CONVEX OPTIMIZATION PROBLEMS

MAT 220

Dr.C. Rajan (8113053359) Off: S109E

Support Vector Machines

Slides from Alexy Skoutnev;
Tan, Steinbach, and Kumar;

O.L. Mangasarian

SVM applications

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97], principal component analysis [Schölkopf *et al.* '99], etc.
- Most popular optimization algorithms for SVMs use *decomposition* to hill-climb over a subset of α_i's at a time, e.g. SMO [Platt '99] and [Joachims '99]
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.

What are Support Vector Machines Used For?

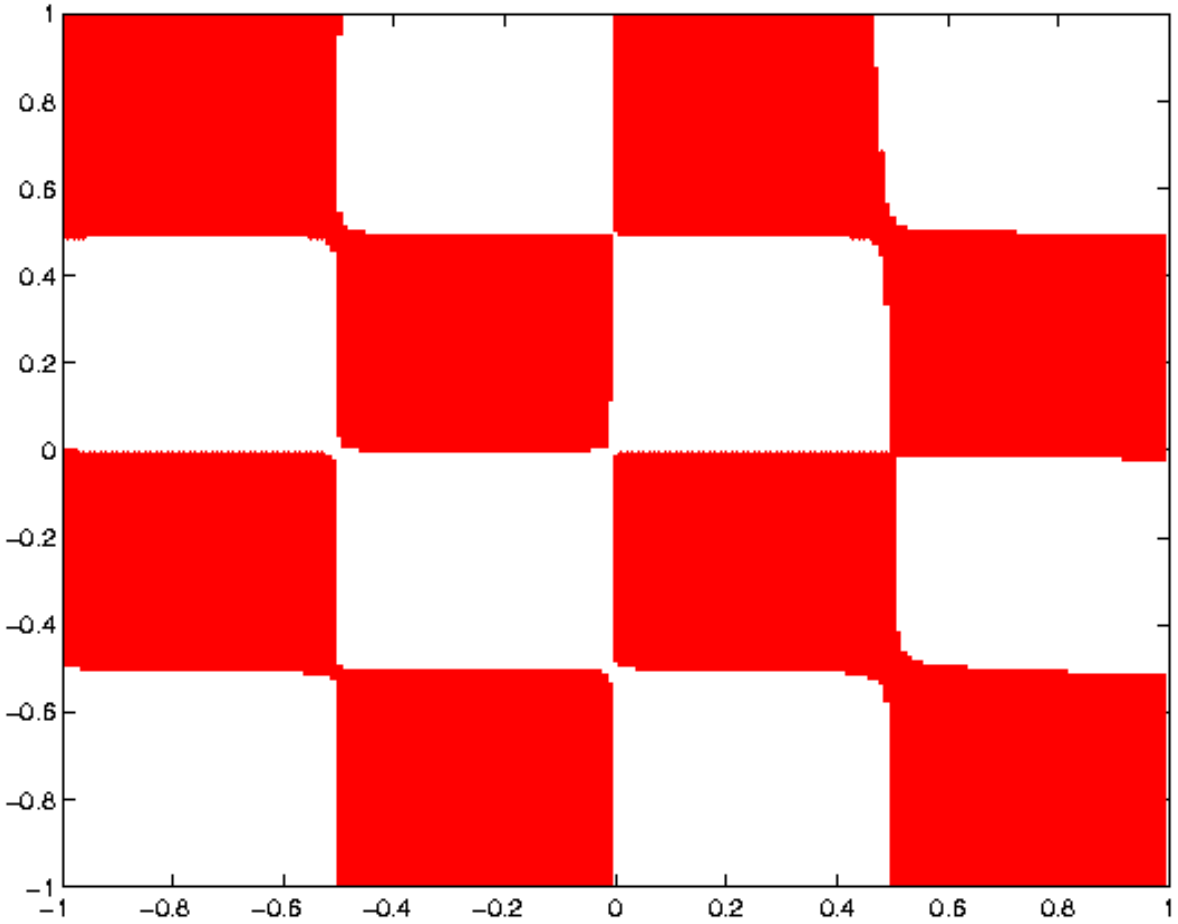
- Classification
- Regression & Data Fitting
- Supervised & Unsupervised Learning

(Will concentrate on classification)

What is a Support Vector Machine?

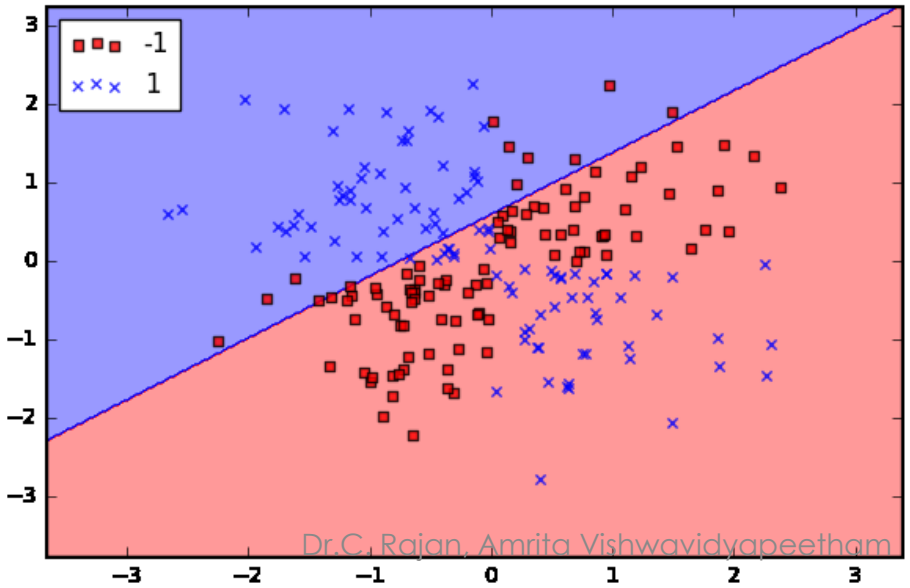
- A ML algorithm to optimally classify data using a surface
- Typically nonlinear in the input space
- Linear in a higher dimensional space (after kernel map)
- Implicitly defined by a kernel function
- Kernel trick not needed if data are already linear separable

Example of Nonlinear Classifier: Checkerboard Classifier

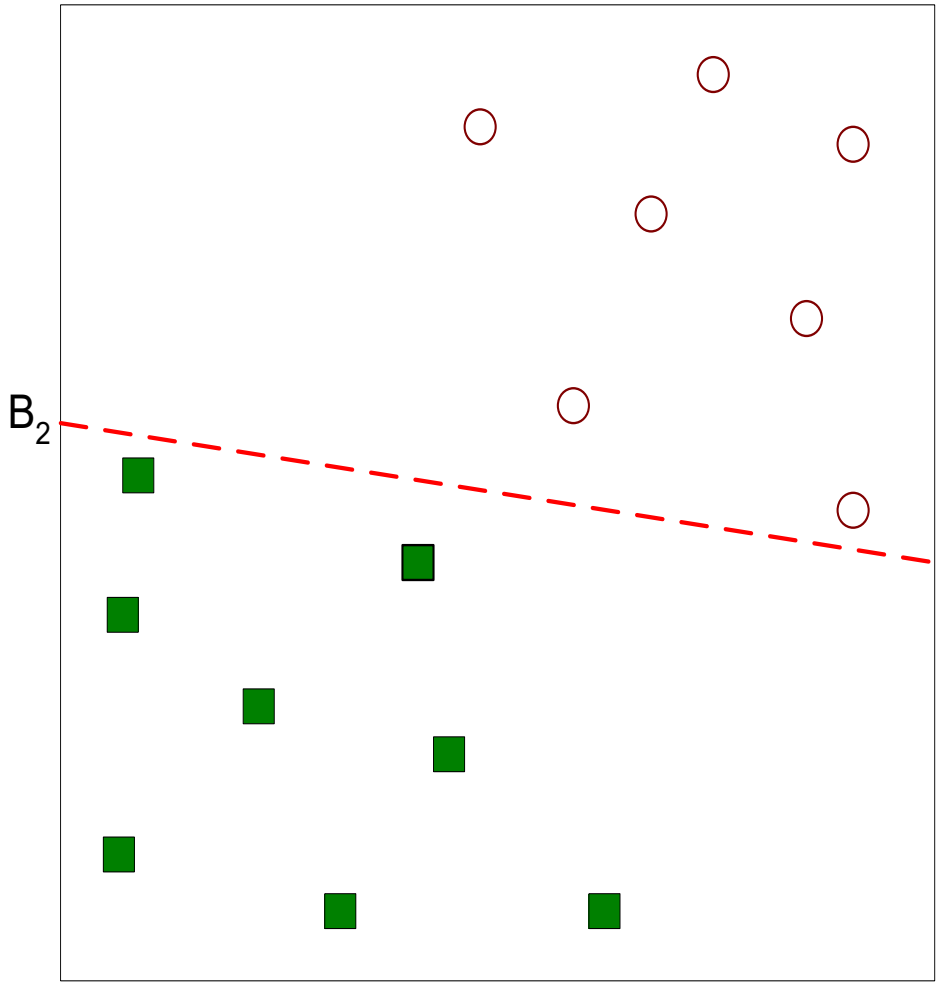


Support Vector Machine Basics

- Supervised learning technique used for classification and regression analysis
  - Input of training data (with known response variable)
  - Creates a decision boundary between data points
  - Separates data into distinct sets

