Support Vector Machines

Educlash
Just Another Way To Learn

Is preprocessing cheating?

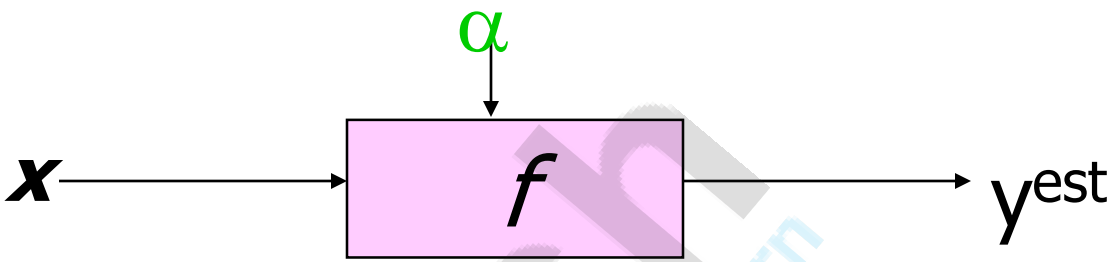
- Its cheating if we use a carefully designed set of task-specific, hand-coded features and then claim that the learning algorithm solved the whole problem.
- Its not cheating if we learn the non-linear preprocessing.
  - This makes learning much more difficult and much more interesting
  - Its not cheating if we use a very big set of non-linear features that is task-independent.
  - Support Vector Machines do this.
  - They have a clever way to prevent overfitting They have a very clever way to use a huge number of features without requiring nearly as much computation as seems to be necessary

Introduction

- In machine learning, **support vector machines** (**SVMs**, also **support vector networks** are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.

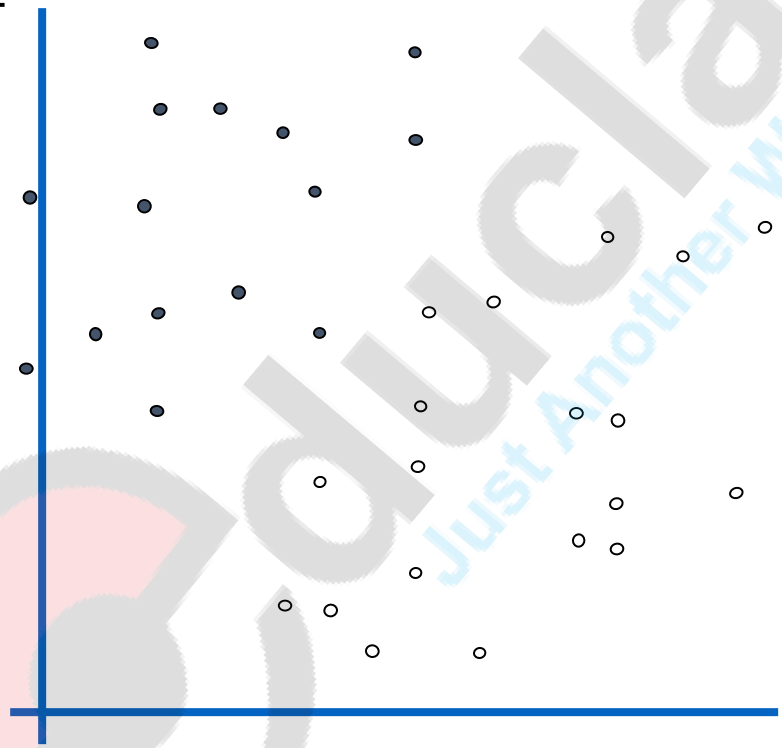
- Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier

Linear Classifiers



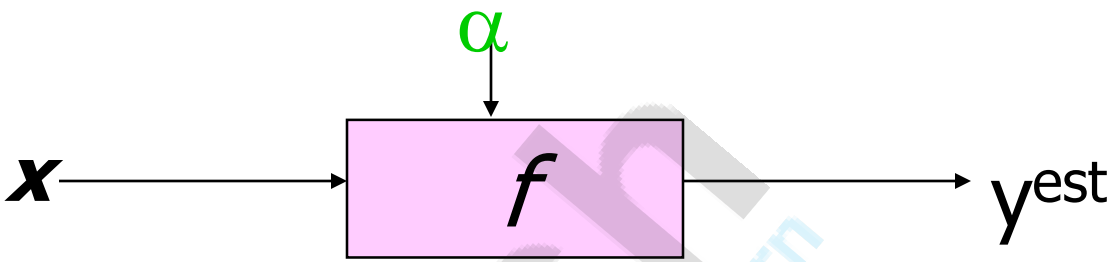
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



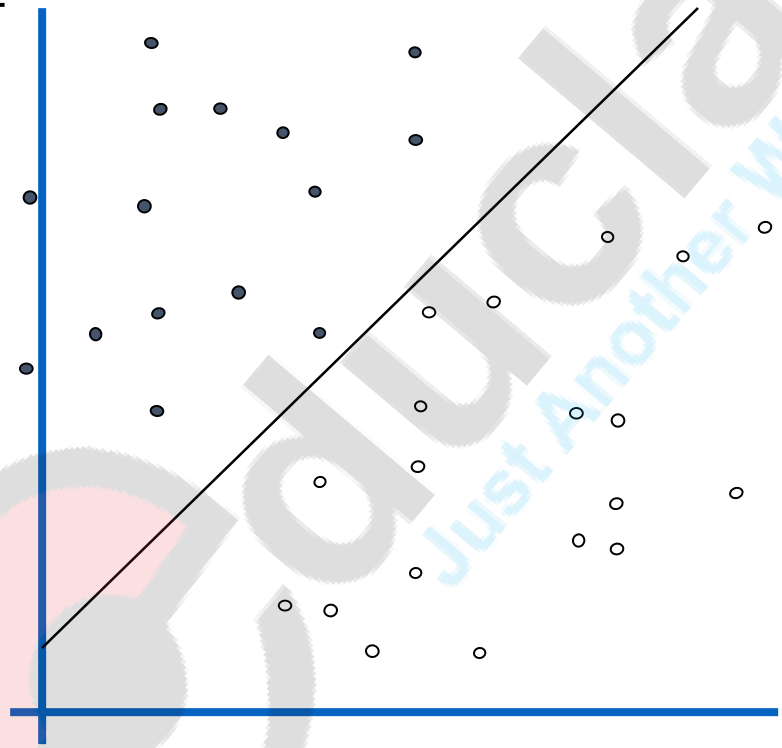
How would you classify this data?

Linear Classifiers



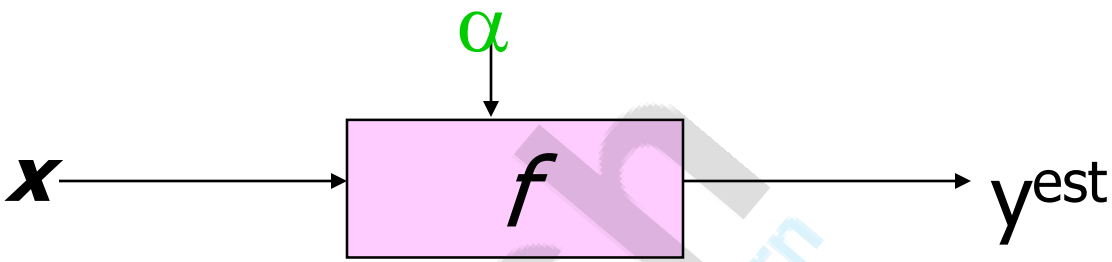
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



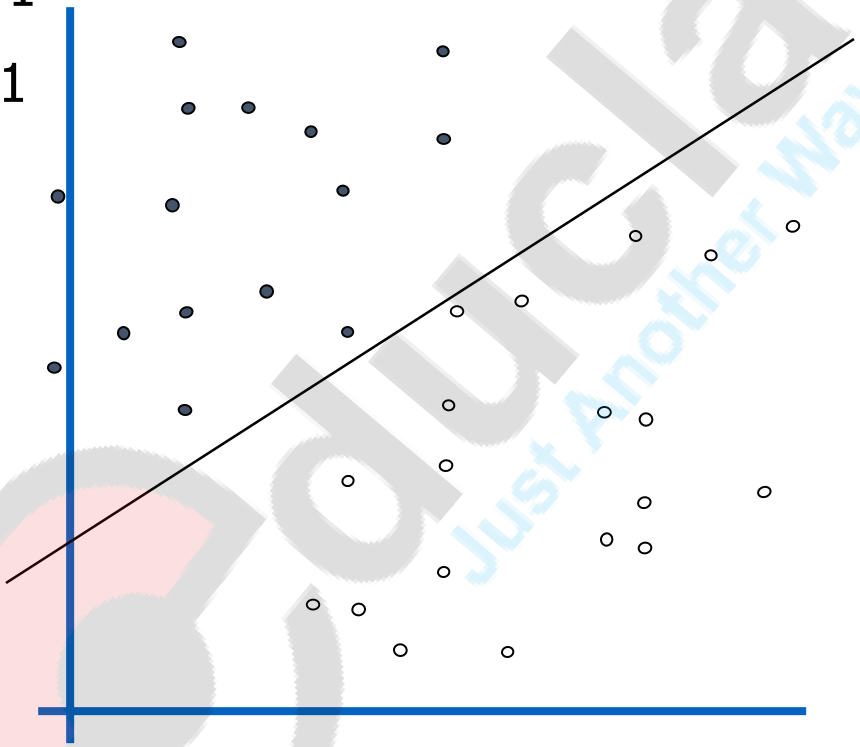
How would you classify this data?

