

Contents

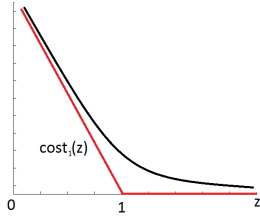
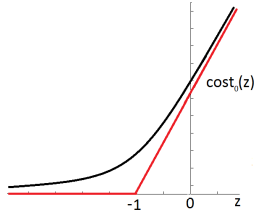
I	Week 7	1
0.1	Optimization objectives	2
0.1.1	cost function	2
0.1.2	SVM Optimization Objective	2
0.2	SVM - Large Margin Classifier	3
0.2.1	constraint for SVM	3
0.2.2	For large C	4
0.2.3	Visualization of the decision boundary for large C	4
0.3	Mathematics behind Large Margin classifier	5
0.4	SVM Kernels I	6
0.4.1	Example	6
0.4.2	landmarks	7
0.4.3	How to choose the landmarks	8
0.4.4	How to choose C and σ	9
0.5	How to use an SVM	9
0.6	Multi-class Classification	9
	Appendices	10
.1	question	11
.2	Vector inner product ($u^T v$)	11
.3	SVM Intro	11

Support Vector Machines
Effective algorithm for non-linear classification
(non-linear decision boundary)

March 26, 2018

Part I

Week 7

	if $y = +1$	if $y = 0$
		
new cost function (SVM)	$(cost_1(z))$ [red line]	$(cost_0(z))$
new condition	(want $\theta^T x \geq 1$)	(want $\theta^T x \leq -1$)

Support Vector Machine is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. Regression cannot deal with complex data. SVM works well on small dataset, but can be stronger and more powerful in building complex models.

0.1 Optimization objectives

0.1.1 cost function

- For Logistic regression:

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}} \begin{cases} \text{if } y = 1, \text{ we would want } h_{\theta}(x) \approx 1 \Rightarrow \theta^T x \gg 0 \\ \text{if } y = 0, \text{ we would want } h_{\theta}(x) \approx 0 \Rightarrow \theta^T x \ll 0 \end{cases} \quad (1)$$

- In Logistic Regression, for each training example, the cost function is :

$$\begin{aligned} cost &= -y \text{Log}(h_{\theta}(x)) - (1 - y) \text{Log}(1 - h_{\theta}(x)) \\ &= -y \text{Log}\left(\frac{1}{1 + e^{-\theta^T x}}\right) - (1 - y) \text{Log}\left(1 - \frac{1}{1 + e^{-\theta^T x}}\right) \end{aligned} \quad (2)$$

- SVM: to build the SVM, we modify the cost function

$$cost = cost_1(\theta^T x) + cost_0(\theta^T x) \quad (3)$$

0.1.2 SVM Optimization Objective

- For Logistic Regression, the optimization objective is:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} (-\text{Log} h_{\theta}(x^{(i)})) + (1 - y^{(i)}) (-\text{Log}(1 - h_{\theta}(x^{(i)}))) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2 \quad (4)$$

- For Support Vector Machine, the formulation of the optimization objective is:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2 \quad (5)$$

SVM uses different conventions than Linear Regression

- $\frac{1}{m}$ term is eliminated
- λ is replaced by C which now acts on the main term, rather than the regularization term:

$$\min_{\theta} C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2 \quad (6)$$

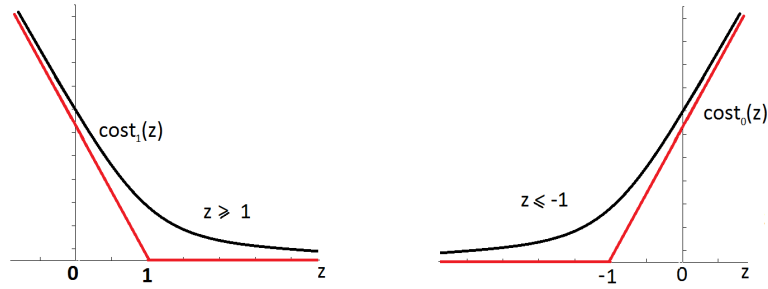
Unlike Linear Regression, SVM does not output a probability but values $\in \{0, 1\}$:

0.2 SVM - Large Margin Classifier

0.2.1 constraint for SVM

- Optimization objective:

$$\min_{\theta} C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{\lambda}{2} \sum_{j=1}^n \theta_j^2 \quad (7)$$



