Support Vector Machines

(Introduction, Linear Model)

CSED, TIET

Logistic Regression- Recap

Hypothesis function is:

$$y^{\wedge} = f(x) = \frac{1}{1 + e^{-z}}$$

and $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \ldots\ldots\ldots\ldots + \beta_k x_k$

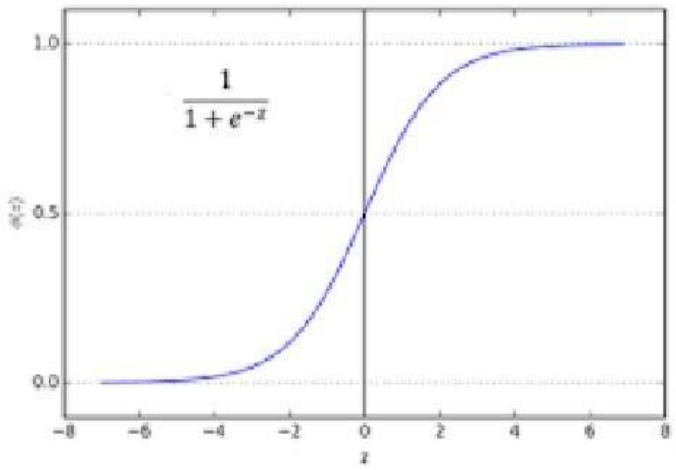
Or in matrix form , $z = X\beta$

where $X = \begin{bmatrix} 1 & x_{11} & x_{12} \cdots & x_{1k} \\ 1 & x_{21} & x_{22} \cdots & x_{2k} \\ \vdots & \vdots & \vdots \ \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \cdots & x_{nk} \end{bmatrix}$; $\beta = \begin{bmatrix} \beta_0 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}$

$$\text{Predicted } Label = \begin{cases} 1 \ if \ y^{\wedge} \geq 0.5 \\ 0 \ if \ y^{\wedge} < 0.5 \end{cases}$$

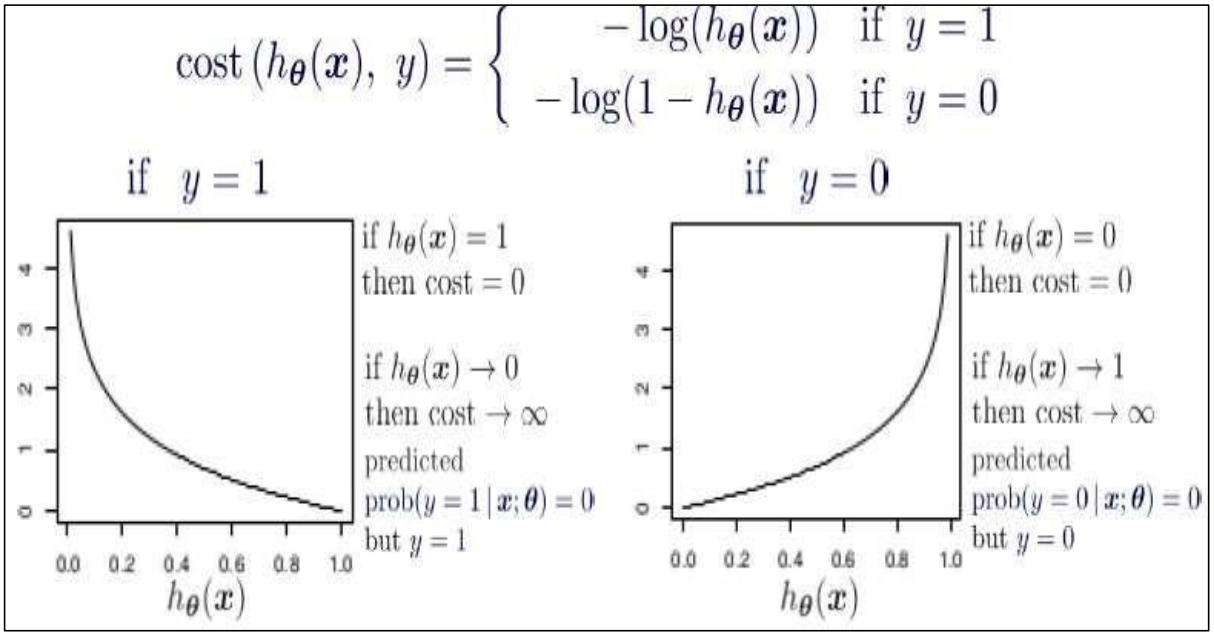
Therefore, The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit.