Linear Classifiers



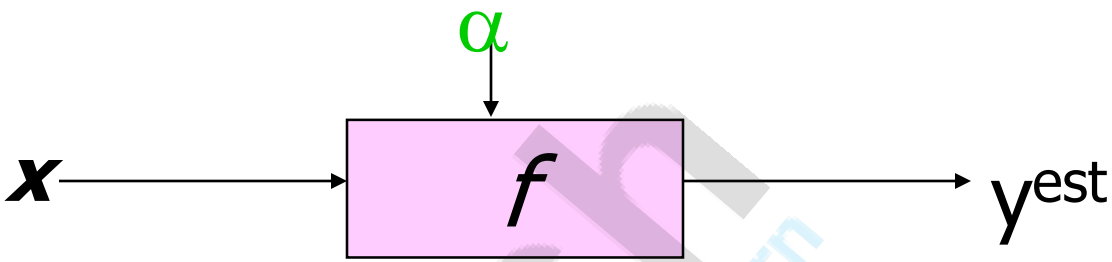
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



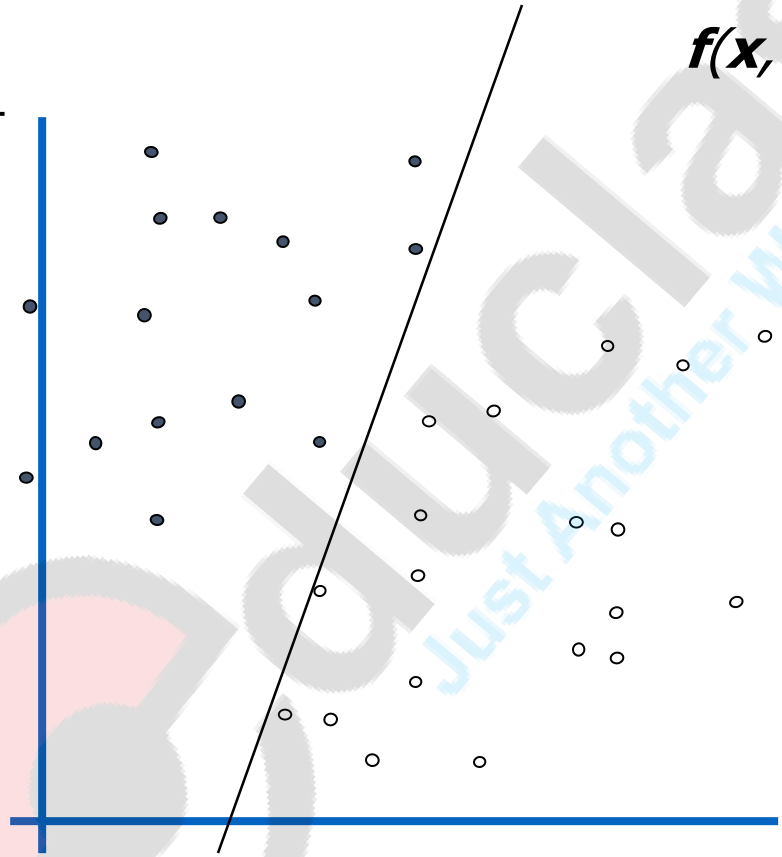
How would you classify this data?

Linear Classifiers



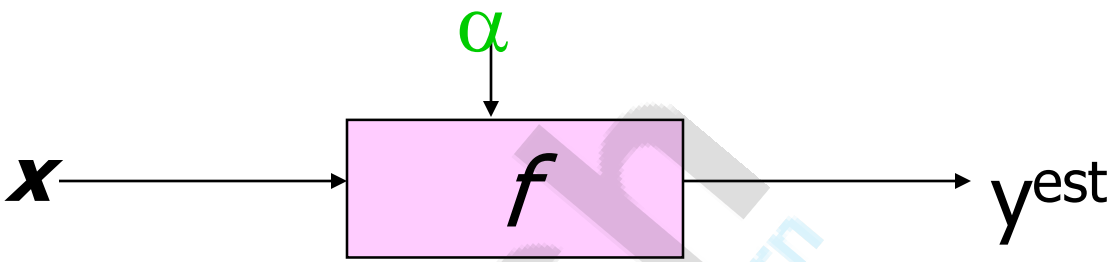
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



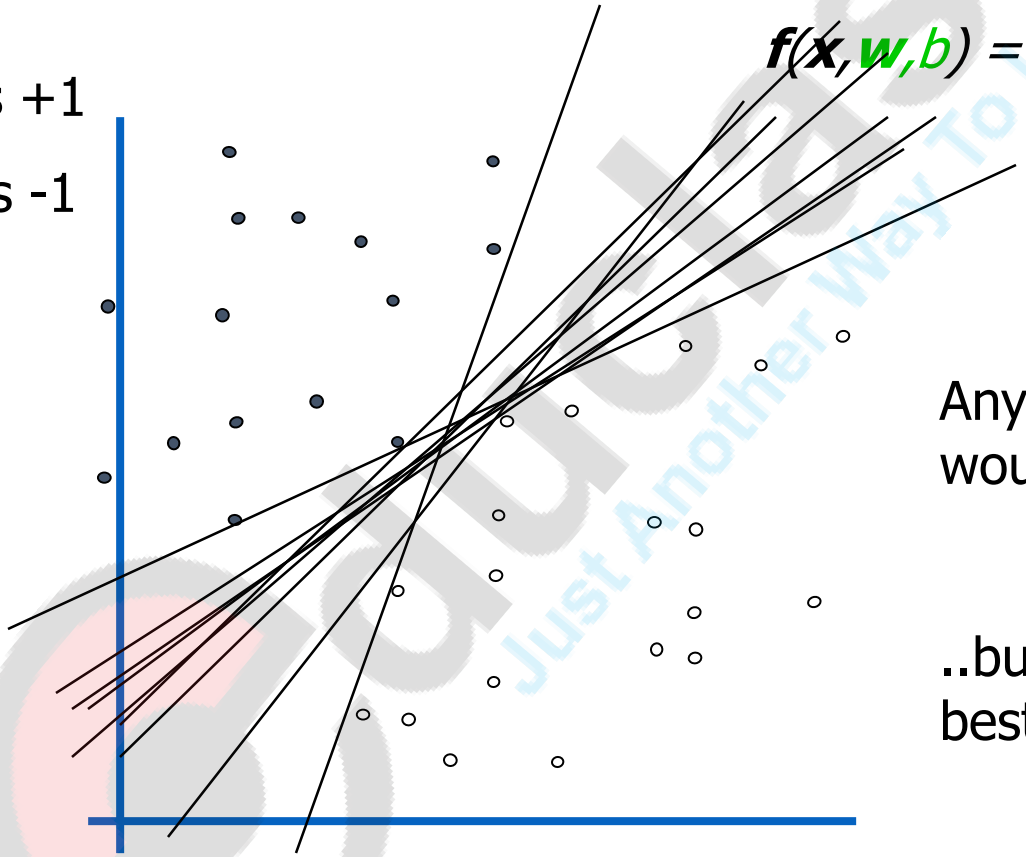
How would you classify this data?

Linear Classifiers



$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

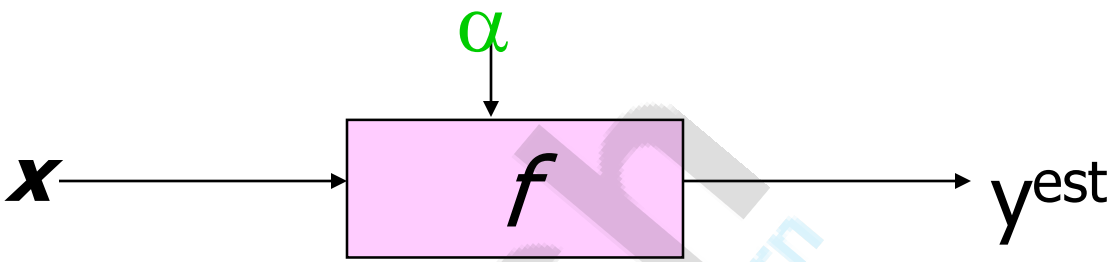
- denotes +1
- denotes -1



Any of these would be fine..