- what does it take to make the cost function small ($\min_{\theta} \text{cost} = 0$)

$$\begin{cases} \text{if } y = 1 \text{ we want } & \text{cost}_1 \theta^T x = 0 \Rightarrow \theta^T x \geq 1 \\ \text{if } y = 0 \text{ we want } & \text{cost}_0 \theta^T x = 0 \Rightarrow \theta^T x \leq -1 \end{cases} \quad (8)$$

Note that in contrast, for Linear Regression we just needed $\theta^T x \geq 0$ if $y = 1$, and $\theta^T x \leq 0$ if $y = 0$

0.2.2 For large C

- SVM Optimization objective

$$\min_{\theta} C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2 \quad (9)$$

In order to minimize the cost for large C, we would want:

$$\left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] = 0 \quad (10)$$

so:

$$\begin{cases} \text{whenever } y = 1 \text{ we want } \text{cost}_1 \theta^T x = 0 \Rightarrow \theta^T x \geq 1 \\ \text{whenever } y = 0 \text{ we want } \text{cost}_0 \theta^T x = 0 \Rightarrow \theta^T x \leq -1 \end{cases} \quad (11)$$

so, the optimization problems becomes:

$$\min_{\theta} C \times 0 + \frac{1}{2} \sum_{i=1}^n \theta_j^2 \quad (12)$$

with the constraint:

$$\begin{cases} \theta^T x^{(i)} \geq 1 \text{ if } y^{(i)} = 1 \\ \theta^T x^{(i)} \leq -1 \text{ if } y^{(i)} = 0 \end{cases} \quad (13)$$

0.2.3 Visualization of the decision boundary for large C

In this linearly separable case, the 2 dashed lines are possible boundary decisions. However, SVM would choose the decision boundary resulting in the largest margin possible. Hence, SVM is also called 'Large Margin Classifier'.

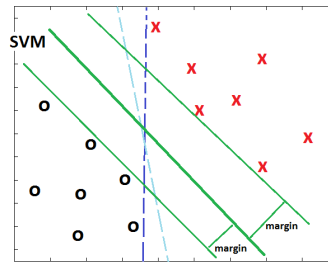


Figure 1: linearly separable data

- Large Margin Classifier in presence of outliers (very large C)

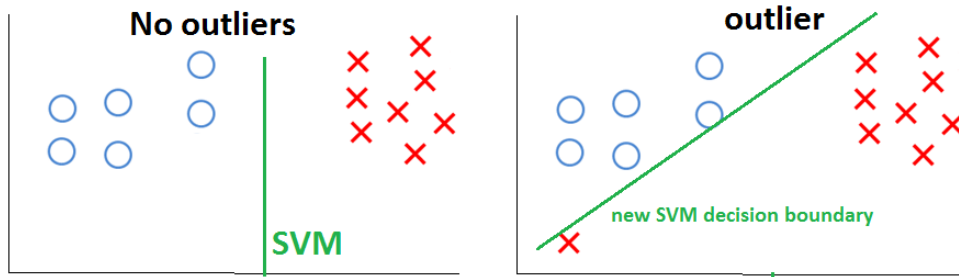


Figure 2: if C is large, SVM will take into account the presence of the outlier. But if C is not too large, there is less overfitting.

0.3 Mathematics behind Large Margin classifier

- Optimization Objective for SVM (large C)

$$\min_{\theta} = \frac{1}{2} \sum_{j=1}^n \theta_j^2 \quad \text{with} \quad \begin{cases} \theta^T x^{(i)} \geq 1 & \text{if } y^{(i)} = 1 \\ \theta^T x^{(i)} \leq -1 & \text{if } y^{(i)} = 0 \end{cases} \quad (14)$$

- Assumptions:

- $\theta_0 = 0$
- only 2 features ($n=2$)

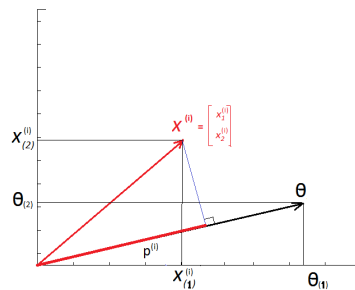


Figure 3

- Optimization objective:

$$\min_{\theta} \frac{1}{2} (\theta_1^2 + \theta_2^2) = \frac{1}{2} (\sqrt{\theta_1^2 + \theta_2^2})^2 = \frac{1}{2} \|\theta\|^2 \quad (15)$$

where $\theta = [\theta_1 \ \theta_2]$

- Vector Inner Product $\theta^T x^{(i)} = ?$

$$\theta^T x^{(i)} = p^{(i)} \|\theta\| = \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} \quad (16)$$

