes and class variable is non-deterministic.
- Why?
  - Noisy data
  - Confounding and interaction of factors
  - Relevant variables not included in the data
- Scenario
  - Risk of heart disease based on individual's diet and workout frequency
  - Most people who "work out" and have a healthy diet don't get heart disease. Yet, some healthy individuals still do: Smoking, alcohol abuse

Gaussian Naive Bayes with Scikit-Learn

Gaussian Naive Bayes

```
# Gaussian Naive Bayes
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
# load the iris datasets
dataset = datasets.load_iris()
# fit a Naive Bayes model to the data
model = GaussianNB()
model.fit(dataset.data, dataset.target)
print(model)
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

(Ref: Machine Learning Algorithm Recipes in scikit-learn - Jason Brownlee)

Thanks . . .

- Search **"Yogesh Haribhau Kulkarni"** on Google and follow me on LinkedIn and Medium
- Office Hours: Saturdays, 2 to 5pm (IST); Free-Open to all; email for appointment.
- Email: yogeshkulkarni at yahoo dot com