Optimization Objective

Support Vector Machine (SVM) is yet another type of supervised machine learning algorithm. It is sometimes cleaner and more powerful.

Now, we'll modify some term in our default cost function for logistic regression.

Instead of using sigmoid function we should use "HINGE LOSS" Function. This is more realistic.

Now our modified cost function:

$$J(\theta) = C \sum_{i=1}^{m} y^{(i)} \, cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) \, cost_0(\theta^T x^{(i)}) + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

$$\left[C = \frac{1}{\lambda}\right]$$

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

Large Margine

If $y=1$, we want $\theta^T x \geq 1$

$y=0$, we want $\theta^T x \leq -1$

If $C >> 100{,}000$

$J(\theta) = \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$ → large margine is achieved.

But if we have outlier examples that we don't want to affect the decision boundary, then we can reduce C.

Data is linearly separable when a straight line can separate the positive and negative example.

Large margine Classification:

$u^T v = p \cdot \|u\| = u_1 v_1 + u_2 v_2$

↳ projection of v onto u.

$$J(\theta) = \min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 = \frac{1}{2} \|\theta\|^2$$

$$\theta^T(x^i) = p^{(i)} \cdot \|\theta\|$$

The reason behind 'large margin' is because the vector for θ is perpendicular to the decision boundary. In order for our optimization objective to hold true, we need the absolute value of our projections $p^{(i)}$ to be as large as possible.

Kernels allow us to make complex, non-linear classifiers using support vector machines.

Now, for x. we need to compute new feature depending on proximity to landmarks $l^{(1)}, l^{(2)}, l^{(3)}$

$$f_i = \text{similiarity}(x, l^{(i)}) = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$$

↳ Gaussian Kernel.

Properties of Kernel:

i) if $x \approx l^{(i)}$ then $f_i \approx 1$

ii) if x far from $l^{(i)}$ then $f_i \approx 0$

Each landmark gives us the features in our hypothesis.

$$l^{(i)} \rightarrow f_i$$

$$h_\theta(x) = \sum_{i=1}^{m} \theta_i f_i$$

Now, one way to get landmarks is to put them in the exact same locations as all the training examples.

For example, for x

$f_1^{(i)} = \text{similiarity}(x, l^{(1)})$ ... $f_m^{(i)} = \text{similiarity}(x, l^{(m)})$

$f_{(i)} \rightarrow$ feature vector of x_i

Then we substitute f_i with x_i in our cost function.

Now, Kernel is always used with SVM because of computational optimization.

Properties of SVM parameters

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

$C \rightarrow$ large $\rightarrow$ high variance, low bias

$C \rightarrow$ small $\rightarrow$ low variance, high bias

$\sigma^2 \rightarrow$ large $\rightarrow f_i$ vary more smoothly $\rightarrow$ high bias, low variance

$\sigma^2 \rightarrow$ small $\rightarrow f_i$ vary less smoothly $\rightarrow$ low bias, high variance.

Using An SVM

SVM Library → liblinear
libsvm

In practicle Application

→ Choice of parameter C
→ Choice of Kernel (similiarity function)

→ No Kernel ("linear Kernel") → n large m small
→ Gaussian Kernel (need to choose σ) → n small m medium
↳ feature scaling necessary

n small m large → add more feature, use logistic regression.

To make a similiarity function to a valid Kernal, it must be satisfy "Mercer's Theorem"

" A NEURAL NETWORK IS LIKELY TO WORK WELL FOR ANY OF THESE SITUATIONS, BUT MAY BE SLOWER TO TRAIN"