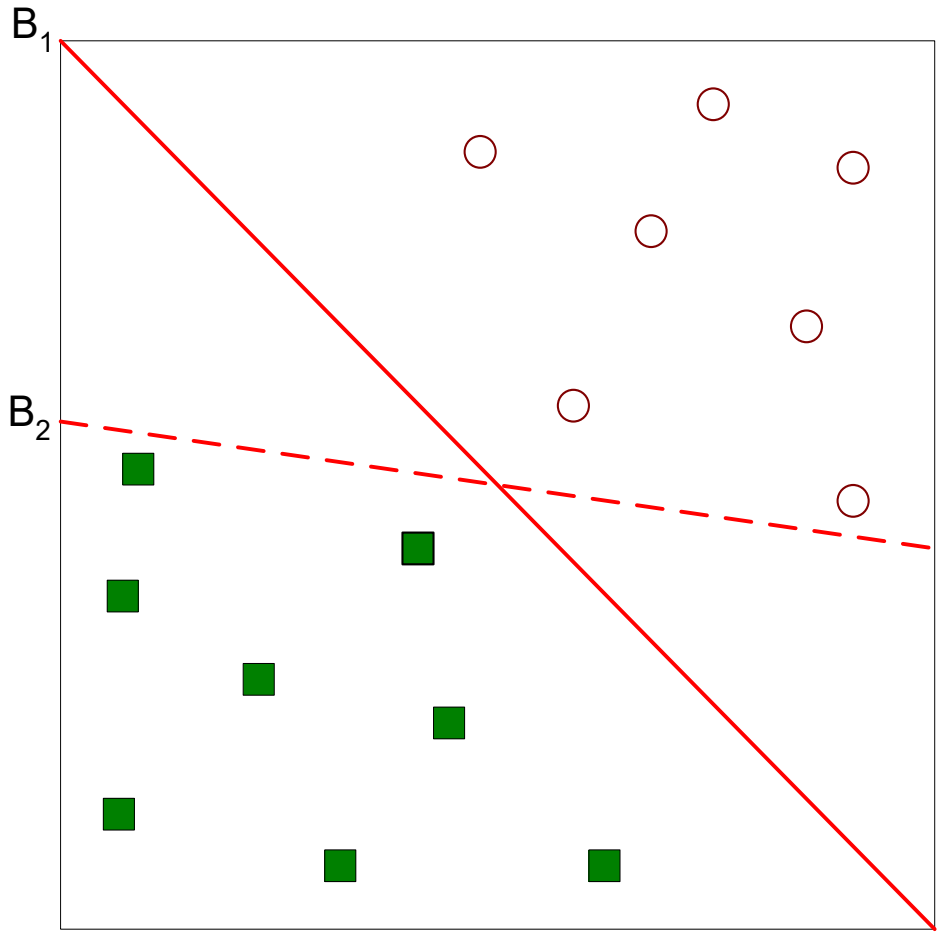


Support Vector Machines



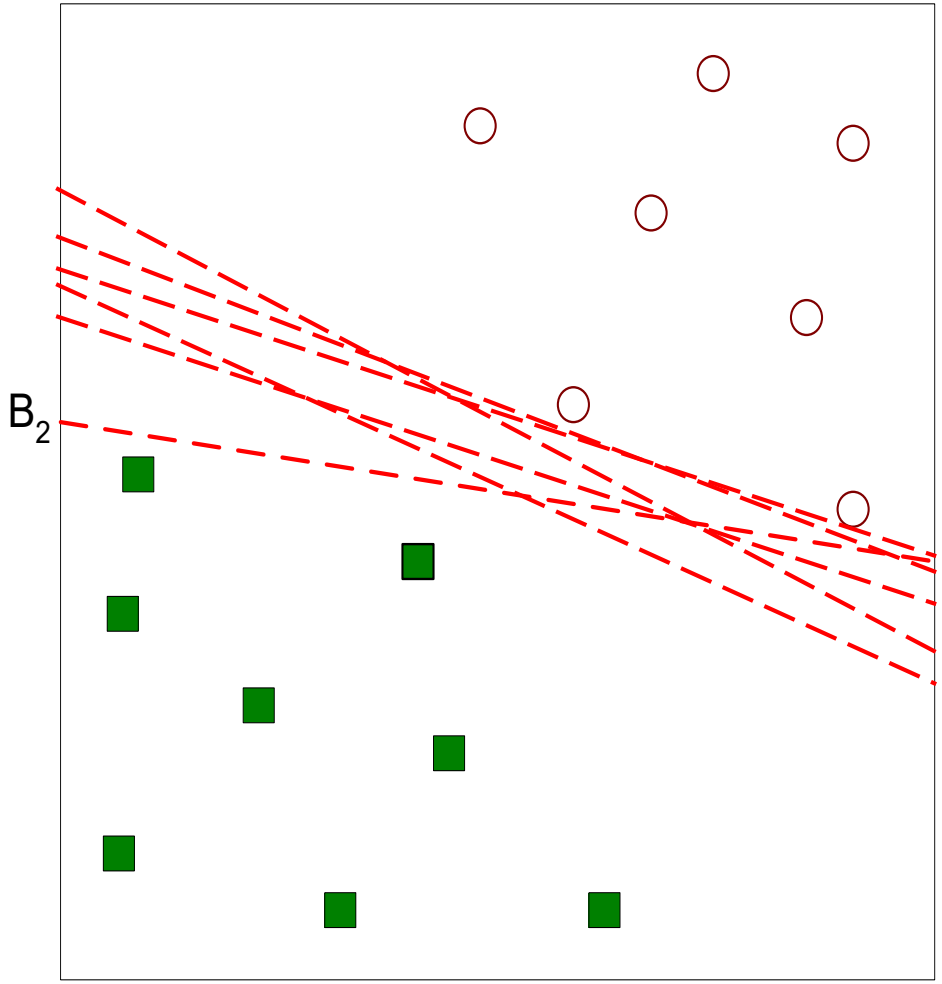
- Another possible solution

Support Vector Machines



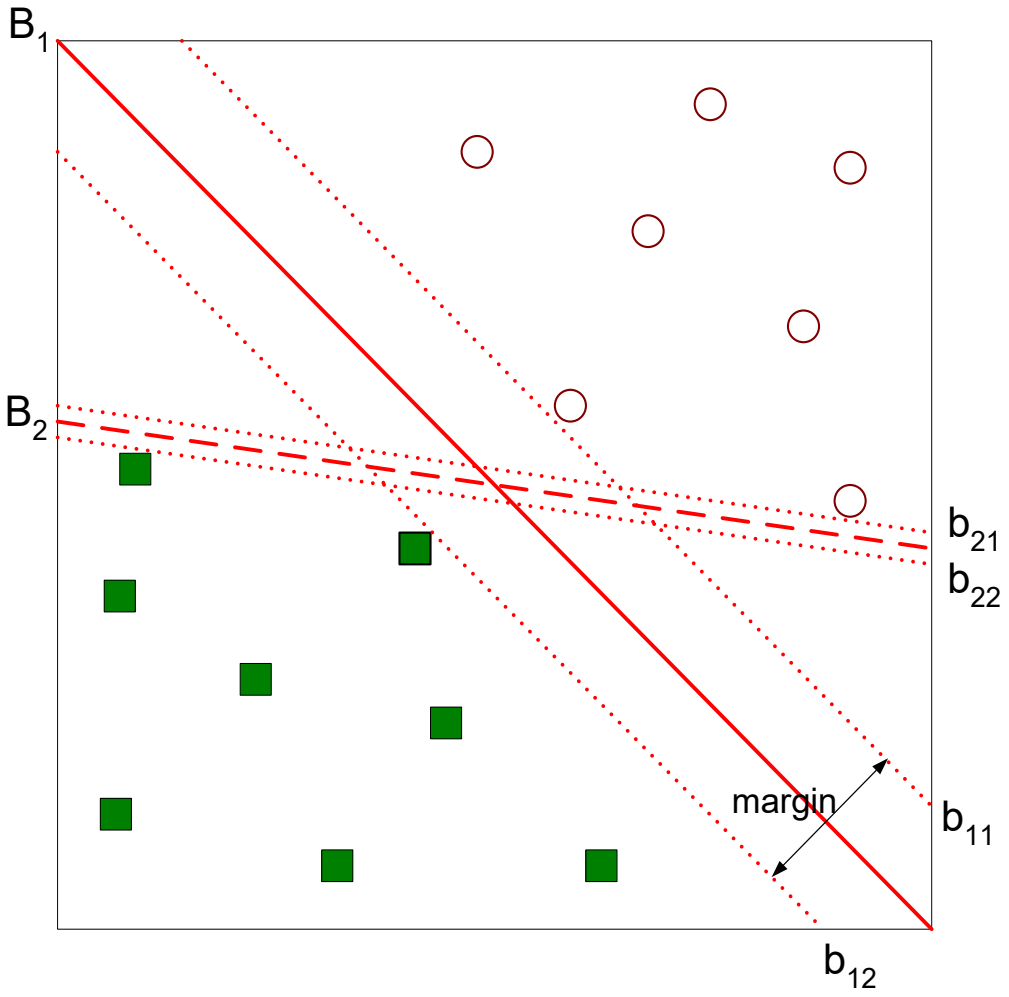
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Other possible solutions

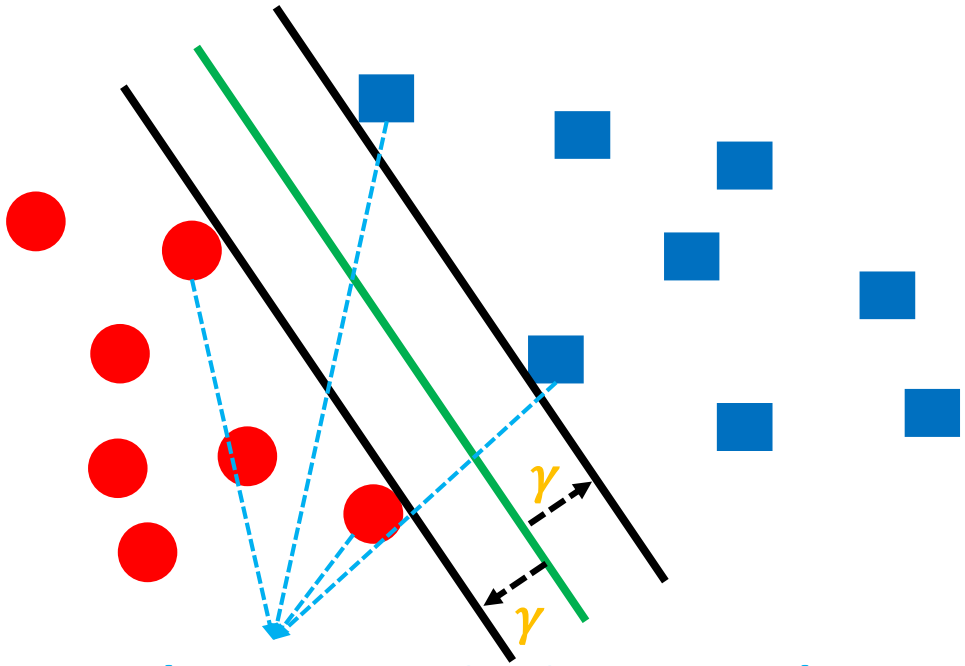
Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines

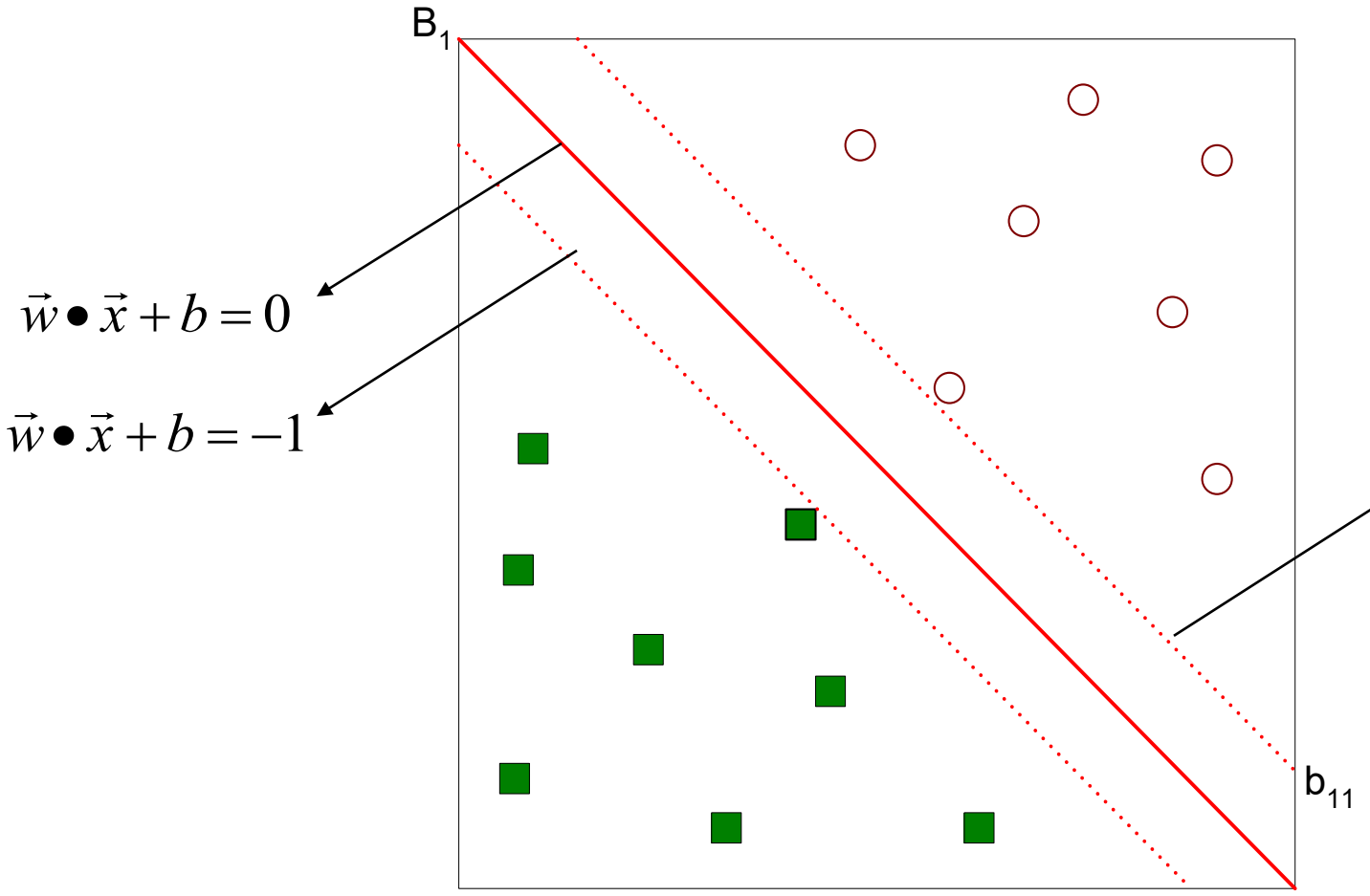
- A **Support Vector Machine** (**SVM**) is an improvement over a perceptron



Support Vectors (points on the boundary)

An SVM selects one particular hyperplane (the green line in the figure) that not only separates the examples into two classes, but does so in a way that maximizes the margin (γ in the figure), which is the distance between the hyperplane and the closest examples of the training set

Support Vector Machines



Examples are;