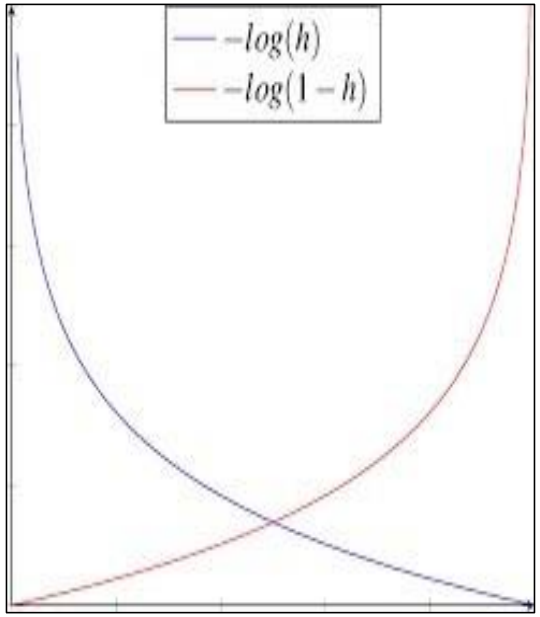
So, it forms a curve like the "S" form.



Logistic Regression Cost Function-Recap

$$\text{cost}\left(h_\theta(x),\ y\right) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





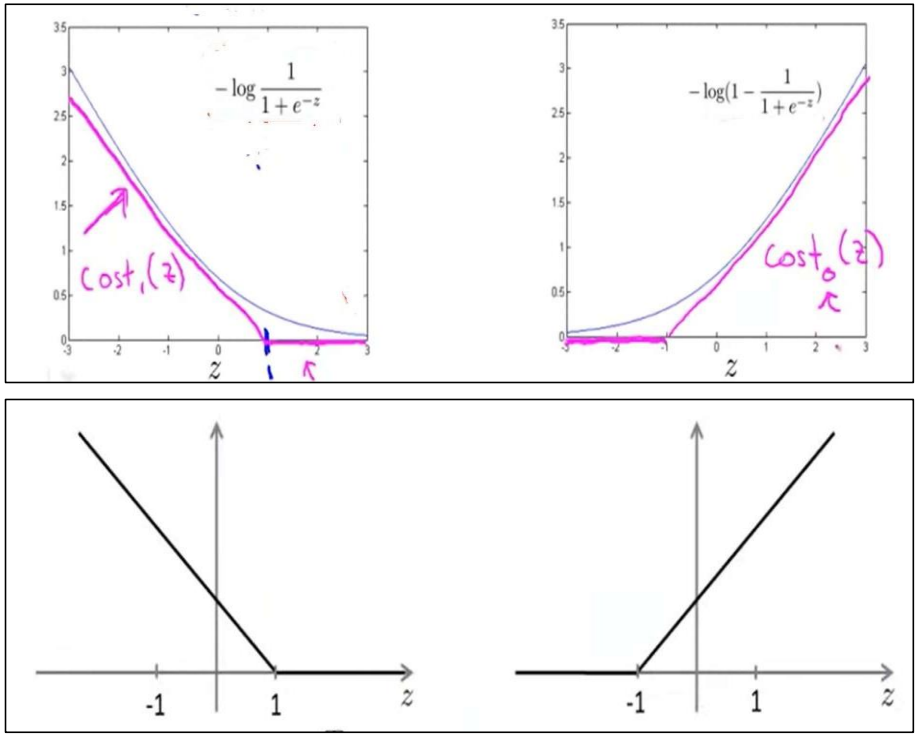
$$J = -\frac{1}{n}\sum_{i=1}^{n} y_i \log\left(f(x_i)\right) + (1 - y_i)\log\left(1 - f(x_i)\right) + \frac{1}{2n}\lambda \sum_{j=0}^{k} \beta_j^2$$

Support Vector Machine

For Support Vector Function, we modify the cost function of Logistic Regression a bit (as shown in figures).

The only difference is that it consist of two straight lines (shown as pink lines in the figure).

- One which is flat and the other which is straight line (quite close to the curved cost of logistic regression).
- Let these cost for y=1 and y=0 be named as $cost_1(z)$ and $cost_0(z)$ respectively.



