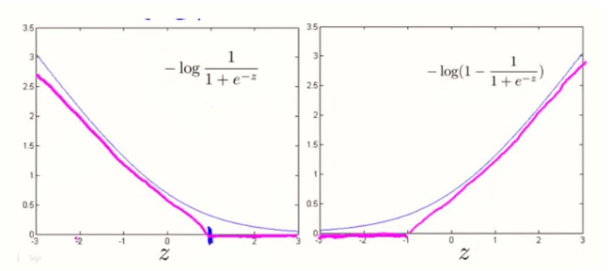
Large Margin Classification (SVM)

Large Margin Classification

One of the widely used algorithm in supervised learning is support vector machine (SVM). SVM gives cleaner way of non-linear learning in compare to neural network and logistic regression. By few modification of logistic regression, we can obtain the hypothesis for SVM. The blue line is showing the overall cost function for logit in image below and the purple line is showing the cost function for SVM.



Instead of a curved line in logit we create two straight lines (Flat portion that's equal to 0 and straight angled line) which acts as an approximation to the logistic regression function. It gives much better optimization and computational cost. We show the left figure as $cost_0(x)$ indicating y=0 and right plot $cost_1(x)$, indicating y=1. The cost function for SVM is as follow:

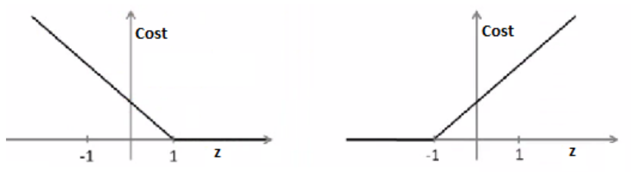
$$\min_\theta \frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{\lambda}{2m}\sum_{i=1}^{n}\theta_j^2$$

By convention, we change the formula to slightly different ones as follow where C=1/λ:

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

Sometimes SVM is referred as large margin classifier. You'll understand the reason behind naming in next few paragraphs. Cost function for SVM will be zero:

- If y=1 only when z≥ 1
- If y=0 only when z<-1

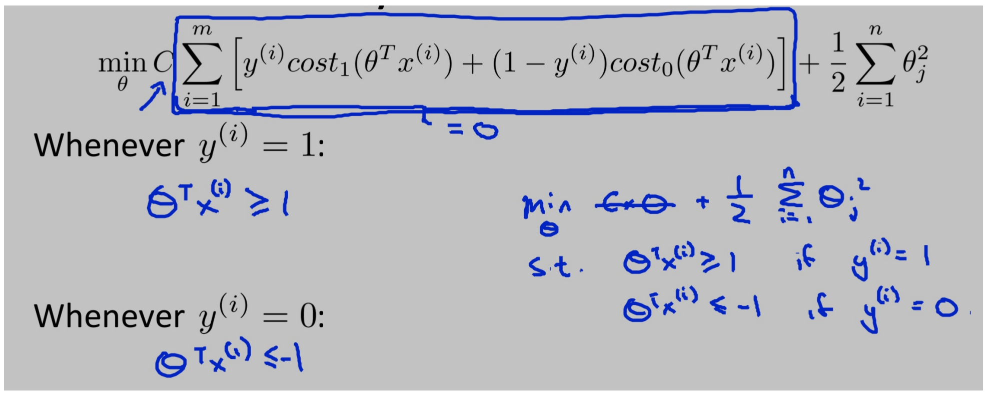


If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)
If $y = 0$, we want $\theta^T x \leq -1$ (not just < 0)

The previous models, for previous model we only need z to be greater or equal to 0, but SVM doesn't just want to get it right but also the value should be quite bigger than 0. This is an extra safety margin factor.

If C is very large value, we are motivated to the make the first summation equal to 0 in optimization problem. In equation below we can see how it happens in different situation. Finally, we end up with this term $\frac{1}{2}\sum_{i=1}^{n}\theta_j^2$. This term gives us a boundary decision for SVM.