(x1,..,xn,y) with

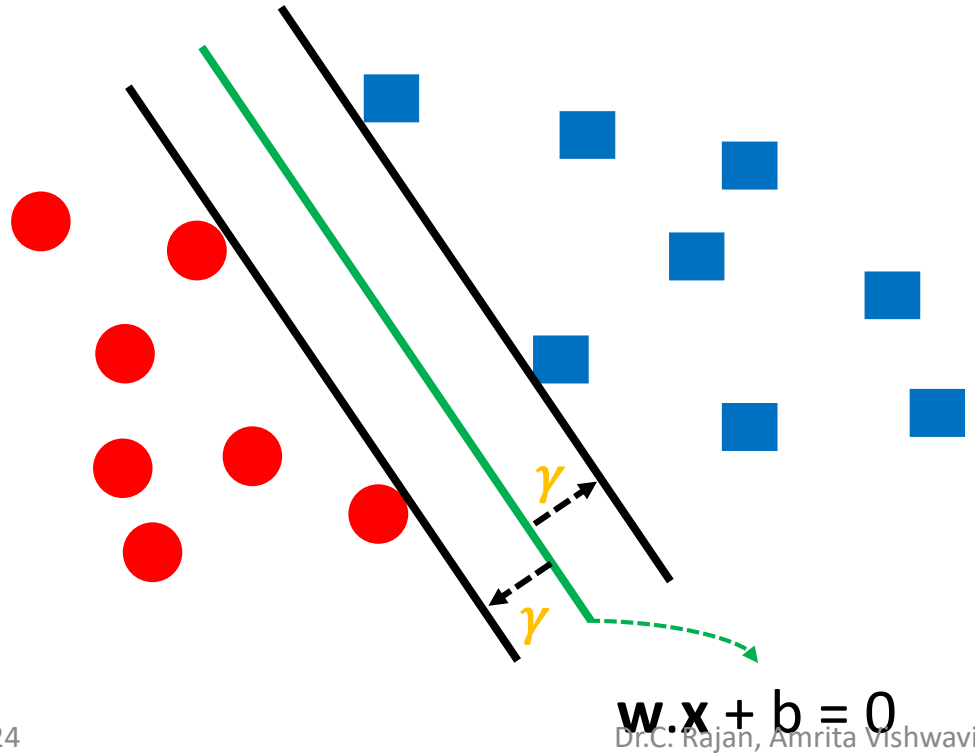
y∈{-1.1}

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|^2}$$

What is the Objective of SVM?

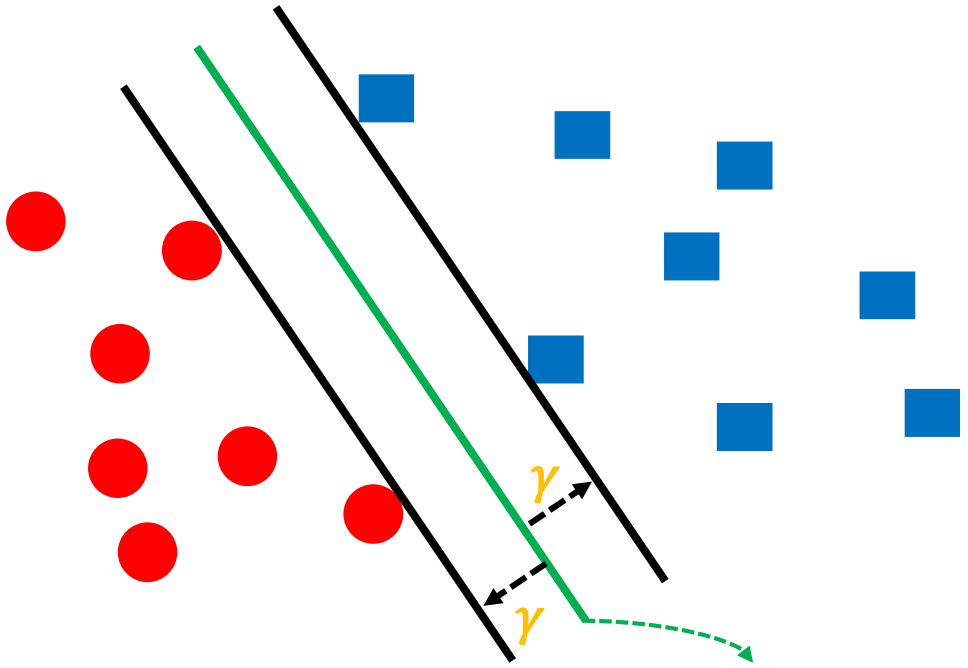
- The objective of an SVM is to select a hyperplane **w**.**x** + b = 0 that maximizes the distance, γ, between the hyperplane and any example in the training set



Intuitively, we are more certain of the class of examples that are far from the separating hyperplane than we are of examples near to that hyperplane

What is the Objective of SVM?

- The objective of an SVM is to select a hyperplane **w**.**x** + b = 0 that maximizes the distance, γ, between the hyperplane and any example in the training set



Thus, it is desirable that all the training examples be as far from the hyperplane as possible (*but on the correct side of that hyperplane, of course!*)

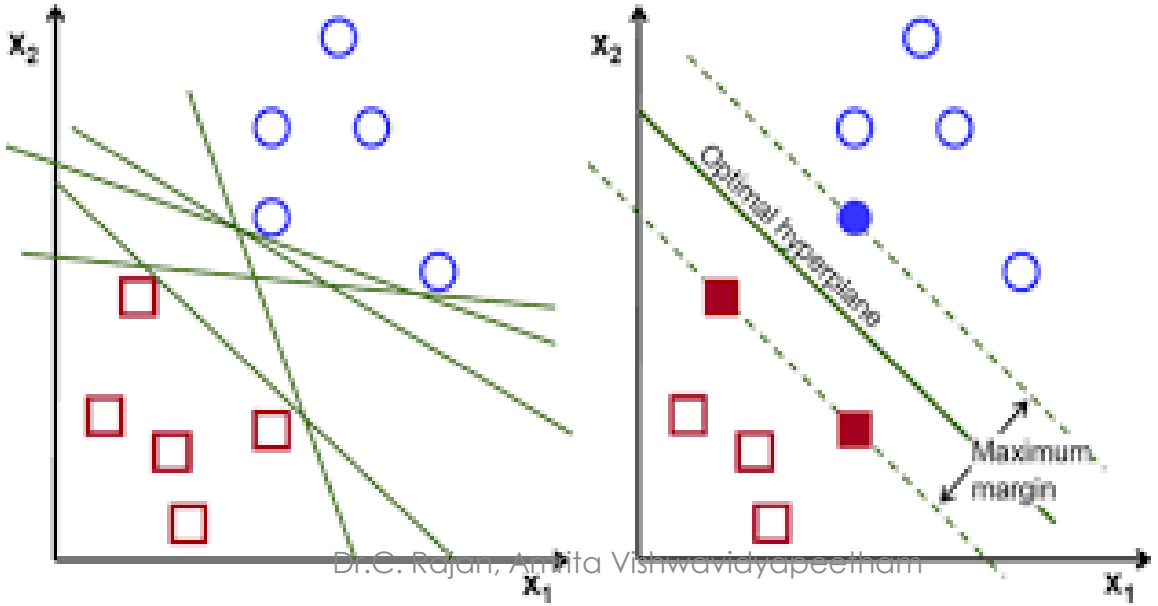
Support Vector Machine Hyperplanes

- A hyperplane is a subspace whose dimension is n-1 than the number of variables in the dataset
- In 3-dimensional datasets, 2-dimensional planes are used to separate data into two distinct groups
- There are many different hyperplanes that can classify data; however we want to find a hyperplane with the maximum margin

$$\underline{\text{p-dimensional hyperplane}}$$
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

Maximum-margin Classifier

- Creates separating hyperplane that is farthest from the first closest training observation
- There is a boundary between the sides of the hyperplane and the width is known as the “margin”, which is maximized



What is the Objective of SVM?

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, maximize γ (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$y^i(w.x^i + b) \geq \gamma$$

Notice that y^i, which must be +1 or −1, determines which side of the hyperplane the point x^i must be on, so the ≥ relationship to γ is always correct!

What is the Objective of SVM?

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, maximize γ (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$y^i(w.x^i + b) \geq \gamma \quad \equiv \quad \begin{cases} w.x^i + b \geq \gamma \text{ if } y^i = +1 \\ w.x^i + b \leq -\gamma \text{ if } y^i = -1 \end{cases}$$

But, by increasing **w** and b, we can always allow a larger value of γ (e.g., If we replace **w** by 2**w** and b by 2b, then for all i, $y^i((2w).x^i + 2b) \geq 2\gamma$

What is the Objective of SVM?

- More formally, the goal of any SVM can be stated as follows:

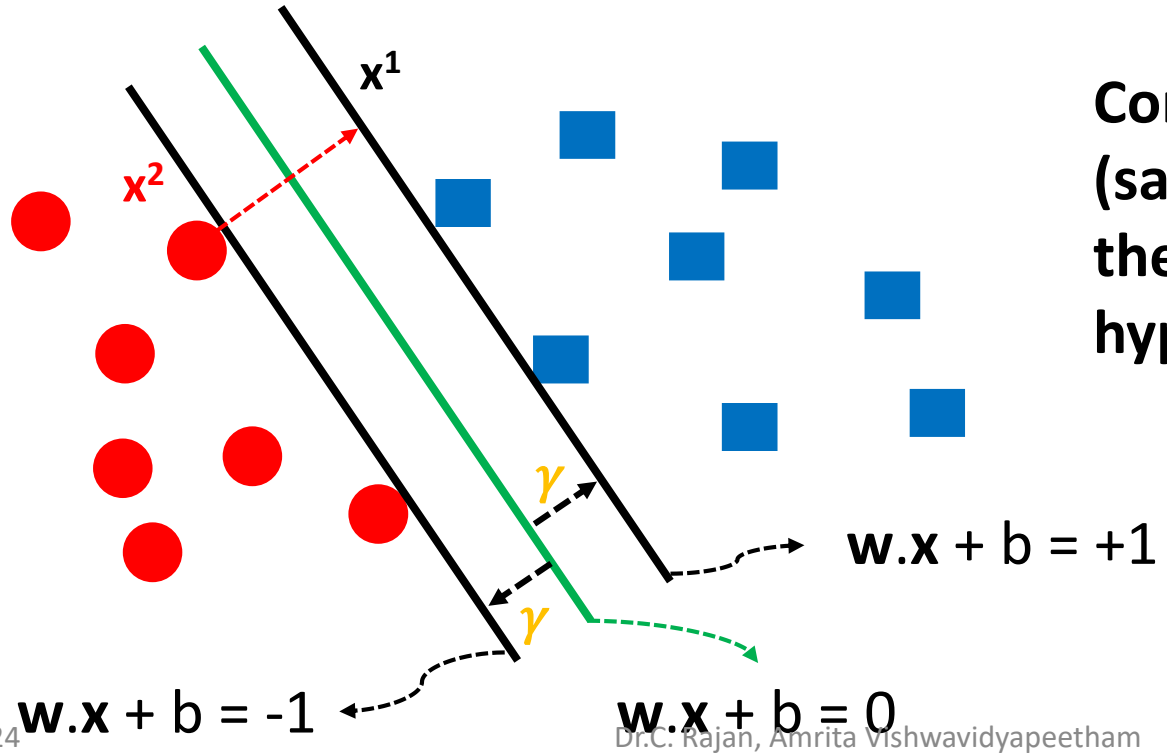
Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, maximize γ (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$y^i(w.x^i + b) \geq \gamma \quad \equiv \quad \begin{cases} w.x^i + b \geq \gamma \text{ if } y^i = +1 \\ w.x^i + b \leq -\gamma \text{ if } y^i = -1 \end{cases}$$

Hence, 2**w** and 2b are always better than w and b, so there is no best choice and no maximum γ!

