Support Vector Machines and KNN & their Comparison with LDA & QDA

Instructor : Dr. Deepayan Sarkar

Stat-Math Unit, Indian Statistical Institute Delhi.

Group III

Debarshi Chakraborty
Arijit Naskar

Spandan Ghoshal

Contents

Introduction

- In this presentation we try to introduce the seperating hyperplane classifiers and the basic idea behind their working mechanism.

Introduction

- In this presentation we try to introduce the seperating hyperplane classifiers and the basic idea behind their working mechanism.
- Thereby we move to the more useful support vector classifiers for non-seperable datasets and their further modification using kernel functions.

Introduction

- In this presentation we try to introduce the seperating hyperplane classifiers and the basic idea behind their working mechanism.
- Thereby we move to the more useful support vector classifiers for non-seperable datasets and their further modification using kernel functions.
- Thereafter, we give a brief idea of KNN classifiers.

Introduction

- In this presentation we try to introduce the seperating hyperplane classifiers and the basic idea behind their working mechanism.
- Thereby we move to the more useful support vector classifiers for non-seperable datasets and their further modification using kernel functions.
- Thereafter, we give a brief idea of KNN classifiers.
- Lastly, we compare these newly introduced classifiers with more other conventional techniques such as LDA, QDA etc.

Training Data and Seperating Hyperplane

- Suppose our training dataset consists of N pairs $(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_N}, y_N)$, with the covariates or input features $\boldsymbol{x_i} \in \mathbb{R}^p$ and each of the observations being classified to either of the two categories $\{-1, 1\}$ i.e. $y_i = \{-1, 1\}$.

Training Data and Seperating Hyperplane

- Suppose our training dataset consists of N pairs $(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_N}, y_N)$, with the covariates or input features $\boldsymbol{x_i} \in \mathbb{R}^p$ and each of the observations being classified to either of the two categories $\{-1, 1\}$ i.e. $y_i = \{-1, 1\}$.
- We define a hyperplane by the set of values

$$L = \left\{\boldsymbol{x} : f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\beta} + \beta_0 = 0\right\}. \tag{2.1}$$

which we will use to classify our training observations.

Training Data and Seperating Hyperplane

- Suppose our training dataset consists of N pairs $(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_N}, y_N)$, with the covariates or input features $\boldsymbol{x_i} \in \mathbb{R}^p$ and each of the observations being classified to either of the two categories $\{-1, 1\}$ i.e. $y_i = \{-1, 1\}$.
- We define a hyperplane by the set of values

$$L = \left\{\boldsymbol{x} : f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\beta} + \beta_0 = 0\right\}. \tag{2.1}$$

  which we will use to classify our training observations.
- Now a classification rule induced by $f(\boldsymbol{x})$ is :-

$$G(\boldsymbol{x}) = \text{sign}\left[\boldsymbol{x}^T \boldsymbol{\beta} + \beta_0\right] \tag{2.2}$$

  i.e, a new observation with observed covariate values $\boldsymbol{x}$ is classified as belonging to category -1 if $\boldsymbol{x}^T \boldsymbol{\beta} + \beta_0 < 0$ and to category 1 otherwise.

Seperation Boundary and Margin

- So this means that we can consider the hyperplane $L = \left\{ \boldsymbol{x} : f\left(\boldsymbol{x}\right) = \boldsymbol{x}^T \boldsymbol{\beta} + \beta_0 = 0 \right\}$ as the seperation boundary for the two classes.

Seperation Boundary and Margin

- So this means that we can consider the hyperplane $L = \left\{ \boldsymbol{x} : f\left(\boldsymbol{x}\right) = \boldsymbol{x}^T \boldsymbol{\beta} + \beta_0 = 0 \right\}$ as the seperation boundary for the two classes.
- Also, we note that the signed distance of any point $\boldsymbol{x}_0$ from L is

$$\frac{1}{||\boldsymbol{\beta}||} \left(\boldsymbol{\beta}^T \boldsymbol{x}_0 + \beta_0 \right) = \frac{f\left(\boldsymbol{x}_0\right)}{||f'\left(\boldsymbol{x}_0\right)||} \tag{2.3}$$

.

Seperation Boundary and Margin

- So this means that we can consider the hyperplane $L = \left\{ \boldsymbol{x} : f\left(\boldsymbol{x}\right) = \boldsymbol{x}^T \boldsymbol{\beta} + \beta_0 = 0 \right\}$ as the seperation boundary for the two classes.
- Also, we note that the signed distance of any point $\boldsymbol{x}_0$ from L is

$$\frac{1}{||\boldsymbol{\beta}||} \left(\boldsymbol{\beta}^T \boldsymbol{x}_0 + \beta_0 \right) = \frac{f\left(\boldsymbol{x}_0\right)}{||f'\left(\boldsymbol{x}_0\right)||} \tag{2.3}$$

.

- If the classes are **linearly seperable**, then we can find a function $f\left(\boldsymbol{x}\right)$ such that

$$y_i \frac{f\left(\boldsymbol{x}_i\right)}{||f'\left(\boldsymbol{x}_i\right)||} > 0 \iff y_i f\left(\boldsymbol{x}_i\right) > 0 \tag{2.4}$$

$\forall \ i = 1, 2, \ldots, N$.

Seperation Boundary and Margin

- So this means that we can consider the hyperplane $L = \left\{ \boldsymbol{x} : f\left(\boldsymbol{x}\right) = \boldsymbol{x}^T \boldsymbol{\beta} + \beta_0 = 0 \right\}$ as the seperation boundary for the two classes.
- Also, we note that the signed distance of any point $\boldsymbol{x}_0$ from L is

$$\frac{1}{||\boldsymbol{\beta}||} \left(\boldsymbol{\beta}^T \boldsymbol{x}_0 + \beta_0 \right) = \frac{f\left(\boldsymbol{x}_0\right)}{||f'\left(\boldsymbol{x}_0\right)||} \tag{2.3}$$

.

- If the classes are **linearly seperable**, then we can find a function $f\left(\boldsymbol{x}\right)$ such that

$$y_i \frac{f\left(\boldsymbol{x}_i\right)}{||f'\left(\boldsymbol{x}_i\right)||} > 0 \iff y_i f\left(\boldsymbol{x}_i\right) > 0 \tag{2.4}$$

$\forall\ i = 1, 2, \ldots, N$.

- But we can find infinitely many such function $f\left(\boldsymbol{x}\right)$ which can separate out the two classes.

Many seperating hyperplanes can be drawn

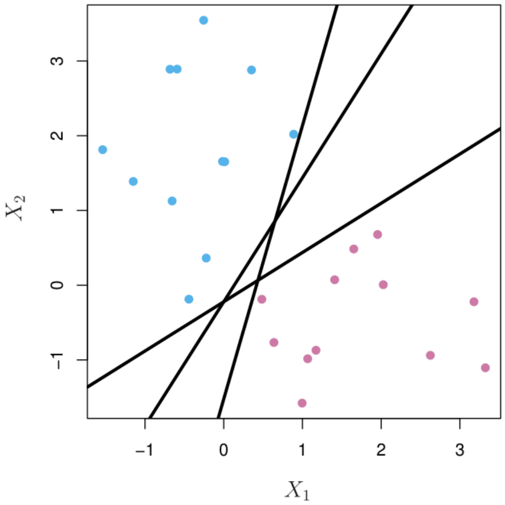

Figure: But infinitely many such hyperplanes can be drawn

Seperation Boundary and Margin

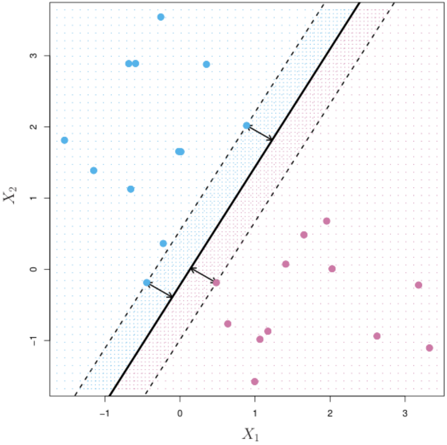

Figure: Here the margin is maximum !

Optimal Seperation Boundary

- Hence, we can find the hyperplane that creates the biggest margin between the training points for class 1 and -1. This can be formulated in the form of the following optimization problem :-

$$\max_{\beta_0,\boldsymbol{\beta},||\boldsymbol{\beta}||=1} M \tag{2.5}$$

$$\text{subjectto } y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right) \geq M, i=1,\ldots,N \tag{2.6}$$

Optimal Seperation Boundary

- Hence, we can find the hyperplane that creates the biggest margin between the training points for class 1 and -1. This can be formulated in the form of the following optimization problem :-

$$\max_{\beta_0, \boldsymbol{\beta}, ||\boldsymbol{\beta}||=1} M \tag{2.5}$$

$$\text{subjectto } y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq M, i = 1, \ldots, N \tag{2.6}$$

- We can get rid of the constraint $||\boldsymbol{\beta}|| = 1$ by replacing the conditions with :-

$$\max_{\beta_0, \boldsymbol{\beta}} M \tag{2.7}$$

$$\text{subjectto } \frac{1}{||\boldsymbol{\beta}||} y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq M, i = 1, \ldots, N \tag{2.8}$$

which is equivalent to

$$\max_{\beta_0, \boldsymbol{\beta}} M \tag{2.9}$$

$$\text{subjectto } y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq M ||\boldsymbol{\beta}||, i = 1, \ldots, N \tag{2.10}$$

Optimal Seperation Boundary

- Here, we can arbitarily put $M = \frac{1}{||\boldsymbol{\beta}||}$ in the inequality constraints to reformulate the problem as :-

$$\min_{\beta_0, \boldsymbol{\beta}} ||\boldsymbol{\beta}|| \tag{2.11}$$

$$\text{subjectto } y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq 1, i = 1, \ldots, N \tag{2.12}$$

and it's computationally more convinient to express this in the following manner :-

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2} ||\boldsymbol{\beta}||^2 \tag{2.13}$$

$$\text{subjectto } y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq 1, i = 1, \ldots, N \tag{2.14}$$

Optimal Seperation Boundary

- Here, we can arbitarily put $M = \frac{1}{||\boldsymbol{\beta}||}$ in the inequality constraints to reformulate the problem as :-

$$\min_{\beta_0, \boldsymbol{\beta}} ||\boldsymbol{\beta}|| \tag{2.11}$$

$$\text{subjectto } y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq 1, i = 1, \ldots, N \tag{2.12}$$

and it's computationally more convinient to express this in the following manner :-

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2} ||\boldsymbol{\beta}||^2 \tag{2.13}$$

$$\text{subjectto } y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq 1, i = 1, \ldots, N \tag{2.14}$$

- This is a convex optimization problem (quadratic criterion with linear inequality constraint). Before we go into the specific details of the optimization procedure, it's good to know about the basics of Lagrange Dual problem, weak & strong duality and lastly the KKT conditions for strong duality.

Lagrangian

- Suppose we have the following optimization problem :-

$$
\begin{aligned}
\text{minimize} f_0(\boldsymbol{x}) \quad & (2.15) \\
\text{subject to} f_i(\boldsymbol{x}) \leq 0, i = 1, 2, \ldots, m \quad & (2.16) \\
h_i(\boldsymbol{x}) = 0, i = 1, 2, \ldots, p \quad & (2.17)
\end{aligned}
$$

with variable $\boldsymbol{x} \in \mathbb{R}^n$, domain $\mathcal{D}$ and optimal value p^*.

Lagrangian

- Suppose we have the following optimization problem :-

$$\text{minimize} f_0(\boldsymbol{x}) \tag{2.15}$$

$$\text{subject to} f_i(\boldsymbol{x}) \leq 0, i = 1, 2, \ldots, m \tag{2.16}$$

$$h_i(\boldsymbol{x}) = 0, i = 1, 2, \ldots, p \tag{2.17}$$

  with variable $\boldsymbol{x} \in \mathbb{R}^n$, domain $\mathcal{D}$ and optimal value p^*.

- Then the lagrangian of this problem is defined as $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ with domain $\mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$, such that :-

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{m} \nu_i h_i(\boldsymbol{x}) \tag{2.18}$$

Lagrange dual function

- Now, the lagrange dual function associated with the lagrangian is $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ defined as :-

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x} \in \mathcal{D}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \tag{2.19}$$

$$= \inf_{\boldsymbol{x} \in \mathcal{D}} \left(f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{m} \nu_i h_i(\boldsymbol{x}) \right) \tag{2.20}$$

and it can be shown that g being a pointwise infimum of a affine function is concave and can be $-\infty$ for some $\boldsymbol{\lambda}, \boldsymbol{\nu}$.