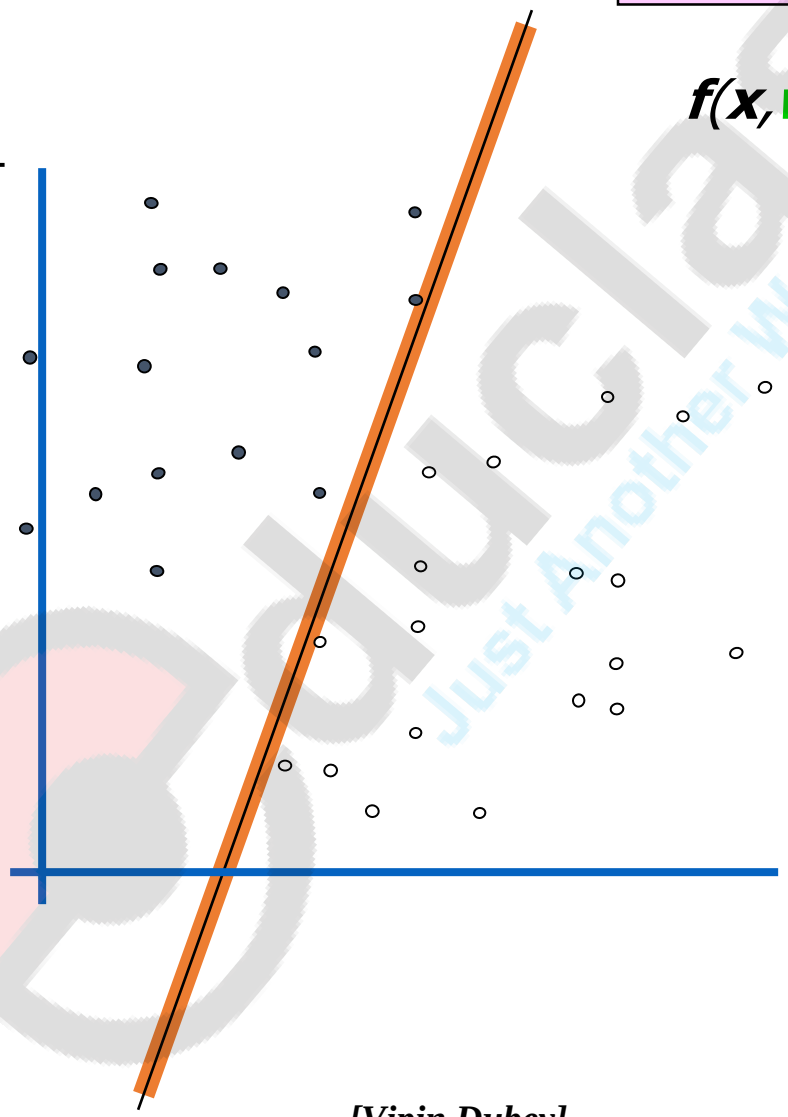
..but which is best?

Classifier Margin



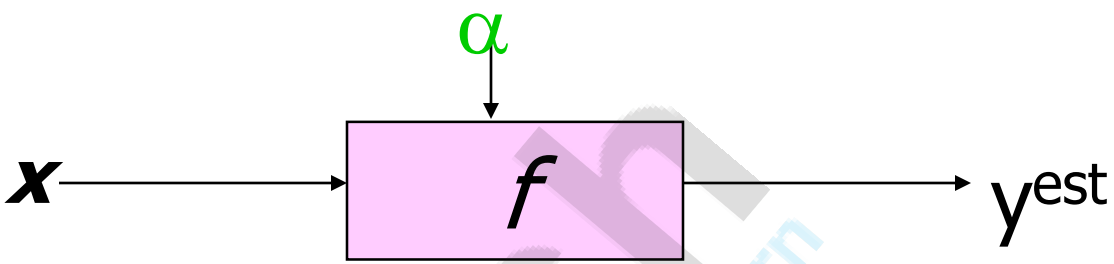
$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



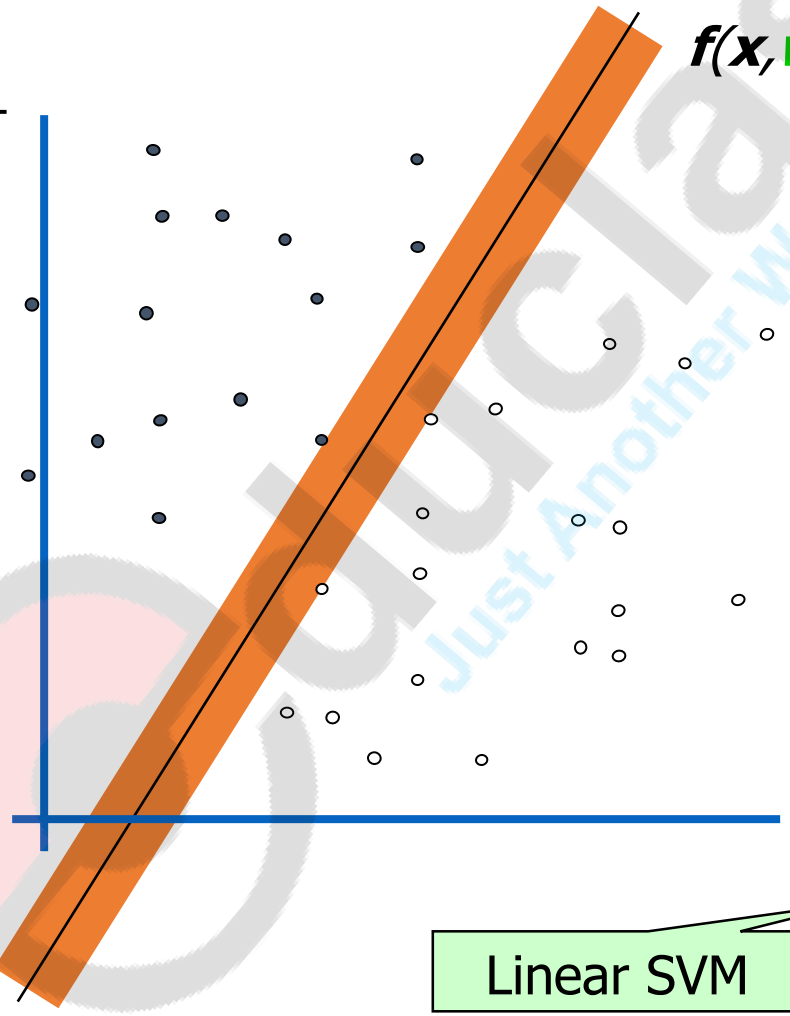
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

- denotes +1
- denotes -1



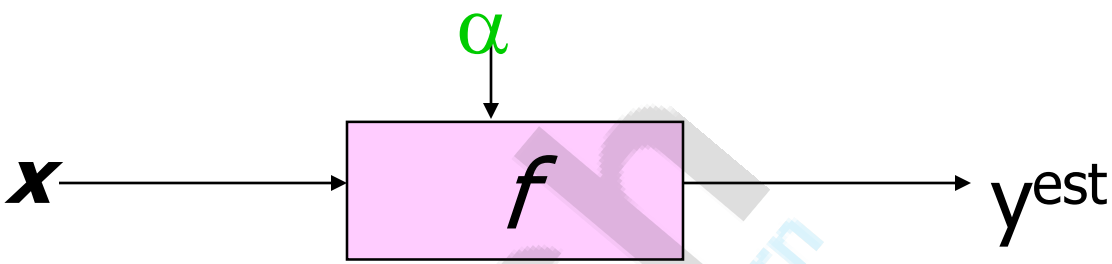
The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

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

Linear SVM

[Vipin Dubey]

Maximum Margin



$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}. \mathbf{x} - b)$

- denotes +1
- denotes -1

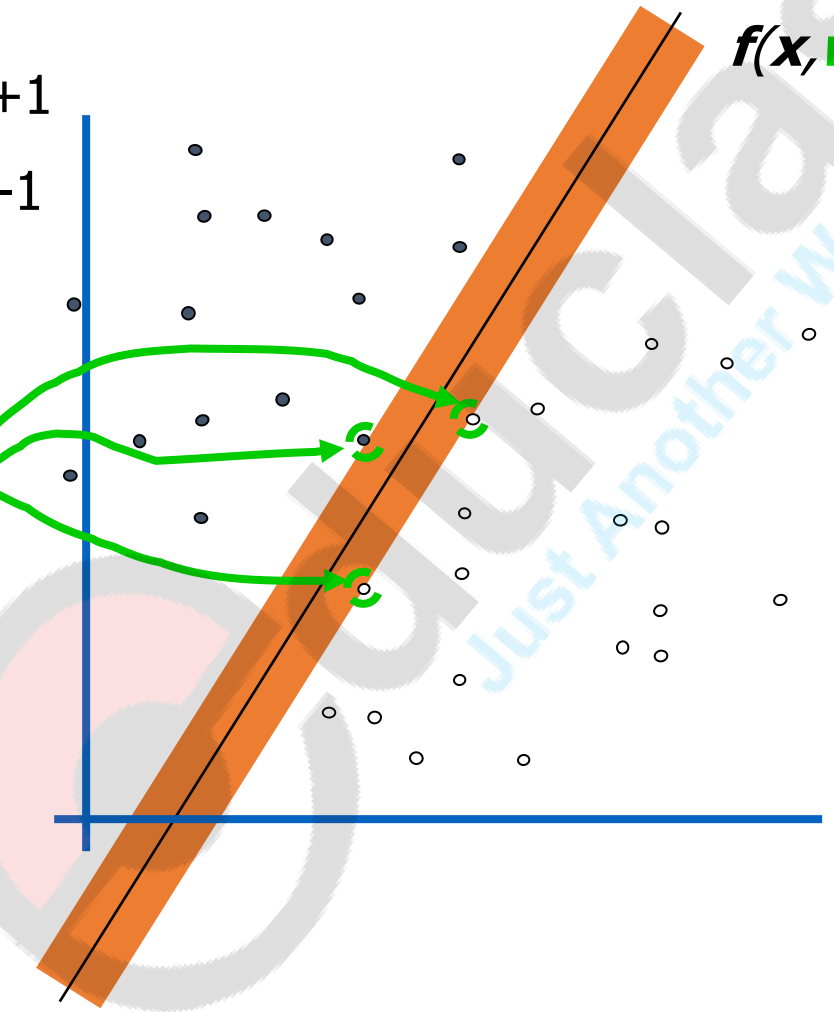
Support Vectors are those datapoints that the margin pushes up against

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

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

Linear SVM

Why Maximum Margin?