The Objective of SVM

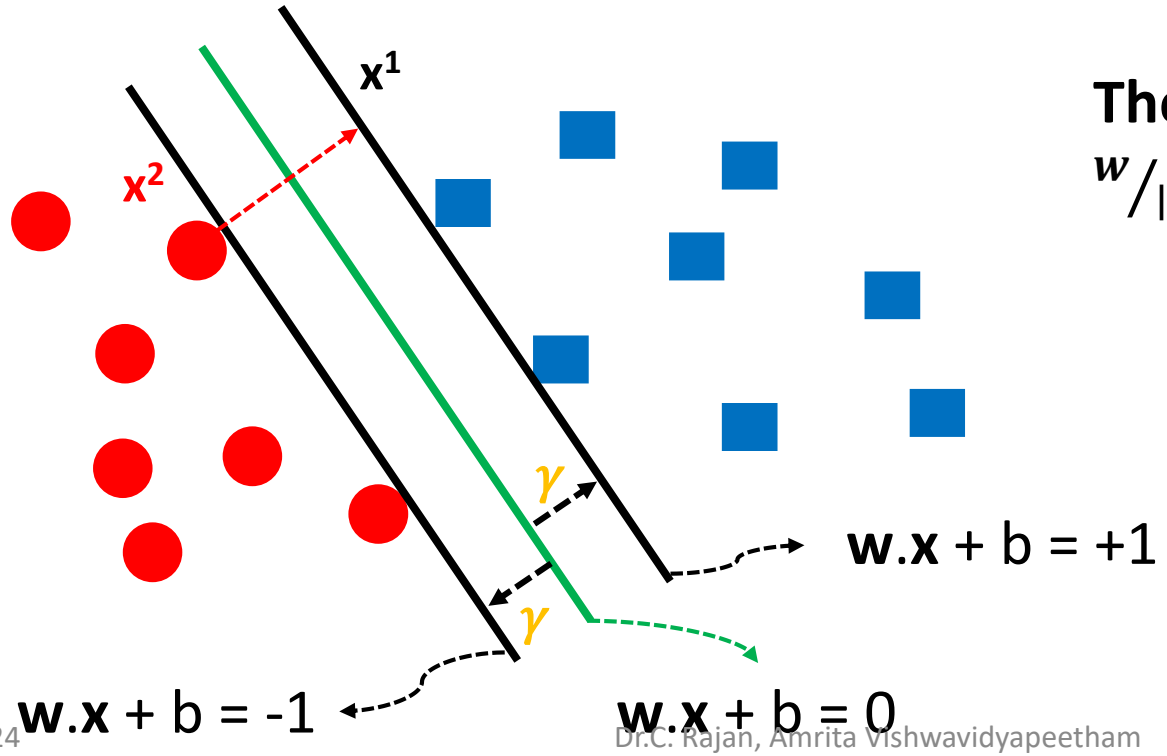
- How can we solve this problem?
  - By normalizing the weight vector w (i.e., $w/\|w\|$), thus making it a unit vector



Consider one of the support vectors, (say, x^2, in the figure) and let x^1 be the projection of x^2 to the upper hyperplane

The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector w (i.e., $w/\|w\|$), thus making it a unit vector

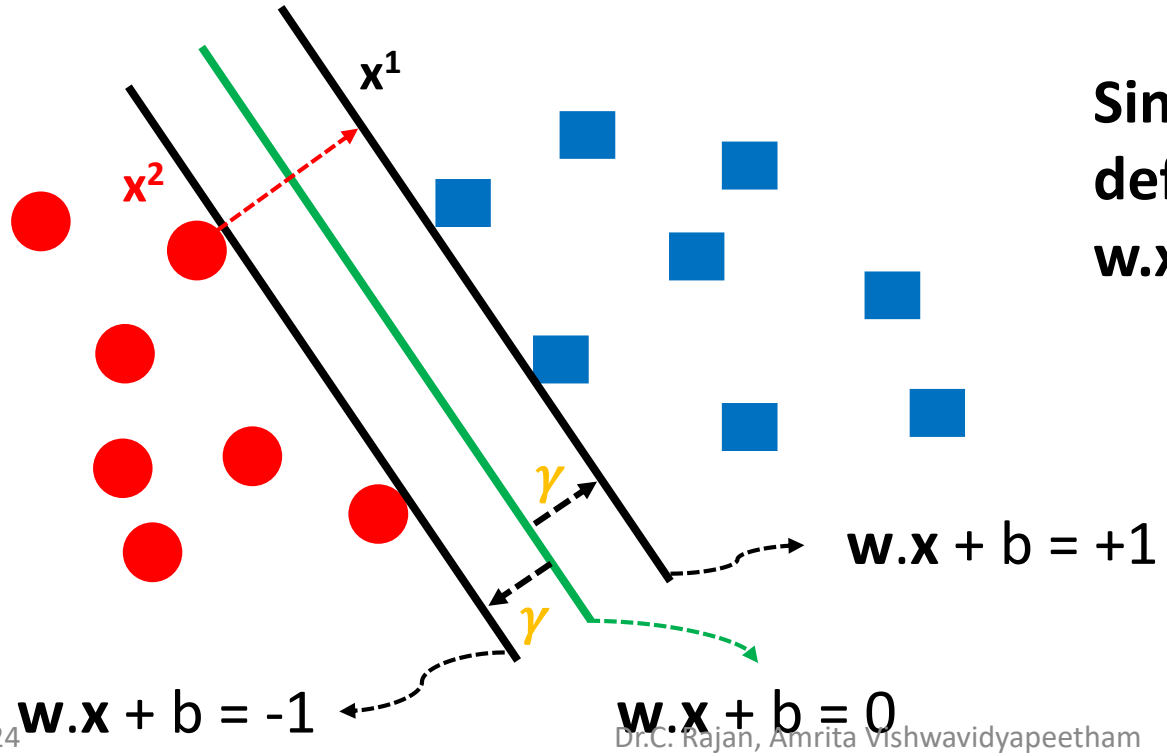


The distance from x^2 to x^1 in units of $w/\|w\|$ is 2γ. That is,

$$x^1 = x^2 + 2\gamma \, w/\|w\|$$

The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector w (i.e., $w/\|w\|$), thus making it a unit vector



Since x^1 is on the hyperplane defined by w.x + b = +1, we know that $w.x^1 + b = 1$. If we substitute for x^1:

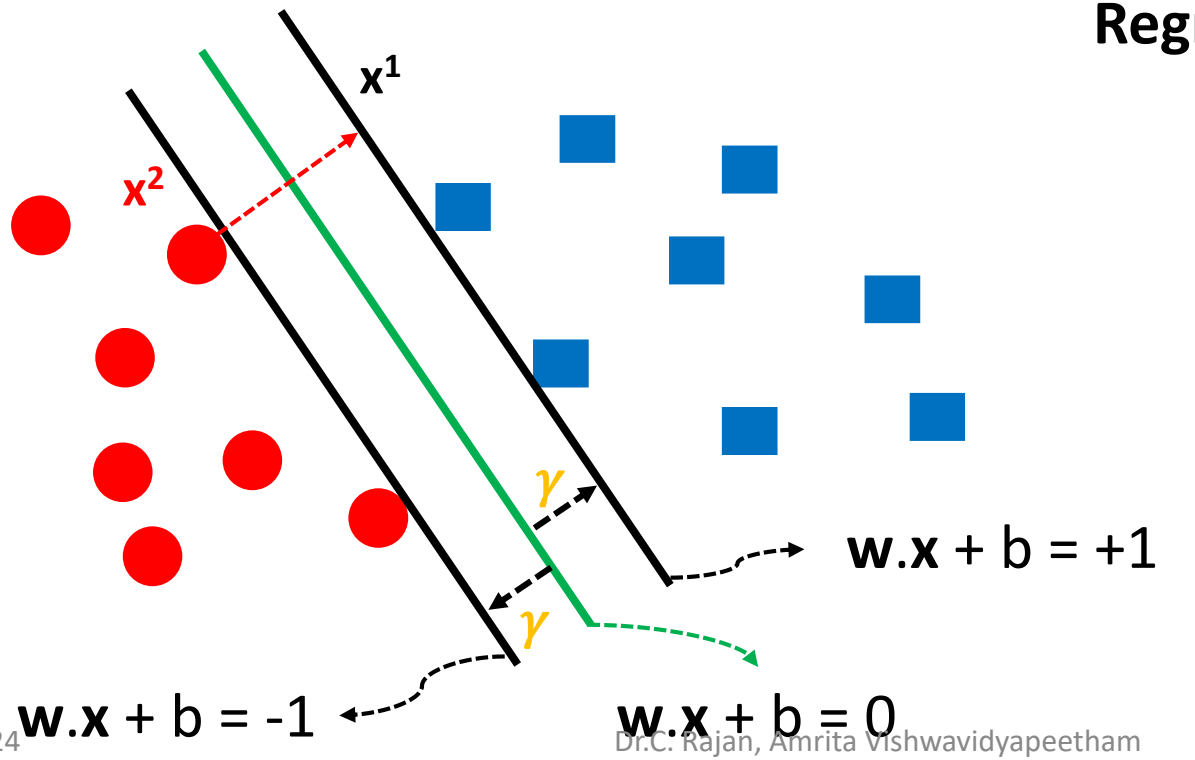
$$x^1 = x^2 + 2\gamma \, w/\|w\|$$

$$\equiv$$

$$w.(x^2 + 2\gamma \, w/\|w\|) + b = 1$$

The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector w (i.e., $w/\|w\|$), thus making it a unit vector



Regrouping terms, we see:

$$x^1 = x^2 + 2\gamma \frac{w}{\|w\|}$$

$$\equiv$$

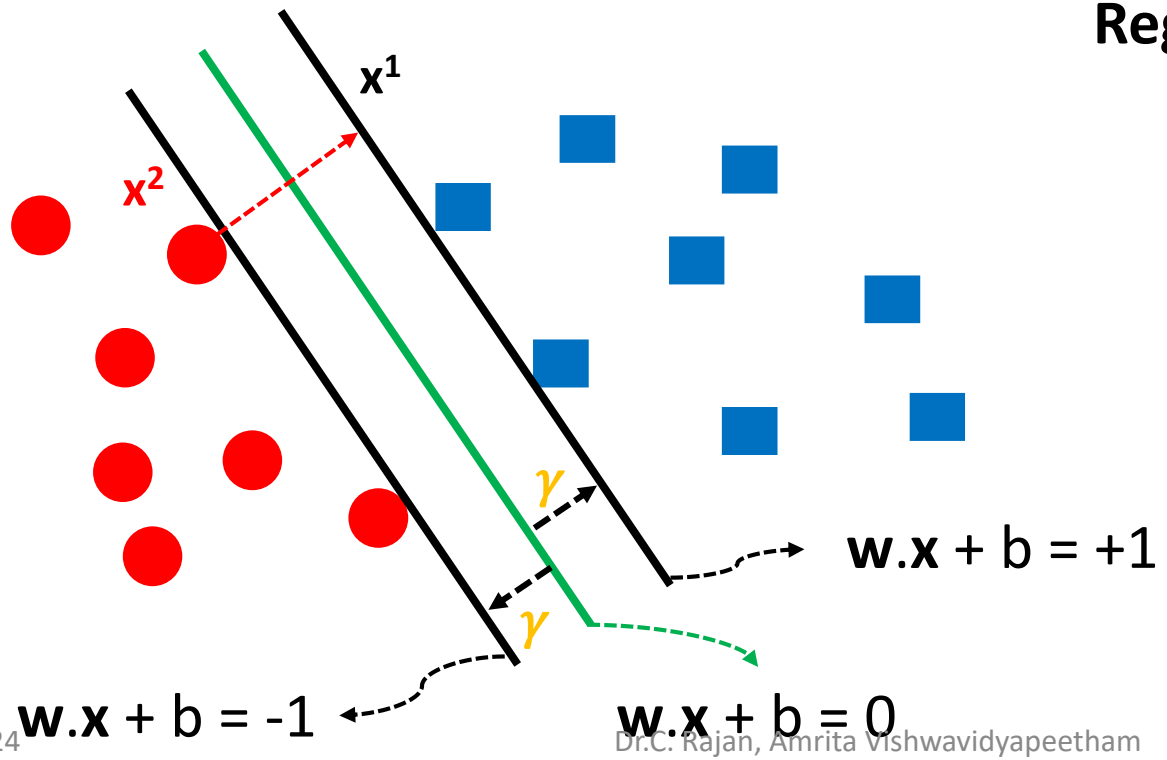
$$w.(x^2 + 2\gamma \frac{w}{\|w\|}) + b = 1$$

$$\equiv$$

$$\underbrace{w.x^2 + b}_{-1} + 2\gamma \frac{w.w}{\|w\|} = 1$$

The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector w (i.e., $w/\|w\|$), thus making it a unit vector