Lagrange dual function

- Now, the most important property of g is that if $\boldsymbol{\lambda} \succeq \mathbf{0}$, then $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^*$ since, if $\tilde{\boldsymbol{x}}$ is a feasible solution, then

$$f_0(\tilde{\boldsymbol{x}}) \geq L(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \geq \inf_{\boldsymbol{x} \in \mathcal{D}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \tag{2.21}$$

Lagrange dual function

- Now, the most important property of g is that if $\boldsymbol{\lambda} \succeq \mathbf{0}$, then $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^*$ since, if $\tilde{\boldsymbol{x}}$ is a feasible solution, then

$$f_0(\tilde{\boldsymbol{x}}) \geq L(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \geq \inf_{\boldsymbol{x} \in \mathcal{D}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \tag{2.21}$$

- and then,

$$\min_{\tilde{\boldsymbol{x}} \in \mathcal{F}} f_0(\tilde{\boldsymbol{x}}) \geq g(\boldsymbol{\lambda}, \boldsymbol{\nu})$$

($\mathcal{F}$ being the set of all feasible solutions.)

$$\Longrightarrow p^* \geq g(\boldsymbol{\lambda}, \boldsymbol{\nu})$$

now, in order to find the best lower bound of p^*, we maximize $g(\boldsymbol{\lambda}, \boldsymbol{\nu})$ subject to $\boldsymbol{\lambda} \succeq \mathbf{0}$ which is a convex optimization problem.

Weak & Strong Duality

- If we denote the maximum obtained value of $g(\boldsymbol{\lambda}, \boldsymbol{\nu})$ by d^*, then trivially, $d^* \leq p^*$ which is called **weak duality** and this always holds. Whereas exact equality $(d^* = p^*)$ does not hold in general but usually holds for convex problems.

Weak & Strong Duality

- If we denote the maximum obtained value of $g(\boldsymbol{\lambda}, \boldsymbol{\nu})$ by d^*, then trivially, $d^* \leq p^*$ which is called **weak duality** and this always holds. Whereas exact equality $(d^* = p^*)$ does not hold in general but usually holds for convex problems.
- In order to gurantee the **strong duality** (exact equality) we impose something called the KKT (Karush-Kuhn-Tucker) conditions which are :-

$1)$The functions f_i and h_i are differentiable

$2)$primal constraints : $f_i(\boldsymbol{x}) \leq 0, \forall i$

$h_i(\boldsymbol{x}) = 0, \forall i$

$3)$dual constraints : $\boldsymbol{\lambda} \succeq \mathbf{0}$

$4)$complementary slackness : $\lambda_i f_i(\boldsymbol{x}) = 0, \forall i$

$5)$gradient of Lagrangian with respect to $\boldsymbol{x}$ vanishes:

$$\nabla f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \nabla f_i(\boldsymbol{x}) + \sum_{i=1}^{m} \nu_i \nabla h_i(\boldsymbol{x}) = 0$$

Back to our buisness!

- Hence, for the optimal seperating hyperplane, our primal function which is to be minimized is :-

$$L_p\left(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0\right) - 1\right] \qquad (2.22)$$

Back to our buisness!

- Hence, for the optimal seperating hyperplane, our primal function which is to be minimized is :-

$$L_p(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) - 1 \right] \quad (2.22)$$

- The corresponding dual function is obtained as :-

$$L_D(\boldsymbol{\alpha}) = \inf_{\beta_0, \boldsymbol{\beta}} L_p(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) \quad (2.23)$$

$$= L_p(\beta_0^*, \boldsymbol{\beta}^*, \boldsymbol{\alpha}) \quad (2.24)$$

where β_0^* and $\boldsymbol{\beta}^*$ are obtained by setting the gradient of L_p wrt $\beta_0, \boldsymbol{\beta}$, equal to 0.

Back to our buisness!

- Hence, for the optimal seperating hyperplane, our primal function which is to be minimized is :-

$$L_p\left(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0\right) - 1\right] \tag{2.22}$$

- The corresponding dual function is obtained as :-

$$L_D\left(\boldsymbol{\alpha}\right) = \inf_{\beta_0, \boldsymbol{\beta}} L_p\left(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) \tag{2.23}$$

$$= L_p\left(\beta_0^*, \boldsymbol{\beta}^*, \boldsymbol{\alpha}\right) \tag{2.24}$$

where β_0^* and $\boldsymbol{\beta}^*$ are obtained by setting the gradient of L_p wrt $\beta_0, \boldsymbol{\beta}$, equal to 0.

- Thus, $\nabla_{\beta_0} L_p = 0 \implies \sum_{i=1}^{N} \alpha_i y_i = 0$ and

  $\nabla_{\boldsymbol{\beta}} L_p = 0 \implies \boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$.

Back to our buisness!

- Hence, for the optimal seperating hyperplane, our primal function which is to be minimized is :-

$$L_p\left(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0\right) - 1\right] \quad (2.22)$$

- The corresponding dual function is obtained as :-

$$L_D\left(\boldsymbol{\alpha}\right) = \inf_{\beta_0, \boldsymbol{\beta}} L_p\left(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) \quad (2.23)$$

$$= L_p\left(\beta_0^*, \boldsymbol{\beta}^*, \boldsymbol{\alpha}\right) \quad (2.24)$$

where β_0^* and $\boldsymbol{\beta}^*$ are obtained by setting the gradient of L_p wrt $\beta_0, \boldsymbol{\beta}$, equal to 0.

- Thus, $\nabla_{\beta_0} L_p = 0 \implies \sum_{i=1}^{N} \alpha_i y_i = 0$ and

$\nabla_{\boldsymbol{\beta}} L_p = 0 \implies \boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i.$

- Putting this we get :-

$$L_D\left(\boldsymbol{\alpha}\right) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \quad (2.25)$$

Back to our buisness!

- now, we maximize this dual function w.r.t $\boldsymbol{\alpha}$ under the KKT conditions which includes

$$
\begin{aligned}
\text{primal constraints : } & y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right)-1\geq 0\\
\text{dual constraints : } & \alpha_i\geq 0\\
\text{complementary slackness :} & \alpha_i\left[y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right)-1\right]=0\\
& \text{for } i=1,2,\ldots,N\\
\text{zero gradient wrt } (\boldsymbol{\beta_0},\boldsymbol{\beta}) & \begin{cases}\sum_{i=1}^{N}\alpha_i y_i=0\\ \boldsymbol{\beta}=\sum\limits_{i=1}^{N}\alpha_i y_i\boldsymbol{x}_i\end{cases}
\end{aligned}
$$

Back to our buisness!

- now, we maximize this dual function w.r.t $\boldsymbol{\alpha}$ under the KKT conditions which includes

$$\begin{aligned}
\text{primal constraints :}\ & y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right)-1\geq 0\\
\text{dual constraints :}\ & \alpha_i\geq 0\\
\text{complementary slackness :}& \alpha_i\left[y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right)-1\right]=0\\
&\text{for } i=1,2,\ldots,N\\
\text{zero gradient wrt }\ (\boldsymbol{\beta_0},\boldsymbol{\beta})&\begin{cases}\sum_{i=1}^{N}\alpha_i y_i=0\\ \boldsymbol{\beta}=\sum\limits_{i=1}^{N}\alpha_i y_i\boldsymbol{x}_i\end{cases}
\end{aligned}$$

- These conditions are needed in order to ensure strong duality of the optimization problem.

Back to our buisness!

- now, we maximize this dual function w.r.t $\boldsymbol{\alpha}$ under the KKT conditions which includes

$$\begin{aligned}
\text{primal constraints}: & \; y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right)-1\geq 0\\
\text{dual constraints}: & \; \alpha_i \geq 0\\
\text{complementary slackness}: & \alpha_i\left[y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right)-1\right]=0\\
& \text{for } i=1,2,\ldots,N\\
\text{zero gradient wrt } (\boldsymbol{\beta_0},\boldsymbol{\beta}) & \begin{cases} \sum_{i=1}^{N}\alpha_i y_i = 0\\ \boldsymbol{\beta}=\sum\limits_{i=1}^{N}\alpha_i y_i \boldsymbol{x}_i \end{cases}
\end{aligned}$$

- These conditions are needed in order to ensure strong duality of the optimization problem.
- This is a comparatively simpler convex optimization problem for which standard software can be used.

Why "Support" Vectors ?

- From conditions (20), we can see that

Why "Support" Vectors ?

- From conditions (20), we can see that
  - if $\alpha_i > 0$, then $y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) - 1 = 0$ i.e. the $\boldsymbol{x}_i$ is on the boundary of the slab;

Why "Support" Vectors ?

- From conditions (20), we can see that
  - if $\alpha_i > 0$, then $y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) - 1 = 0$ i.e. the $\boldsymbol{x}_i$ is on the boundary of the slab;
  - on the other hand if $y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) > 1$ i.e. it's not on the boundary of the slab, then $\alpha_i = 0$.

Why "Support" Vectors ?

- From conditions (20), we can see that
  - if $\alpha_i > 0$, then $y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) - 1 = 0$ i.e. the $\boldsymbol{x}_i$ is on the boundary of the slab;
  - on the other hand if $y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) > 1$ i.e. it's not on the boundary of the slab, then $\alpha_i = 0$.
  - And since, the solution vector $\boldsymbol{\beta}$ equals to $\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$ under the KKT conditions, we see that it's determined by only the points $\boldsymbol{x}_i$, $i \in \mathcal{S}$ where $\mathcal{S}$ denotes the set of points which lie on the boundary of the slab. (Hence here, $\widehat{\boldsymbol{\beta}} = \sum_{i \in \mathcal{S}} \widehat{\alpha}_i y_i \boldsymbol{x}_i$)

Why "Support" Vectors ?

- From conditions (20), we can see that
  - if $\alpha_i > 0$, then $y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) - 1 = 0$ i.e. the $\boldsymbol{x}_i$ is on the boundary of the slab;
  - on the other hand if $y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) > 1$ i.e. it's not on the boundary of the slab, then $\alpha_i = 0$.
  - And since, the solution vector $\boldsymbol{\beta}$ equals to $\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$ under the KKT conditions, we see that it's determined by only the points $\boldsymbol{x}_i$, $i \in \mathcal{S}$ where $\mathcal{S}$ denotes the set of points which lie on the boundary of the slab. (Hence here, $\widehat{\boldsymbol{\beta}} = \sum_{i \in \mathcal{S}} \widehat{\alpha}_i y_i \boldsymbol{x}_i$)
- The points in $\mathcal{S}$ are called support vectors as they determine the nature of the optimal seperating hyperplane.

Why "Support" Vectors ?

- From conditions (20), we can see that
  - if $\alpha_i > 0$, then $y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) - 1 = 0$ i.e. the $\boldsymbol{x}_i$ is on the boundary of the slab;
  - on the other hand if $y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) > 1$ i.e. it's not on the boundary of the slab, then $\alpha_i = 0$.
  - And since, the solution vector $\boldsymbol{\beta}$ equals to $\sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$ under the KKT conditions, we see that it's determined by only the points $\boldsymbol{x}_i$, $i \in \mathcal{S}$ where $\mathcal{S}$ denotes the set of points which lie on the boundary of the slab. (Hence here, $\widehat{\boldsymbol{\beta}} = \sum_{i \in \mathcal{S}} \widehat{\alpha}_i y_i \boldsymbol{x}_i$)
- The points in $\mathcal{S}$ are called support vectors as they determine the nature of the optimal seperating hyperplane.
- Thus, we obtain the optimal seperating hyperplane which produces a function $\widehat{f}(\boldsymbol{x}) = \widehat{\beta}_0 + \widehat{\boldsymbol{\beta}}^T \boldsymbol{x}$ for classifying a new observation $\boldsymbol{x}$ as :-

$$G(\boldsymbol{x}) = \text{sign } \widehat{f}(\boldsymbol{x})$$

What if the classes are not linearly seperable ?!

- Now, we consider the case when the classes are **not** linearly seperable i.e there is some overlap of the classes in the feature space.

What if the classes are not linearly seperable ?!

- Now, we consider the case when the classes are **not** linearly seperable i.e there is some overlap of the classes in the feature space.
- One way to deal with overlap is to still maximize M but allow for some points to be on the wrong side of the margin.

What if the classes are not linearly seperable ?!

- Now, we consider the case when the classes are **not** linearly seperable i.e there is some overlap of the classes in the feature space.
- One way to deal with overlap is to still maximize M but allow for some points to be on the wrong side of the margin.
- Hence, we define slack variables $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_N)$ corresponding to each of the observations.

What if the classes are not linearly seperable ?!

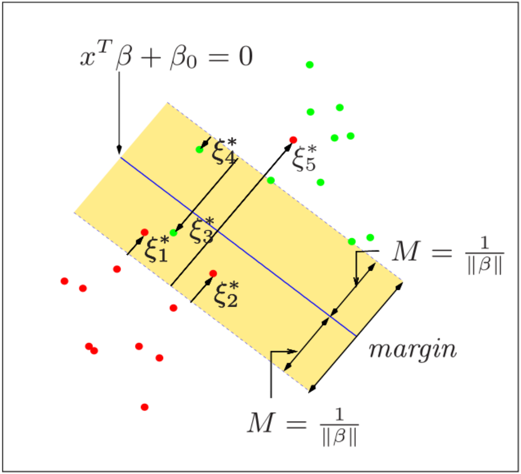

Figure: here the points are not linearly seperable

What if the classes are not linearly seperable ?!

- Then we can modify the optimization problem in two possible ways firstly :-

$$\max_{\beta_0, \boldsymbol{\beta}, ||\boldsymbol{\beta}||=1} M \tag{3.1}$$

$$\text{subjectto } y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq M - \xi_i, i = 1, \ldots, N \tag{3.2}$$

$$\text{also } \xi_i \geq 0 \ \forall i \text{ and } \sum_{i=1}^{N} \xi_i \leq C \tag{3.3}$$

What if the classes are not linearly seperable ?!

- Then we can modify the optimization problem in two possible ways firstly :-

$$\max_{\beta_0,\boldsymbol{\beta},||\boldsymbol{\beta}||=1} M \tag{3.1}$$

$$\text{subjectto } y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right) \geq M-\xi_i, i=1,\ldots,N \tag{3.2}$$

$$\text{also } \xi_i \geq 0 \ \forall i \text{ and } \sum_{i=1}^{N}\xi_i \leq C \tag{3.3}$$

- Or we can write it as :-

$$\max_{\beta_0,\boldsymbol{\beta},||\boldsymbol{\beta}||=1} M \tag{3.4}$$

$$\text{subjectto } y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right) \geq M\left(1-\xi_i\right), i=1,\ldots,N \tag{3.5}$$

$$\text{also } \xi_i \geq 0 \ \forall i \text{ and } \sum_{i=1}^{N}\xi_i \leq C \tag{3.6}$$

What if the classes are not linearly seperable ?!

- Then we can modify the optimization problem in two possible ways firstly :-

$$\max_{\beta_0, \boldsymbol{\beta}, ||\boldsymbol{\beta}||=1} M \tag{3.1}$$

$$\text{subjectto } y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq M - \xi_i, i = 1, \ldots, N \tag{3.2}$$

$$\text{also } \xi_i \geq 0 \ \forall i \text{ and } \sum_{i=1}^{N} \xi_i \leq C \tag{3.3}$$

- Or we can write it as :-

$$\max_{\beta_0, \boldsymbol{\beta}, ||\boldsymbol{\beta}||=1} M \tag{3.4}$$

$$\text{subjectto } y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq M \left(1 - \xi_i \right), i = 1, \ldots, N \tag{3.5}$$

$$\text{also } \xi_i \geq 0 \ \forall i \text{ and } \sum_{i=1}^{N} \xi_i \leq C \tag{3.6}$$

- We generally work with the second choice as the first one results in a nonconvex optimization problem whereas the second one is convex.

Reforming our optimization problem

- As done before we can drop the norm constraint $||\beta|| = 1$, define $M = \frac{1}{||\boldsymbol{\beta}||}$ and rewrite the optimization problem as :-

$$\min_{\beta_0, \boldsymbol{\beta}} ||\boldsymbol{\beta}|| \tag{3.7}$$

$$\text{subjectto } \begin{cases} y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq 1 - \xi_i & , i = 1, \dots, N \\ \xi_i \geq 0, & \sum_{i=1}^{N} \xi_i \leq K \end{cases} \tag{3.8}$$

we note that whenever $\xi_i > 1$ for some i, the corresponding observation is misclassified. Hence, bounding $\sum\limits_{i=1}^{N} \xi_i$ at value K means bounding total number of misclassification at K.

Reforming our optimization problem

- As done before we can drop the norm constraint $||\boldsymbol{\beta}|| = 1$, define $M = \frac{1}{||\boldsymbol{\beta}||}$ and rewrite the optimization problem as :-

$$\min_{\beta_0, \boldsymbol{\beta}} ||\boldsymbol{\beta}|| \tag{3.7}$$

$$\text{subjectto} \begin{cases} y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq 1 - \xi_i & , i = 1, \ldots, N \\ \xi_i \geq 0, & \sum_{i=1}^{N} \xi_i \leq K \end{cases} \tag{3.8}$$

  we note that whenever $\xi_i > 1$ for some i, the corresponding observation is misclassified. Hence, bounding $\sum_{i=1}^{N} \xi_i$ at value K means bounding total number of misclassification at K.

- We similarly, re-express the above problem in the more convinient & equivalent form :-

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2} ||\boldsymbol{\beta}||^2 + C \sum_{i=1}^{N} \xi_i \tag{3.9}$$

$$\text{subject to} \xi_i \geq 0, y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) \geq 1 - \xi_i, \forall i \tag{3.10}$$

  where the constant C is called the "cost" of missclassification and seperable case corresponds to $C = \infty$.

Lagrange Primal Function

- For this formulation, we can write the Lagrange primal function as :-

$$L_P\left(\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}\right) =$$

$$\frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) - (1 - \xi_i)\right] - \sum_{i=1}^{N}\mu_i\xi_i$$

which is minimized w.r.t $\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}$. By setting the respective derivatives equal to zero, we get :-

$$\boldsymbol{\beta} = \sum_{i=1}^{N}\alpha_i y_i \boldsymbol{x}_i \tag{3.11}$$

$$0 = \sum_{i=1}^{N}\alpha_i y_i \tag{3.12}$$

$$C - \mu_i = \alpha_i \ \forall i \tag{3.13}$$

Dual Problem

- Putting the constraints we get the corresponding dual objective function :-

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \tag{3.14}$$

which is to be maximized under the KKT conditions :-

$$\text{primal constraints :} \begin{cases} y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) - (1 - \xi_i) \geq 0 \\ \xi_i \geq 0 \ \forall \ i = 1(1)N. \end{cases} \tag{3.15}$$

$$\text{dual constraints :} \ \alpha_i, \mu_i \geq 0 \ \forall \ i = 1(1)N. \tag{3.16}$$

$$\text{complementary slackness :} \begin{cases} \alpha_i \left[y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) - (1 - \xi_i) \right] = 0 \\ \mu_i \xi_i = 0 \ \forall \ i = 1(1)N. \end{cases} \tag{3.17}$$

$$\text{zero gradient wrt } (\boldsymbol{\beta_0}, \boldsymbol{\beta}, \boldsymbol{\xi}) \begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0, \boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \\ \alpha_i = C - \mu_i \ \forall \ i = 1(1)N. \end{cases} \tag{3.18}$$

Classifier depends on support points only

- Here, only the points which satisfy the condition $y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right)-(1-\xi_i)=0$, determine the fitted value of $\boldsymbol{\beta}$. These ponits are called the support vectors. Among these support vectors, some lie on the edge of the margin $\left(\widehat{\xi}_i=0\right)$, for them we get $0<\widehat{\alpha}_i<C$ and for the remainder $\left(\widehat{\xi}_i>0\right)$ have $\widehat{\alpha}_i=C$. The points on the margin can be used to solve for β_0.

Classifier depends on support points only

- Here, only the points which satisfy the condition $y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0\right)-(1-\xi_i)=0$, determine the fitted value of $\boldsymbol{\beta}$. These ponits are called the support vectors. Among these support vectors, some lie on the edge of the margin $\left(\widehat{\xi}_i=0\right)$, for them we get $0<\widehat{\alpha}_i<C$ and for the remainder $\left(\widehat{\xi}_i>0\right)$ have $\widehat{\alpha}_i=C$. The points on the margin can be used to solve for β_0.
- Maximizing the dual (12.13) is a simpler convex quadratic programming problem than the primal , and can be solved with standard techniques (Murray et al., 1981, for example).

Going beyond "linear" boundaries

- The support vector classifier described so far creates linear boundaries in the input feature space which may or maynot seperate the training data completely.

Going beyond “linear” boundaries

- The support vector classifier described so far creates linear boundaries in the input feature space which may or maynot seperate the training data completely.
- Hence, we can instead fit non-linear boundaries which will classify the training observations better and hence would also be expected to perform better as classifiers.

Going beyond "linear" boundaries

- The support vector classifier described so far creates linear boundaries in the input feature space which may or maynot seperate the training data completely.
- Hence, we can instead fit non-linear boundaries which will classify the training observations better and hence would also be expected to perform better as classifiers.
- We fit the SV classifier using input features $\boldsymbol{h}(\boldsymbol{x}_i) = (h_1(\boldsymbol{x}_i), h_2(\boldsymbol{x}_i), \ldots, h_M(\boldsymbol{x}_i))$ for $i = 1, 2, \ldots, N$.

Going beyond "linear" boundaries

- The support vector classifier described so far creates linear boundaries in the input feature space which may or maynot seperate the training data completely.
- Hence, we can instead fit non-linear boundaries which will classify the training observations better and hence would also be expected to perform better as classifiers.
- We fit the SV classifier using input features $\boldsymbol{h}(\boldsymbol{x}_i) = (h_1(\boldsymbol{x}_i), h_2(\boldsymbol{x}_i), \ldots, h_M(\boldsymbol{x}_i))$ for $i = 1, 2, \ldots, N$.
- Then fit a linear seperation hyperplane in this enlarged input feature space which if converted back to the original space, gives a non-linear seperation boundary.

Going beyond "linear" boundaries

- The support vector classifier described so far creates linear boundaries in the input feature space which may or maynot seperate the training data completely.
- Hence, we can instead fit non-linear boundaries which will classify the training observations better and hence would also be expected to perform better as classifiers.
- We fit the SV classifier using input features $\boldsymbol{h}(\boldsymbol{x}_i) = (h_1(\boldsymbol{x}_i), h_2(\boldsymbol{x}_i), \ldots, h_M(\boldsymbol{x}_i))$ for $i = 1, 2, \ldots, N$.
- Then fit a linear seperation hyperplane in this enlarged input feature space which if converted back to the original space, gives a non-linear seperation boundary.
- We demonstrate the idea using an example.

Example

- Consider this scatterplot where the points are colored according to their class with two features x and y.
- We can clearly see that they are not linearly seperable.

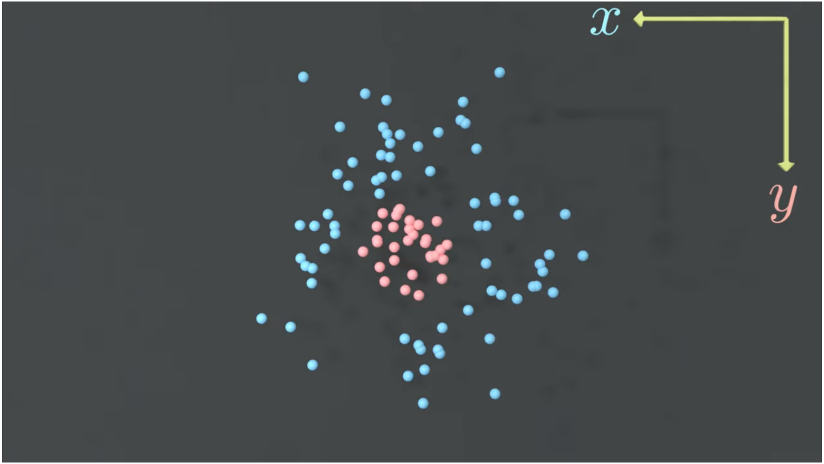

Figure: Here also the classes are not linearly seperable.

Example

- But if we rather transform the points into an enlarged space as $\boldsymbol{h}:\begin{pmatrix} x_i \\ y_i \end{pmatrix} \to \begin{pmatrix} x_i \\ y_i \\ x_i^2 + y_i^2 \end{pmatrix}$, then we can easily draw a seperating hyperplane in this 3 dimensional feature space.

Example

- But if we rather transform the points into an enlarged space as $\boldsymbol{h} : \begin{pmatrix} x_i \\ y_i \end{pmatrix} \to \begin{pmatrix} x_i \\ y_i \\ x_i^2 + y_i^2 \end{pmatrix}$, then we can easily draw a seperating hyperplane in this 3 dimensional feature space.
- Using the same optimization technique, we will fit an optimal seperating hyperplane $\widehat{f^*}(x, y, z) = \widehat{\beta}_1 x + \widehat{\beta}_2 y + \widehat{\beta}_3 z + \widehat{\beta}_0$ for any point $(x, y, z)^T \in \mathbb{R}^3$ but if we get back to the original feature space by putting $z = x^2 + y^2$ and write the original fitted function as $\widehat{f}(x, y) = \widehat{\beta}_0 + \widehat{\beta}_1 x + \widehat{\beta}_2 y + \widehat{\beta}_3 (x^2 + y^2)$ and hence the classifier $G(x, y) = \text{Sign} \widehat{f}(x, y)$.

Example

- But if we rather transform the points into an enlarged space as $\boldsymbol{h} : \begin{pmatrix} x_i \\ y_i \end{pmatrix} \to \begin{pmatrix} x_i \\ y_i \\ x_i^2 + y_i^2 \end{pmatrix}$, then we can easily draw a seperating hyperplane in this 3 dimensional feature space.
- Using the same optimization technique, we will fit an optimal seperating hyperplane $\widehat{f^*}(x, y, z) = \widehat{\beta}_1 x + \widehat{\beta}_2 y + \widehat{\beta}_3 z + \widehat{\beta}_0$ for any point $(x, y, z)^T \in \mathbb{R}^3$ but if we get back to the original feature space by putting $z = x^2 + y^2$ and write the original fitted function as $\widehat{f}(x, y) = \widehat{\beta}_0 + \widehat{\beta}_1 x + \widehat{\beta}_2 y + \widehat{\beta}_3 (x^2 + y^2)$ and hence the classifier $G(x, y) = \text{Sign}\widehat{f}(x, y)$.
- Then this classifier creates a non-linear classification boundary on the original feature space.

Example

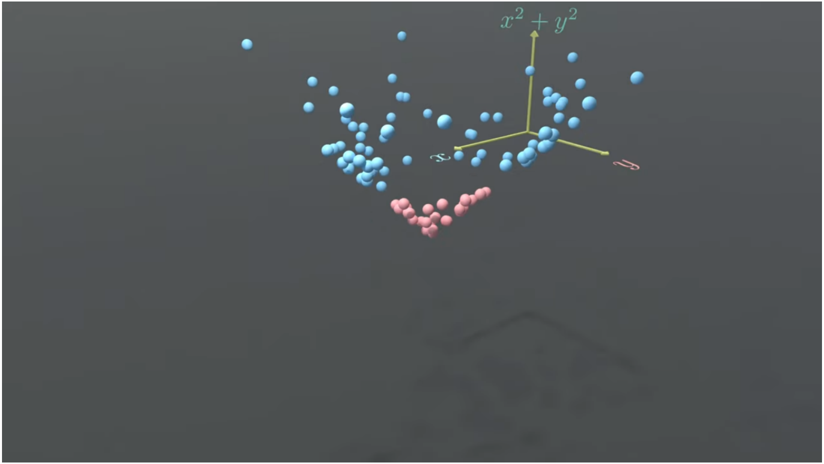

Figure: So we transform them!

Example

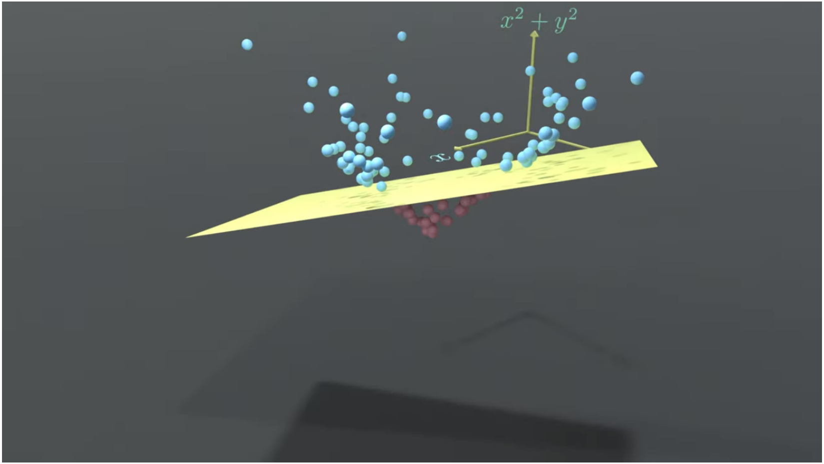

Figure: And now they are linearly seperable!

Example

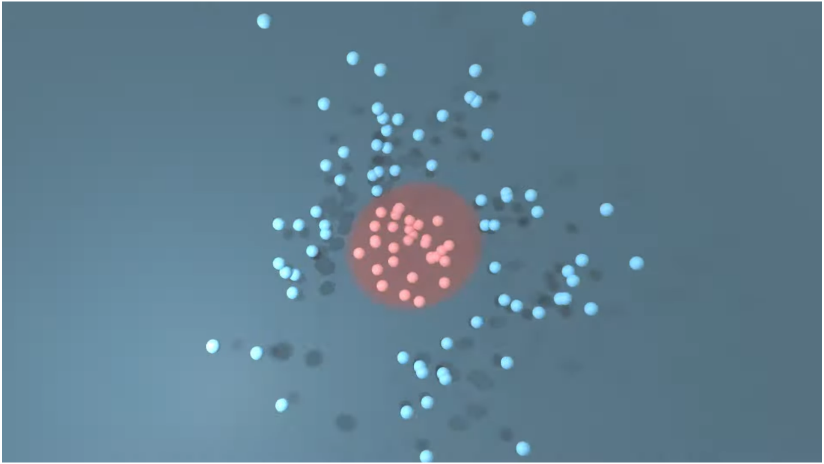

Figure: After the job is done, we get back to original feature space!

Some “comments”

- So far, we have considered the “dual” problem as our optimization problem due to many advantages as it explains why the solutions depend only on the support points and much more.

Some “comments”

- So far, we have considered the “dual” problem as our optimization problem due to many advantages as it explains why the solutions depend only on the support points and much more.
- But, in terms of optimization, sometimes, it can be more efficient to solve the primal problem than the dual one.

Some "comments"

- So far, we have considered the "dual" problem as our optimization problem due to many advantages as it explains why the solutions depend only on the support points and much more.
- But, in terms of optimization, sometimes, it can be more efficient to solve the primal problem than the dual one.
- Note that we can write the optimization problem :-

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2 + C \sum_{i=1}^{N} \xi_i \tag{4.1}$$

$$\text{subject to } \xi_i \geq 0, y_i f\left(\boldsymbol{x}_i\right) \geq 1 - \xi_i, \forall i \tag{4.2}$$

Some “comments”

- So far, we have considered the “dual” problem as our optimization problem due to many advantages as it explains why the solutions depend only on the support points and much more.
- But, in terms of optimization, sometimes, it can be more efficient to solve the primal problem than the dual one.
- Note that we can write the optimization problem :-

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N} \xi_i \tag{4.1}$$

$$\text{subject to } \xi_i \geq 0, y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i, \forall i \tag{4.2}$$

- As optimizing a more familiar objective function of the form loss+penalty (commonly used in statisical literature) :-

$$\min_{\beta_0, \boldsymbol{\beta}} C\sum_{i=1}^{N} L(y_i, f(\boldsymbol{x}_i)) + \frac{1}{2}||\boldsymbol{\beta}||^2$$

where, L is the “hinge” loss function defined as $L(y_i, f(\boldsymbol{x}_i)) = \max\{0, 1 - y_i f(\boldsymbol{x}_i)\}$.

Some "comments"

- It can be shown that this is also a convex objective function which can be optimized using standard techniques.

Some "comments"

- It can be shown that this is also a convex objective function which can be optimized using standard techniques.
- On the other hand, as we saw earlier, the dual problem invloves N parameters α_i where as the primal problem involves d many parameters $\beta_0, \boldsymbol{\beta}$. Hence, if $d < N$, then it is more efficient to solve the primal problem than the dual one.

Some "comments"

- It can be shown that this is also a convex objective function which can be optimized using standard techniques.
- On the other hand, as we saw earlier, the dual problem invloves N parameters α_i where as the primal problem involves d many parameters $\beta_0, \boldsymbol{\beta}$. Hence, if $d < N$, then it is more efficient to solve the primal problem than the dual one.
- But now, we will see that sometimes $N \ll d$ where the dimension of the problem gets (or made) very large, then its better to work with the dual problem.

Some "comments"

- It can be shown that this is also a convex objective function which can be optimized using standard techniques.
- On the other hand, as we saw earlier, the dual problem invloves N parameters α_i where as the primal problem involves d many parameters $\beta_0, \boldsymbol{\beta}$. Hence, if $d < N$, then it is more efficient to solve the primal problem than the dual one.
- But now, we will see that sometimes $N \ll d$ where the dimension of the problem gets (or made) very large, then its better to work with the dual problem.
- Also, one main thing to note is that, the dual form involves the feature inputs only in the form of dot products $(\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)$ which gives the main motivation in creating the "best" version of SV classifiers, formerly known as the Support Vector Machines.

Some “comments”

- It can be shown that this is also a convex objective function which can be optimized using standard techniques.
- On the other hand, as we saw earlier, the dual problem invloves N parameters α_i where as the primal problem involves d many parameters $\beta_0, \boldsymbol{\beta}$. Hence, if $d < N$, then it is more efficient to solve the primal problem than the dual one.
- But now, we will see that sometimes $N \ll d$ where the dimension of the problem gets (or made) very large, then its better to work with the dual problem.
- Also, one main thing to note is that, the dual form involves the feature inputs only in the form of dot products $(\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle)$ which gives the main motivation in creating the “best” version of SV classifiers, formerly known as the Support Vector Machines.
- We explain the idea clearly in the next few slides.

Support Vector "Machines"

- The **support vector machine** classifier is an extension of this idea, where the dimension of the enlarged space is allowed to get very large, infinite in some cases.

Support Vector "Machines"

- The **support vector machine** classifier is an extension of this idea, where the dimension of the enlarged space is allowed to get very large, infinite in some cases.
- We can represent the original optimization problem (3.14) and its solution in a special way that only involves the input features via inner products.

Support Vector "Machines"

- The **support vector machine** classifier is an extension of this idea, where the dimension of the enlarged space is allowed to get very large, infinite in some cases.
- We can represent the original optimization problem (3.14) and its solution in a special way that only involves the input features via inner products.
- We do this directly for the transformed feature vectors $\boldsymbol{h}(\boldsymbol{x}_i)$.

Support Vector "Machines"

- The **support vector machine** classifier is an extension of this idea, where the dimension of the enlarged space is allowed to get very large, infinite in some cases.
- We can represent the original optimization problem (3.14) and its solution in a special way that only involves the input features via inner products.
- We do this directly for the transformed feature vectors $\boldsymbol{h}(\boldsymbol{x}_i)$.
- We then see that for particular choices of $\boldsymbol{h}$, these inner products can be computed very cheaply.

Support Vector "Machines"

- Here, our optimization problem is of a similar form :-

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2 + C \sum_{i=1}^{N} \xi_i \tag{4.3}$$

$$\text{subject to } \xi_i \geq 0, y_i \left(\boldsymbol{h} \left(\boldsymbol{x}_i \right)^T \boldsymbol{\beta} + \beta_0 \right) \geq 1 - \xi_i, \forall i \tag{4.4}$$

Support Vector "Machines"

- Here, our optimization problem is of a similar form :-

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N}\xi_i \tag{4.3}$$

$$\text{subject to } \xi_i \geq 0, y_i\left(\boldsymbol{h}\left(\boldsymbol{x}_i\right)^T\boldsymbol{\beta} + \beta_0\right) \geq 1 - \xi_i, \forall i \tag{4.4}$$

- which becomes equivalent to minimizing the lagrange dual function :-

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left\langle \boldsymbol{h}\left(\boldsymbol{x}_i\right), \boldsymbol{h}\left(\boldsymbol{x}_j\right)\right\rangle \tag{4.5}$$

under the KKT conditions.

Support Vector "Machines"

- Here, our optimization problem is of a similar form :-

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N}\xi_i \tag{4.3}$$

$$\text{subject to } \xi_i \geq 0, y_i\left(\boldsymbol{h}\left(\boldsymbol{x}_i\right)^T\boldsymbol{\beta} + \beta_0\right) \geq 1 - \xi_i, \forall i \tag{4.4}$$

- which becomes equivalent to minimizing the lagrange dual function :-

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \left\langle \boldsymbol{h}\left(\boldsymbol{x}_i\right), \boldsymbol{h}\left(\boldsymbol{x}_j\right)\right\rangle \tag{4.5}$$

  under the KKT conditions.
- Note that we are writing the dual function using inner products of feature space only instead of explicitly writing $\boldsymbol{\beta}$.

Support Vector "Machines"

- In fact, the solution function $f(\boldsymbol{x})$ can also be written more conviniently as :-

$$\widehat{f}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})^T \widehat{\boldsymbol{\beta}} + \widehat{\beta}_0 \tag{4.6}$$

$$= \sum_{i=1}^{N} \widehat{\alpha}_i y_i \langle \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}_i) \rangle + \widehat{\beta}_0 \tag{4.7}$$

with similar constraints as before.

Support Vector "Machines"

- In fact, the solution function $f(\boldsymbol{x})$ can also be written more conviniently as :-

$$\widehat{f}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})^T \widehat{\boldsymbol{\beta}} + \widehat{\beta}_0 \tag{4.6}$$

$$= \sum_{i=1}^{N} \widehat{\alpha}_i y_i \langle \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}_i) \rangle + \widehat{\beta}_0 \tag{4.7}$$

  with similar constraints as before.
- The ingenious idea behind using kernels is that, we don't need to specify the transformation $\boldsymbol{h}(\boldsymbol{x})$ at all, but require only the form of the inner products $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}') \rangle$. Only, we need that the function K is positive definite.