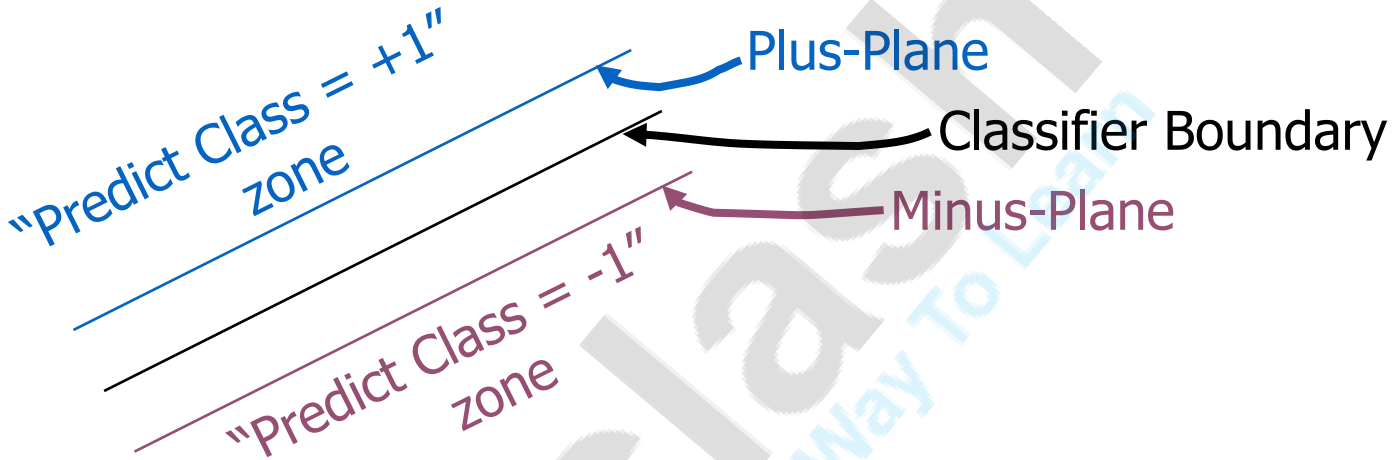


$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w}.\ \mathbf{x} - b)$

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

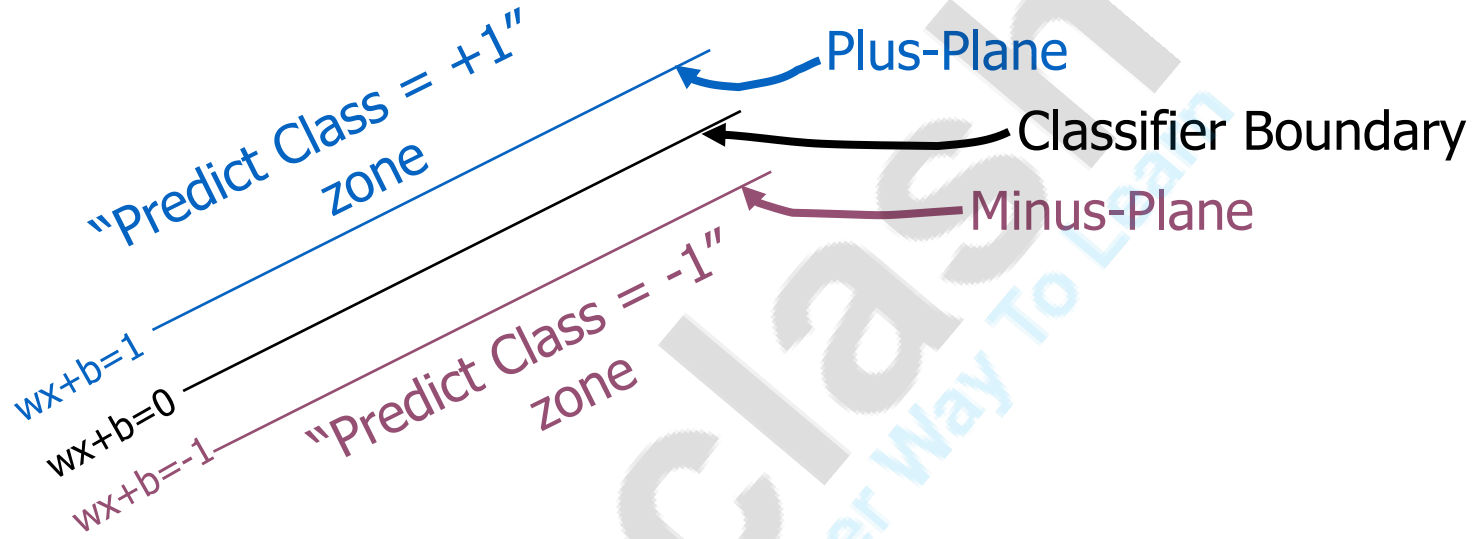
This is the simplest kind of SVM (Called an LSVM)

Specifying a line and margin



- How do we represent this mathematically?
- …in *m* input dimensions?

Specifying a line and margin

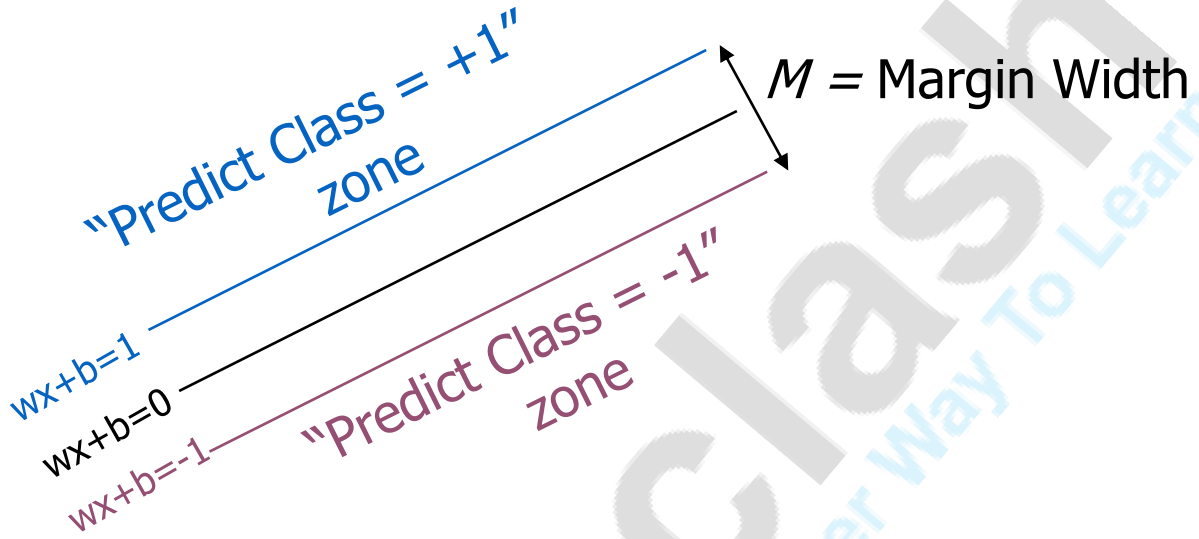


- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} \cdot \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} \cdot \boldsymbol{x} + b = -1 \}$

Classify as..

+1	if	$\boldsymbol{w} \cdot \boldsymbol{x} + b >= 1$
-1	if	$\boldsymbol{w} \cdot \boldsymbol{x} + b <= -1$
Universe explodes	if	$-1 < \boldsymbol{w} \cdot \boldsymbol{x} + b < 1$

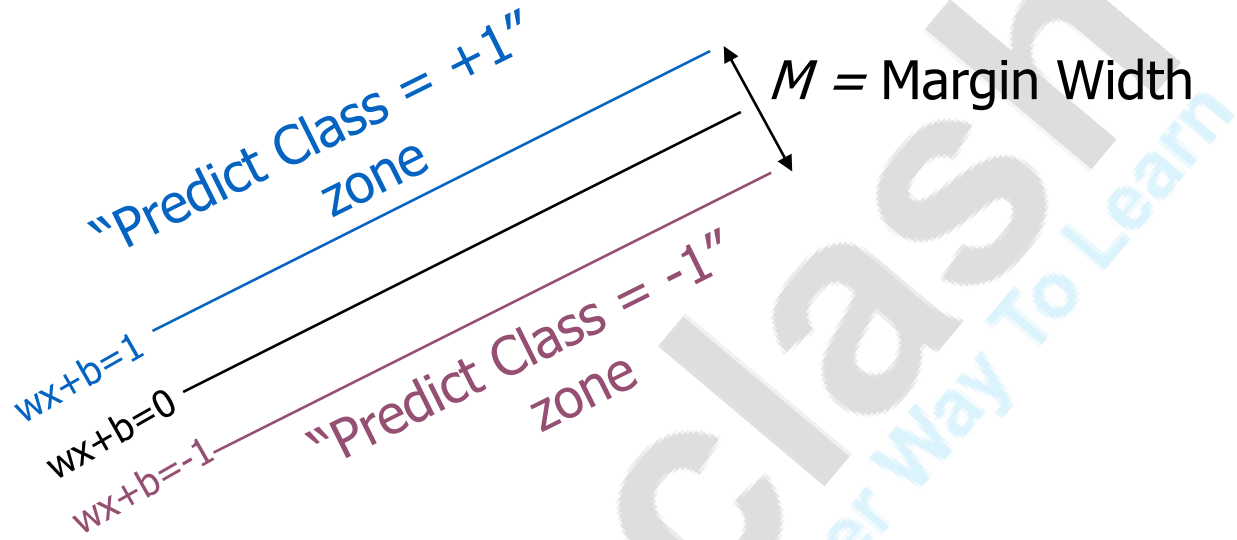
Computing the margin width



- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} \cdot \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} \cdot \boldsymbol{x} + b = -1 \}$

Claim: The vector **w** is perpendicular to the Plus Plane. Why?

