1 Support vector machine (SVM)

1.1 Cost function

Recall the cost function of logistic regression:

$$\begin{aligned} J(\theta) =& \frac{1}{m}\sum_{i=1}^{m}\left(-y^{(i)}\log\frac{1}{1+e^{-\theta^{\mathsf{T}}x^{(i)}}} - (1-y^{(i)})\log\left(1-\frac{1}{1+e^{-\theta^{\mathsf{T}}x^{(i)}}}\right)\right) \\ &+ \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \end{aligned} \tag{1}$$

For a training example (x, y) with $y = 1$, we expect $\theta^{\mathsf{T}}x \gg 0$ in order to minimize the cost function. Similarly, we expect $\theta^{\mathsf{T}}x \ll 0$ for an example with $y = 0$. The graph of $f_1(z) = -\log\frac{1}{1+e^{-z}}$ is provided with the blue line in Figure 1, while the graph of $f_0(z) = -\log\left(1-\frac{1}{1+e^{-z}}\right)$ is provided with the blue line in Figure 2.

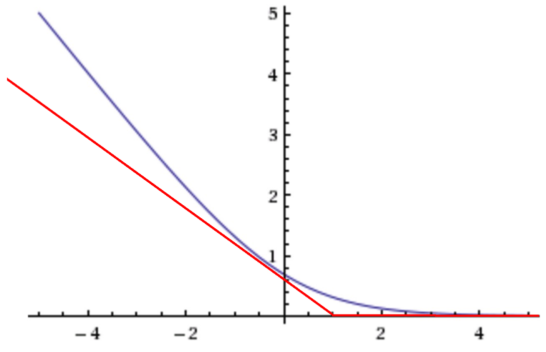

Figure 1: Image of cost1(z)

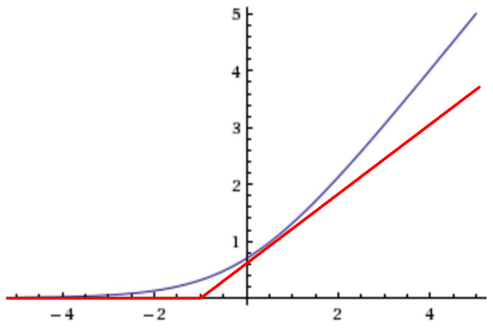

Figure 2: Image of cost0(z)

In SVM, we will use new functions $cost1(z), cost0(z)$ as depicted by the red lines in Figure 1 and Figure 2 to substitute $f_1(z)$ and $f_0(z)$. The new cost function is

$$J(\theta) = C\sum_{i=1}^{m}\Big(y^{(i)}cost1(\theta^{\mathsf{T}}x^{(i)}) + (1-y^{(i)})cost0(\theta^{\mathsf{T}}x^{(i)})\Big) + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 \tag{2}$$

Note that for the reason of convention, m is dropped and rather than writing the function as $A+\lambda B$, we are now writing it as $CA+B$. C has the same effect as the original $\frac{1}{\lambda}$ when it comes to its effect on regularization.

Unlike logistic regression, in which $h_\theta(x)$ is interpreted as the probablity of $y=1$, SVM has the following hypothesis:

$$h_\theta(x) = \begin{cases} 1, \text{ if } \theta^{\mathsf{T}}x \geq 0 \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

1.2 Large margin classification

It is easy to tell from the graphs of $cost1$ and $cost0$ that for a training example with $y=1$, in order to minimize the cost function, we expect $\theta^{\mathsf{T}}x \geq 1$(not just ≥ 0). If $y=0$, we expect $\theta^{\mathsf{T}}x \leq -1$(not just < 0). This builds an extra "safety margin factor" for the SVM. Geometrically, in a linearly separable case, it is ralated to choosing the decision boundary that maximizes its distance from the examples, i.e. conducts the "large margin classification". Figure 3 demonstrates a simple case. Here, large margin classification results in the green line as the decision boundary, rather than the black one or the orange one.

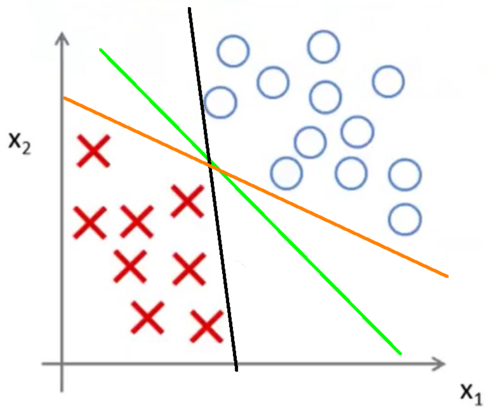

Figure 3: Intuition of large margin classification

When there exist outliners, the regularization factor C ensures that the algorithm does not overfit the examples. Obviously C cannot be too large.