Support Vector “Machines”

- In fact, the solution function $f\left(\boldsymbol{x}\right)$ can also be written more conviniently as :-

$$\widehat{f}\left(\boldsymbol{x}\right) = \boldsymbol{h}\left(\boldsymbol{x}\right)^{T}\widehat{\boldsymbol{\beta}} + \widehat{\beta}_0 \tag{4.6}$$

$$= \sum_{i=1}^{N} \widehat{\alpha}_i y_i \left\langle \boldsymbol{h}\left(\boldsymbol{x}\right), \boldsymbol{h}\left(\boldsymbol{x}_i\right)\right\rangle + \widehat{\beta}_0 \tag{4.7}$$

  with similar constraints as before.
- The ingenious idea behind using kernels is that, we don't need to specify the transformation $\boldsymbol{h}\left(\boldsymbol{x}\right)$ at all, but require only the form of the inner products $K\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \left\langle \boldsymbol{h}\left(\boldsymbol{x}\right), \boldsymbol{h}\left(\boldsymbol{x}'\right)\right\rangle$. Only, we need that the function K is positive definite.
- The three most popular choices of K are :-

$$d\text{th degree polynomial:} K\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \left(1 + \left\langle \boldsymbol{x}, \boldsymbol{x}'\right\rangle\right)^{d} \tag{4.8}$$

$$\text{Radial Basis:} K\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \exp\left(-\gamma ||\boldsymbol{x} - \boldsymbol{x}'||^2\right) \tag{4.9}$$

$$\text{Neural Network:} K\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \tanh\left(\kappa_1 \left\langle \boldsymbol{x}, \boldsymbol{x}'\right\rangle + \kappa_2\right) \tag{4.10}$$

$$\text{where } \tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{4.11}$$

Relation of Kernel and Feature Space

- We give an example to illustrate the relation between inner product and input feature space. Suppose we consider a classification problem with two covariates x_1 and x_2, then the inner product for the choice $K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2$ where $d = 2$ can be written as

$$
\begin{aligned}
(1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 &= (1 + x_1 x_1' + x_2 x_2')^2 \\
&= 1 + 2(x_1 x_1') + 2(x_2 x_2') + \\
&2(x_1 x_2)(x_1' x_2') + \left(x_1^2 x_1'^2\right) + \left(x_2^2 x_2'^2\right) \\
&= \langle \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}') \rangle
\end{aligned}
$$

where, $\boldsymbol{h}(\boldsymbol{x}) = \begin{pmatrix} 1 & \sqrt{2}x_1 & \sqrt{2}x_2 & \sqrt{2}x_1 x_2 & x_1^2 & x_2^2 \end{pmatrix}$ is a $6-$dimensional input feature space.

Relation of Kernel and Feature Space

- We give an example to illustrate the relation between inner product and input feature space. Suppose we consider a classification problem with two covariates x_1 and x_2, then the inner product for the choice $K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2$ where $d = 2$ can be written as

$$\begin{aligned} (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 &= (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2(x_1 x_1') + 2(x_2 x_2') + \\ & 2(x_1 x_2)(x_1' x_2') + \left(x_1^2 x_1'^2\right) + \left(x_2^2 x_2'^2\right) \\ &= \langle \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}') \rangle \end{aligned}$$

  where, $\boldsymbol{h}(\boldsymbol{x}) = \begin{pmatrix} 1 & \sqrt{2}x_1 & \sqrt{2}x_2 & \sqrt{2}x_1 x_2 & x_1^2 & x_2^2 \end{pmatrix}$ is a $6-$dimensional input feature space.
- Similarly for other two choices of K, the input feature space is infinite dimensional.

Interpreting the Radial Kernel

- The radial kernel is of the form

$$K\left(\boldsymbol{x}_i, \boldsymbol{x}_i'\right) = \exp\left(-\gamma ||\boldsymbol{x}_i - \boldsymbol{x}_i'||^2\right) \tag{4.12}$$

$$= \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij} - x_{ij}'\right)^2\right) \tag{4.13}$$

and we also know that the fitted classifier can be written in terms of the kernel function as :-

$$\widehat{f}\left(\boldsymbol{x}\right) = \widehat{\beta}_0 + \sum_{i=1}^{N} \widehat{\alpha}_i y_i K\left(\boldsymbol{x}, \boldsymbol{x}_i\right)$$

Interpreting the Radial Kernel

- Hence, if a test observation $\boldsymbol{x}^*$ is far from a training observation $\boldsymbol{x}_i$ in terms of Euclidean distance, then $\sum_{j=1}^{p}\left(x_{ij}^* - x_{ij}\right)^2$ will be large, and so $K\left(\boldsymbol{x}^*, \boldsymbol{x}_i\right) = \exp\left(-\gamma \sum_{j=1}^{p}\left(x_{ij}^* - x_{ij}\right)^2\right)$ will be tiny.

Interpreting the Radial Kernel

- Hence, if a test observation $\boldsymbol{x}^*$ is far from a training observation $\boldsymbol{x}_i$ in terms of Euclidean distance, then $\sum_{j=1}^{p} \left(x_{ij}^* - x_{ij}\right)^2$ will be large, and so $K\left(\boldsymbol{x}^*, \boldsymbol{x}_i\right) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij}^* - x_{ij}\right)^2\right)$ will be tiny.
- This means that in , $\boldsymbol{x}_i$ will play virtually no role in the estimated value $\widehat{f}\left(\boldsymbol{x}^*\right)$.

Interpreting the Radial Kernel

- Hence, if a test observation $\boldsymbol{x}^*$ is far from a training observation $\boldsymbol{x}_i$ in terms of Euclidean distance, then $\sum_{j=1}^{p} \left(x_{ij}^* - x_{ij}\right)^2$ will be large, and so $K\left(\boldsymbol{x}^*, \boldsymbol{x}_i\right) = \exp\left(-\gamma \sum_{j=1}^{p} \left(x_{ij}^* - x_{ij}\right)^2\right)$ will be tiny.
- This means that in , $\boldsymbol{x}_i$ will play virtually no role in the estimated value $\widehat{f}\left(\boldsymbol{x}^*\right)$.
- In other words, training observations that are far from $\boldsymbol{x}^*$ will play essentially no role in the predicted class label for $\boldsymbol{x}^*$.

Regularization

- The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there.

Regularization

- The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there.
- A large value of C will discourage any positive ξ_i , and lead to an overfit wiggly boundary in the original feature space; however a small value of C will encourage the boundary to be smoother and will allow more positive ξ_i values.

Regularization

- The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there.
- A large value of C will discourage any positive ξ_i , and lead to an overfit wiggly boundary in the original feature space; however a small value of C will encourage the boundary to be smoother and will allow more positive ξ_i values.
- The "optimal" value between a set of values of C is determined by choosing the C for which the cross-validation error is minimum.

Regularization

- The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there.
- A large value of C will discourage any positive ξ_i , and lead to an overfit wiggly boundary in the original feature space; however a small value of C will encourage the boundary to be smoother and will allow more positive ξ_i values.
- The "optimal" value between a set of values of C is determined by choosing the C for which the cross-validation error is minimum.
- Similarly, for radial kernel, the optimal value of C depends on the value of γ parameter. For example if the value of γ gets larger, the kernels become more and more narrow peaked and hence the seperation boundaries get much more wiggly. To balance, that C shuold become smaller in order to decrease the test error.

Regularization

- The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there.
- A large value of C will discourage any positive ξ_i , and lead to an overfit wiggly boundary in the original feature space; however a small value of C will encourage the boundary to be smoother and will allow more positive ξ_i values.
- The "optimal" value between a set of values of C is determined by choosing the C for which the cross-validation error is minimum.
- Similarly, for radial kernel, the optimal value of C depends on the value of γ parameter. For example if the value of γ gets larger, the kernels become more and more narrow peaked and hence the seperation boundaries get much more wiggly. To balance, that C shuold become smaller in order to decrease the test error.
- Similar situation arise when we use polynomial kernel. There also, we have to choose the degree d of the kernel.

Regularization

- The role of the parameter C is clearer in an enlarged feature space, since perfect separation is often achievable there.
- A large value of C will discourage any positive ξ_i , and lead to an overfit wiggly boundary in the original feature space; however a small value of C will encourage the boundary to be smoother and will allow more positive ξ_i values.
- The "optimal" value between a set of values of C is determined by choosing the C for which the cross-validation error is minimum.
- Similarly, for radial kernel, the optimal value of C depends on the value of γ parameter. For example if the value of γ gets larger, the kernels become more and more narrow peaked and hence the seperation boundaries get much more wiggly. To balance, that C shuold become smaller in order to decrease the test error.
- Similar situation arise when we use polynomial kernel. There also, we have to choose the degree d of the kernel.
- We explain these using visuals.

Demonstration Using Examples

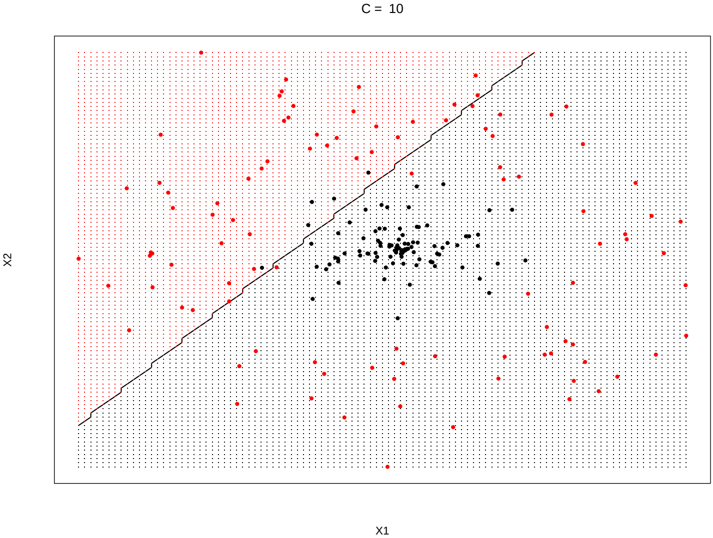

Figure: Fitting a linear classifier with $C = 10$

Demonstration Using Examples

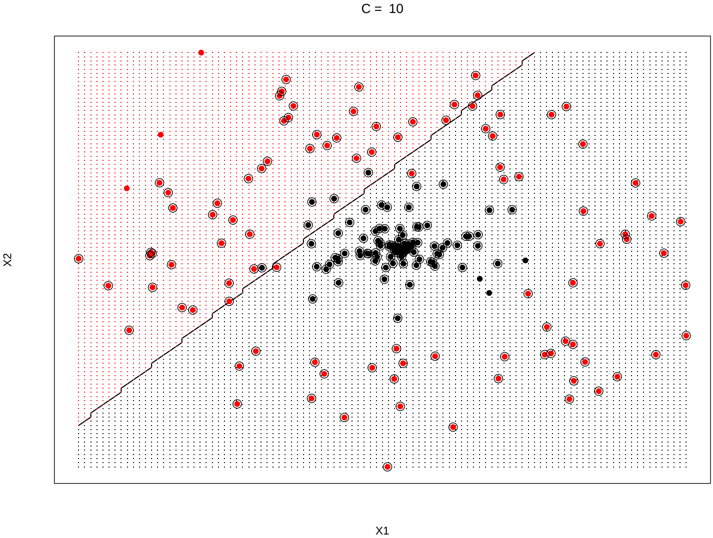

Figure: All the support points are marked.

Demonstration Using Examples

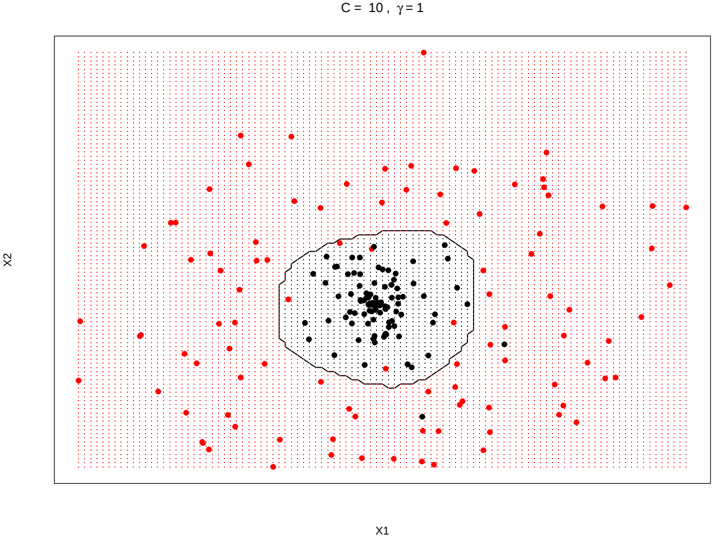

Figure: Then we fit SVM with radial kernel.

Demonstration Using Examples

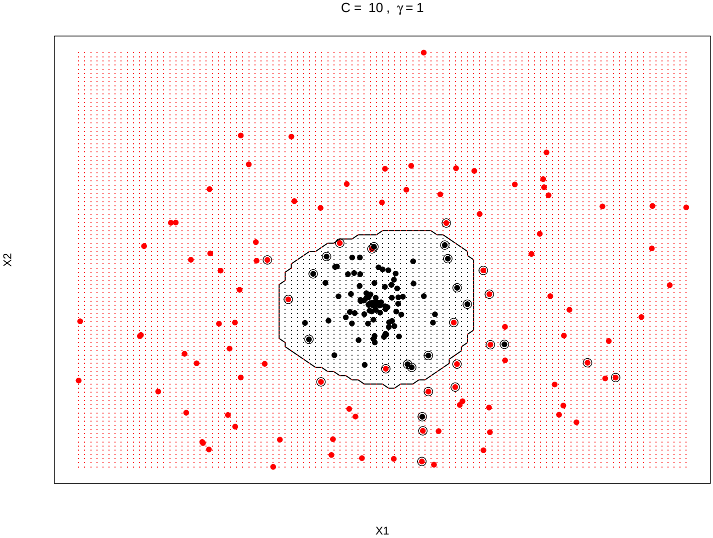

Figure: And these are the support points here.