Computing the margin width



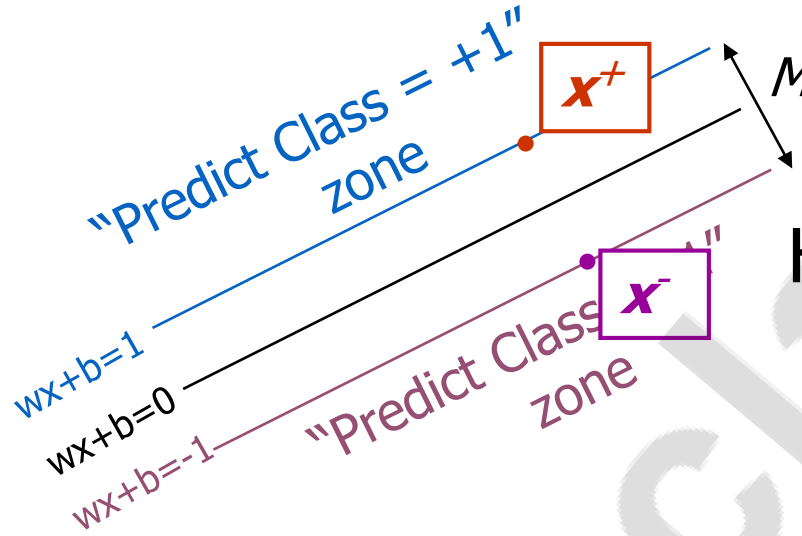
- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = -1 \}$

Claim: The vector **w** is perpendicular to the Plus Plane. Why?

Let **u** and **v** be two vectors on the Plus Plane. What is $\boldsymbol{w} . (\boldsymbol{u} - \boldsymbol{v})$?

And so of course the vector **w** is also perpendicular to the Minus Plane

Computing the margin width

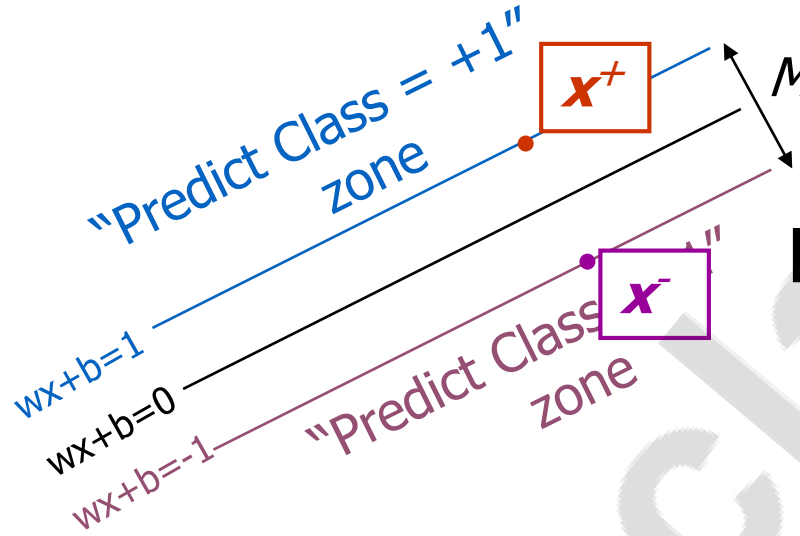


How do we compute *M* in terms of ***w*** and *b*?

- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = -1 \}$
- The vector **w** is perpendicular to the Plus Plane
- Let $\boldsymbol{x}^-$ be any point on the minus plane
- Let $\boldsymbol{x}^+$ be the closest plus-plane-point to $\boldsymbol{x}^-$.

Any location in R^m: not necessarily a datapoint

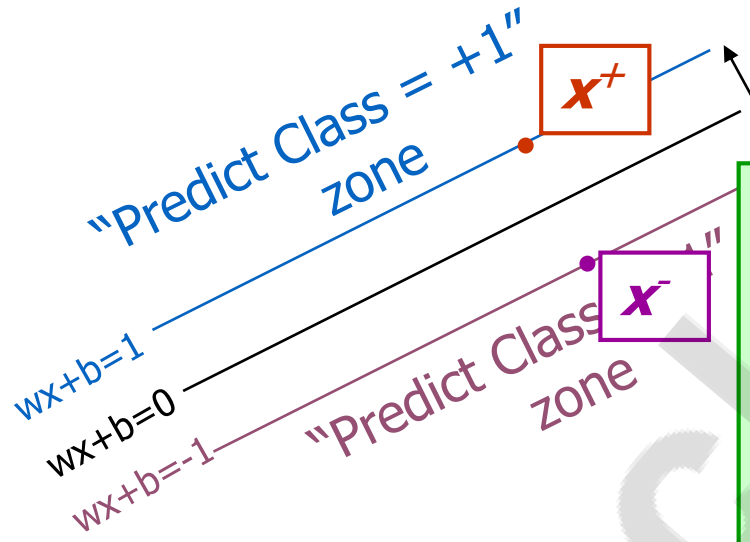
Computing the margin width



How do we compute *M* in terms of ***w*** and *b*?

- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = -1 \}$
- The vector **w** is perpendicular to the Plus Plane
- Let $\boldsymbol{x}^-$ be any point on the minus plane
- Let $\boldsymbol{x}^+$ be the closest plus-plane-point to $\boldsymbol{x}^-$.
- Claim: $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{w}$ for some value of λ. Why?

Computing the margin width

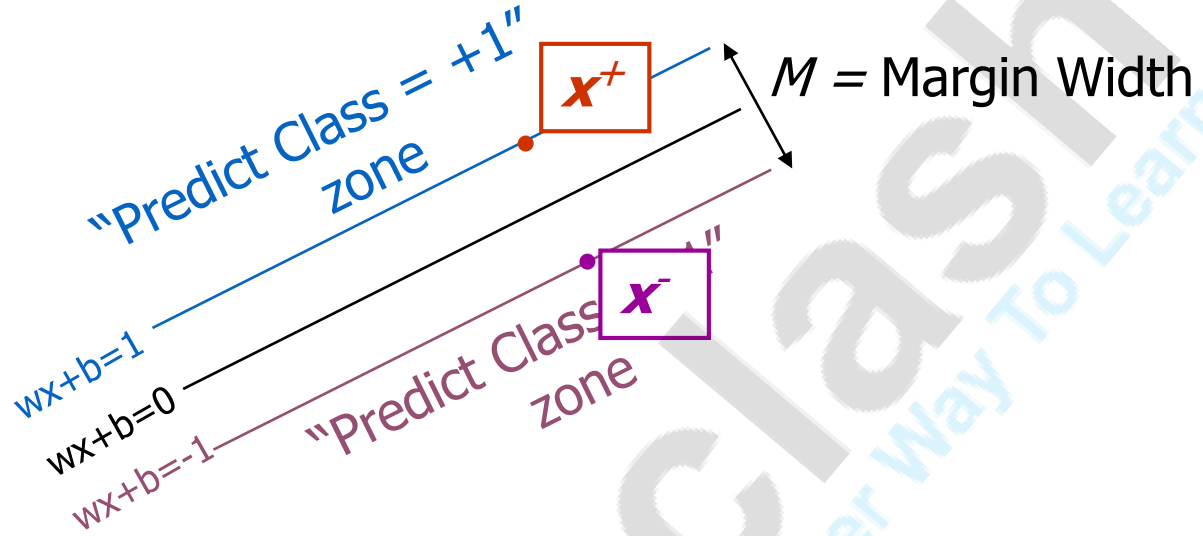


The line from x^- to x^+ is perpendicular to the planes.

So to get from x^- to x^+ travel some distance in direction w.

- Plus-plane = $\{ x : w \cdot x + b = +1 \}$
- Minus-plane = $\{ x : w \cdot x + b = -1 \}$
- The vector **w** is perpendicular to the Plus Plane
- Let x^- be any point on the minus plane
- Let x^+ be the closest plus-plane-point to x^-.
- Claim: $x^+ = x^- + \lambda w$ for some value of λ. Why?