Regrouping terms, we see:

$$x^1 = x^2 + 2\gamma \frac{w}{\|w\|}$$

$$\equiv$$

$$w.(x^2 + 2\gamma \frac{w}{\|w\|}) + b = 1$$

$$\equiv$$

$$w.x^2 + b + 2\gamma \frac{w.w}{\|w\|} = 1$$

$$\equiv$$

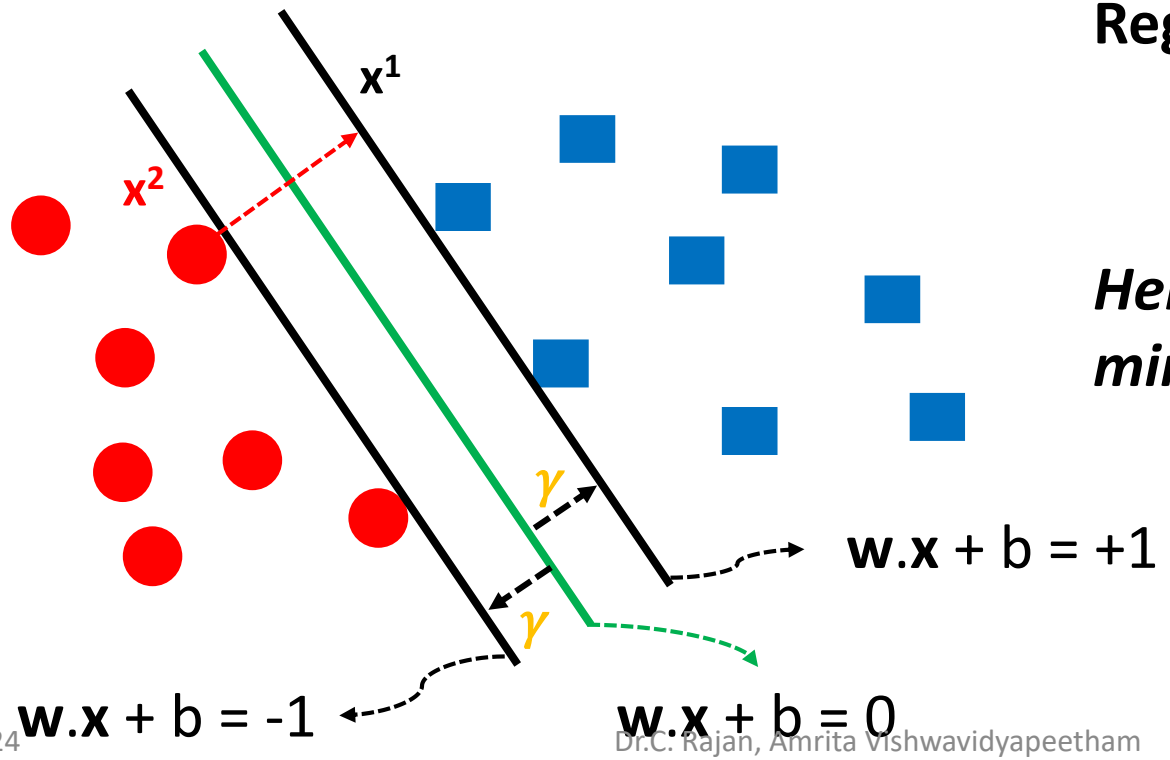
$$\gamma \frac{\|w\|^2}{\|w\|} = 1$$

$$\equiv$$

$$\gamma = \frac{1}{\|w\|}$$

The Objective of SVM

- How can we solve this problem?
  - By normalizing the weight vector w (i.e., $^{w}/_{\|w\|}$), thus making it a unit vector



Regrouping terms, we see:

$$\gamma = {}^{1}/_{\|w\|}$$

Hence, maximizing γ is the same as minimizing $\| w \|$

The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

  Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **maximize** γ (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} w.x^i + b \geq \gamma \text{ if } y^i = +1 \\ w.x^i + b \leq \gamma \text{ if } y^i = -1 \end{cases}$$

Support Vector Machines

- We want to maximize: $$\text{Margin} = \frac{2}{\| \vec{w} \|^2}$$
  - Which is equivalent to minimizing: $$L(w) = \frac{\| \vec{w} \|^2}{2}$$
  - But subjected to the following N constraints:

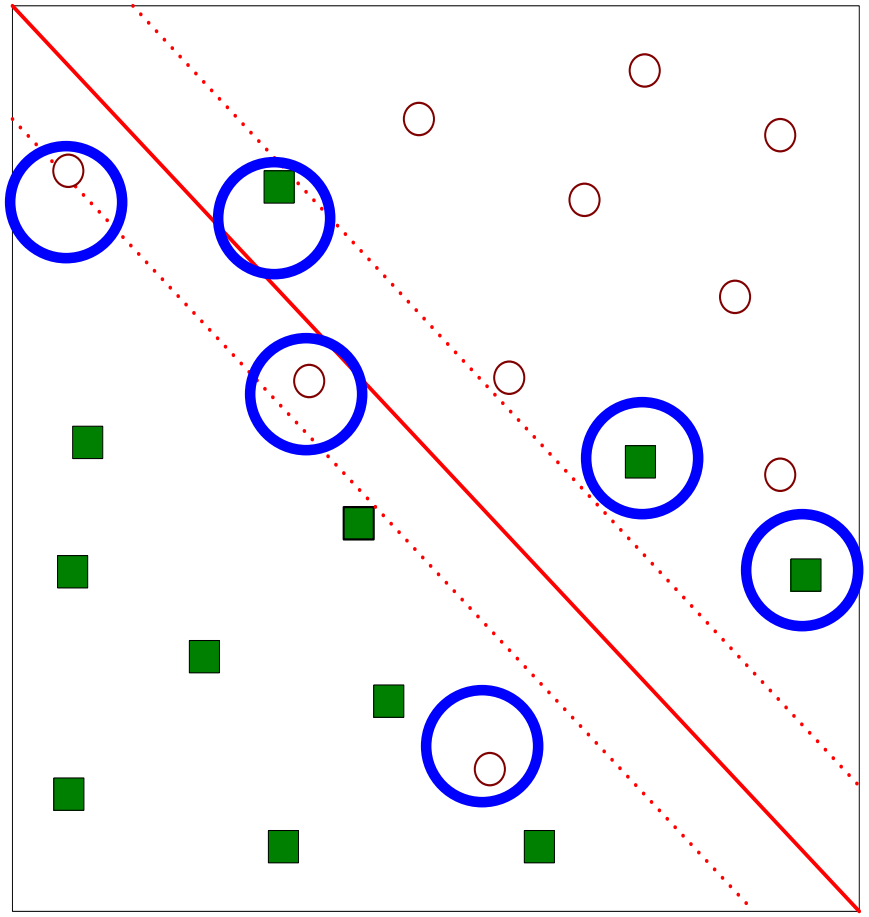
$$\begin{cases} y_i(\vec{w} \bullet \vec{x}_i + b) \geq 1 \; i = 1,.., N \end{cases}$$

◆ This is a constrained convex quadratic optimization problem that can be solved in polynominal time

- Numerical approaches to solve it (e.g., quadratic programming) exist
- The function to be optimized has only a single minimum →no local minimum problem

Support Vector Machines

- What if the problem is not linearly separable?

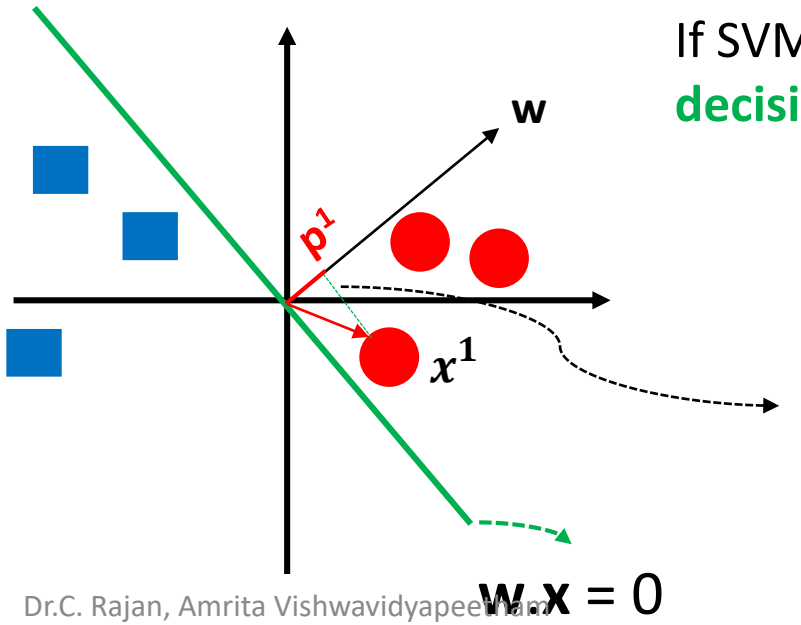


The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **minimize $\| \boldsymbol{w} \|$** (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} \mathbf{p}^i . \| \boldsymbol{w} \| \geq 1 \text{ if } y^i = +1 \\ \mathbf{p}^i . \| \boldsymbol{w} \| \leq -1 \text{ if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

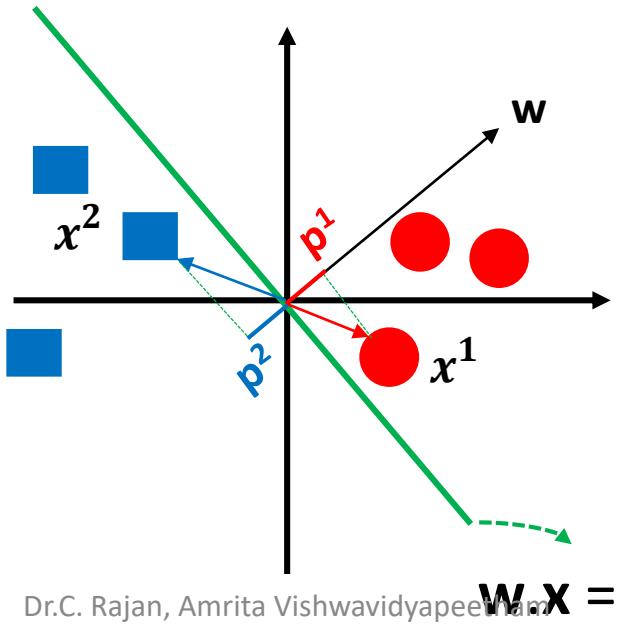
Project x^1 onto **w**

The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **minimize $\| \boldsymbol{w} \|$** (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} \mathbf{p^i}. \| \boldsymbol{w} \| \geq 1 \text{ if } y^i = +1 \\ \mathbf{p^i}. \| \boldsymbol{w} \| \leq -1 \text{ if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

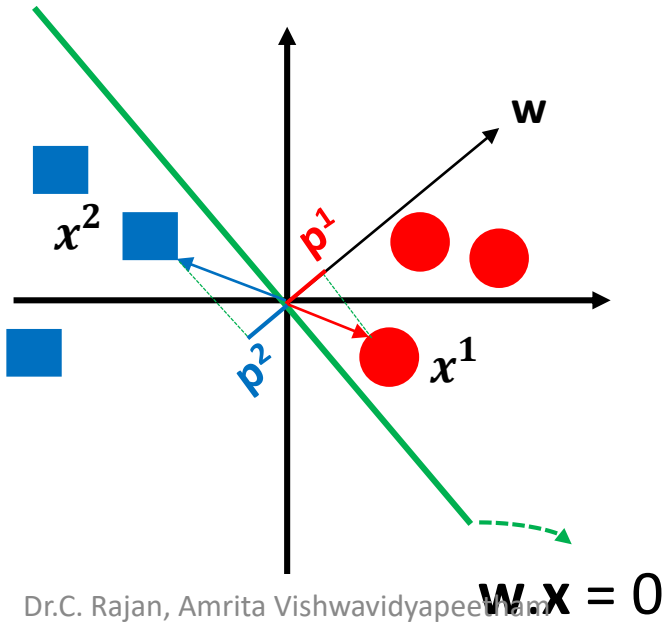
p^1 is positive and very small, hence, for $\mathbf{p^1}. \| \boldsymbol{w} \|$ to be ≥ 1, $\| \boldsymbol{w} \|$ has to be very large!