Demonstration Using Examples

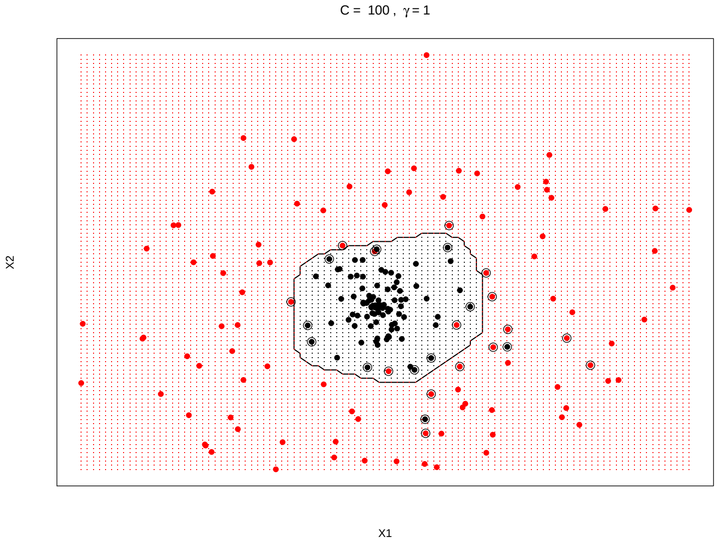

Figure: As we increase the cost, the margin shrinks decreasing the no of support points

Demonstration Using Examples

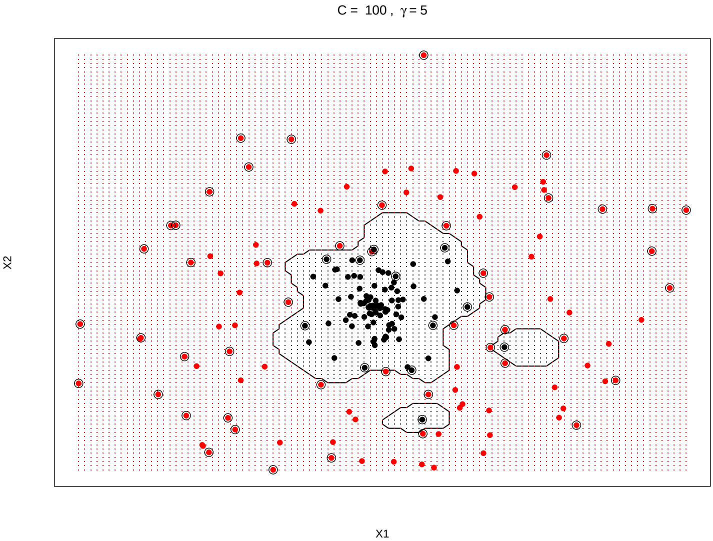

Figure: Increasing the value of γ also overfits the training set.

Demonstration Using Examples

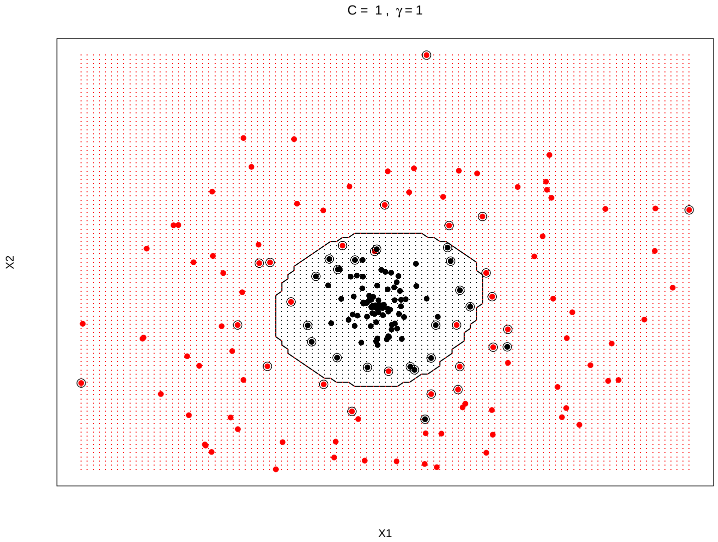

Figure: Optimal choice of C & γ using Cross-validation

Demonstration Using Examples

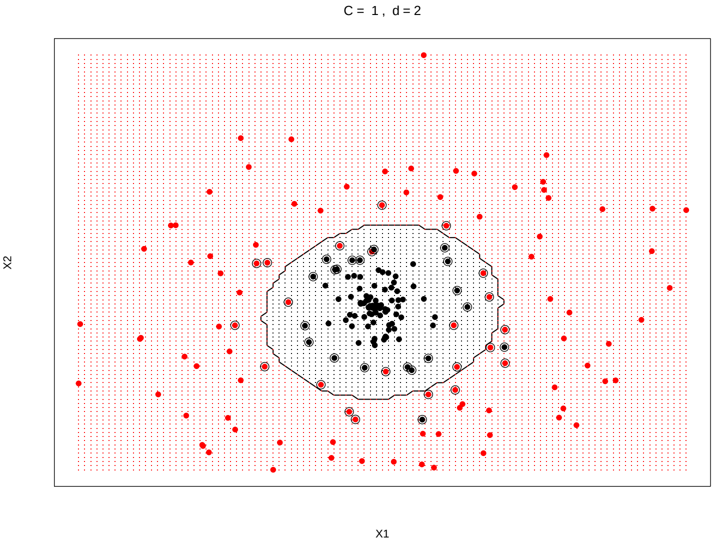

Figure: Polynomial Kernel of 2^{nd} degree

Demonstration Using Examples

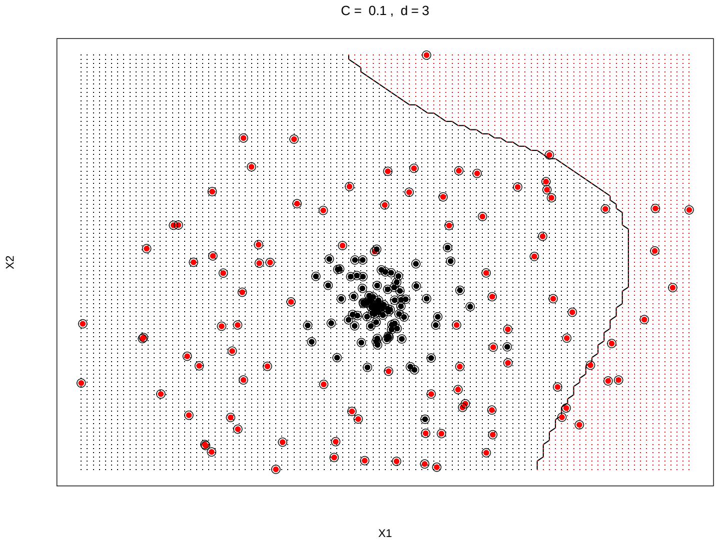

Figure: For degree 3, the polynomial kernel gives a very poor classification!

Demonstration Using Examples

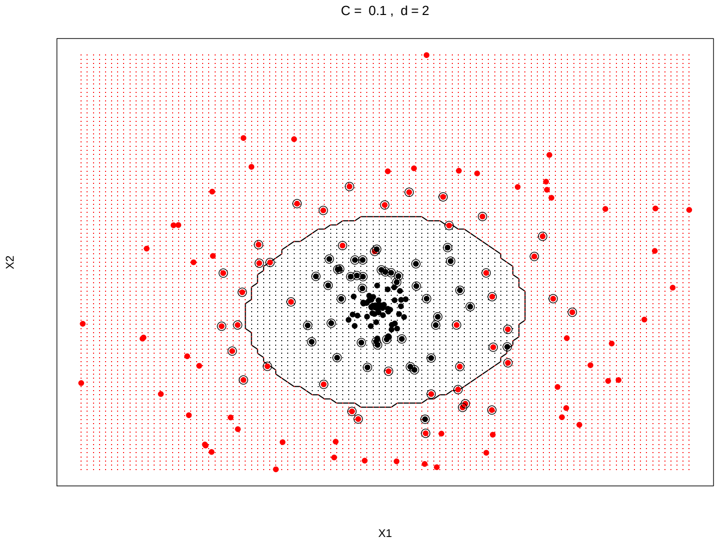

Figure: For optimal value of d and C

SVMs with More than Two Classes

- Our discussion so far, has been limited to the case of binary classification only i.e., classification where total number of classes is equal to 2.

SVMs with More than Two Classes

- Our discussion so far, has been limited to the case of binary classification only i.e., classification where total number of classes is equal to 2.
- Hence, we can't use SVM directly for classification problem with more than two classes.

SVMs with More than Two Classes

- Our discussion so far, has been limited to the case of binary classification only i.e., classification where total number of classes is equal to 2.
- Hence, we can't use SVM directly for classification problem with more than two classes.
- But we can extend the idea behid SVM to more general case where we have any arbitrary number of classes.

SVMs with More than Two Classes

- Our discussion so far, has been limited to the case of binary classification only i.e., classification where total number of classes is equal to 2.
- Hence, we can't use SVM directly for classification problem with more than two classes.
- But we can extend the idea behid SVM to more general case where we have any arbitrary number of classes.
- Though a number of proposals for extending SVMs to the K-class case have been made, the two most popular are the one-versus-one and one-versus-all approaches. We briefly discuss those two approaches here.

One-Versus-One Classification

- Suppose that we would like to perform classification using SVMs, and there are $K > 2$ classes.

One-Versus-One Classification

- Suppose that we would like to perform classification using SVMs, and there are $K > 2$ classes.
- Here, we construct $\binom{K}{2}$ SVMs, each of which compares a pair of classes, say k and k' coded as $+1$ and -1 respectively.

One-Versus-One Classification

- Suppose that we would like to perform classification using SVMs, and there are $K > 2$ classes.
- Here, we construct $\binom{K}{2}$ SVMs, each of which compares a pair of classes, say k and k' coded as $+1$ and -1 respectively.
- We classify a test observation using each of the $\binom{K}{2}$ classifiers and count the number of times the observation is classified into each of the K classes.

One-Versus-One Classification

- Suppose that we would like to perform classification using SVMs, and there are $K > 2$ classes.
- Here, we construct $\binom{K}{2}$ SVMs, each of which compares a pair of classes, say k and k' coded as $+1$ and -1 respectively.
- We classify a test observation using each of the $\binom{K}{2}$ classifiers and count the number of times the observation is classified into each of the K classes.
- Finally, we classify the observation to the class to which it was most frequently assigned in these $\binom{K}{2}$ pairwise classifications.

One-Versus-All Classification

- We fit K SVMs each time comparing one of the K classes to the remaining $K - 1$ classes.

One-Versus-All Classification

- We fit K SVMs each time comparing one of the K classes to the remaining $K-1$ classes.
- Suppose we denote the estimated parameters as $\widehat{\beta}_{0k}, \widehat{\beta}_{1k}, \ldots, \widehat{\beta}_{pk}$ that result from fitting an SVM comparing the K^{th} class (coded as $+1$) to the other $K-1$ classes (all coded as -1).

One-Versus-All Classification

- We fit K SVMs each time comparing one of the K classes to the remaining $K-1$ classes.
- Suppose we denote the estimated parameters as $\widehat{\beta}_{0k}, \widehat{\beta}_{1k}, \ldots, \widehat{\beta}_{pk}$ that result from fitting an SVM comparing the K^{th} class (coded as $+1$) to the other $K-1$ classes (all coded as -1).
- Then for a test observation $\boldsymbol{x}^*$, we assign it to the class for which $\widehat{f}(\boldsymbol{x}^*) = \widehat{\beta}_{0k} + \sum_{i=1}^{p} \widehat{\beta}_{ik} x_i^*$ is largest.

The KNN Classifier

- Next, we give a brief introduction of the KNN classifier.

The KNN Classifier

- Next, we give a brief introduction of the KNN classifier.
- Suppose, we have a similar setup and dataset consisting of N pairs $(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_N}, y_N)$, with the covariates or input features $\boldsymbol{x_i} \in \mathbb{R}^p$ and each of the observations being classified to say g categories i.e. $y_i \in \{1, 2, \ldots, g\}$ for all $i = 1, 2, \ldots, N$.

The KNN Classifier

- Next, we give a brief introduction of the KNN classifier.
- Suppose, we have a similar setup and dataset consisting of N pairs $(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_N}, y_N)$, with the covariates or input features $\boldsymbol{x_i} \in \mathbb{R}^p$ and each of the observations being classified to say g categories i.e. $y_i \in \{1, 2, \ldots, g\}$ for all $i = 1, 2, \ldots, N$.
- Then, the $k-$Nearest Neighbor method use the observations in the training set to classify a new observation $\boldsymbol{x}$. To be specific, it predicts the probability that the observation belongs to the class j, as :-

$$\mathsf{Pr}\left(y\left(\boldsymbol{x}\right) \in j\right) = \frac{1}{k} \sum_{i \in N_k(\boldsymbol{x})} I\left(y_i = j\right)$$

  where, $N_k\left(\boldsymbol{x}\right)$ denotes the set of K closest training observation wrt to some choice of metric. (commonly Euclidean distance).Then the point $\boldsymbol{x}$ is classified to the class j^* which has the largest value of the probabiliy.