Large Margin Classification (SVM)



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

$= 0$

Whenever $y^{(i)} = 1$:

$\theta^T x^{(i)} \geq 1$

$$\min_{\theta} \cancel{C \times 0} + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

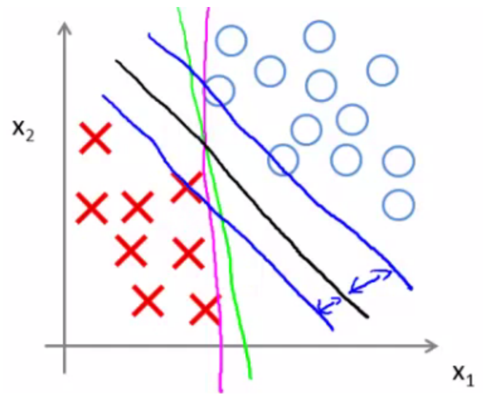
s.t. $\theta^T x^{(i)} \geq 1$ if $y^{(i)} = 1$

$\theta^T x^{(i)} \leq -1$ if $y^{(i)} = 0$

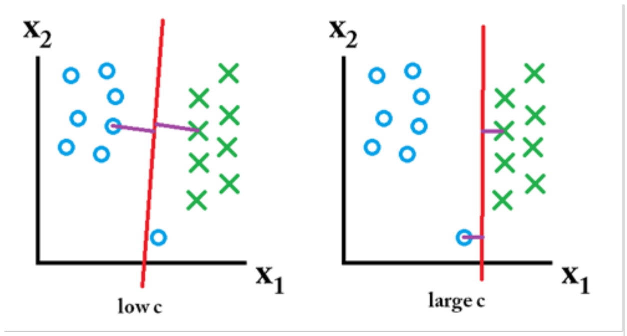
Whenever $y^{(i)} = 0$:

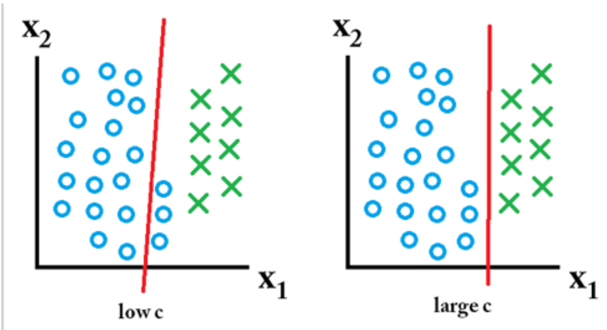
$\theta^T x^{(i)} \leq -1$

It turns out solving this equation give us boundary decision for SVM as shown in image below. The green and magenta might be drawn when using logistic regression. The black line is obtained through the SVM modeling which gives us more robust separator. The black line gives maximum margin in compare to the blue lines.

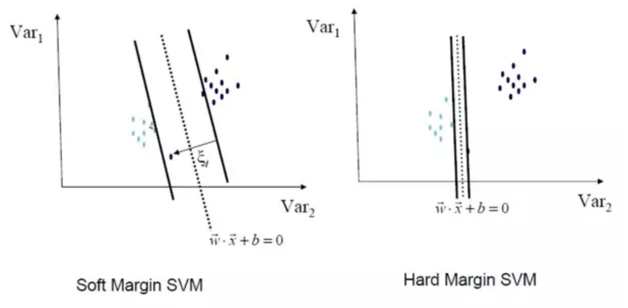


<u>The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example</u>. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable. Here are some examples shown in graph below.