So the **constraint** $\theta^T x^{(i)} \geq 1$ becomes :

$$p^{(i)} \|\theta\| \geq 1 \quad (17)$$

- SVM Decision Boundary: What SVM is more likely to choose for the decision boundary
 - case 1:

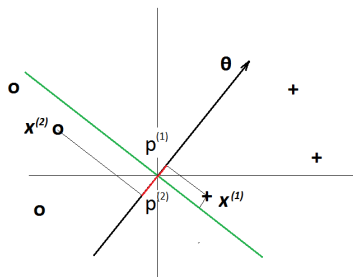


Figure 4

The projection of $x^{(1)}$ and $x^{(2)}$ on θ axis gives: $p^{(1)}$ and $p^{(2)}$. Those 2 length are small. So to satisfy the constraint $p^{(1)}\|\theta\| \geq 1$ ($y=1$), we would need $\|\theta\|$ to be large. Similarly, to satisfy the constraint $p^{(2)}\|\theta\| \leq 1$ ($y=-1$), we would need $\|\theta\|$ to be large also. However this goes opposite to the optimization objective $\min_{\theta}\|\theta\|^2$ which requires $\|\theta\|$ to be small. So this decision boundary is not a good start.

- case 2:

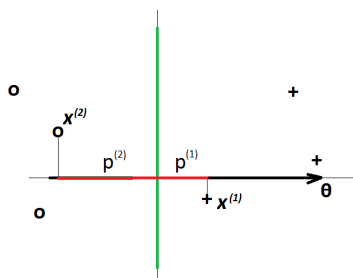


Figure 5

In this case, $p^{(1)}$ and $p^{(2)}$ are large. So, the constraint $p^{(1)}\|\theta\| \geq 1$ requires $\|\theta\|$ to be small, which does go against along a minimization of $\|\theta\|^2$ for the optimization objective.

SVM is trying to maximize $p^{(i)}$.

Note that with the restriction $\theta_0 = 0$, it just means that the decision boundary must pass through 0.

0.4 SVM Kernels I

The Kernels enable to develop complex non linear classifier.

0.4.1 Example

For non-linear decision boundary, a first approach could be the use of polynomial features : $\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 \dots$, where $\theta^T x \geq 1$ predicts $y = 1$.

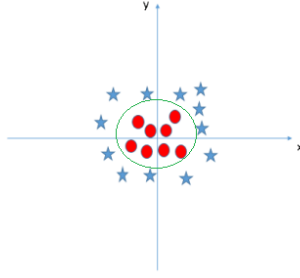


Figure 6

This can also be written:

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 \dots \quad (18)$$

where $f_1 = x_1$, $f_2 = x_2$, $f_3 = x_1 x_2$, etc...

But, is there a better set of features?

0.4.2 landmarks

Given x , compute new feature depending on proximity to landmarks $l^{(1)}$, $l^{(2)}$ and $l^{(3)}$.

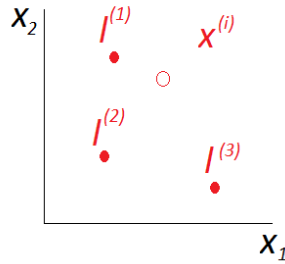


Figure 7

$$\begin{aligned} f_1 &= \text{measure of similarity of } (x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right) \\ f_2 &= \text{measure of similarity of } (x, l^{(2)}) = \exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right) \\ f_3 &= \text{measure of similarity of } (x, l^{(3)}) = \exp\left(-\frac{\|x - l^{(3)}\|^2}{2\sigma^2}\right) \end{aligned} \quad (19)$$

$\exp\left(-\frac{\|x-l\|^2}{2\sigma^2}\right)$ is called the Kernel function (similarity function), and in this particular instance, we have a Gaussian Kernel. The Kernel between x and l is also noted $K(x, l)$

$$f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^n (x_j - l_j^{(1)})^2}{2\sigma^2}\right) \quad (20)$$

If $x \approx l^{(1)}$ (x is close to $l^{(1)} \Rightarrow f_1 \approx \exp(0) \approx 1$

If x is far from $l^{(1)} \Rightarrow f_1 \approx \exp(-(large Nbr)/2\sigma^2) \approx 0$

- Example:

- Hypothesis: Predict 1 when $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$

Let's say the minimization gives: $\theta_0 = -0.5$, $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = 0$