The KNN Classifier

- Hence, if we have $g = 2$ and we take the value of k to be 1 then the $1-$NN classifier will essentially classify each observation $\boldsymbol{x}$ to the class y_i corresponding to the point $\boldsymbol{x}_i$ that is closest in the feature space to $\boldsymbol{x}$.

The KNN Classifier

- Hence, if we have $g = 2$ and we take the value of k to be 1 then the $1-$NN classifier will essentially classify each observation $\boldsymbol{x}$ to the class y_i corresponding to the point $\boldsymbol{x}_i$ that is closest in the feature space to $\boldsymbol{x}$.
- We draw the classification boundary using $1-$NN classifier for the same dataset used before in SVM :-

The KNN Classifier

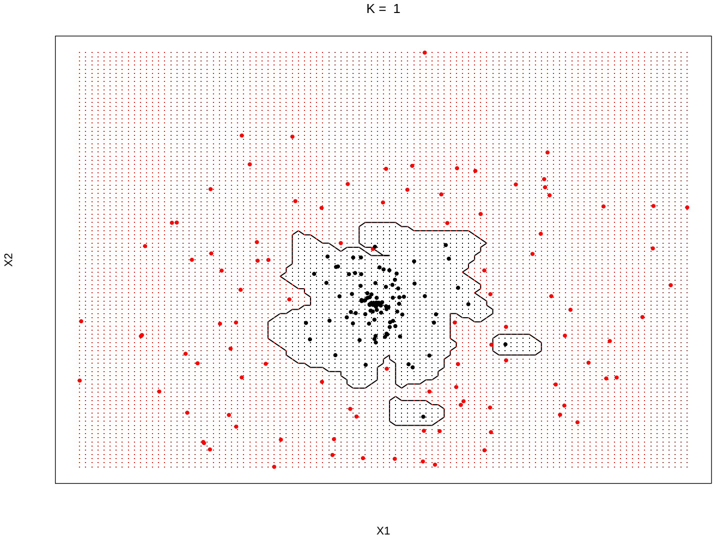

Figure: The KNN Classifier creates a wiggly boundary for $K = 1$

The KNN Classifier

- As we can see that the 1-NN classifier creates a wiggly boundary, we increase the value of k to get a smoother boundary for example for $k = 20$ we get :-

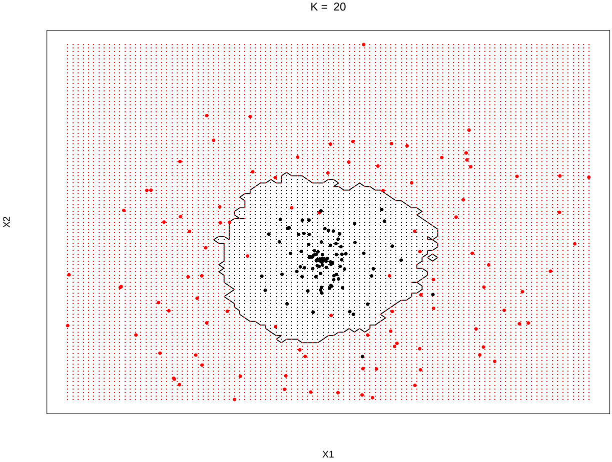

Figure: The boundary is smoother for $K = 20$

The KNN Classifier

- So in order to find an optimal value of k, we choose a set of values of k in the range $1:90$ and for each of them we plot the average test error for 5 validation sets that the data is split into.

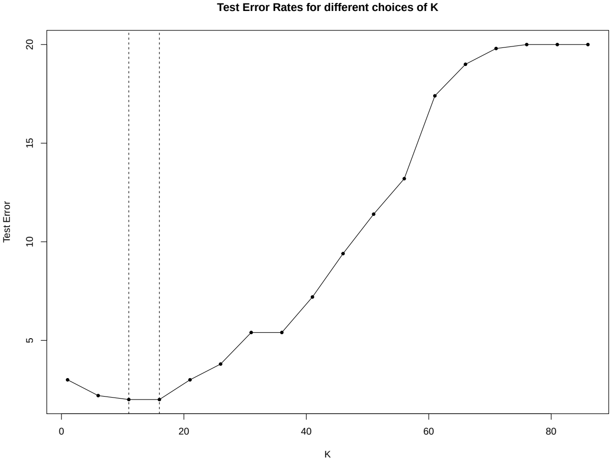

Figure:

Comparison of SVM with KNN, LDA & QDA

- In order to compare SVM, KNN and LDA as classification algorithms we simulate some artificial examples and then compare them empirically.

Comparison of SVM with KNN, LDA & QDA

- In order to compare SVM, KNN and LDA as classification algorithms we simulate some artificial examples and then compare them empirically.
- First we simulate a dataset where we have p input variables $X_1, X_2, \ldots, X_p$ and $n = 200$ observations where half of the observations belong to a class labeled as -1 and the other half as $+1$.

Comparison of SVM with KNN, LDA & QDA

- In order to compare SVM, KNN and LDA as classification algorithms we simulate some artificial examples and then compare them empirically.
- First we simulate a dataset where we have p input variables $X_1, X_2, \ldots, X_p$ and $n = 200$ observations where half of the observations belong to a class labeled as -1 and the other half as $+1$.
- Then we apply the different classification techniques to compare between their performance.

Comparison of SVM with KNN, LDA & QDA

- In order to compare SVM, KNN and LDA as classification algorithms we simulate some artificial examples and then compare them empirically.
- First we simulate a dataset where we have p input variables $X_1, X_2, \ldots, X_p$ and $n = 200$ observations where half of the observations belong to a class labeled as -1 and the other half as $+1$.
- Then we apply the different classification techniques to compare between their performance.
- In our first example, we classify first 100 random samples from $N_p\left(\boldsymbol{\mu}_1^{p\times 1}, \boldsymbol{\Sigma}_1^{p\times p}\right)$ where $\boldsymbol{\mu}_1^{p\times 1} = 5\mathbf{1}_p$ and $\boldsymbol{\Sigma}_1^{p\times p} = \Sigma^{p\times p} = \text{diag}\left(1, 2, \ldots, p\right)$, classify them as -1 and second 100 random samples from $N_p\left(\boldsymbol{\mu}_2^{p\times 1}, \boldsymbol{\Sigma}_2^{p\times p}\right)$ where $\boldsymbol{\mu}_2^{p\times 1} = 9\mathbf{1}_p$ and $\boldsymbol{\Sigma}_2^{p\times p} = 4\Sigma^{p\times p}$ which we classify as $+1$.

Comparison of SVM with KNN, LDA & QDA

- In order to compare SVM, KNN and LDA as classification algorithms we simulate some artificial examples and then compare them empirically.
- First we simulate a dataset where we have p input variables $X_1, X_2, \ldots, X_p$ and $n = 200$ observations where half of the observations belong to a class labeled as -1 and the other half as $+1$.
- Then we apply the different classification techniques to compare between their performance.
- In our first example, we classify first 100 random samples from $N_p\left(\boldsymbol{\mu}_1^{p\times 1}, \boldsymbol{\Sigma}_1^{p\times p}\right)$ where $\boldsymbol{\mu}_1^{p\times 1} = 5\mathbf{1}_p$ and $\boldsymbol{\Sigma}_1^{p\times p} = \Sigma^{p\times p} = \text{diag}\,(1, 2, \ldots, p)$, classify them as -1 and second 100 random samples from $N_p\left(\boldsymbol{\mu}_2^{p\times 1}, \boldsymbol{\Sigma}_2^{p\times p}\right)$ where $\boldsymbol{\mu}_2^{p\times 1} = 9\mathbf{1}_p$ and $\boldsymbol{\Sigma}_2^{p\times p} = 4\Sigma^{p\times p}$ which we classify as $+1$.
- Then we vary the value of p from 5 to 50 and for each value of p, we simulate observations from both classes and then fit a classifier using different methods and calculate the total number of misclassifications for each of the methods at each values of p to compare.

Comparison of SVM with KNN, LDA & QDA

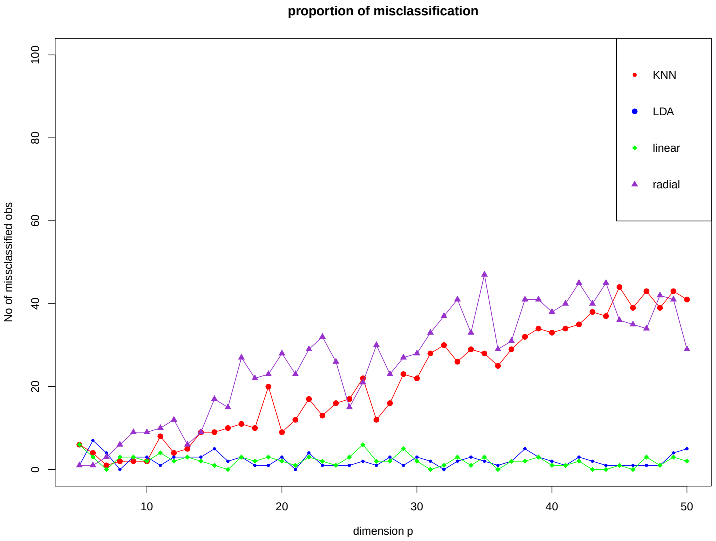

Figure: Test errors for different values of p for different classifiers

Comparison of SVM with KNN, LDA & QDA

- As we see KNN does not perform well as the dimension of the problem increases which is quite expected.

Comparison of SVM with KNN, LDA & QDA

- As we see KNN does not perform well as the dimension of the problem increases which is quite expected.
- Rather LDA and Support Vector Classifier (linear classifier with soft margin) are the best performing classifiers here.

Comparison of SVM with KNN, LDA & QDA

- As we see KNN does not perform well as the dimension of the problem increases which is quite expected.
- Rather LDA and Support Vector Classifier (linear classifier with soft margin) are the best performing classifiers here.
- We see that the observations in the feature space are generated randomly from two normal populations where the mean vectors are quite far apart though the variances for each of the covariates also increase. But they will be well seperated linearly for the first few covariates which makes the data clouds linearly seperable that is the reason for the better performance of the linear discriminant & support vector classifiers.

Comparison of SVM with KNN, LDA & QDA

- As we see KNN does not perform well as the dimension of the problem increases which is quite expected.
- Rather LDA and Support Vector Classifier (linear classifier with soft margin) are the best performing classifiers here.
- We see that the observations in the feature space are generated randomly from two normal populations where the mean vectors are quite far apart though the variances for each of the covariates also increase. But they will be well seperated linearly for the first few covariates which makes the data clouds linearly seperable that is the reason for the better performance of the linear discriminant & support vector classifiers.
- But SVM with radial kernel couldn't perform well here.

Comparison of SVM with KNN, LDA & QDA

- Next we perform the same comparison but with a new pair of random samples for each category.

Comparison of SVM with KNN, LDA & QDA

- Next we perform the same comparison but with a new pair of random samples for each category.
- In this example, we classify first 100 random samples from $N_p\left(\mathbf{0}^{p\times 1}, \boldsymbol{\Sigma}_1^{p\times p}\right)$ where $\boldsymbol{\Sigma}_1^{p\times p} = \Sigma^{p\times p}$, classify them as -1 and second 100 random samples from $N_p\left(\mathbf{0}^{p\times 1}, \boldsymbol{\Sigma}_2^{p\times p}\right)$ where $\boldsymbol{\Sigma}_2^{p\times p} = 4\Sigma^{p\times p}$ which we classify as $+1$.

Comparison of SVM with KNN, LDA & QDA

- Next we perform the same comparison but with a new pair of random samples for each category.
- In this example, we classify first 100 random samples from $N_p\left(\mathbf{0}^{p\times 1}, \mathbf{\Sigma}_1^{p\times p}\right)$ where $\mathbf{\Sigma}_1^{p\times p} = \Sigma^{p\times p}$, classify them as -1 and second 100 random samples from $N_p\left(\mathbf{0}^{p\times 1}, \mathbf{\Sigma}_2^{p\times p}\right)$ where $\mathbf{\Sigma}_2^{p\times p} = 4\Sigma^{p\times p}$ which we classify as $+1$.
- Then we similarly, vary the value of p from 5 to 50 and for each value of p, we simulate observations from both classes and then fit a classifier using different methods and calculate the total number of misclassifications for each of the methods at each values of p to compare.