Support Vector Machine- Cost Function

- Therefore the cost function of support vector machine is modified as:

$$J = \sum_{i=1}^{n} (y_i cost_1(z) + (1 - y_i)\, cost_0(z)) + \frac{1}{2}\lambda \sum_{j=0}^{k} \beta_j^2$$

where $z = f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \ldots \ldots \ldots \ldots + \beta_k x_k = X\beta$

- The expression is not divided by n because the point of minimum for J or J/n would be same.
- The value of λ controls the relative weight between the training set error (cross entropy loss) and the regularization factor.
- So instead of using λ we use a parameter C in SVM such that (it is just different way of representation)

$$J = C \sum_{i=1}^{n} (y_i\, cost_1(z) + (1 - y_i)\, cost_0(z)) + \frac{1}{2} \sum_{j=0}^{k} \beta_j^2$$

SVM- Cost Function (Contd…)

- Now Generally, the value of C is taken as quite high. Then, in order to minimize the cost function, the factor

$$\sum_{i=1}^{n} (y_i cost_1(z) + (1 - y_i)\, cost_0(z)) \; should\ approximate\ to\ zero$$

- Therefore, the cost function of an SVM classifier is given by:

$$J = \frac{1}{2} \sum_{j=0}^{k} \beta_j^2$$

- Another, assumption that the SVM classifier makes is that the positive labels are mapped to 1 and the negative labels are mapped to -1.
- Therefore, y_i=1 if and only if $f(x_i) \geq 1$ and y_i=-1 if and only if $f(x_i) \leq -1$

$$f(x_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots \ldots\ldots\ldots\ldots + \beta_k x_{ik} = X\beta$$

- So, both the constraints mentioned above can be combined in a single equation as $y_i f(xi) \geq 1$

SVM -Cost function Summary

- Thus for a linear separable data with two classes positive (+1) and negative (-1), the objective is to

$$Minimize\ J(\beta) = \frac{1}{2}\sum_{j=0}^{k} \beta_j^2$$

$$subject\ to\ y_i f(x_i) \geq 1\ for\ all\ i = 1,2 \ldots . n$$

$$such\ that\ y_i \in \{1, -1\}$$

- Once, we find the value of β with some optimization technique (Stochastic Gradient Descent, Lagrange Multiplier), each test case is labelled according to the sign of $f(x_i)$

$$\text{i.e. } label_i = sign(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots \ldots \ldots \ldots \ldots . . + \beta_k x_{ik})$$

- This variant of SVM which is valid for linear separable data is called **Hard Linear SVM Model (where C is very very large such that $\sum_{i=1}^{n}(y_i cost_1(z) + (1 - y_i)\ cost_0(z))$ approximate to zero).**

Hard Linear SVM- Intuition

- The objective function of SVM (discussed in the previous slide) is a quadratic function and when it is optimized subject to given constraints, it gives a very interesting decision boundary.
- Let's consider the linearly separable case (as shown in Fig (a)) which means there exists straight lines (as shown in Fig (b)) that can separate the data into its categories.
- A SVM Classifier from many linear decision boundaries (shown as pink, green, black etc.) will give the black decision boundary – which seems to be much better decision boundary.

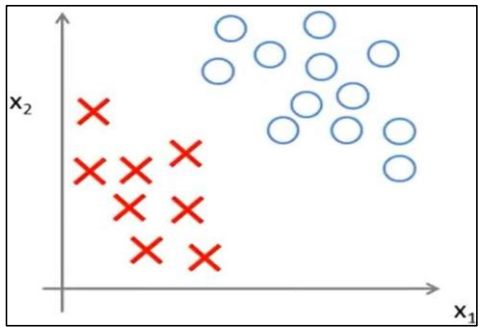


Fig. (a)