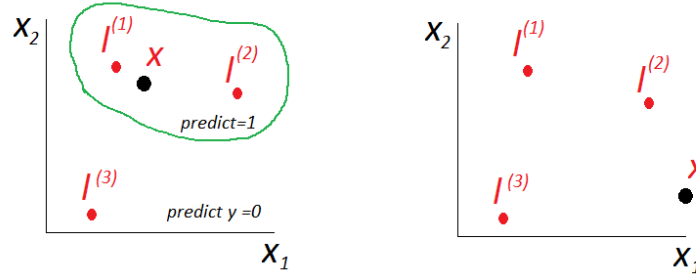


Figure 8

- Let's consider a training example x . Because x is close to $l^{(1)}$ and far from $l^{(2)}$ and $l^{(3)}$, $f_1 \approx 1$, $f_2 \approx 0$ and $f_3 \approx 0$.

$$\theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0 = 0.5 \geq 0 \Rightarrow \text{we predict } y = 1 \quad (21)$$

- Let's consider another training example $x^{(2)}$. Because $x^{(2)}$ is far from all landmarks $f_1 \approx 0$, $f_2 \approx 0$ and $f_3 \approx 0$.

$$\theta_0 + \theta_1 \times 0 + \theta_2 \times 0 + \theta_3 \times 0 = -0.5 \leq 0 \Rightarrow \text{we predict } y = 0 \quad (22)$$

We can also notice that for points close to $l^{(1)}$ and $l^{(2)}$, we end up predicting positive, and for points far away, we end up predicting negative. So the decision boundary would be as shown by the green line.

0.4.3 How to choose the landmarks

In practice the landmarks will be at exactly the same position as the training examples.

- Given a set of training examples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \dots, (x^{(m)}, y^{(m)})$
- Choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$
- Given an example $x^{(i)} \in \mathbb{R}^{n+1}$ (or $\in \mathbb{R}^n$):
 $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$

$$\begin{aligned} f_1 &= \text{similarity}(x^{(i)}, l^{(1)}) \\ f_2 &= \text{similarity}(x^{(i)}, l^{(2)}) \\ &\dots \\ f_i &= \text{similarity}(x^{(i)}, l^{(i)}) = 1 \\ &\dots \\ f_m &= \text{similarity}(x^{(i)}, l^{(m)}) \end{aligned} \quad (23)$$

So the new feature vector $f^{(i)} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_m \end{bmatrix}$ with $f_0 = 1$.

- To get θ , we use the SVM minimization algorithm:

$$\min_{\theta} C \sum_i = 1^m [y^{(i)} \text{cost}_1(\theta^T f^{(i)} + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)})] + \frac{1}{2} \sum_{j=1}^m \theta_j^2 \quad (24)$$

Note that most SVM packages replace $\theta^2 = \theta^T \theta$ by $\theta^T M \theta$, where M is a matrix that depends on the Kernel used.

0.4.4 How to choose C and σ

- How to choose C(= 1/ λ):
 - Large C: lower bias, high variance
 - small C: large bias, low variance
- How to choose σ^2 :
 - Large σ^2 : features f_i vary more smoothly \Rightarrow higher bias and lower variance
 - small σ^2 : features f_i vary less smoothly \Rightarrow higher variance and lower bias

0.5 How to use an SVM

- use SVM packages ('liblinear', 'libsvm'...) to solve for parameters θ
 - Need to specify:
 - Choice of parameter C
 - Choice of kernel (similarity function)
 - * when n large and m small: No kernel ("linear" kernel) (=standard linear classifier)
 - * when n small and m large: use Gaussian Kernel – need to choose σ^2
- Note: do perform feature scaling before using the Gaussian Kernel.
 Not all similarity functions are valid kernels. They must satisfy "Mercer's Theorem," which guarantees that the SVM package's optimizations run correctly and do not diverge. You want to train C and σ using the training and cross-validation datasets.

0.6 Multi-class Classification

- Many SVM libraries have multi-class classification already built-in.
- You can use the one-vs-all method
- Logistic Regression vs. SVMs
 - if n large (relative to m) \Rightarrow use logistic regression, or linear Kernel
 - if n small and m is intermediate \Rightarrow use SVM with a Gaussian Kernel
 - if n small and m large, \Rightarrow manually create/add more features, \Rightarrow use logistic regression or SVM without a kernel. Note: a neural network is likely to work well for any of these situations, but may be slower to train.

Appendices

.1 question

Consider the training set, where 'x' denotes positive examples ($y = 1$) and 'o' denotes negative examples ($y = 0$). Suppose you train an SVM (which will predict 1 when $\theta_0 + \theta_1 x_1 + \theta_2 x_2 \geq 0$). What values might the SVM give for θ_0 , θ_1 and θ_2 ?