Comparison of SVM with KNN, LDA & QDA

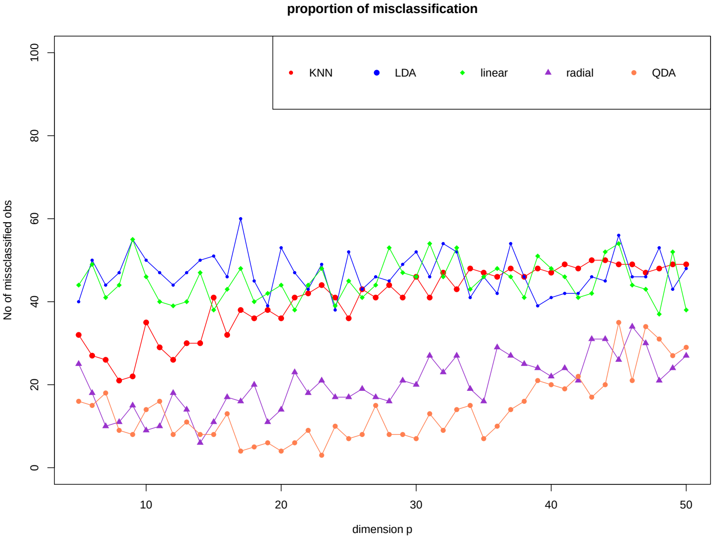

Figure: Test errors for different values of p for different classifiers

Comparison of SVM with KNN, LDA & QDA

- Here also, KNN does not perform well as the dimension of the problem increases which is quite expected.

Comparison of SVM with KNN, LDA & QDA

- Here also, KNN does not perform well as the dimension of the problem increases which is quite expected.
- Infact LDA also fails to capture the non-linear seperation boundary which is quite obvious since the very basic assumptions of LDA are violated here. (different covariance matrices)

Comparison of SVM with KNN, LDA & QDA

- Here also, KNN does not perform well as the dimension of the problem increases which is quite expected.
- Infact LDA also fails to capture the non-linear seperation boundary which is quite obvious since the very basic assumptions of LDA are violated here. (different covariance matrices)
- And hence it's expected that QDA classifier will perform well here and it does perform well.

Comparison of SVM with KNN, LDA & QDA

- Here also, KNN does not perform well as the dimension of the problem increases which is quite expected.
- Infact LDA also fails to capture the non-linear seperation boundary which is quite obvious since the very basic assumptions of LDA are violated here. (different covariance matrices)
- And hence it's expected that QDA classifier will perform well here and it does perform well.
- Though Support vector classifier didn't perform well here, SVM classifier with radial kernel performed much better than most of the other classifiers and is comparable with the ideal classifier here, (i.e. the QDA classifier).

Conclusion

- Though Support Vector Classifiers and SVMs are not outperforming in each of the cases.

Conclusion

- Though Support Vector Classifiers and SVMs are not outperforming in each of the cases.
- But we can see that they do reasonably good when compared to other classifiers since we have the flexibility of changing the kernel function in SVM which covers a large variety of different situations where one can classify.

Conclusion

- Though Support Vector Classifiers and SVMs are not outperforming in each of the cases.
- But we can see that they do reasonably good when compared to other classifiers since we have the flexibility of changing the kernel function in SVM which covers a large variety of different situations where one can classify.
- The idea of finding a hyperplane that separates the data as well as possible, while allowing some violations to this separation, is distinctly different from classical approaches for classification, such as logistic regression and linear discriminant analysis.

Conclusion

- Though Support Vector Classifiers and SVMs are not outperforming in each of the cases.
- But we can see that they do reasonably good when compared to other classifiers since we have the flexibility of changing the kernel function in SVM which covers a large variety of different situations where one can classify.
- The idea of finding a hyperplane that separates the data as well as possible, while allowing some violations to this separation, is distinctly different from classical approaches for classification, such as logistic regression and linear discriminant analysis.
- Moreover, the idea of using a kernel to expand the feature space in order to accommodate non-linear class boundaries appears to be a unique and valuable characteristic.

Conclusion

- Though Support Vector Classifiers and SVMs are not outperforming in each of the cases.
- But we can see that they do reasonably good when compared to other classifiers since we have the flexibility of changing the kernel function in SVM which covers a large variety of different situations where one can classify.
- The idea of finding a hyperplane that separates the data as well as possible, while allowing some violations to this separation, is distinctly different from classical approaches for classification, such as logistic regression and linear discriminant analysis.
- Moreover, the idea of using a kernel to expand the feature space in order to accommodate non-linear class boundaries appears to be a unique and valuable characteristic.
- Hence, we can consider SVMs as a very flexible and powerful tool in the toolbox of classifiers.

Case Study

- Till now we have shown how LDA,QDA,KNN and SVM performs in different scenarios i.e. when our data at hand has some particular structure.

Case Study

- Till now we have shown how LDA,QDA,KNN and SVM performs in different scenarios i.e. when our data at hand has some particular structure.
- We had created these datasets artificially to demonstrate our findings.

Case Study

- Till now we have shown how LDA,QDA,KNN and SVM performs in different scenarios i.e. when our data at hand has some particular structure.
- We had created these datasets artificially to demonstrate our findings.
- But now,we will compare the methods on a real life dataset - **Mobile Price Range** dataset.

Case Study

- Till now we have shown how LDA,QDA,KNN and SVM performs in different scenarios i.e. when our data at hand has some particular structure.
- We had created these datasets artificially to demonstrate our findings.
- But now,we will compare the methods on a real life dataset - **Mobile Price Range** dataset.
- Here our response variable is the **Price Range of a Mobile Phone** which is divided into four classes namely 0,1,2,3. The predictors/covariates are :-
  1) Battery Power 2) Clock Speed 3) FC 4) Internal Memory
  5) Mobile Depth 6) Mobile Weight 7) Number of Cores 8) PC
  9) px_height 10) px_width 11) RAM 12) sc_h 13) sc_w 14) talk time.

Case Study

- We have 2000 observations in our dataset.

Here is how the data looks like :-

```
  battery_power clock_speed fc int_memory m_dep mobile_wt n_cores pc
1           842         2.2  1          7   0.6       188       2  2
2          1021         0.5  0         53   0.7       136       3  6
3           563         0.5  2         41   0.9       145       5  6
4           615         2.5  0         10   0.8       131       6  9
5          1821         1.2 13         44   0.6       141       2 14
6          1859         0.5  3         22   0.7       164       1  7
  px_height px_width  ram sc_h sc_w talk_time price_range
1        20      756 2549    9    7        19           1
2       905     1988 2631   17    3         7           2
3      1263     1716 2603   11    2         9           2
4      1216     1786 2769   16    8        11           2
5      1208     1212 1411    8    2        15           1
6      1004     1654 1067   17    1        10           1
```

Case Study

- Given these features , our problem is to detect that a particular mobile phone belongs to which class of the price range?

Case Study

- Given these features , our problem is to detect that a particular mobile phone belongs to which class of the price range?
- Basically this is a classification problem.

Case Study

- Given these features , our problem is to detect that a particular mobile phone belongs to which class of the price range?
- Basically this is a classification problem.
- We will compare which one among LDA, QDA, KNN, SVM performs best here.

Case Study

- Given these features , our problem is to detect that a particular mobile phone belongs to which class of the price range?
- Basically this is a classification problem.
- We will compare which one among LDA, QDA, KNN, SVM performs best here.
- Initially, we divide our training set further into two sets containing 1500 and 500 observations respectively for training and validation of our model.

Case Study

- Given these features , our problem is to detect that a particular mobile phone belongs to which class of the price range?
- Basically this is a classification problem.
- We will compare which one among LDA, QDA, KNN, SVM performs best here.
- Initially, we divide our training set further into two sets containing 1500 and 500 observations respectively for training and validation of our model.
- Then using these two datasets, we compare the 4 methods i.e. which model makes the least number of missclassification in the test dataset.

LDA Classifier

We fit the LDA classifier and get the following classification table for the 4 categories :-

```
pred.lda   0   1   2   3
       0 121   3   0   0
       1   6 121  11   0
       2   0   3 118   3
       3   0   0   0 114
```

So, here the classifier performs pretty well with total of just

```
[1] 26
```

misclassified observations in the test set out of 500 observations.

QDA Classifier

We fit the LDA classifier and get the following classification table for the 4 categories :-

```
pred.qda   0   1   2   3
       0 123   8   0   0
       1   4 110   7   0
       2   0   9 117   2
       3   0   0   5 115
```

Here the QDA classifier makes a total of

```
[1] 35
```

misclassified observations in the test set.

KNN Classifier

We fit the KNN classifier and get the following classification table for the 4 categories :-

```
mod.knn   0   1   2   3
      0 125   8   0   0
      1   2 113   9   0
      2   0   6 113   8
      3   0   0   7 109
```

We note that since, the feature space is quite large here, KNN becomes inefficient and slow and we can also see it makes a total of

```
[1] 40
```

misclassified observations in the test set out of 500 observations which is the highest number till now.

SV Classifier

Lastly, we fit the SVM classifier using one-vs-one method and get the following classification table for the 4 categories :-

```
      0   1   2   3
  0 126   3   0   0
  1   1 119   7   0
  2   0   5 122   1
  3   0   0   0 116
```

And quite surprisingly, the SV classifier misclassified only

```
[1] 17
```

many observations in the test set out of 500 observations which is the lowest in all the methods.

Conclusion

- Since, we have just considered one test/validation dataset, we should not conclude anything from this .

Conclusion

- Since, we have just considered one test/validation dataset, we should not conclude anything from this .
- It will be reasonable if we rather split the data into several validation sets (5 to be specific, each being of size 400) and each time fit the classifier using the observations other than a particular validation set (the training set is now of size 1600) and lastly predict the classes for the validation set in order to measure the accuracy of that method.

Conclusion

- Since, we have just considered one test/validation dataset, we should not conclude anything from this .
- It will be reasonable if we rather split the data into several validation sets (5 to be specific, each being of size 400) and each time fit the classifier using the observations other than a particular validation set (the training set is now of size 1600) and lastly predict the classes for the validation set in order to measure the accuracy of that method.
- We repeat this process for all the 5 randomly generated validation sets and plot the total number of misclassifion values for the four different classifiers in order to get better idea about their performance.

Conclusion

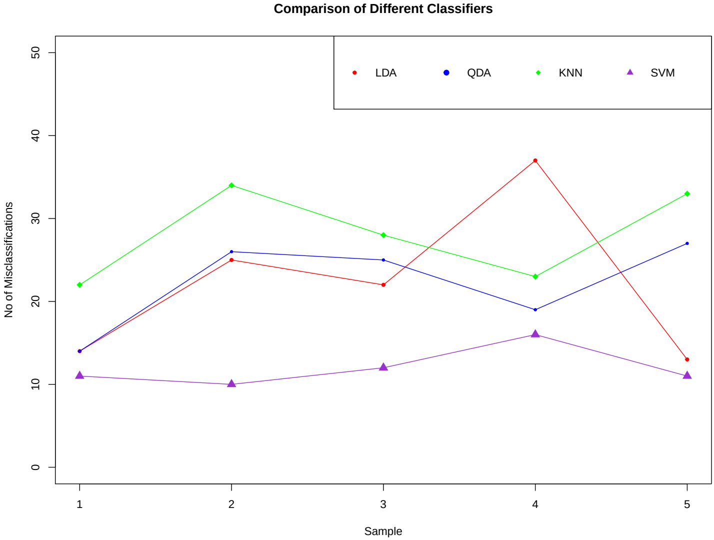

Figure: Test errors for 5 validation sets

Conclusion

- From the plot we can clearly see that the SV classifier performs better throughout the different validation sets which is an evidence in favour of its better performance than other methods.

Books

- Hastie, T., Tibshirani, R.,, Friedman, J. (2001). The Elements of Statistical Learning. New York, NY, USA: Springer New York Inc.
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning : with Applications in R. New York :Springer.
- Vapnik, V. N. (1998). Statistical Learning Theory. Wiley-Interscience.
- Gill, P. E., Murray, W.,, Wright, M. H. (1981). Practical Optimization. New York: Academic.

R Packages

- Sarkar D (2008). Lattice: Multivariate Data Visualization with R. Springer, New York. (Link)
- Venables WN, Ripley BD (2002). Modern Applied Statistics with S. Springer, New York. (Link)
- e1071 : Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. (Link)
- Genz A, Bretz F, Miwa T, Mi X, Leisch F, Scheipl F, Hothorn T (2021). mvtnorm: Multivariate Normal and t Distributions. (Link)

Other resources

- https://hastie.su.domains/MOOC-Slides/svm.pdf.
- https://youtu.be/_YPScrckx28 (Figure 5-9).
- Hastie, T., Tibshirani, R.,, Friedman, J. (2001). The Elements of Statistical Learning. (Figure 1-4)
- Curry, Haskell B. (1944). "The Method of Steepest Descent for Non-linear Minimization Problems". Quart. Appl. Math.
- https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf.
- Bottou, Léon (1998). "Online Algorithms and Stochastic Approximations". Online Learning and Neural Networks. Cambridge University Press.

Acknowledgement

We would like to express our **special thanks of gratitude** to our respected professor **Dr. Deepayan Sarkar** sir, for helping us throught the presentation work and also for giving us this wonderful opportunity as we learned many new & interesting things during the making of this presentation.