The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **minimize $\| \boldsymbol{w} \|$** (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} \mathbf{p}^i . \| \boldsymbol{w} \| \geq 1 \text{ if } y^i = +1 \\ \mathbf{p}^i . \| \boldsymbol{w} \| \leq -1 \text{ if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

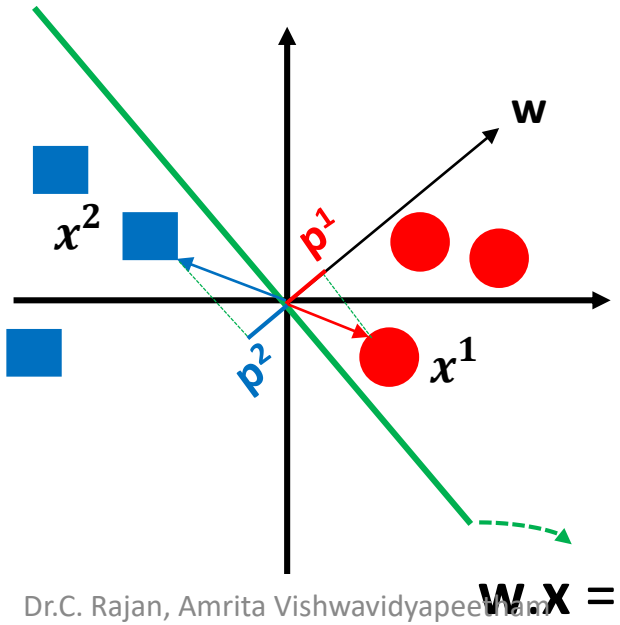
But, the optimization objective is to minimize $\| \boldsymbol{w} \|$, hence, SMV will not prefer this decision boundary

The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **minimize $\| \boldsymbol{w} \|$** (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} \mathbf{p^i}. \| \boldsymbol{w} \| \geq 1 \text{ if } y^i = +1 \\ \mathbf{p^i}. \| \boldsymbol{w} \| \leq -1 \text{ if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

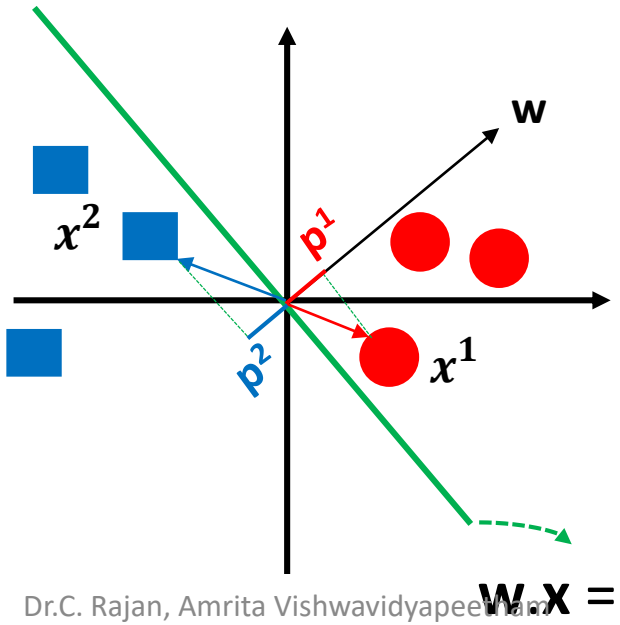
p^2 is negative and very small, hence, for $\mathbf{p^2}. \| \boldsymbol{w} \|$ to be ≤ -1, $\| \boldsymbol{w} \|$ has to be very large!

The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **minimize $\| \boldsymbol{w} \|$** (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} \mathbf{p}^i . \| \boldsymbol{w} \| \geq 1 \text{ if } y^i = +1 \\ \mathbf{p}^i . \| \boldsymbol{w} \| \leq -1 \text{ if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

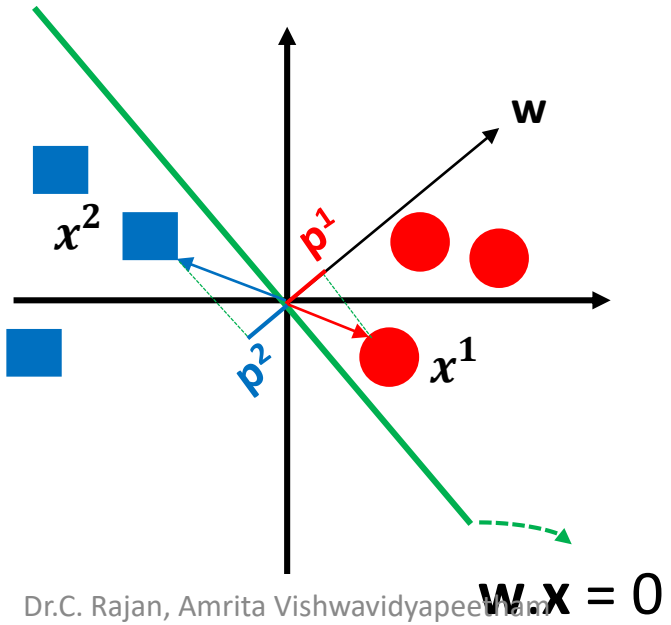
But, the optimization objective is to minimize $\| \boldsymbol{w} \|$, hence, again, SMV will not prefer this decision boundary

The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **minimize $\| \boldsymbol{w} \|$** (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} \mathbf{p}^i . \| \boldsymbol{w} \| \geq 1 \text{ if } y^i = +1 \\ \mathbf{p}^i . \| \boldsymbol{w} \| \leq -1 \text{ if } y^i = -1 \end{cases}$$



If SVM encounters this **green decision boundary**, will it choose it?

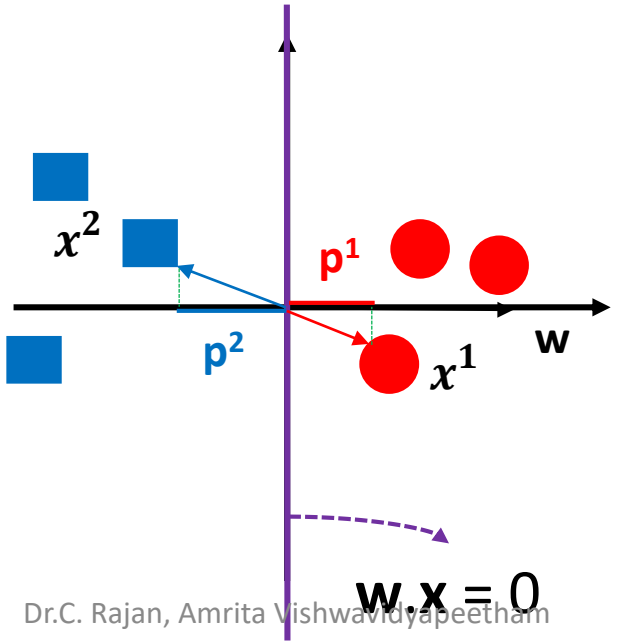
NO

The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **minimize $\| \boldsymbol{w} \|$** (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} \mathbf{p^i}. \| \boldsymbol{w} \| \geq 1 \text{ if } y^i = +1 \\ \mathbf{p^i}. \| \boldsymbol{w} \| \leq -1 \text{ if } y^i = -1 \end{cases}$$



If SVM encounters this **purple decision boundary**, will it choose it?

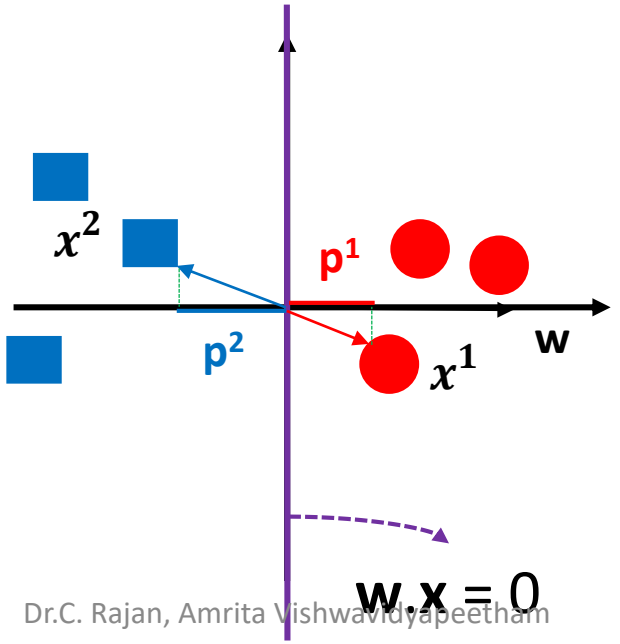
$\mathbf{p^1}$ is positive and bigger now, hence, for $\mathbf{p^1}$. $\| \boldsymbol{w} \|$ to be ≥ 1, $\| \boldsymbol{w} \|$ can be smaller, aligning better with the optimization objective

The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **minimize $\| \boldsymbol{w} \|$** (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} \mathbf{p^i}. \| \boldsymbol{w} \| \geq 1 \text{ if } y^i = +1 \\ \mathbf{p^i}. \| \boldsymbol{w} \| \leq -1 \text{ if } y^i = -1 \end{cases}$$



If SVM encounters this **purple decision boundary**, will it choose it?

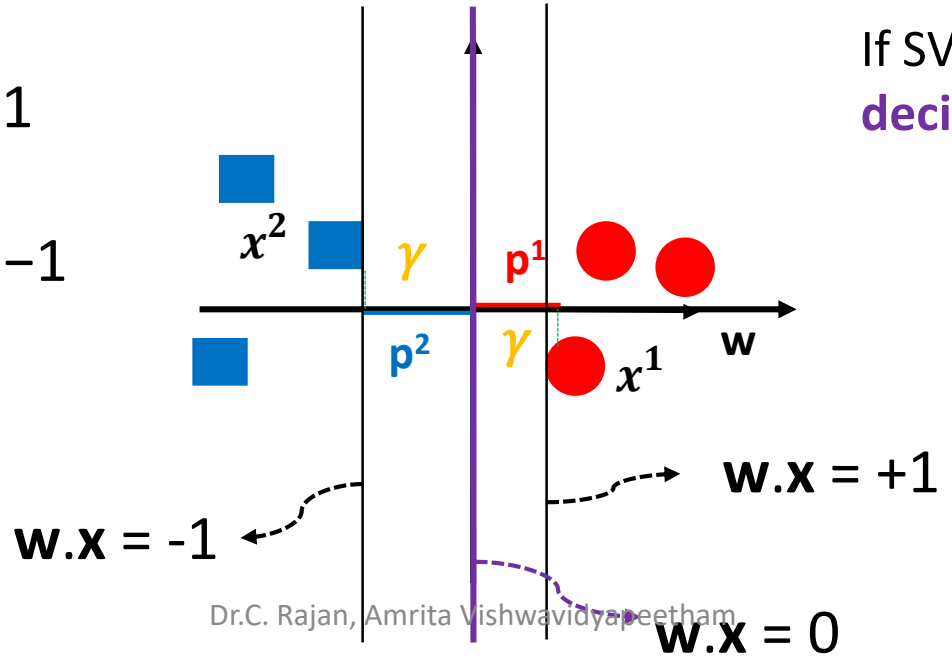
$\mathbf{p^2}$ is negative and bigger now, hence, for $\mathbf{p^2}$. $\| \boldsymbol{w} \|$ to be ≤ -1, $\| \boldsymbol{w} \|$ can be smaller, aligning better with the optimization objective

The Objective of SVM

- More formally, the goal of any SVM can be stated as follows:

Given a training set $(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)$, **minimize $\| \boldsymbol{w} \|$** (by varying **w** and b) subject to the constraint that for all i = 1, 2, . . . , n,

$$\begin{cases} \mathbf{p}^i . \| \boldsymbol{w} \| \geq 1 \text{ if } y^i = +1 \\ \mathbf{p}^i . \| \boldsymbol{w} \| \leq -1 \text{ if } y^i = -1 \end{cases}$$



If SVM encounters this **purple decision boundary**, will it choose it?

YES

Example

- Consider the following training examples, assuming **w** = [u, v] and $\gamma = 1$
  - Our goal is to minimize $\sqrt{u^2 + v^2}$ subject to the constraints $y_i(\mathbf{w}.x_i + b) \geq \gamma$, which can be derived from the training examples

x	y	Constraints
[1, 2]	+1	(+1)(u + 2v + b) = u + 2v + b ≥ 1
[2, 1]	-1	2u + v + b ≤ −1
[3, 4]	+1	3u + 4v + b ≥ 1
[4, 3]	-1	4u + 3v + b ≤ −1

How to solve for u, v, and b?

Example

- Consider the following training examples, assuming **w** = [u, v] and $\gamma = 1$
  - Our goal is to minimize $\sqrt{u^2 + v^2}$ subject to the constraints $y_i(\mathbf{w}.x_i + b) \geq \gamma$, which can be derived from the training examples

x	y	Constraints
[1, 2]	+1	(+1)(u + 2v + b) = u + 2v + b ≥ 1
[2, 1]	-1	2u + v + b ≤ −1
[3, 4]	+1	3u + 4v + b ≥ 1
[4, 3]	-1	4u + 3v + b ≤ −1

In this very simple case, it is easy to see that b = 0 and **w** [-1, +1]

Example

- Consider the following training examples, assuming **w** = [u, v] and $\gamma = 1$
  - Our goal is to minimize $\sqrt{u^2 + v^2}$ subject to the constraints $y_i(\mathbf{w}.x_i + b) \geq \gamma$, which can be derived from the training examples

x	y	Constraints
[1, 2]	+1	(+1)(u + 2v + b) = u + 2v + b ≥ 1
[2, 1]	-1	2u + v + b ≤ −1
[3, 4]	+1	3u + 4v + b ≥ 1
[4, 3]	-1	4u + 3v + b ≤ −1

In general, we can use *gradient descent*!