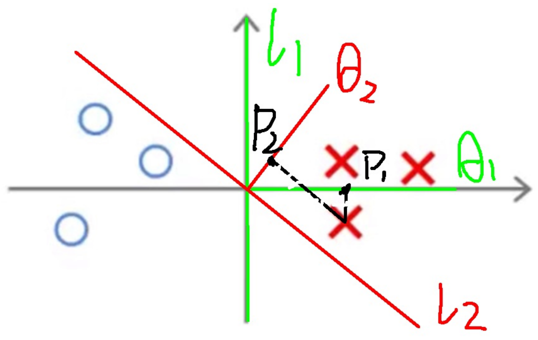

Figure 4: Mathematical background of large margin classification

The mathematical background of large margin classification can be illustrated by Figure 4. In this simple 2-d case, our target becomes

$$\begin{aligned} &\min_\theta \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \min_\theta \frac{1}{2}\|\theta\|^2 \\ &\text{s.t. } \begin{cases} \theta^\intercal x^{(i)} \geq 1, \text{ if } y^{(i)} = 1 \\ \theta^\intercal x^{(i)} \leq -1, \text{ if } y^{(i)} = 0 \end{cases} \end{aligned} \tag{4}$$

Note that

$$\theta^\intercal x = \|\theta\|\|x\|\cos\langle\theta, x\rangle = p \cdot \|\theta\|$$

in which p is the projection of x along the direction of θ. In order to minimize $\|\theta\|$, p should be as large as possible for all samples. From Figure 4, obviously l_1 is a better decision boundary because $p_1 > p_2$.

1.3 Kernals

In order to adapt SVMs to develop complex non-linear classifiers, we have to use **kernals**.

One way to develop non-linear classifiers is to use high degree polynomial features. We will end up with a classifier that predicts $y = 1$ if

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \cdots > 0 \tag{5}$$

(5) can also be written as

$$\theta^\intercal f > 0$$

in which

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ \cdots \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1 x_2 \\ \cdots \end{bmatrix}$$

In complex cases, there could be a lot of polynomial features, causing significant computational difficulty. Our target is to find a better choice of features $f_1, f_2, f_3 \dots$

Kernal introduces the idea to compute features of an example x according to its proximity to a series of landmarks $l^{(i)}$, e.g.

$$f_i = \text{similarity}\left(x, l^{(i)}\right) = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right) \tag{6}$$

Here we are using **Gaussian Kernal**. When x is close to $l^{(i)}$, this kernal returns approximately 1, while when x is far from $l^{(i)}$, it returns approximately 0.

With featurs f_i defined as such, we now have SVMs that can conduct non-linear classification. The SVM predicts $y = 1$ when $\theta^\intercal f \geq 0$, and $y = 0$ otherwise.

As for the choice of landmarks $l^{(i)}$, in practice, we use all examples in the training set as landmarks. Thus we will end up with m features. θ can be trained with

$$\min_\theta C \sum_{i=1}^m \left(y^{(i)} cost1(\theta^\intercal f^{(i)}) + (1 - y^{(i)}) cost0(\theta^\intercal f^{(i)})\right) + \frac{1}{2}\sum_{i=1}^m \theta_i^2 \tag{7}$$

Since we have exactly m features, now we have $n = m$.

As mentioned above, C controls the regularization in the same way as $\frac{1}{\lambda}$ before. Thus with large C, the SVM tends to have high variance and low bias, whereas with small C, it tends to have high bias and low variance. The σ in the definition of Gaussion kernal also affects the bias and variance of the SVM. With large σ, features vary more smoothly, and the SVM tends to have high bias and low variance, while with small σ, features vary more rapidly, and it tends to have high variance and low bias.

When using Gaussian kernal, it is important to do feature scaling. Otherwise $\|x - l\|^2$ will be dominated by the component with largest magnitude.

1.4 Logistic regression v.s. SVM

Some guidelines about the choice between logistic regression and SVM:

- When n is large and m is small: use logistic regression or SVM without a kernal (linear kernal).
- When n is small and m is large: add/create more features and then use logistic regression or SVM without a kernal.
- When n is small and m is intermediate: use SVM with Gaussian kernal.