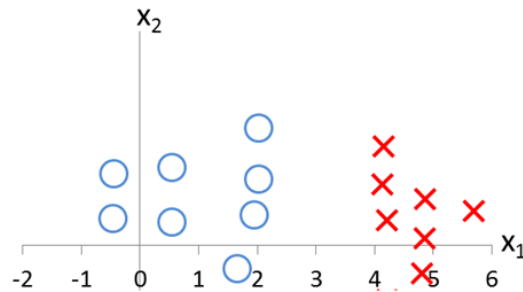


Figure 9

Response: $\theta_0 = -3$, $\theta_1 = 1$ and $\theta_2 = 0$

.2 Vector inner product ($u^T v$)

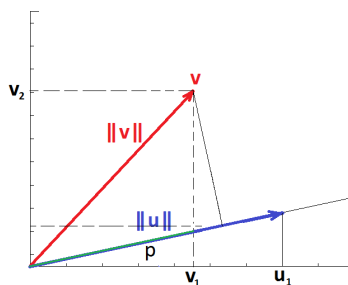


Figure 10: p is the length of projection of v on u (where p is \pm -ive)

Let's consider 2 vectors $u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ and $v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$.
 $\|u\| = \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$.

- p is the length of the projection of v on u , where $p \in \mathbb{R}$ and can be positive or negative

$$u^T v = p \cdot \|u\| = u_1 v_1 + u_2 v_2 \quad (25)$$

.3 SVM Intro

In this algorithm, we plot each data item as a point in n dimensional space (where n is the number of features), with the value of each feature being the value of a particular coordinate. Then we perform classification by finding the hyper-plane that differentiate the 2 classes. Support Vector Machine is a frontier which best segregates the 2 classes (hyper plane line).

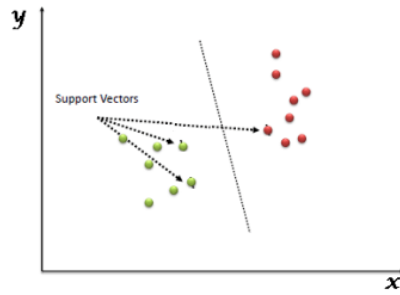
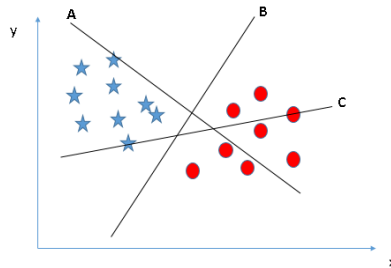


Figure 11: Support vectors are simply the coordinates of individual observation (x, y) .

Let's look at a few examples:

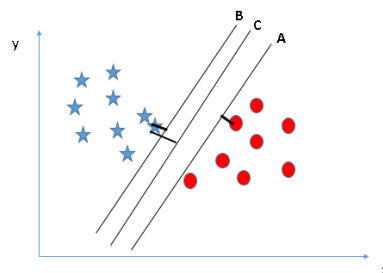
- Scenario 1

Thumb-rule to identify the right hyper-plane: "select the hyper-plane which segregates the 2 classes better". In this scenario, B is the best frontier.



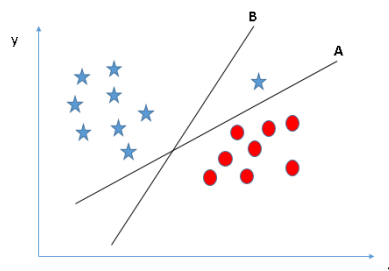
- Scenario 2

Maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called a margin and C has the largest margin.



- Scenario 3

SVM selects the hyper-plane which classifies the classes accurately prior to maximizing the margin. Therefore, the right hyper-plane is A.



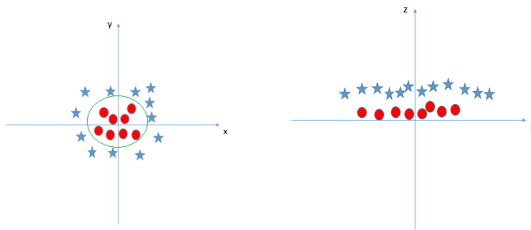
- Scenario 4

One star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, SVM is robust to outliers.



- Scenario 5

A linear hyperplane between the 2 classes cannot be found. In such a case, we need to map those vectors to a higher dimension plane so that they get segregated from each other. Here we will add a new features: $z = x^2 + y^2$ and plot x versus z .



Note that, all values of $z > 0$.

SVM has a technique called the Kernel trick. These are functions which takes takes low dimensional input space and transform it to a higher dimensional space. When we took at the hyper-plane in original input space, it looks like a circle