Computing the margin width

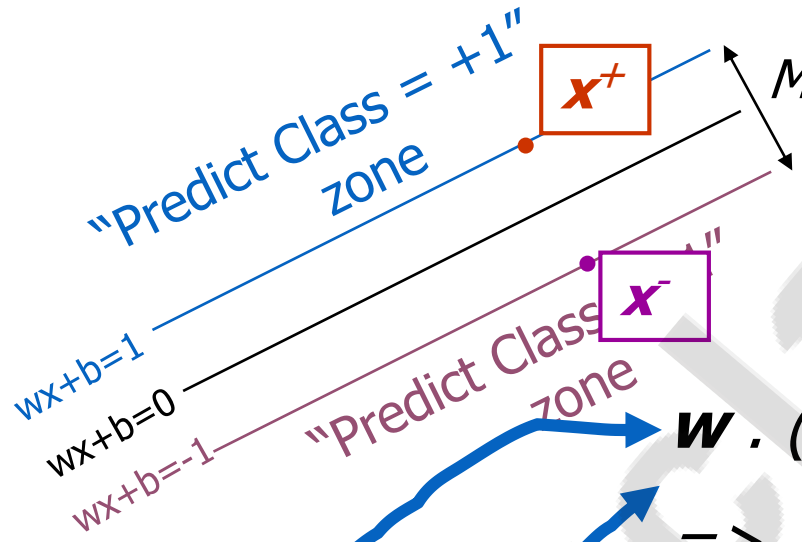


What we know:

- $\boldsymbol{w} \cdot \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \cdot \boldsymbol{x}^- + b = -1$
- $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{w}$
- $|\boldsymbol{x}^+ - \boldsymbol{x}^-| = M$

It's now easy to get M in terms of $\boldsymbol{w}$ and b

Computing the margin width



What we know:

- $\boldsymbol{w} \cdot \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \cdot \boldsymbol{x}^- + b = -1$
- $\boldsymbol{x}^+ = \boldsymbol{x}^- + \lambda \boldsymbol{w}$
- $|\boldsymbol{x}^+ - \boldsymbol{x}^-| = M$

It's now easy to get M in terms of $\boldsymbol{w}$ and b

$$\boldsymbol{w} \cdot (\boldsymbol{x}^- + \lambda \boldsymbol{w}) + b = 1$$

$$\Rightarrow$$

$$\boldsymbol{w} \cdot \boldsymbol{x}^- + b + \lambda \boldsymbol{w} \cdot \boldsymbol{w} = 1$$

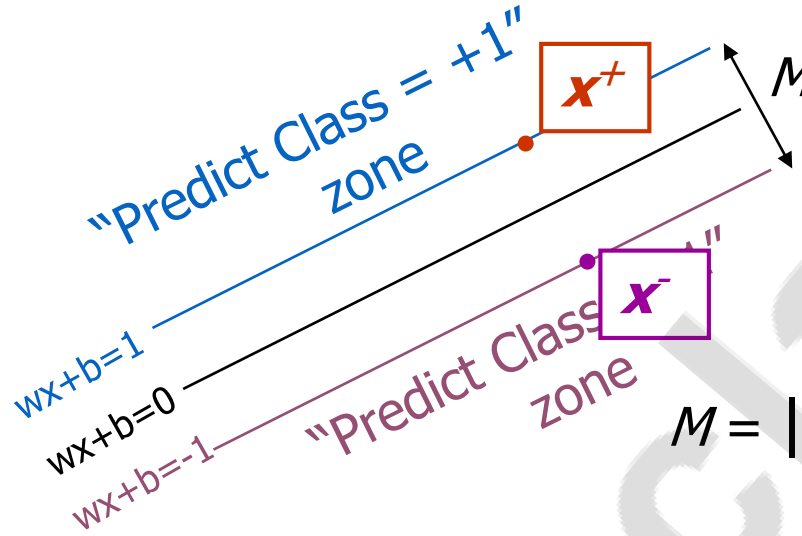
$$\Rightarrow$$

$$-1 + \lambda \boldsymbol{w} \cdot \boldsymbol{w} = 1$$

$$\Rightarrow$$

$$\lambda = \frac{2}{\mathbf{w} \cdot \mathbf{w}}$$

Computing the margin width



M = Margin Width = $\frac{2}{\sqrt{\mathbf{w}.\mathbf{w}}}$

What we know:

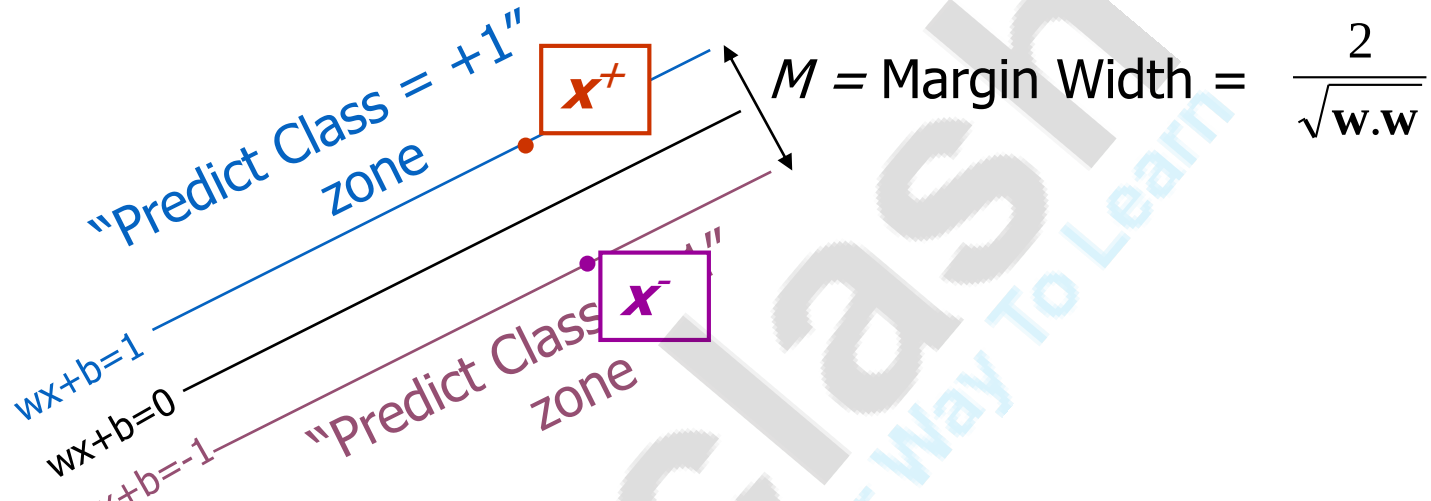
- $\mathbf{w} \cdot \mathbf{x}^+ + b = +1$
- $\mathbf{w} \cdot \mathbf{x}^- + b = -1$
- $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$
- $|\mathbf{x}^+ - \mathbf{x}^-| = M$
- $$\lambda = \frac{2}{\mathbf{w}.\mathbf{w}}$$

$$M = |\mathbf{x}^+ - \mathbf{x}^-| = |\lambda \mathbf{w}| =$$

$$= \lambda |\mathbf{w}| = \lambda \sqrt{\mathbf{w}.\mathbf{w}}$$

$$= \frac{2\sqrt{\mathbf{w}.\mathbf{w}}}{\mathbf{w}.\mathbf{w}} = \frac{2}{\sqrt{\mathbf{w}.\mathbf{w}}}$$

Learning the Maximum Margin Classifier



Given a guess of ***w*** and *b* we can

- Compute whether all data points in the correct half-planes
- Compute the width of the margin

So now we just need to write a program to search the space of **w**'s and *b*'s to find the widest margin that matches all the datapoints. *How?*

Gradient descent? Simulated Annealing? Matrix Inversion? EM? Newton's Method?

SVM Performance

- Anecdotally they work very very well indeed.
- Example: They are currently the best-known classifier on a well-studied hand-written-character recognition benchmark
- Another Example: Andrew knows several reliable people doing practical real-world work who claim that SVMs have saved them when their other favorite classifiers did poorly.
- There is a lot of excitement and religious fervor about SVMs as of 2001.
- Despite this, some practitioners (including your lecturer) are a little skeptical.

Doing multi-class classification

- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- What can be done?
- Answer: with output arity N, learn N SVM's
  - SVM 1 learns "Output==1" vs "Output != 1"
  - SVM 2 learns "Output==2" vs "Output != 2"
  - :
  - SVM N learns "Output==N" vs "Output != N"
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Separating data with the maximum margin

- Support vector machines
- Pros: Low generalization error, computationally inexpensive, easy to interpret results
- Cons: Sensitive to tuning parameters and kernel choice; natively only handles binary
- classification
- Works with: Numeric values, nominal values

SVM Terminologies: Separating data with the maximum margin

- Consider the data in frames A–D in figure 6.1; could you draw a straight line to put all of the circles on one side and all of the squares on another side?