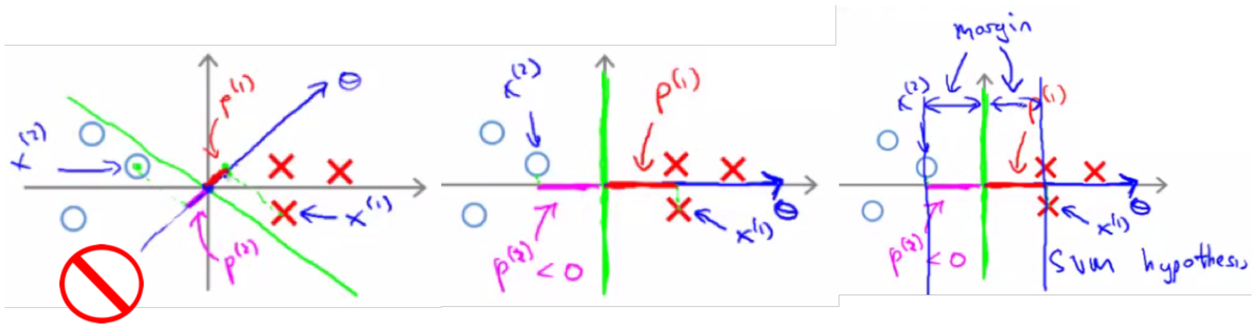
C is essentially a regularization parameter, which controls the trade-off between achieving a low error on the training data and minimizing the norm of the weights. Tuning C correctly is a vital step in best practice in the use of SVM. Soft margin and hard margin SVM are referred to a small and large value of C respectively.



Large Margin Classification (SVM)

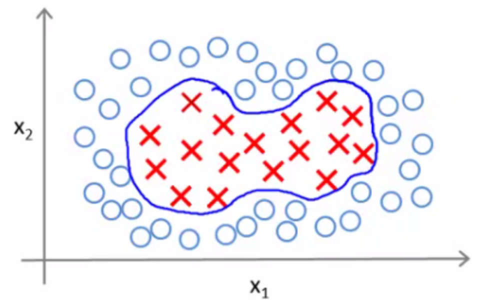
Despite the popularity of SVM, it has a serious drawback which is sensitivity to outliers in training samples. If we have few outliers, we still can use SVM. In case of large amount of outlier, we may seek using other algorithms.

Here is how SVM chooses the boundary. In specific example below, SVM tries to find the biggest p and smallest theta. So, by maximizing these p values we minimize ||θ||.



Kernels (Adapting SVM to nonlinear classifier)

Sometimes we have more complex model and cannot fit a linear line between features using SVM we described above. In this case we need to make some adjustments to use SVM for polynomial features and define a polynomial as follows.

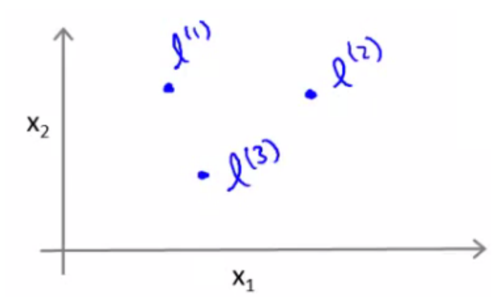


Predict $y = 1$ if

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots \geq 0$$

$$h_\theta(x) = \begin{cases} 1 & \text{if } \theta_0 + \theta_1 x_1 + \cdots \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Another way is to define $f_1 = x_1, f_2 = x_1 x_2$and so on. Is there any better choice of features than what we learned before? Imagine x is given and we try to compute new features depends on proximity to landmarks l^1, l^2, l^3 as follow:



$$f_1 = similarity(x, l^1) = \exp\left(-\frac{\left\|x - l^1\right\|^2}{2\sigma^2}\right)$$

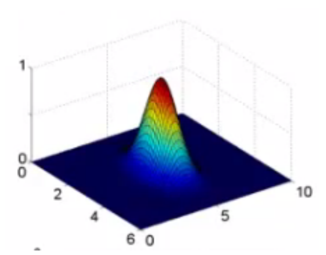
$$f_2 = similarity(x, l^2) = \exp\left(-\frac{\left\|x - l^2\right\|^2}{2\sigma^2}\right)$$

Large Margin Classification (SVM)

The similarity functions are kernels. Kernels define new features of f and acts as below:

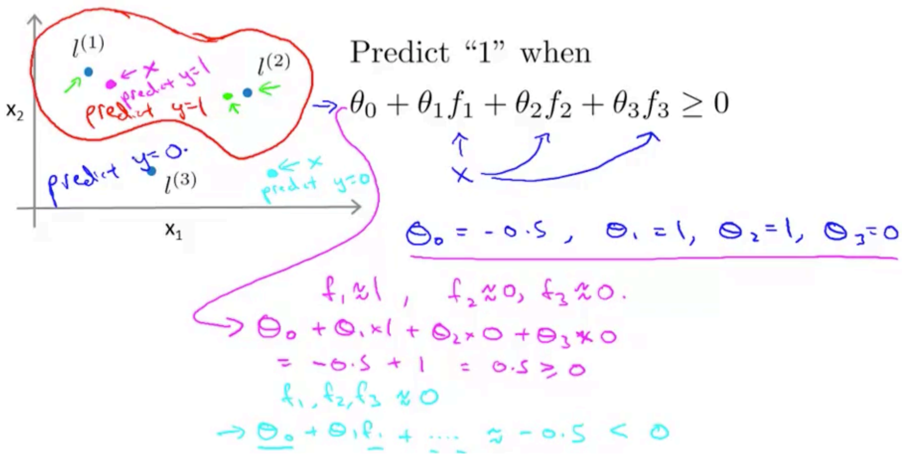
- When x=l^1: Euclidean distance will be zero, and the kernel will be 1.
- When x is far from l^1: Euclidian distance will be large, therefore the kernel will be 0.

The f function is shown in graph on the right. The y axis is f and x axis is the training examples ($x_1 x_2$, ...). This graph shows how close the value of training example is to the landmarks. So, we predict 1 when:



$$\boxed{\theta_0 = \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0}$$

In example below, we got values of theta from training set. If we have a new data shown in magenta color, we get 1 for the f closest to the point and the reset of f will be zero.



$l^{(1)}$
$l^{(2)}$
$l^{(3)}$
x_2
x_1
predict y=1
predict y=1
predict y=0.
predict y=0
Predict "1" when
$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$
$\theta_0 = -0.5, \ \theta_1 = 1, \ \theta_2 = 1, \ \theta_3 = 0$
$f_1 \approx 1, \ f_2 \approx 0, \ f_3 \approx 0.$
$\theta_0 + \theta_1 \times 1 + \theta_2 \times 0 + \theta_3 \times 0$
$= -0.5 + 1 = 0.5 \geq 0$
$f_1, f_2, f_3 \approx 0$
$\theta_0 + \theta_1 f_1 + \dots \approx -0.5 < 0$

Finally, we end up with y=1 inside of the boundary and y=0 outside of the boundary in this specific example.

Landmarks in Kernels

Given machine problem, we have some negative and some positive examples. We choose the landmarks exactly same location as training examples. We end up with m (training size) landmarks. In summary, here's how we apply kernels in SVM:

- Given a training set
- Choose landmark exactly similar to the training set
- Use kernels to find similarity between dataset
- Predict 1 if $\theta^T f \geq 0$
- Minimize the cost function

In SVM if:

- C is large: low bias, high variance
- C is small: high bias, low variance
- σ^2 is large: smoother function, higher bias and low variance.

Large Margin Classification (SVM)

SVM in practice

We can use linear kernel (no kernel) when n is large and m is small. We us Gaussian kernels when n is small and m (training set) is large. Remember to perform feature scaling before using kernels. You may use any other kernels that satisfy the Mercer's theorem such as:

- Polynomial kernels: $(x^T l + C)^{degree}$: it's only used when all examples are non negative.
- String kernels
- Chi-square kernels
- Histogram kernels

For multiclass classification, you may use the function already built in Python or use one-vs-all method.

Logistic regression vs SVM use case

- Logistic Regression or SVM with no kernel: If n is large in compare to m (like n=10,000, m= 1000)
- SVM with Gaussian kernel: when n is small and m is intermediate (n=1000, m=10,000)
- Logistic Regression or SVM with no kernel: If n is small and m is large (like n=1000, m= 50,000+)
- Neural network likely works in all cases above, but may be slower to train