Statistical Learning– MATH 6333
Set 5 (Support Vector Machines - SVM)

Tamer Oraby

UTRGV
tamer.oraby@utrgv.edu

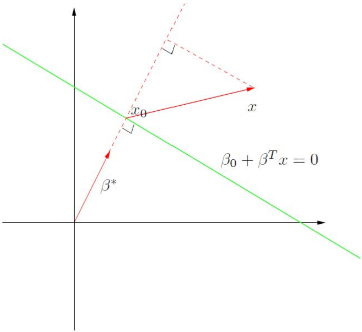
* Last updated October 27, 2021

Recall ...

From linear algebra ...

- $\beta^* = \frac{\beta}{||\beta||}$ is orthonormal to the separating hyperplane

$$L = \{x : \beta_0 + x^T\beta = 0\}$$



Recall ...

From linear algebra ...

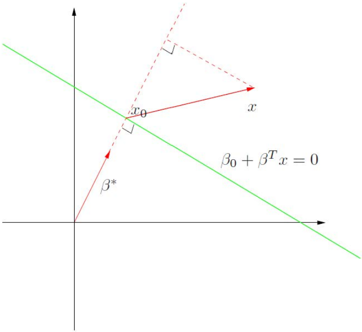
- $\beta^* = \frac{\beta}{\|\beta\|}$ is orthonormal to the separating hyperplane

  $$L = \{x : \beta_0 + x^T\beta = 0\}$$

  if

  $$(x_1 - x_0)^T\beta^* = 0$$

  for any $x_0, x_1 \in L$.



Recall ...

From linear algebra ...

- $\beta^* = \frac{\beta}{||\beta||}$ is orthonormal to the separating hyperplane

$$L = \{x : \beta_0 + x^T\beta = 0\}$$

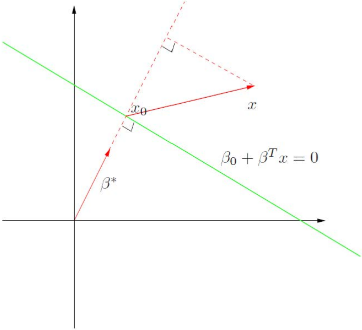
if

$$(x_1 - x_0)^T\beta^* = 0$$

for any $x_0, x_1 \in L$.

- For $x \notin L$, the signed distance of x to L is

$$(x - x_0)^T\beta^* = \frac{\beta_0 + x^T\beta}{||\beta||} \propto \beta_0 + x^T\beta$$



Recall ...

From linear algebra ...

- $\beta^* = \frac{\beta}{\|\beta\|}$ is orthonormal to the separating hyperplane

$$L = \{x : \beta_0 + x^T\beta = 0\}$$

if

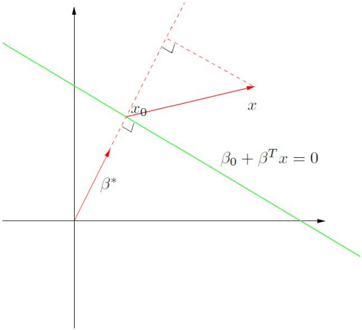
$$(x_1 - x_0)^T\beta^* = 0$$

for any $x_0, x_1 \in L$.

- For $x \notin L$, the signed distance of x to L is

$$(x - x_0)^T\beta^* = \frac{\beta_0 + x^T\beta}{\|\beta\|} \propto \beta_0 + x^T\beta$$

- Note that, signed distance of $x_1 \in L$ is zero.



Recall ...

From linear algebra ...

- The "actual" distance between two hyperplanes

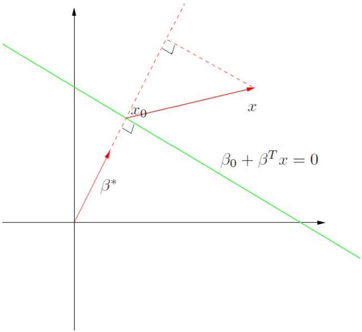
$$L_1 = \{x : \beta_{0,1} + x^T\beta = 0\}$$

and

$$L_2 = \{x : \beta_{0,2} + x^T\beta = 0\}$$

is

$$\frac{|\beta_{0,1} - \beta_{0,2}|}{||\beta||}$$



Other Classification Methods

Other Classification Methods

1. Maximal Margin Classifier (*aka* Optimal Separating Hyperplane)
2. Support Vector Classifier (*aka* Soft Margin Classifier)
3. Support Vector Machine
4. Flexible Discriminant Methods