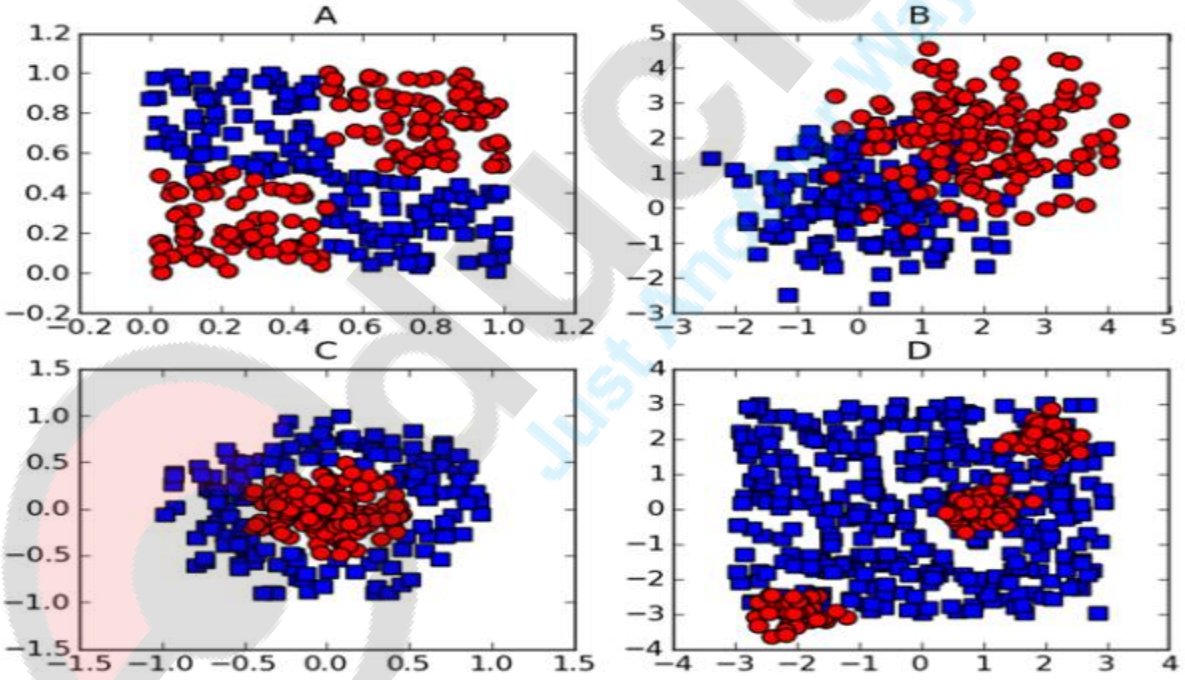


Figure 6.1 Four examples of datasets that aren't linearly separable

SVM Terminologies

- Now consider the data in figure 6.2, frame A. There are two groups of data, and the data points are separated enough that you could draw a straight line on the figure with all the points of one class on one side of the line and all the points of the other class on the other side of the line. If such a situation exists, we say the data is **linearly separable**.

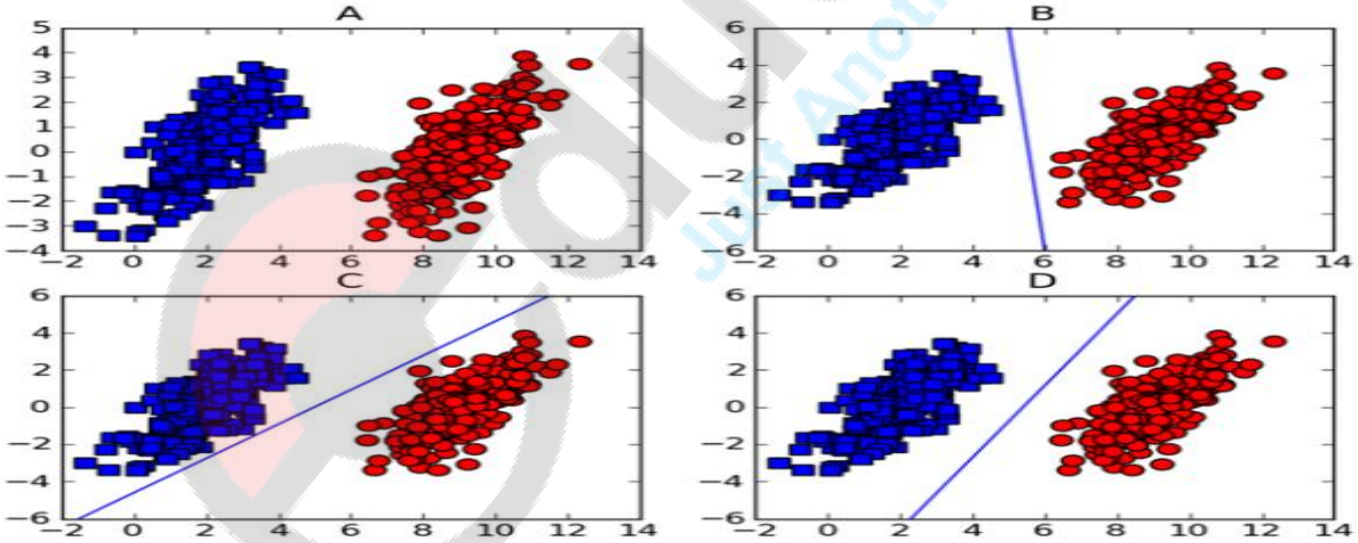


Figure 6.2 **Linearly separable data is shown in frame A. Frames B, C, and D show possible valid lines separating the two classes of data.**

SVM Terminologies

- The line used to separate the dataset is called a separating hyperplane. In our simple 2D plots, it's just a line. But, if we have a dataset with three dimensions, we need a plane to separate the data; and if we have data with 1024 dimensions, we need something with 1023 dimensions to separate the data.
- What do you call something with 1023 dimensions? How about N-1 dimensions? It's called **a hyperplane**. The hyperplane is our decision boundary. Everything on one side belongs to one class, and everything on the other side belongs to a different class.

- We'd like to make our classifier in such a way that the farther a data point is from the decision boundary, the more confident we are about the prediction we've made consider the plots in figure 6.2, frames B–D. They all separate the data, but which one does it best? Should we minimize the average distance to the separating hyperplane?

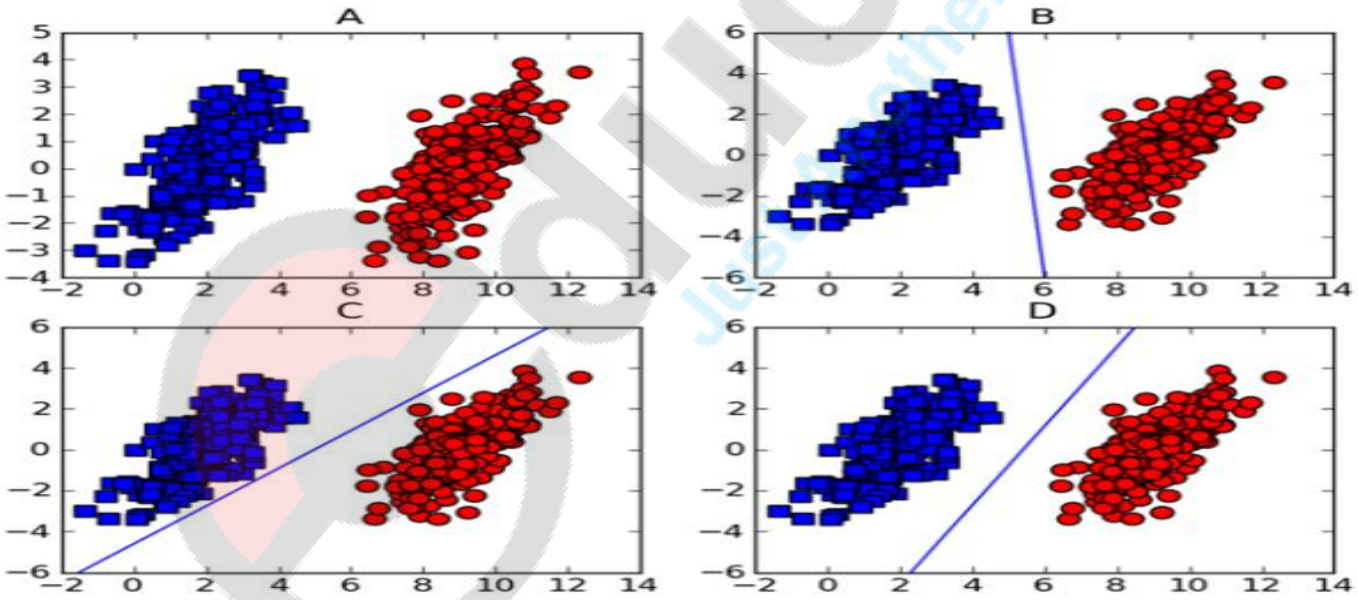


Figure 6.2 **Linearly separable data is shown in frame A. Frames B, C, and D show possible valid lines separating the two classes of data.**

- In that case, are frames B and C any better than frame D in figure 6.2? Isn't something like that done with best-fit lines? Yes, but it's not the best idea here. We'd like to find the point closest to the separating hyperplane and make sure this is as far away from the separating line as possible.
- This is known as **margin**. We want to have the greatest possible margin, because if we made a mistake or trained our classifier on limited data, we'd want it to be as robust as possible. The points closest to the separating hyperplane are known as **support vectors**.