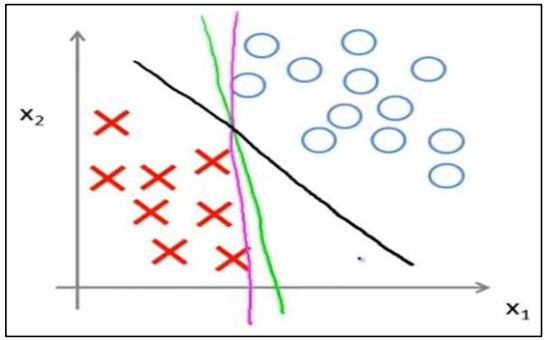
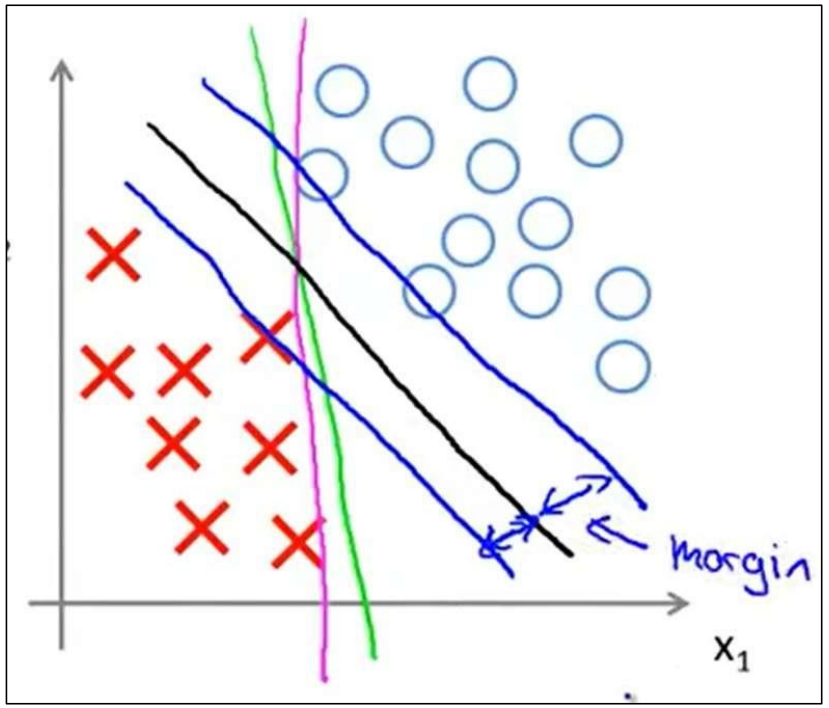


Fig. (b)

Hard Linear SVM- Intuition (Contd…..)

- The black decision boundary is better because it has larger distance from the nearest training examples of both categories.
- This distance between the distance boundary and the nearest training examples of both categories is called **margin** and the nearest training examples are called **support vectors**.
- Since, the support vector machines tends to maximize the margin, therefore, these are also sometimes called as **Maximum Margin Classifiers** or **Optimal Margin Classifiers**.



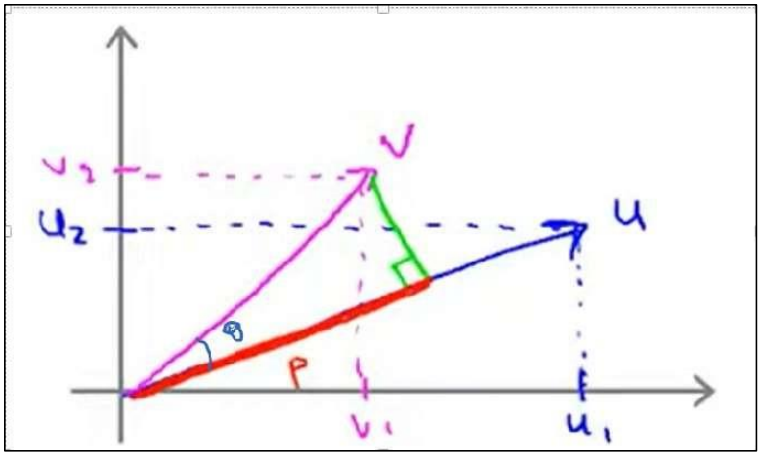
How Hard SVM Optimization leads to large margin classifier?

- In order to understand, how the objective function of Hard SVM leads to a maximum margin classifier, we need to understand the concept of Vector Inner Product.

$$Vector\ Inner\ Product\ between\ u\ \&\ v$$
$$= u.v = |u||v|cos\theta$$
$$= |u|(|v|cos\theta)$$
$$= |u|Projection\ of\ v\ on\ u$$
$$= |u| \times P$$

where P is signed, i.e., when θ is greater than 90, then P will be negative.