Linear SVM for Non-linearly Separable Problems

- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize:

$$L(w) = \frac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k \right)$$

Parameter

Inverse size of margin between hyperplanes

Measures testing error

    - Subject to (i=1,..,N):

$$\begin{cases} (1) & y_i * (\vec{w} \bullet \vec{x}_i + b) \geq 1 - \xi_i \\ (2) & 0 \leq \xi_i \end{cases}$$

Slack variable

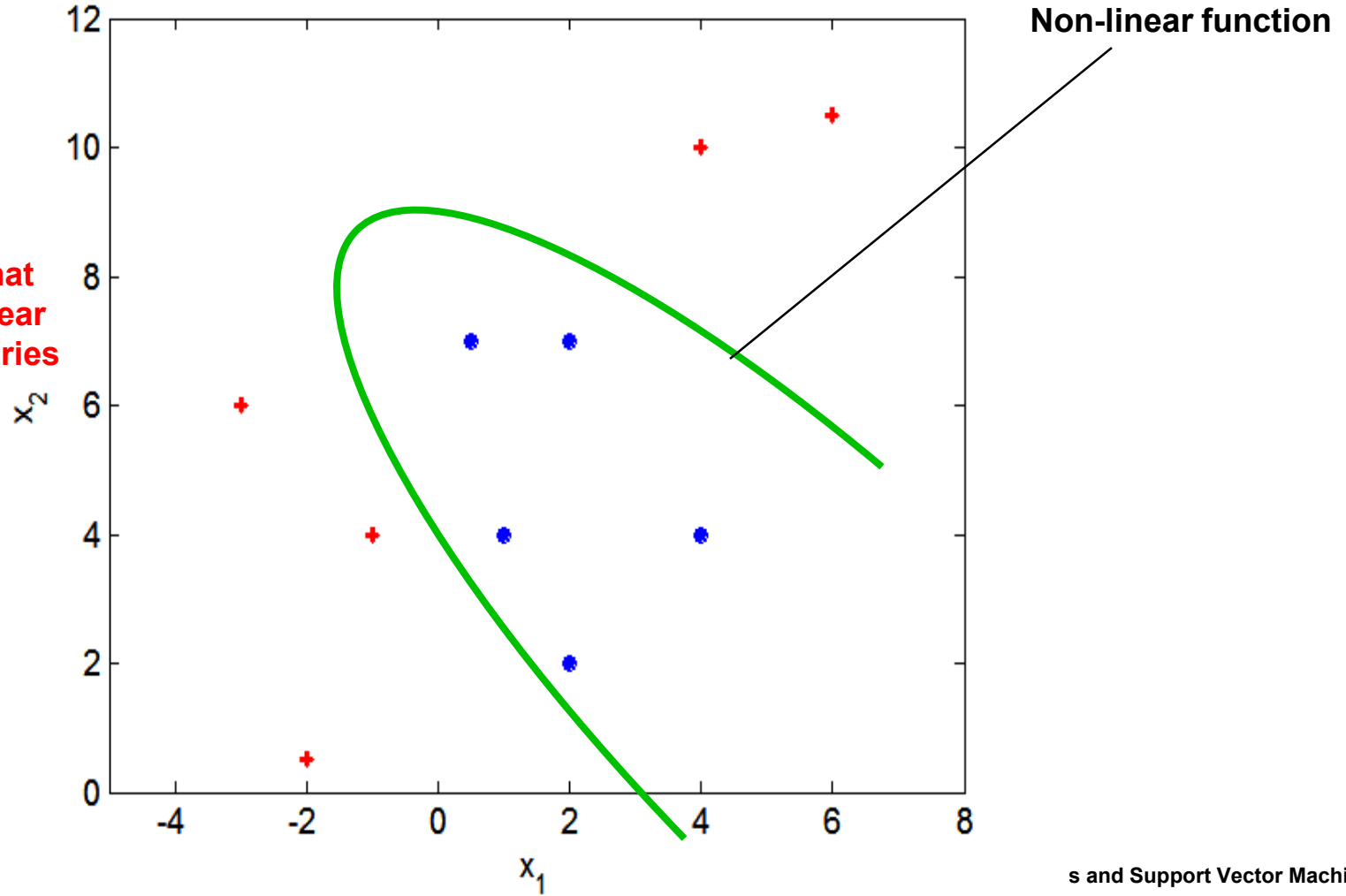
allows constraint violation to a certain degree

    - C is chosen using a validation set trying to keep the margins wide while keeping the training error low.

Nonlinear Support Vector Machines

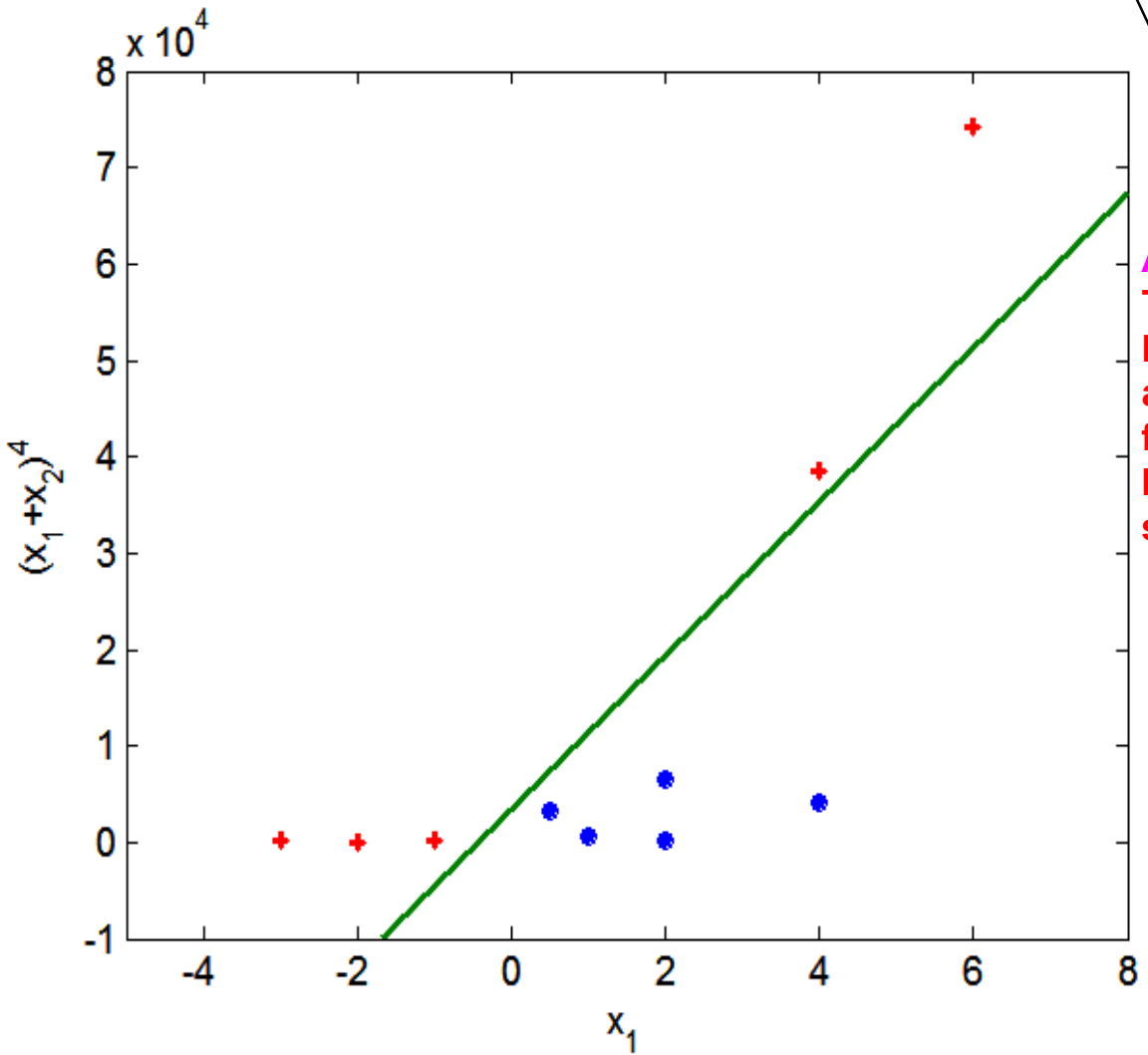
- What if decision boundary is not linear?

Alternative 1:
Use technique that Employs non-linear decision boundaries



Nonlinear Support Vector Machines

1. Transform data into higher dimensional space
2. Find the best hyperplane using the methods introduced earlier



Alternative 2:
Transform into a higher dimensional attribute space and find linear decision boundaries in this space

Nonlinear Support Vector Machines

1. Choose a non-linear kernel function ϕ to transform into a different, usually higher dimensional, attribute space
2. Minimize $L(w) = \frac{\| \vec{w} \|^2}{2}$

Find a good hyperplane
In the transformed space

– but subjected to the following N constraints:

$$\begin{cases} y_i(\vec{w} \bullet \phi(\vec{x}_i) + b) \geq 1 \; i = 1,.., N \end{cases}$$

Example: Polynomial Kernal Function

Polynomial Kernel Function:

$\Phi(x1,x2)=(x1^2,x2^2,sqrt(2)*x1,sqrt(2)*x2,1)$

$K(u,v)=\Phi(u)\bullet\Phi(v)= (u\bullet v + 1)^2$

Kernel function trick: perform computations in the original space, although we solve an optimization problem in the transformed space → more efficient!!

A Support Vector Machine with polynomial kernel function classifies a new example z as follows:

$sign((\Sigma\ \lambda_i y_i * \Phi(x_i)\bullet\Phi(z))+b)$ =

$sign((\Sigma\ \lambda_i y_i * (x_i\bullet z +1)^2))+b)$

Remark: λ_i and b are determined using the methods for linear SVMs that were discussed earlier

Summary Support Vector Machines

- Support vector machines learn hyperplanes that separate two classes maximizing the *margin between them* (*the empty space between the instances of the two classes)*.
- Support vector machines introduce slack variables, in the case that classes are not linear separable and trying to maximize margins while keeping the training error low.
- The most popular versions of SVMs use non-linear kernel functions to map the attribute space into a higher dimensional space to facilitate finding “good” linear decision boundaries in the modified space.
- Support vector machines find “margin optimal” hyperplanes by solving a convex quadratic optimization problem. However, this optimization process is quite slow and support vector machines tend to fail if the number of examples goes beyond 500/2000/5000…
- In general, support vector machines accomplish quite high accuracies, if compared to other techniques.

Useful Support Vector Machine Links

Lecture notes are much more helpful to understand the basic ideas:
http://www.ics.uci.edu/~welling/teaching/KernelsICS273B/Kernels.html
http://cerium.raunvis.hi.is/~tpr/courseware/svm/kraekjur.html

Some tools are often used in publications

livsvm: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

spider: http://www.kyb.tuebingen.mpg.de/bs/people/spider/index.html

Tutorial Slides: http://www.support-vector.net/icml-tutorial.pdf

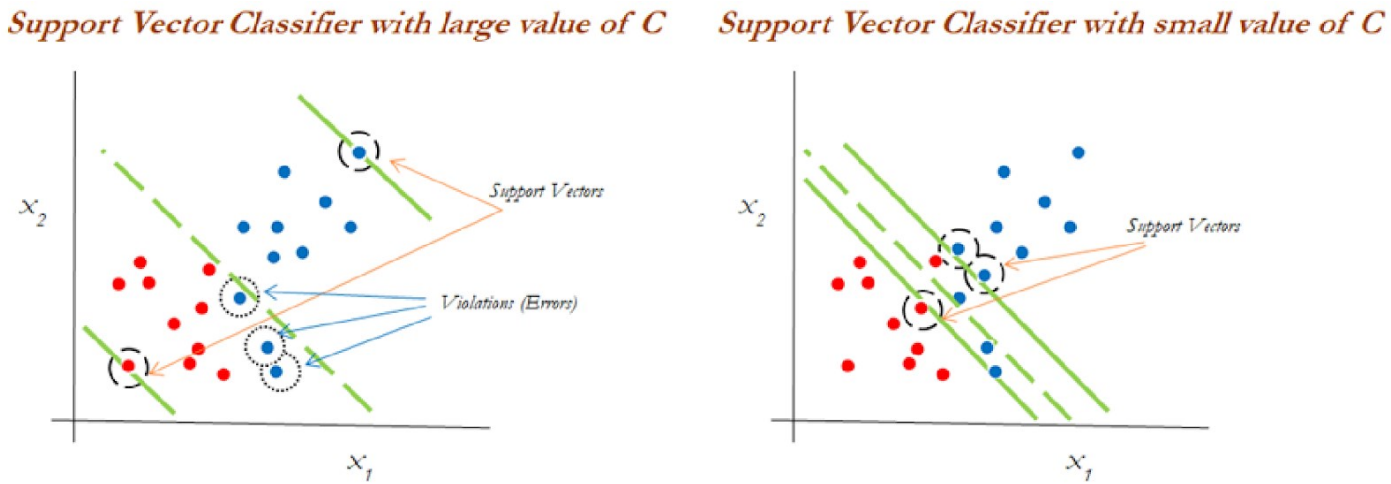
Surveys: http://www.svms.org/survey/Camp00.pdf

More General Material:
http://www.learning-with-kernels.org/
http://www.kernel-machines.org/
http://kernel-machines.org/publications.html
http://www.support-vector.net/tutorial.html

Remarks: Thanks to Chaofan Sun for providing these links!

