Support Vector Machine (SVM)

SVM support vector machine. It's similar to logistic regression but with few changes It's said that sometimes cleaner and more powerful than other algorithms (aledgedly).
We first will start off with the hypothesis first.
Hypothesis of the Support Vector Machine is not interpreted as the probability of y being 1 or 0 (as it is for the hypothesis of logistic regression). Instead, it outputs either 1 or 0. (In technical terms, it is a discriminant function.)

1 Hypothesis

In short No more X's, but changed into kernel. Features into kernels. That means that, instead of $h_\theta(X) = g(\theta^T X)$. X's replace with kernel. that's with F is a vector's of f's:

$$h_\theta(X) = \theta^T * F \tag{1}$$

And this $h_\theta(X)$ will be

$$h_\theta(X) = \begin{cases} 1, & \text{for } \theta^T F \geq 0 \\ 0, & \text{for 0 otherwise} \end{cases}$$

2 Kernel:Linear, gaussian kernel

What is kernel ? Kernel is a way of selecting features for classfication problems, that's anotherway of trying to fit a function around a data sets. Instead of polynomials, it's using kernels to define the similarity between x and a point that they call landmark. And in practice, people have like alot of others kerels to use.
Note: 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.
Kernel function/similarity functions denoted:

$$k(x_1, x_2)$$

The most widely used kernels are linear(Not using any kernels which is $h_\theta(X) = \theta^T X$ and gaussian kernel.
The **gaussian kernel** is define:
X: data with m observations
L: land marks

$$\text{Guassian kernel } = k(X, L) = exp(-\frac{(X-L)^2}{2\sigma^2}) \tag{2}$$

We then define now, our features and F vectors of f's to be:

$$f_1 = k(x, l^{(1)}), f_2 = k(x, l^{(2)})..., f_m = k(x, l^{(m)})$$

Or this can also be written as:

$$f_i = similarity(x, l^{(i)}) = exp(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(i)})^2}{2\sigma^2})$$

This will have our hypothesis to look like this with m observations from X:

$$h_\theta(X) = \theta_0 * f_0 + \theta_1 * f_1 + ... + \theta_m * f_m = \theta^T * F$$

Couple of properties of these is if:

$$x \sim l^{(i)}$$

then

$$f_i = exp(-\frac{\sim 0^2}{2\sigma^2}) \sim 1$$

And if

$$x \neq l^{(i)}$$

then

$$f_i = exp(-\frac{\text{Large fking number}}{2\sigma^2}) \sim 0$$

To get predictions, plugs in X to calculates F then do a vector inner product of $h_\theta(X) = \theta^T F$. If it's larger than 0, it's 1, less than 0.It's 0

*I can hear you screaming. But how the fuck does changing from polynomials into a kernel "features" Will give out a hypothesis that will sensbily fits our function ?
(This shit I dont' know, get on this).*

Choosing landmarks:

One way to get the landmarks is to put them in the exact same locations as all the training examples. This gives us m landmarks, with one landmark per training example.

By calculating the differences some will equal to 1 some will equal to zeros. Then from that you can draw a boundary lines around those (I think, I'm not so sure here).

sigma in gaussian kernel:

σ^2 is a parameter of the Gaussian Kernel, and it can be modified to increase or decrease the drop-off of our feature f_i. Combined with looking at the values inside θ, we can choose these landmarks to get the general shape of the decision boundary.

Other algorightm application Using kernels to generate f_i's is not exclusive to SVMs and may also be applied to logistic regression. However, because of computational optimizations on SVMs, kernels combined with SVMs is much faster than with other algorithms, so kernels are almost always found combined only with SVMs.