Vector inner product between two vectors is thus projection of one vector multiplied by the length of another vector.



How Hard SVM Cost Function leads to large margin classifier? (Contd…)

We know, for Hard SVM cost function is given by:

$$Minimize\ J(\beta) = \frac{1}{2}\sum_{j=0}^{k} \beta_j^2$$

$$subject\ to\ y_i f(x_i) \geq 1\ for\ all\ i = 1,2 \ldots . n$$

$$such\ that\ y_i \in \{1, -1\}$$

Just for simplification, let us consider k=2 i.e., 2 dimensional feature space and β_0=0.

Therefore, $J(\beta) = \frac{1}{2}\sum_{k=1}^{2} \beta_j^2 = \frac{1}{2}(\beta_1^2 + \beta_2^2) = \frac{1}{2}\left(\sqrt{(\beta_1^2 + \beta_2^2)}\right)^2 = \frac{1}{2}|\beta|^2$

- Subject to $X_i\beta \geq 1\ if\ yi = 1$
  $and\ X_i\beta \leq -1\ if\ yi = -1$

How Hard SVM Cost Function leads to large margin classifier? (Contd…)

Now, We know

$$X_i\beta = \text{projection of Xi on } \beta \; vector \times |\beta|$$

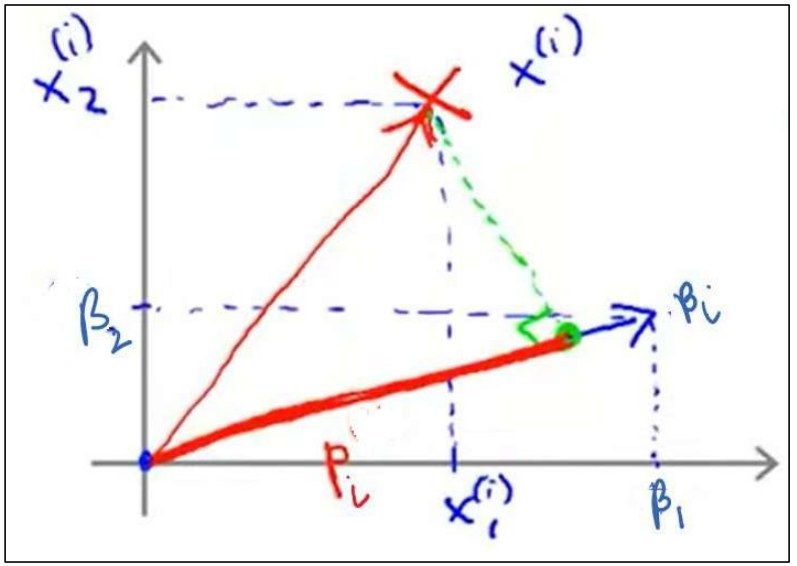
$$= p_i \times |\beta|$$

Therefore, the constraints can be reformulated as:

$$p_i \times |\beta| \geq 1 \; if \; yi = 1$$

$$and \; p_i \times |\beta| \leq -1 \; if \; yi = -1$$

For all i=1,2,3…..n



How Hard SVM Cost Function leads to large margin classifier? (Contd…)

Now lets consider, we have to find decision boundary for data shown in figure.

Lets it sets the green line decision boundary.

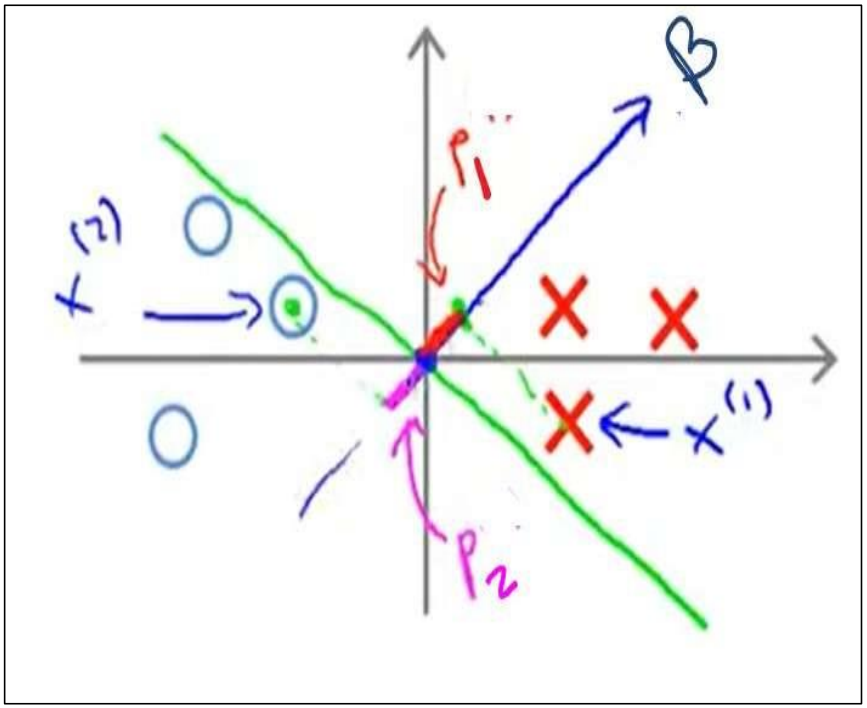
For a training example, marked as $x^{(1)}$, the projection is marked as p_1. which is quite small positive quantity .

Now for p_1 $|\beta| \geq 1$; $|\beta|$ must be large.

For a training example, marked as $x^{(2)}$, the projection is marked as p_2. which is quite small negative quantity .

Now for p_2 $|\beta| \leq -1$; $|\beta|$ must be large.

But, since we need to minimize $|\beta|^2$, so a large value is not possible. Hence this decision boundary will not be returned.



How Hard SVM Cost Function leads to large margin classifier? (Contd…)

Now, if it sets the green line decision boundary (shown in the figure).

For a training example, marked as $x^{(1)}$, the projection is marked as p_1. which is a large positive quantity .

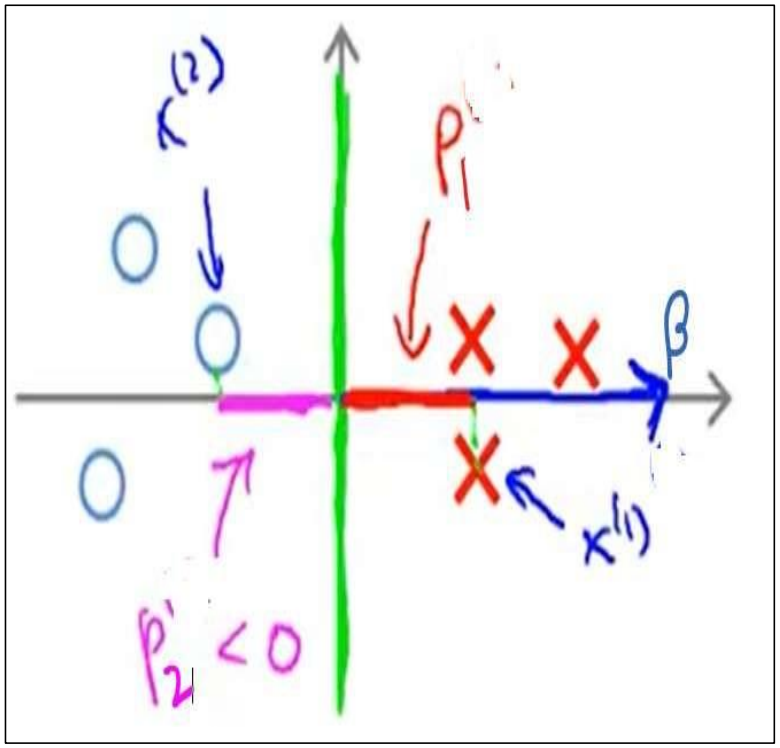
Now for $p_1 \, |\beta| \geq 1$; $|\beta|$ can be small.

For a training example, marked as $x^{(2)}$, the projection is marked as p_2. which is large negative quantity .

Now for $p_2 \, |\beta| \leq -1$; $|\beta|$ can be small.

Since we need to minimize $|\beta|^2$, so it is possible only if projection of examples on β are large (as shown in the figure).

Hence, SVM always return maximum margin classifier.



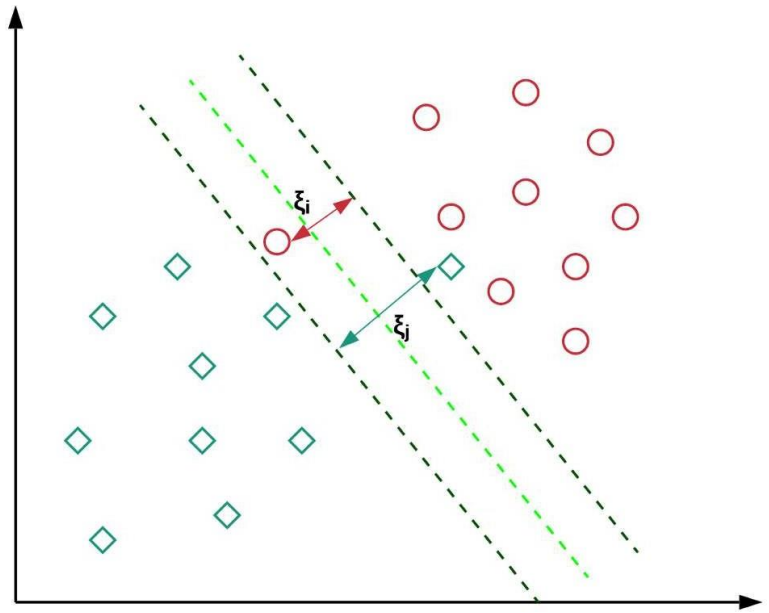
Soft SVM Classifier

- In hard SVM classifier, we consider the value of the parameter C quite high, due to which the training error is reduced to zero.
- Therefore, in hard SVM classifier focus is more on maximizing the margin, at the at the expense of minimizing the classification mistakes.
- Soft allows SVM to make a certain number of mistakes and keep margin as wide as possible so that other points can still be classified correctly. This can be done simply by modifying the objective of SVM.
- So, in soft SVM classifier, the hyperparameter C is set so that, it maintains the trade-off between maximizing the margin and minimizing the mistakes.

$$J = \frac{1}{2}\sum_{j=0}^{k} |\beta_j|^2 + C(misclassifciation\ error)$$

Soft SVM Classifier (Contd….)

- However, not all mistakes are equal.
- Data points that are far away on the wrong side of the decision boundary should incur more penalty as compared to the ones that are closer.
- The idea is: for every data point x_i, we introduce a slack variable ξ_i.
- The value of ξ_i is the distance of x_i from the corresponding class's margin if x_i is on the wrong side of the margin, otherwise zero.
- Thus the points that are far away from the margin on the wrong side would get more penalty.



Soft SVM Classifier –Cost Function

- Therefore, cost function of Soft SVM classifier is modified as:

$$J = \frac{1}{2}\sum_{j=0}^{k} \beta_j^2 + C\sum_{i=1}^{n} \xi_i$$

where $\xi_i \geq 0$

subject to $y_i f(x_i) \geq 1 - \xi_i$

Here, the left-hand side of the inequality could be thought of like the confidence of classification. Confidence score ≥ 1 suggests that classifier has classified the point correctly. However, if confidence score ≤ 1, it means that classifier did not classify the point correctly and incurring a linear penalty of ξ_i

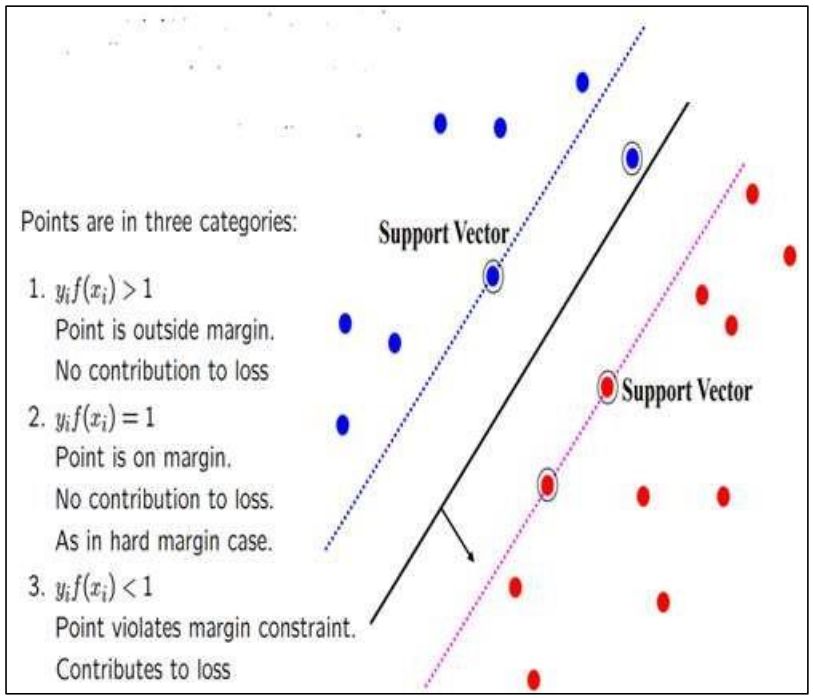
which, together with $\xi_i \geq 0$, is equivalent to

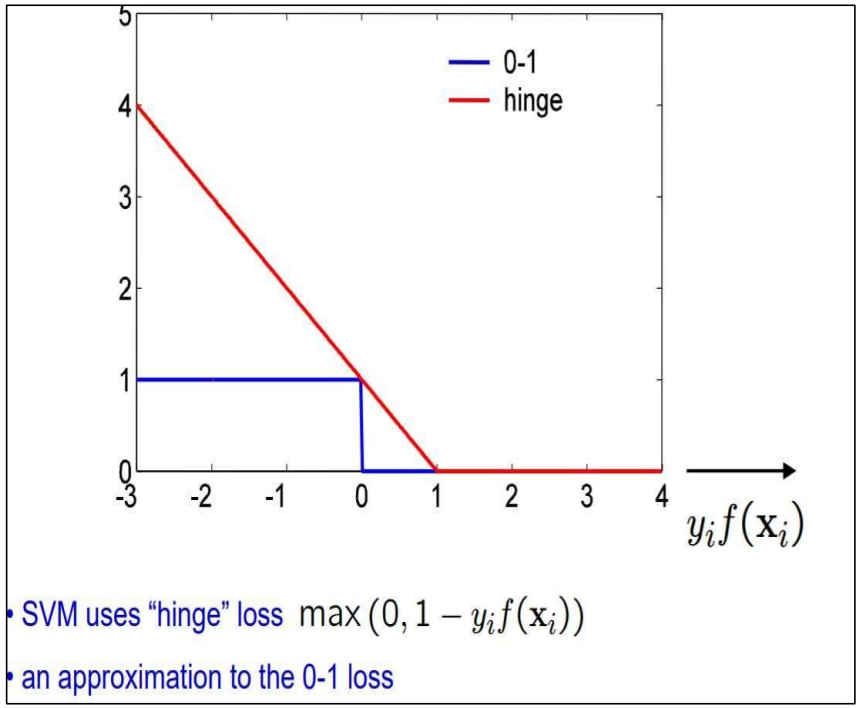
$$\xi_i = \max(0, 1 - y_i f(x_i))$$

- Hence, the learning problem is equivalent to the unconstrained optimization problem over β

$$\boldsymbol{J = \frac{1}{2}\sum_{j=0}^{k} \beta_j^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i f(x_i))}$$

Soft SVM Classifier –Cost Function





Soft SVM-Optimizing Cost Function

- The cost function for soft SVM classifier (discussed in previous slide), is convex and can be optimized using optimization algorithm such as Gradient Descent.

$$J = \frac{1}{2}\sum_{j=0}^{k} \beta_j^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i f(x_i))$$

- If $\max(0, 1 - y_i f(x_i)) = 0$, then

$$J = \frac{1}{2}\sum_{j=0}^{k} \beta_j^2 \quad and \quad \frac{\partial J}{\partial \beta_j} = \beta_j$$

- Therefore, β_j is updated as:

$$\beta_j = \beta_j - \alpha\beta_j$$

- If $\max(0, 1 - y_i f(x_i)) = 1 - y_i f(x_i)$, then

$$J = \frac{1}{2}\sum_{j=0}^{k} \beta_j^2 + C\sum_{i=1}^{n} (1 - y_i f(x_i))$$

$$and \quad \frac{\partial J}{\partial \beta_j} = \beta_j - C\sum_{i=1}^{n} y_i x_{ij}$$

- Therefore, β_j is updated as:

$$\beta_j = \beta_j - \alpha\left(\beta_j - C\sum_{i=1}^{n} y_i x_{ij}\right)$$

The gradient descent algorithm, would be slow, and may not converge. So, we apply *Stochastic Gradient Descent* (that do not work on all the data samples; it randomly select one or few data samples at each iteration).