Support Vector Macines:

What is a Support Vector Machine?
Support Vector Machines, or Large Margin Classifiers are a Learning Algorithm that take reference from the concept of orthogonality in Linear Algebra, as well as the Sigmoid Function.

At an abstract level, SVMs view data as vectors in $\mathbb{R}^{n+1}$ dimension space, and calculate the relative length of projection of that data to a mapping between {0,1} indicating the prediction of that class or not.

Under the hood:
SVMs minimize the length of projection of a new set of data to that of a previous set of data points, and based on that predicts the classification.

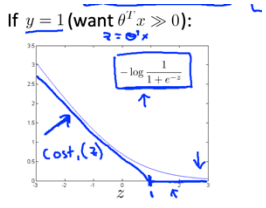
To properly understand SVM and the mechanics behind how it works, a level of understanding of orthogonality is required to understand how projections work with respect to vectors and dot products.

Cost Funtion:
From Logistic Regression, the Logistic Regression cost function is:

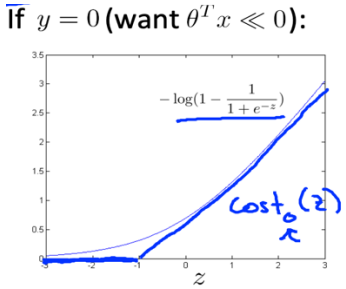
$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log\left(h\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h\left(x^{(i)}\right)\right)]$$

Looking more closely at the terms:
Plotting the graphs of:

$-y^{(i)} \log\left(h(z)\right)$



and $-\left(1 - y^{(i)}\right) \log\left(1 - h(z)\right)$



We can derive $cost_0(z)\ and\ cost_1(z)\ as\ functions$

With Hypothesis:

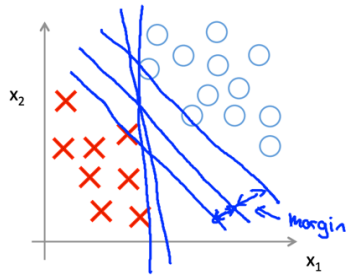
$$h_\theta = \begin{cases} 1\ if\ \theta^T x \geq 0 \\ 0\ otherwise \end{cases}$$

Minimization Objective:

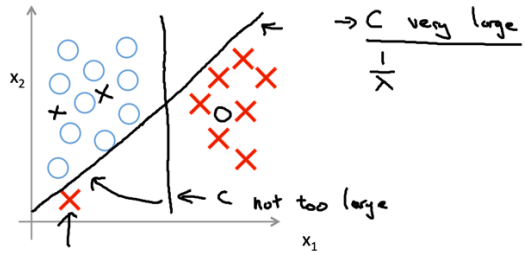
$$\min \; C\sum_{i=1}^{m}\left[y^{(i)}cost_1\left(\theta^T x^{(i)}\right)+\left(1-y^{(i)}\right)cost_0\left(\theta^T x^{(i)}\right)\right]+\frac{1}{2}\sum_{j=1}^{n}{\theta_j}^2$$

$$C=\frac{1}{\lambda}$$

When data is Linearly Seperable/ No Kernel:



In presence of outliers?

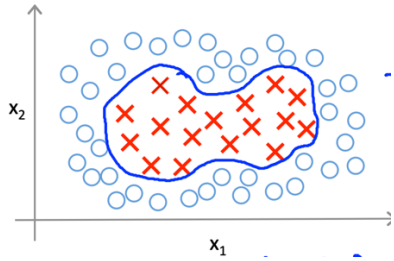


C is used to control the sensitivity of the decision boundary to outliers in the dataset.

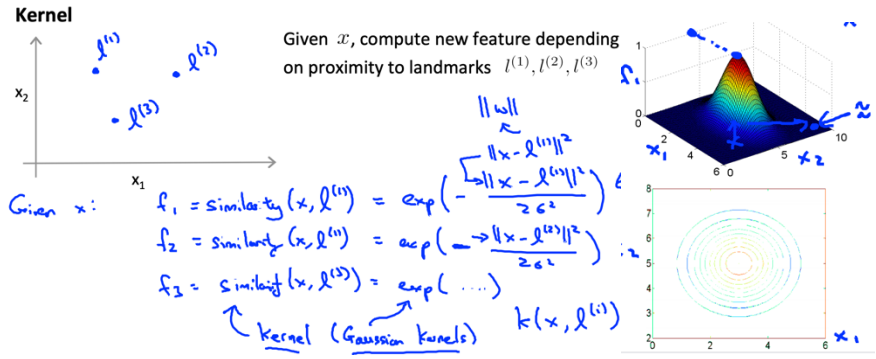
Now:

What makes SVMs really distinguishable from other Learning Algorithms is how it performs on data with non linear decision boundaries. Furthermore, each input x is considered as a vector $x \in \mathbb{R}^{n+1}$ where n = num features and +1 including the bias vector.

Non-linear Decision Boundary



Through the use of the kernel function, each single training data point is considered as a certain landmark, at which there is an abstract contour to which the new datapoint x is compared against, and through some threshold, is evaluated and predicted as a 1 or a 0.



Kernels can be understood to be the function that decide the decision boundary
The most widely used kernel is the Gaussian Kernel:

Given:

$$(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$$

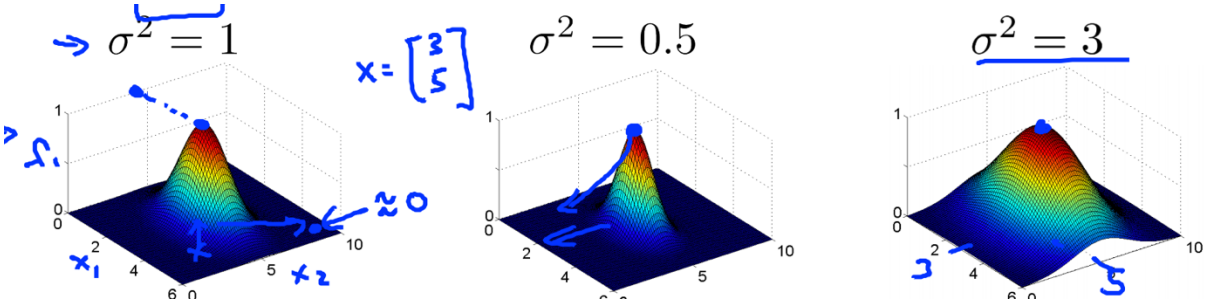
$$Choose:$$

$$l^{(1)} = x^{(1)}, \ldots, l^{(m)} = x^{(m)}$$

Let x = new data to be classified

$$f_1 = similarity(x, l^{(1)})$$
$$f_2 = similarity(x, l^{(2)})$$
$$\vdots$$
$$similarity(x, l^{(j)}) = \ker(x, l^{(j)})$$
$$\ker(x, l^{(j)}) = e^{-\frac{\|x - l^{(j)}\|^2}{2\sigma^2}}$$
$$\sigma \text{ is an adjustable parameter}$$

The effect of σ:



Large σ increases the sensitivity of the kernel
Smaller σincreases the precision of the kernel

Large C/ Small σ => Lower Bias, Higher Variance
Small C/ Large σ=> Higher Bias, Lower Variance

Note: Other Kernels are applicable for use as long as they pass the Mercer's Theorem
On the application of SVMs:

Feature Scaling IS CRITICAL when using SVMs

Multiclassification problems can be resolved by using a 'ovr' mapping similar to that of Logistic regression, or the implementation library will usually have such functionality already included.

Logistic Regression vs SVMs:
SVMs are relatively computationally expensive to use compared to other methods and thus do not scale as well as other learning algorithms to large datasets.
n = num features ($x \in \mathbb{R}^{n+1}, m = number\ of\ training\ egs$
if n is large relative to m, eg n = 10000 m = 10 – 1000

- Use Logistic Regression or SVM with no kernel(ie linear kernel)

If n is small, m is intermediate, eg n = 1-1000, m = 10-10000

- Use SVM with gaussian kernel.

If n is small, m is large: n = 1-1000, m = 50000+

- Create/add more features , then use logistic regression or SVM w/o kernel

Note: Neural Networks are also likely to work well for most of these settings but may be much slower to train/fine-tune.

Overview:
SVMs are one of the most effective and quick to implement learning algorithms currently, and in the use cases where they are applicable, are some of the most effective non deep learning algorithms to use.