Support Vectors

- We find that observations that lie on the margin, or those who violate the margin, are the only observations that affect the hyperplane
- Observations that lie on the correct side of the margin do not influence the hyperplane at all
- These observations on the wrong side of the margin are called “Support Vectors”



Creating the Maximum Margin Hyperplane

- An optimization problem that has a constraint,

$$y_i\left(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{p1}\right) \geq M \; \forall i = 1, 2, \ldots$$

- Where we want to maximize M subject to,

$$\sum_{j=1}^{p} \beta^2{}_j = 1$$

- The optimization of the Maximum Margin Hyperplane is handled by R

Support Vector Machine Optimization

- The linear support classifier has a solution function of,

$$f(x) = \beta_0 + \sum_{j=1}^{n} \alpha_j \langle x, x_j \rangle$$

- Where α_j and β_0 are parameters measured by all the pairs of the inner products of the training data
- α_j is nonzero only for support vectors and α_j is equal to zero if not
- Thus, our solution function can be rewritten as,

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_j \langle x, x_j \rangle$$

- Where S is the collection of support vectors which results in fewer computations

Linear SVMs Mathematically

- We can formulate the *quadratic optimization problem:*

  Find **w** and *b* such that $\rho = \frac{2}{||w||}$ is maximized

  and for all ($\mathbf{x}_i$, y_i), i=1..n : $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- Which can be reformulated as:

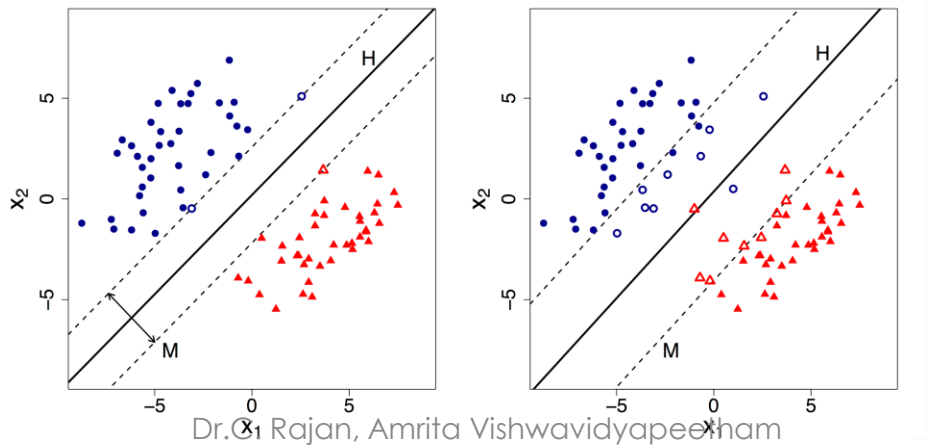
  Find w and *b* such that

  $\Phi(w) = ||w||^2 = w^T w$ is minimized

  and for all (x_i, y_i), i=1..n : $y_i(w^T x_i + b) \geq 1$

Support Vector Classifiers

- Support Vector Classifier (soft margin classifier) is more robust and tends to be a better classifier than maximum margin classifier
- Sometimes a perfect separation of data is not possible, therefore some observations could be on the incorrect side of the margin
- We introduce a new slack variable ϵ_i that allows individual data points to be in the wrong side of the margin.



Support Vector Classifiers Optimization

- An optimization problem that has a constraint,

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{p1}) \geq M(1 - \varepsilon_i)\ \forall i = 1, 2, \ldots$$

- Subject to,

$$\sum_{j=1}^{p} {\beta^2}_j = 1$$

- And a turning parameter C,

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i < C$$

- C bounds the sums of ϵ_i and determines the number and severity of the violations in the margin region.

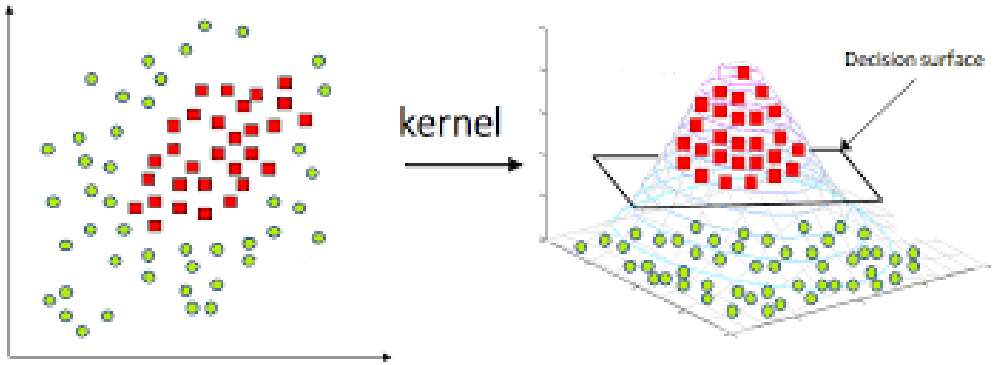
Support Vector Classifiers Optimization

- C is treated as a tuning parameter
- C balances the bias-variance trade-off seen in the dataset

Small Tuning Parameter	Large Tuning Parameter
- Narrow Margins	- Wider Margins
- Few Violations	- More Violations
- More Biased	- Less Biased
- Less Variance	- High Variance

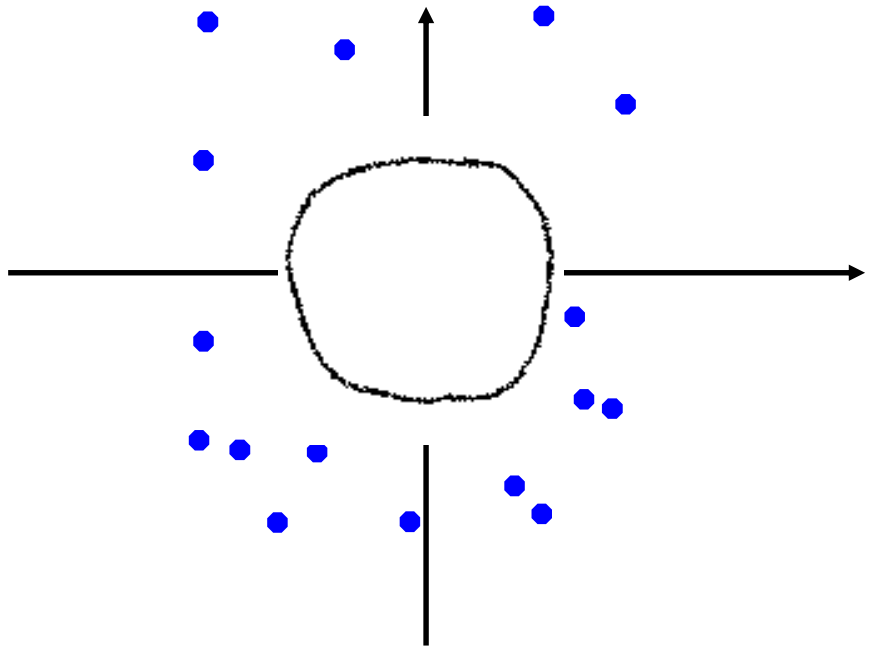
Support Vector Machine

- Support Vector Machine is an extension of support vector classifier
- Support Vector Machine expands the feature space by introducing kernels
- Kernels removes the computational requirements for higher dimension vector spaces and allows us to deal with non-linear data
- There are three common types of kernels: linear, polynomial, radial



Problems with Linear SVM

- What if the decision function is not a linear?



General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable.

Kernel Trick

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$:

  $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} =$

  $= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$

  $= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ where $\phi(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

Kernels

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space

Kernel examples

- Linear: $K(\mathbf{x}_i,\mathbf{x}_j)= \mathbf{x}_i^T\mathbf{x}_j$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ is $\mathbf{x}$ itself

- Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j)= (1+ \mathbf{x}_i^T\mathbf{x}_j)^p$
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ has dimensions

- Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j)$ =
  - Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$, where $\boldsymbol{\phi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

- Higher-dimensional space still has *intrinsic* dimensionality *d* (the mapping is not *onto*), but linear separators in it correspond to *non-linear* separators in original space.

Generalization of Kernels

- We can generalize our inner product by using a kernel seen as,

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

- The kernel function can be transformed into a polynomial kernel of degree d,

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} (x_{ij} x_{i'j}))^d$$

- The solution function has the form,

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_j \, K(x_i, x_{i'})$$

Generalization of Kernels

- The kernel function can also be transformed into a radical kernel,

$$K(x_i, x_{i'}) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

- Where the solution function has the form,

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_j \, K(x_i, x_{i'})$$

Using Support Vector Machine in R

- I used a dataset named spam7 that contains data whether an email is spam or not
- The 6 predictors of the dataset were
  - crl.tot: total length of words in capitals
  - dollar: number of occurrences of the \$ symbol
  - bang: number of occurrences of the ! symbol
  - money: number of occurrences of the word ‘money’
  - n000: number of occurrences of the string ‘000’
  - make: number of occurrences of the word ‘make’
- The response variable “yesno” was the indication whether it is spam or not
  - yesno: outcome variable, a factor with levels n not spam, y spam

Sorting Train and Test data in R

```
#packages
library(DAAG) #Spam7 Data Set
library(e1071) #Support Vector Machine Library

#Error Function - confusion matrix
compute_error = function(a,b)
{
  tmp = table(a,b)
  out = ( sum(tmp) - sum(diag(tmp)) ) / sum(tmp)
  out
}

#Test if Factor
class(train)
?spam7
set.seed(667)
ind = sample(1:5, size = nrow(spam7), replace = TRUE)
#test dataset is 1/5 size of total dataset
#train data set is 4/5 size of total dataset

test_index =  which(ind == 1)
train      =  spam7[ -test_index, ]
test       =  spam7[  test_index, ]

dim(train)
dim(test)
train
test
```

First Prediction of Support Vector Machine Code

Support Vector Machine Code

```
#SVM Model
svm.spam = svm(yesno ~ . , data = train, kernel = 'linear', gamma = 1, cost = 1e5)
summary(svm.spam)
svm.spam$index
ypredict_1 = predict(svm.spam, test)
table(predict = ypredict_1, truth = test$yesno)
#Error of prediction
svm_svm_error = compute_error(test$y, ypredict_1)
svm_svm_error
```

Result

- 1399 Support Vectors
- Linear Kernel
- A confusion matrix is a table that displays the incorrect matches between the actual response and predicted response

	True Value		
Predicted Value		No	Yes
	No	542	153
	Yes	19	208

- From the confusion matrix, we can calculate the test error which are the diagonal points 19 and 153 over the total number of data points
- We received a test error of 18.65%

Tuned Support Vector Machine Code

- R supports many tuning functions to reduce the test error in our models

Tuned Code

```
#Tuned model through cross validation
tune.out=tune(svm ,yesno ~ . ,data=train ,kernel = 'linear',
              ranges=list(cost=c(.0001, 0.001, 0.01, 0.1, 1,5,10,100), gamma = c(0.001,
0.01, .1, 1) ))
tune.out
summary(tune.out)
bestmod = tune.out$best.model
summary(bestmod)
#Prediction through test data set
ypredict = predict(bestmod, test)
table(predict = ypredict, truth = test$yesno)
#Error of prediction
svm_bestmodel_error = compute_error(test$y, ypredict )
svm_bestmodel_error
```

Result

- 1531 Support Vectors
- Polynomial Kernel
- Best parameters obtained through using tuning; cost = 100 and Gamma = 0.001

	True Value		
Predicted Value		No	Yes
	No	541	114
	Yes	20	247

- After using our tuning function, we lowered our test error down to 14.52%
- This meant that almost 9/10 predictions were correct using the support vector machine algorithm!