3 Cost function

Similar to logistic regresision this now will have the cost when y = 0 , and y = 1 changed to this, this is also called a Hinge loss function:

With $z = \theta^T F$

When y = 1

$$\text{cost}_1 = max(0, \text{k}(1 - z)) \tag{3}$$

When y = 0

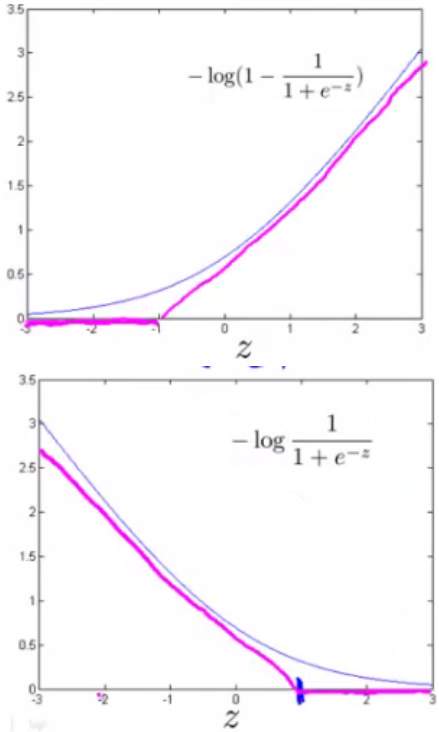
$$\text{cost}_0 = max(0, \text{k}(1 + z)) \tag{4}$$

In general we can combine it and we will have J

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

$$= C[y^T \text{cost}_1(z) + (1 - y)^T \text{cost}_0(z)] + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2 \tag{5}$$

graphs on cost_0 and cost_1 in magenta line, blue line is cost of logistic regression



As you can see, if y = 1 and $\theta^T F$ is larger than 1, the cost is 0. And, if y = 0 and $\theta^T F$ is less than -1, the cost is 0. Otherwise it's additioning the line with k as an abitrary number.

This is also the reason why the SVM is sometimes called the large margin classifer. Lies in the vector inner product, to understand further on why it's called a large margin classifer. Watch Mr andrew Ng video again on week 7 coursera on why it's called a largemargin classifer.

4 Implementation

Now, Professor Andrew Ng doesn't mention what we are using to optimized this, but instead he points to an off the shelf library. That people have already optimized the shit out of. You just need to use, Don't worry about the details. But when using the library he suggests a some listed here:
`https://www.csie.ntu.edu.tw/~cjlin/liblinear/`
`https://www.csie.ntu.edu.tw/~cjlin/libsvm/`
To evaluates which C or sigma, he suggests we check out the lowest error in cross validation set.
A few C's and sigma's to try out. Test each C with each sigma for best effect Below are the octave code

```
C_vec = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30];
sigma_vec = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30];
```

Error cross validation is

```
error_cv = mean(double(predictions ~= y));
```

Bias and variacne controlling with C and sigma
C

- C large: Lower bias, high variacne
- C small: High bias, low variacne.

sigma

- sigma large: Feature f_i, vary more smoothly, Higher bias, lower variacne.
- sigma small: Feature f_i, vary less smoothly, Lower bias, higher variacne.

5 Using SVM library

When using SVM library here are the list of things that you need to provide to the algorithm :

- Choice of kernel (Linear kernel y = 1 if $\theta^T x \geq 0$) or gaussian (And be sure it's satisfied Mercer's theorm)
- Choice of C

If you choose gaussian kernel, prepare gaussian kernel as gaussian(x_1, x_2) then

- Choice of σ
- Perform feature scaling (Read on this if needed) Really needed, if not shit will not work as advertised

Professor said that he doesn't used alot of other kernels but here they are: polynomial kernel, string kernel, chi-square kernel, histogram kernel, intersection kernel,...

6 Multiclassification

If your library doesn't have multiclassification purpose, just 1vsall. let y = to each group. If there are 3 group, then to get θ for group 1 let y = 1 if y = group 1 and let y = 0 if y $\neq$ group 1. Do that for for all 3 groups, then we should have 3 θ's for 3 group.

7 SVM or Logistic regression ? Cons, pros

Okay, so when should we use logistic regression ? and when should we use SVM ?
n = numbers of features ($x \in R^{n+1}$), m = numbers of observations
Case #1: n is large (relative to m): $n \geq m$, n = 10,000 , m = 10,...,1000)
Use: Logistic regression, or SVM without a kernel("linear kernel")
Case#2: n is small, m is intermediate (n = 1...1000, m = 10,...,10,000)
Use: SVM with gaussian kernel
Case #3: n is small, m is large
Use: Create/add more features, then use logistic regression or SVM without a kernel
Neural network will likely to work for all of these, but takes more time to train.

8 Additional reference

https://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf