Pattern Recognition - Lecture 6
NBTree, OneR & SVM

Dr. Dewan Md. Farid

Associate Professor, Department of Computer Science & Engineering
United International University, Bangladesh

March 25, 2018

Naïve Bayesian tree

The naïve Bayesian tree (NBTree) is a hybrid learning approach of decision tree and naïve Bayesian classifier.

- In NBTree nodes contain and split as regular decision trees, but the leaves are replaced by naïve Bayesian classifier.
- The advantage of both decision tree and naïve Bayes can be utilised simultaneously.
- Depending on the precise nature of the probability model, NB classifier can be trained very efficiently in a supervised learning.

NBTree (con.)

Adaptive NBtree splits the dataset by applying entropy based algorithm and then used standard NB classifier at the leaf node to handle attributes. It applies strategy to construct decision tree and replaces leaf node with NB classifier.

OneR Classifier

Algorithm 1 OneR

Input: $D = \{x_1, x_2, \cdots, x_n\}$ // Training data.
Output: OneR Model.
Method:

1: **for** each attribute, $A_i \in D$, **do**
2: **for** each attribute value, $A_{ij} \in A_i$, **do**
3: **Make a classification rule:**
4: count how often each class appears
5: find the most frequent class
6: make the rule assign that class to this A_{ij};
7: **end for**
8: Calculate the error rate of this attribute's A_i rule.
9: **end for**
10: Choose the attribute $A_i \in D$ with the smallest error rate.

Table: The playing tennis dataset

Day	Outlook	Temperature	Humidity	Wind	Play
D_1	Sunny	Hot	High	Weak	No
D_2	Sunny	Hot	High	Strong	No
D_3	Overcast	Hot	High	Weak	Yes
D_4	Rain	Mild	High	Weak	Yes
D_5	Rain	Cool	Normal	Weak	Yes
D_6	Rain	Cool	Normal	Strong	No
D_7	Overcast	Cool	Normal	Strong	Yes
D_8	Sunny	Mild	High	Weak	No
D_9	Sunny	Cool	Normal	Weak	Yes
D_{10}	Rain	Mild	Normal	Weak	Yes
D_{11}	Sunny	Mild	Normal	Strong	Yes
D_{12}	Overcast	Mild	High	Strong	Yes
D_{13}	Overcast	Hot	Normal	Weak	Yes
D_{14}	Rain	Mild	High	Strong	No

Support Vector Machines

In pattern recognition, **support vector machines** (SVM) are supervised learning models with associated learning algorithms that analyse data and recognise patterns, used for classification and regression analysis.

Given a set of training vectors, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new vectors into one category or the other, making it a non-probabilistic binary linear classifier.

SVM (con.)

- A SVM model is a representation of the vectors as points in space, mapped so that the vectors of the separate categories are divided by a clear gap that is as wide as possible.
- New vectors are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.
- In addition to performing linear classification, SVM can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

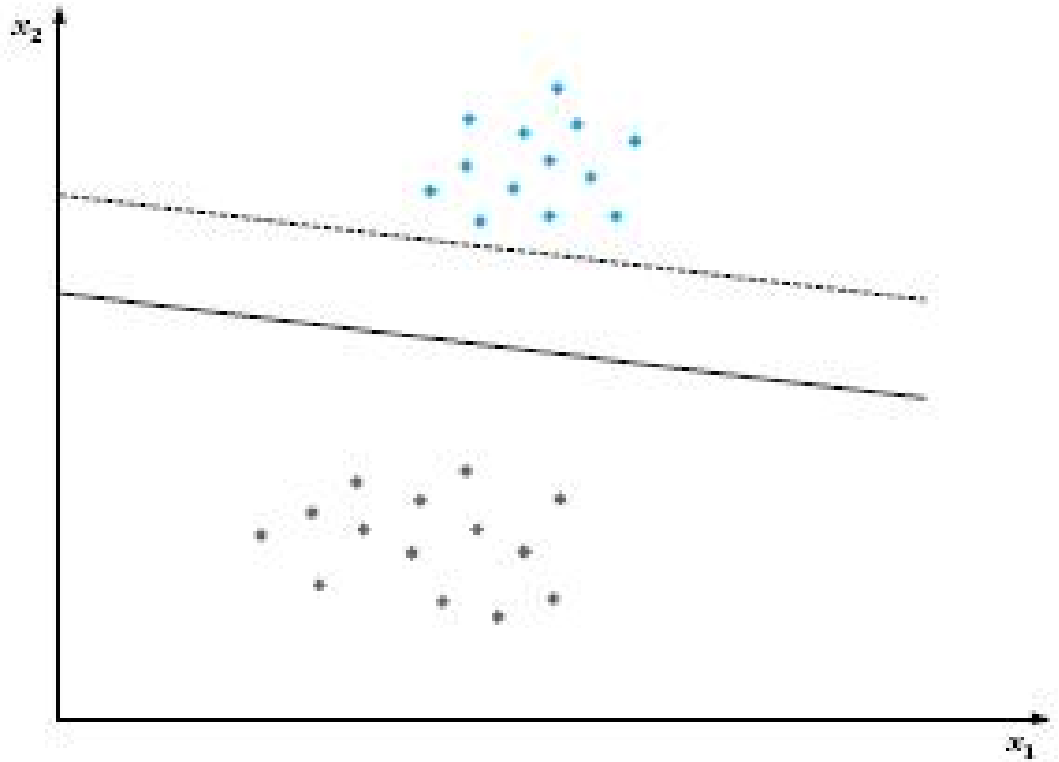

Figure: An example of a linearly separable two-class problem with two possible linear classifiers.

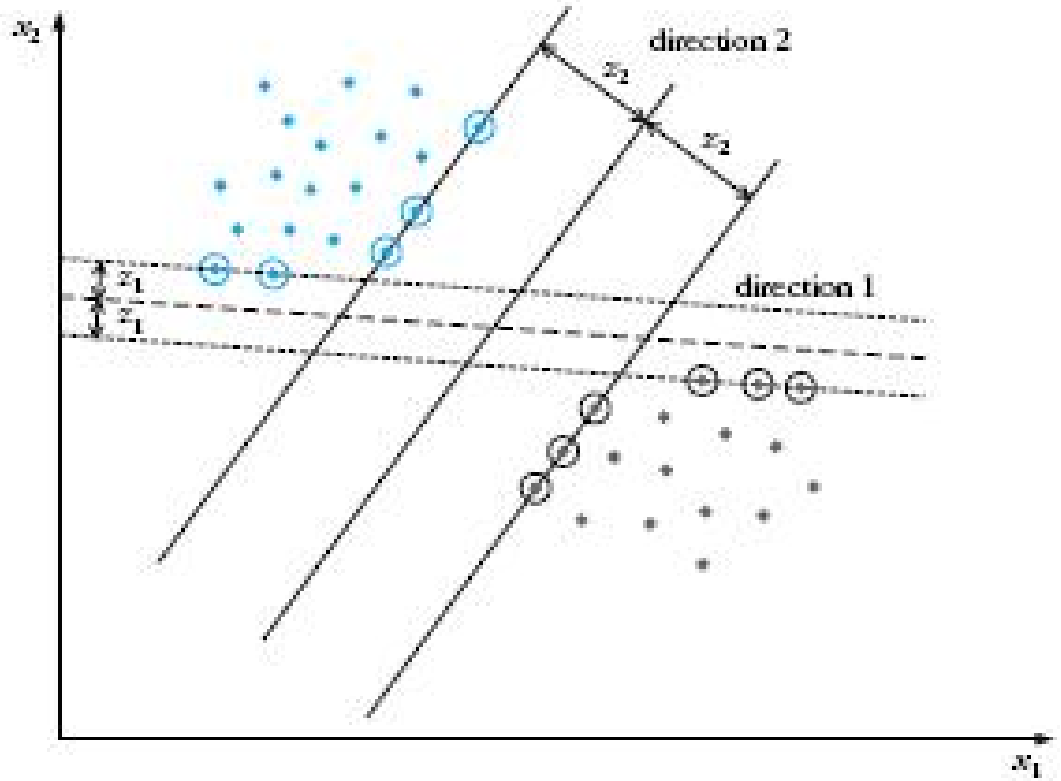

Figure: An example of a linearly separable two-class problem with two possible linear classifiers.

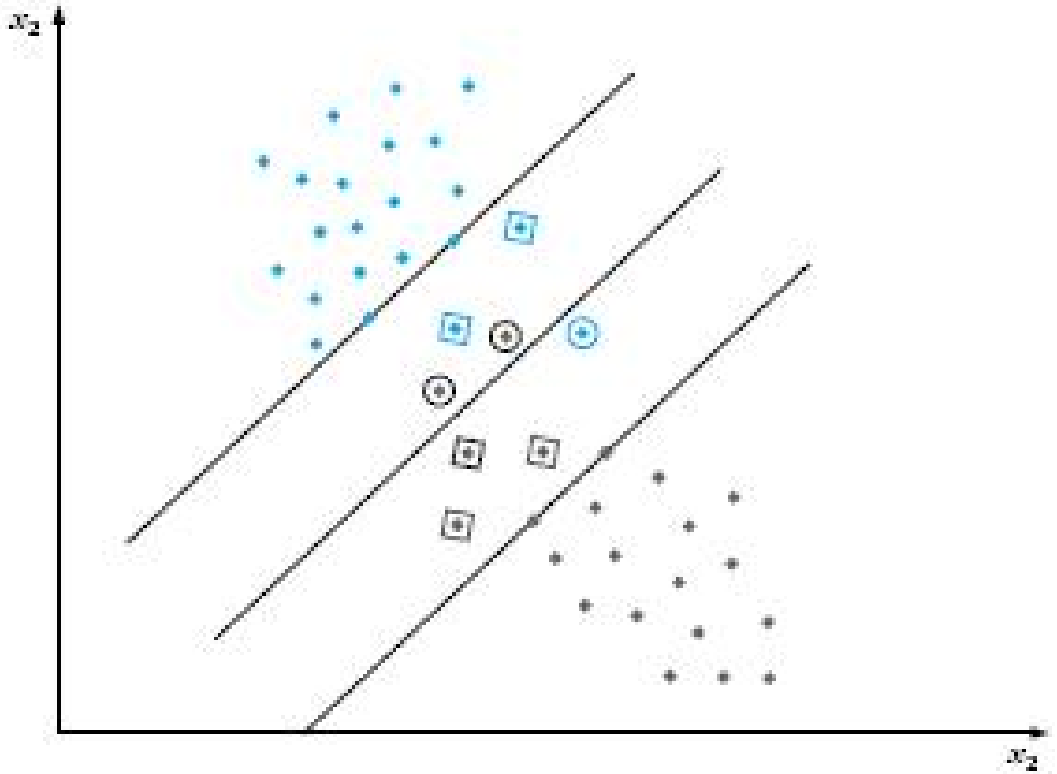

Figure: In the non-separable class case, points fall inside the class separation band.

Vector Inner Product

Let us consider, u and v are the two dimensional vector, $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ and $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$.

$$u^T v = [u_1 u_2] \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = v^T u = p. \parallel u \parallel \tag{1}$$

$$\parallel u \parallel = \sqrt{{u_1}^2 + {u_2}^2} \in \mathbb{R} \tag{2}$$

Vector Inner Product (con.)

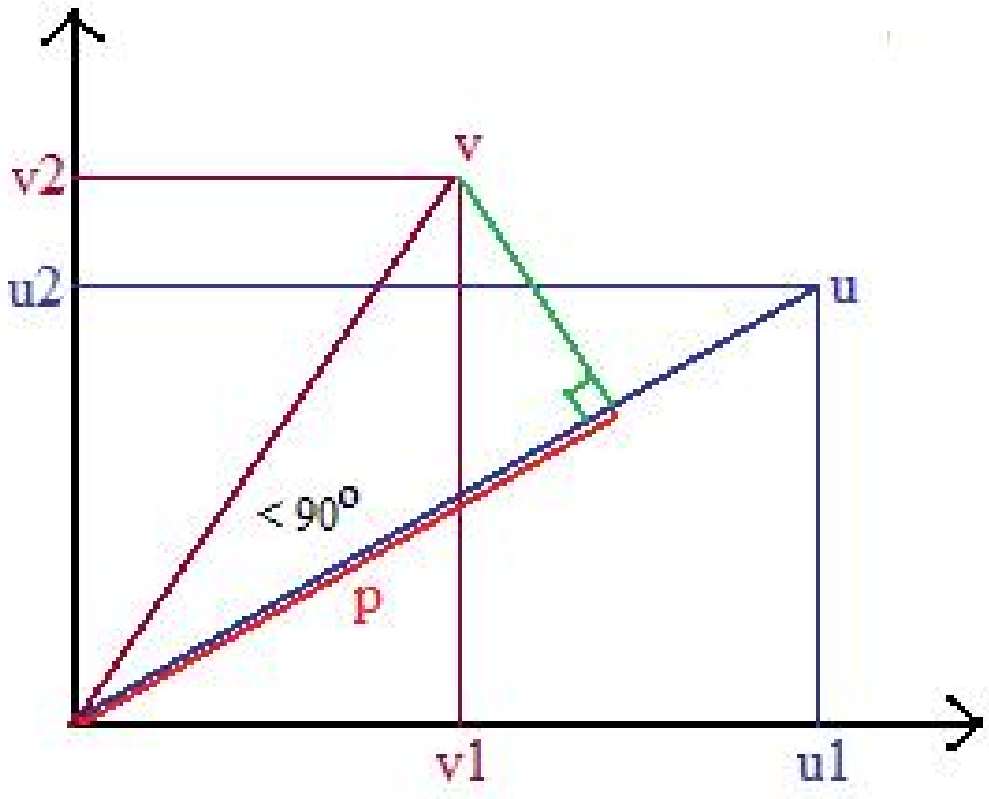

Figure: How to compute the vector inner product between u and v, angle is less than 90^o.

Vector Inner Product (con.)

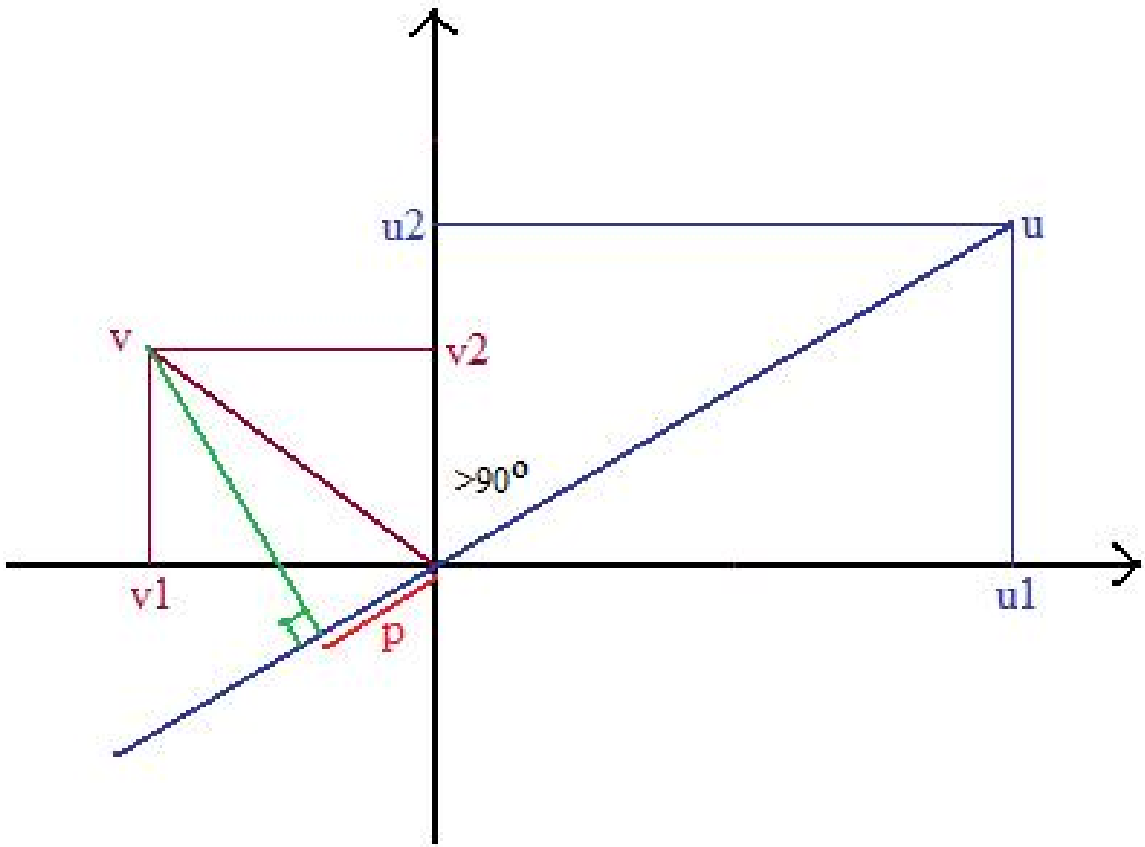

Figure: How to compute the vector inner product between u and v, angle is greater than 90^o.

Vector Inner Product (con.)

$$u^T v = p. \parallel u \parallel \tag{3}$$

$$u^T v = u_1 v_1 + u_2 v_2 \tag{4}$$

So, we can compute the vector inner product using equation 3 and 4.

$$u^T v = v^T u \tag{5}$$

$$u^T v = p. \parallel u \parallel = v^T u = u_1 v_1 + u_2 v_2 \leftarrow p \in \mathbb{R} \tag{6}$$

In equation 6, u_1 is a row number and p is also row number. The angle between u and v is greater than 90^o, where $p < 0$. Inner product of two vector can be negative if the angle is greater than 90^o. Now, we are going to use this properties (**vector inner product**) to try to understand **SVM** optimisation.

SVM Decision Boundary

SVM try to minimise the θ in the following equation.

$$min_\theta \frac{1}{2} \sum_{j=1}^{n} (\theta_j)^2 \tag{7}$$

From equation 1, we can write $\theta^T x^i \geq 1$ if $y^i = 1$ and $\theta^T x^i \leq -1$ if $y^i = 0$. Now simplification is $\theta_0 = 0$, and $n = 2$, number of feature. So, we have only two feature x_1 and x_2.

$$min_\theta \frac{1}{2} \sum_{j=1}^{n} (\theta_j)^2 = \frac{1}{2}({\theta_1}^2 + {\theta_2}^2) = \frac{1}{2}(\sqrt{{\theta_1}^2 + {\theta_2}^2})^2 = \frac{1}{2} \| \theta \|^2 \tag{8}$$

SVM Decision Boundary (con.)

So, all SVM doing in the optimisation objective is to minimising the $\| \theta \|$ of the length of the parameter vector θ. Therefore,

$$\theta^T x^i = p^i . \| \theta \| \tag{9}$$

$$\theta^T x^i = \theta_1 {x_1}^i + \theta_2 {x_2}^i \tag{10}$$

Therefore, $min_\theta \frac{1}{2} \sum_{j=1}^{n} (\theta_j)^2 = \frac{1}{2} \| \theta \|^2$, where $p^i . \| \theta \| \geq 1$, if $y^i = 1$, and $p^i . \| \theta \| \leq -1$, if $y^i = 0$. And p^i is the projection of x^i onto the vector θ.

SVM Decision Boundary (con.)

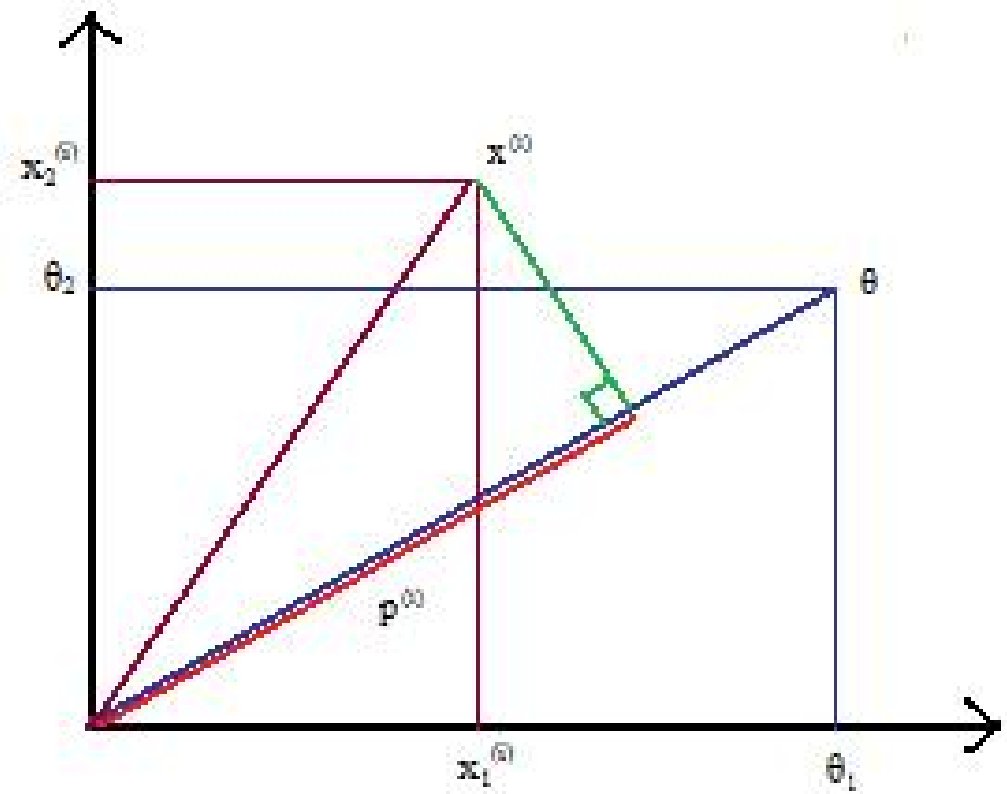

Figure: Projection of p^i, i_{th} training example onto the feature θ.

SVM Decision Boundary (con.)

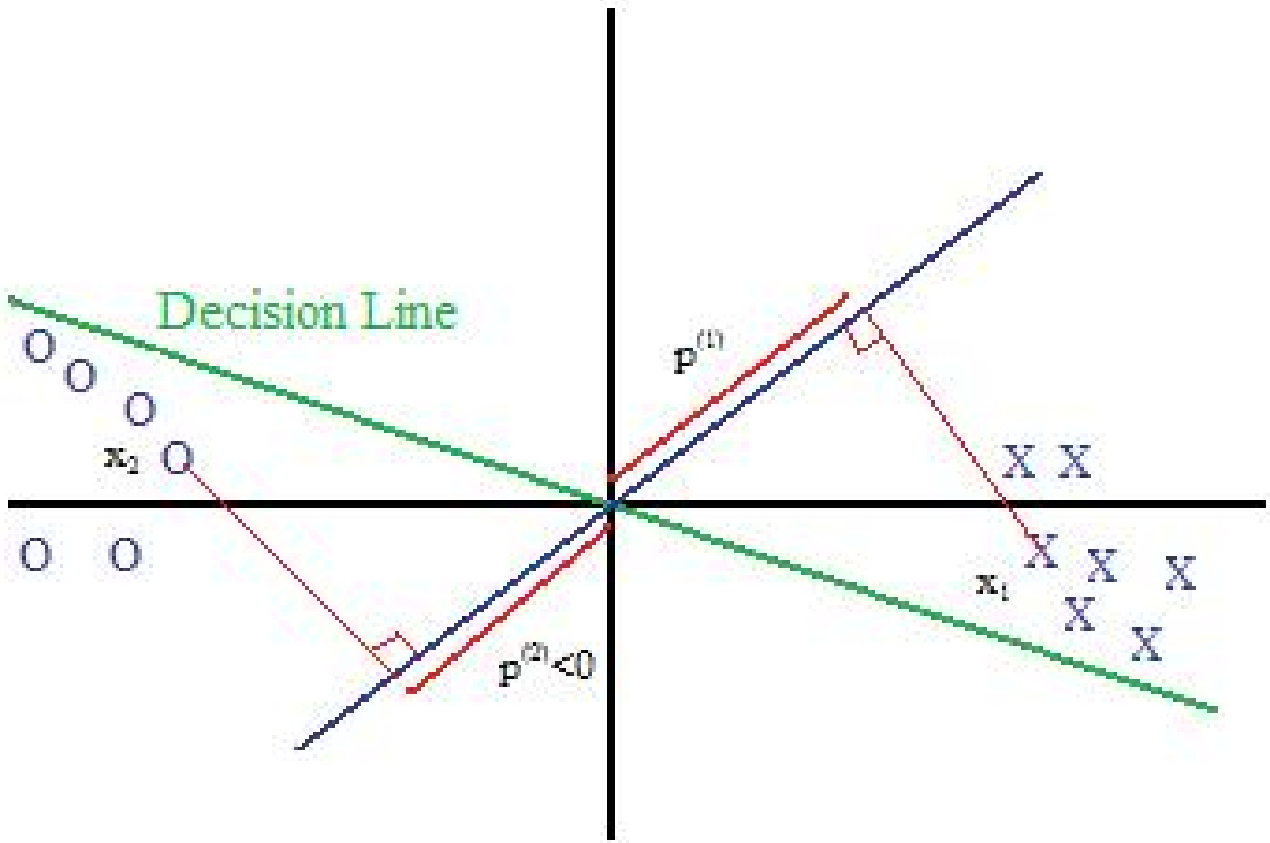

Figure: Not good decision boundary.

SVM Decision Boundary (con.)

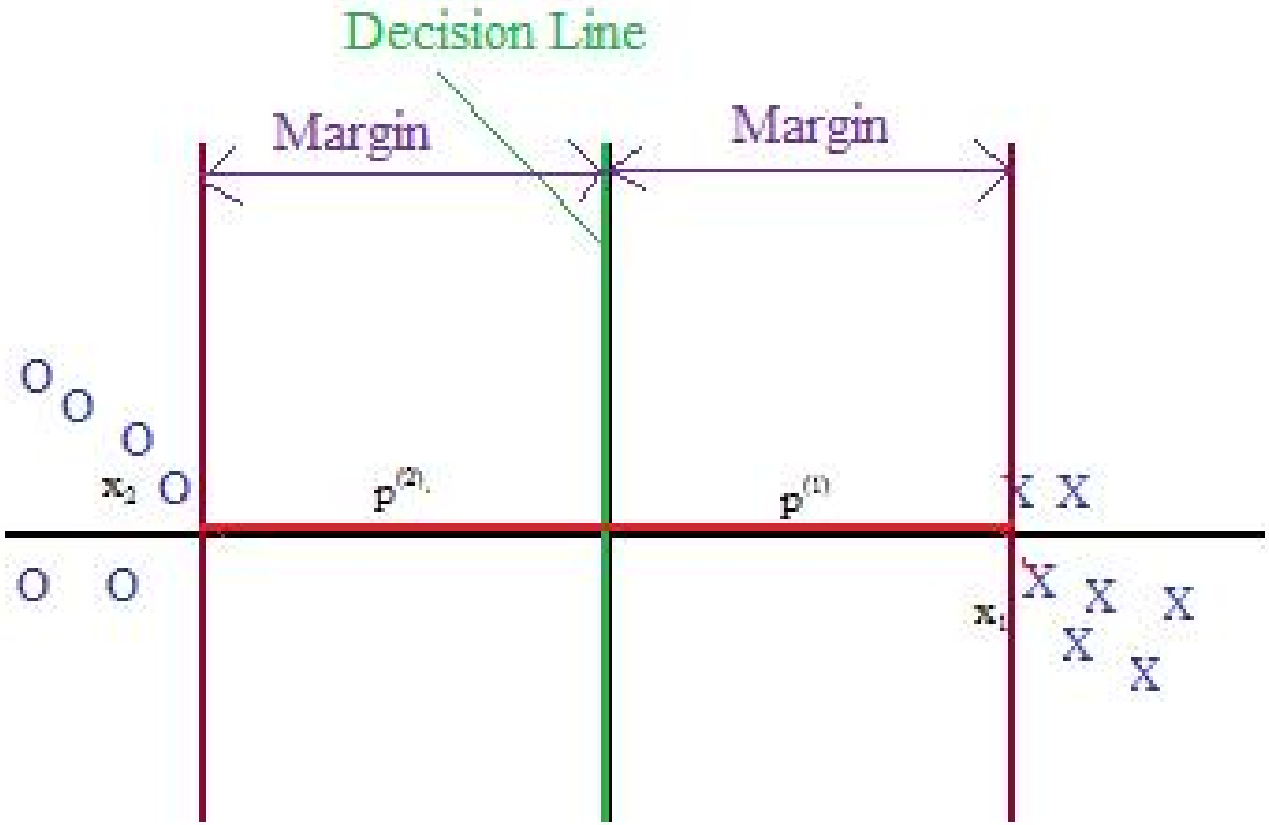

Figure: Good decision boundary

*** THANK YOU ***