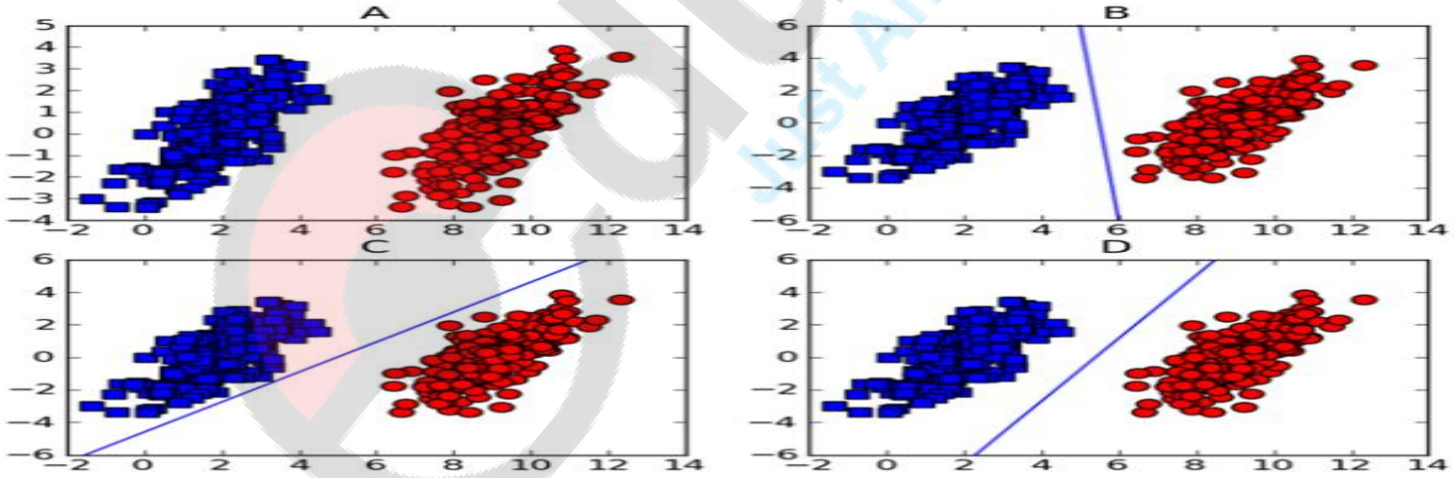


Figure 6.2 Linearly separable data is shown in frame A. Frames B, C, and D show possible valid lines separating the two classes of data.

- Now that we know that we're trying to maximize the distance from the separating line to the support vectors, we need to find a way to optimize this problem.

Finding the maximum margin

- How can we measure the line that best separates the data? To start with, look at figure 6.3. Our separating hyperplane has the form wx+b. If we want to find the distance from A to the separating plane, we must measure normal or perpendicular to the line.

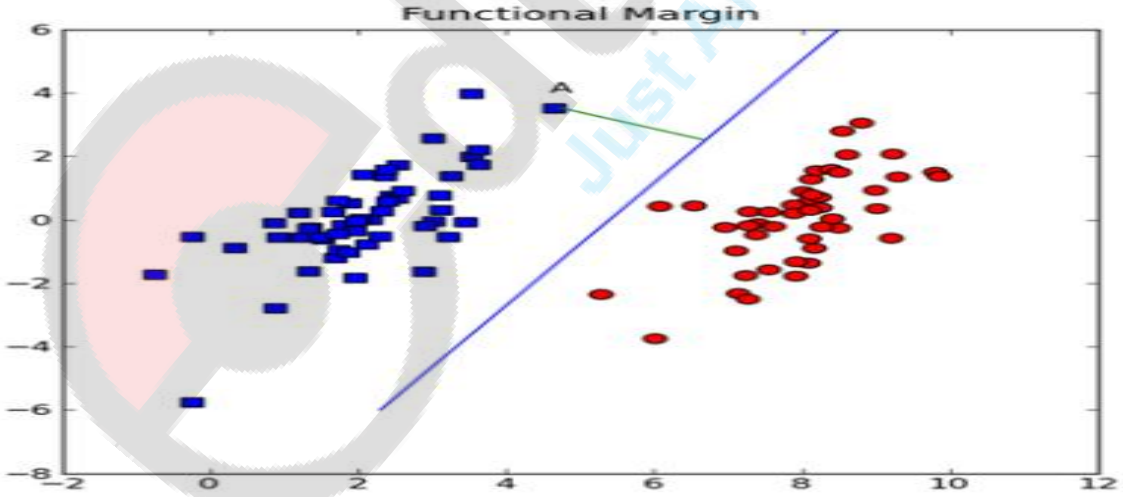


Figure 6.3 The distance from point A to the separating plane is measured by a line normal to the separating plane.

[Vipin Dubey]

Approaching SVMs with our general framework

General approach to SVMs

1. Collect: Any method.

2. Prepare: Numeric values are needed.

3. Analyze: It helps to visualize the separating hyperplane.

4. Train: The majority of the time will be spent here. Two parameters can be adjusted during this phase.

5. Test: Very simple calculation.

6. Use: You can use an SVM in almost any classification problem. One thing to note is that SVMs are binary classifiers.

Efficient optimization with the SMO algorithm

- In 1996 John Platt published a powerful algorithm he called SMO for training the support vector machines.
- Platt's SMO stands for Sequential Minimal Optimization, and it takes the large optimization problem and breaks it into many small problems.
- The small problems can easily be solved, and solving them sequentially will give you the same answer as trying to solve everything together. In addition to getting the same answer, the amount of time is greatly reduced.

Efficient optimization with the SMO algorithm

- The SMO algorithm works to find a set of alphas and b. Once we have a set of alphas, we can easily compute our weights w and get the separating hyperplane. Here's how the SMO algorithm works: it chooses two alphas to optimize on each cycle. Once a suitable pair of alphas is found, one is increased and one is decreased.
- To be suitable, a set of alphas must meet certain criteria. One criterion a pair must meet is that both of the alphas have to be outside their margin boundary. The second criterion is that the alphas aren't already clamped or bounded.

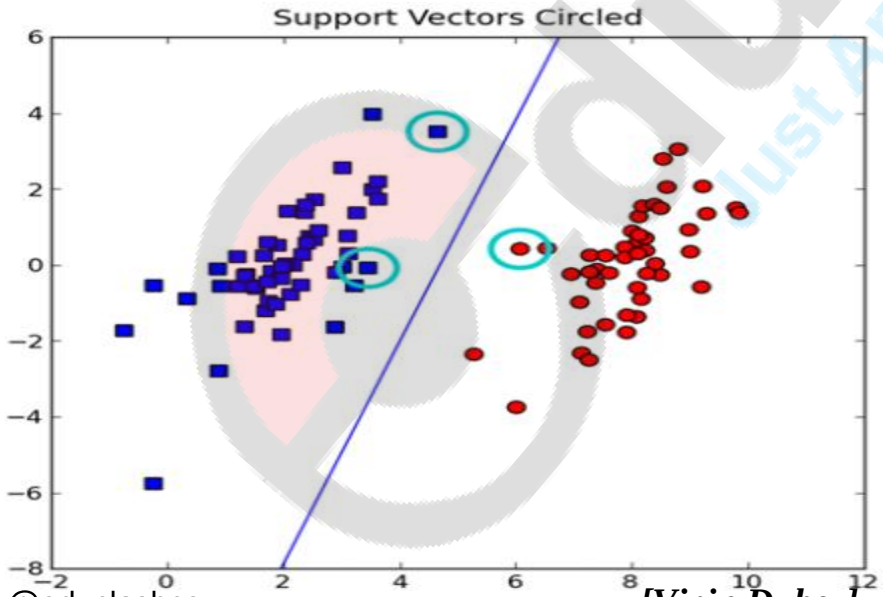


Figure 6.4 SMO sample dataset showing the support vectors circled and the separating hyperplane after the simplified SMO is run on the data

Applications of SVM in Real World

SVMs depends on supervised learning algorithms. The aim of using SVM is to correctly classify unseen data. SVMs have a number of applications in several fields.

Some common applications of SVM are-

- **Face detection** – SVMc classify parts of the image as a face and non-face and create a square boundary around the face.
- **Text and hypertext categorization** – SVMs allow Text and hyper text categorization for both inductive and transductive models. They use training data to classify documents into different categories. It categorizes on the basis of the score generated and then compares with the threshold value.
- **Classification of images** – Use of SVMs provides better search accuracy for image classification. It provides better accuracy in comparison to the traditional query based searching techniques.
- **Bioinformatics** – It includes protein classification and cancer classification. We use SVM for identifying the classification of genes, patients on the basis of genes and other biological problems.
- **Protein fold and remote homology detection** – Apply SVM algorithms for protein remote homology detection.
- **Handwriting recognition** – We use SVMs to recognize hand written characters used widely.
- **Generalized predictive control(GPC)** – Use SVM based GPC to control chaotic dynamics with useful parameters.

Applications of SVM in Real World

Face Detection

- It classifies the parts of the image as face and non-face. It contains training data of n x n pixels with a two-class face (+1) and non-face (-1). Then it extracts features from each pixel as face or non-face. Creates a square boundary around faces on the basis of pixel brightness and classifies each image by using the same process.

Text and Hypertext Categorization

- Allows text and hypertext categorization for both types of models; inductive and transductive. It Uses training data to classify documents into different categories such as news articles, e-mails, and web pages
- **Examples:**
- Classification of news articles into “business” and “Movies”
- Classification of web pages into personal home pages and others