Separating Hyperplanes - Maximal Margin Classifier

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

OSH maximizes the margins (signed distances M) of the slab

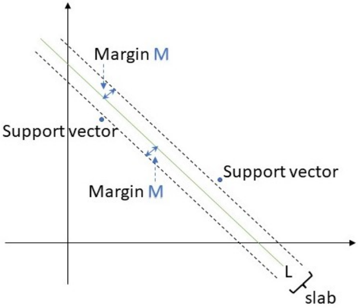
- Solve

$$\max_{\beta_0,\beta} M$$

subject to

$$\frac{1}{\|\beta\|} y_i(\beta_0 + x_i^T \beta) \geq M$$

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



- Set $\|\beta\| = \frac{1}{M}$

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

OSH maximizes the margins (signed distances M) of the slab

- Solve

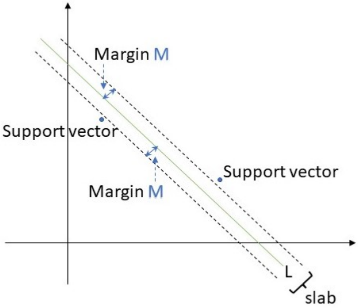
$$\max_{\beta_0,\beta} M$$

subject to

$$\frac{1}{\|\beta\|} y_i(\beta_0 + x_i^T \beta) \geq M$$

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

- Set $\|\beta\| = \frac{1}{M}$



Maximal Margin Classifier

aka Optimal Separating Hyperplanes

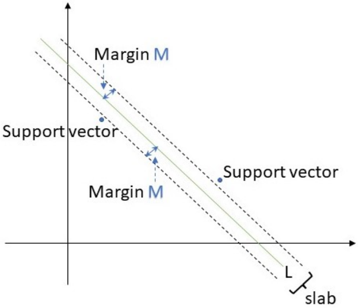
- Then the problem becomes equivalent to the convex optimization problem

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

subject to

$$y_i(\beta_0 + x_i^T\beta) \geq 1$$

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



Maximal Margin Classifier

aka Optimal Separating Hyperplanes

- Step 1: is the Lagrange problem to

$$\min_{\beta_0,\beta} L_p$$

where

$$L_p = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\beta_0 + x_i^T \beta) - 1)$$

s.t. $\alpha_i \geq 0$

- Setting derivatives equal to zero leads to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i x_i = \beta$$

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

- Step 1: is the Lagrange problem to

$$\min_{\beta_0, \beta} L_p$$

where

$$L_p = \frac{1}{2} \|\beta\|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\beta_0 + x_i^T \beta) - 1)$$

s.t. $\alpha_i \geq 0$

- Setting derivatives equal to zero leads to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i x_i = \beta$$

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

- Substituting with those into L_p we get

$$L_p = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

- Step 2: Using Wolfe dual optimization, the problem becomes

$$\max_{\alpha_i} L_D$$

where

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to (the Karush-Kuhn-Tucker conditions)

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i x_i = \beta$$

$$\alpha_i \geq 0$$

and

$$\alpha_i (y_i (\beta_0 + x_i^T \beta) - 1) = 0$$

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

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

- Here, β depends on α through the KKT conditions.
- If optimal $\alpha_i = 0$, then $y_i(\beta_0 + x_i^T \beta) - 1 > 0$ and so the point is not on the margin line.
- If $\alpha_i > 0$, then $y_i(\beta_0 + x_i^T \beta) - 1 = 0$ and so the point is on the margin line and which will contribute to the values of β that will make up the decision boundary based on this support points on the slab's boundaries.
- Separation will occur according to $\hat{G}(x) = sign(\hat{\beta}_0 + x^T \hat{\beta})$.
- where $\hat{\beta} = \sum_{i \in \partial slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T \hat{\beta}$ for any $i \in \partial slab$

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

- Here, β depends on α through the KKT conditions.
- If optimal $\alpha_i = 0$, then $y_i(\beta_0 + x_i^T\beta) - 1 > 0$ and so the point is not on the margin line.
- If $\alpha_i > 0$, then $y_i(\beta_0 + x_i^T\beta) - 1 = 0$ and so the point is on the margin line and which will contribute to the values of β that will make up the decision boundary based on this support points on the slab's boundaries.
- Separation will occur according to $\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta})$.
- where $\hat{\beta} = \sum_{i\in\partial slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T\hat{\beta}$ for any $i \in \partial slab$

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

- Here, β depends on α through the KKT conditions.
- If optimal $\alpha_i = 0$, then $y_i(\beta_0 + x_i^T\beta) - 1 > 0$ and so the point is not on the margin line.
- If $\alpha_i > 0$, then $y_i(\beta_0 + x_i^T\beta) - 1 = 0$ and so the point is on the margin line and which will contribute to the values of β that will make up the decision boundary based on this support points on the slab's boundaries.
- Separation will occur according to $\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta})$.
- where $\hat{\beta} = \sum_{i \in \partial slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T\hat{\beta}$ for any $i \in \partial slab$

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

- Here, β depends on α through the KKT conditions.
- If optimal $\alpha_i = 0$, then $y_i(\beta_0 + x_i^T\beta) - 1 > 0$ and so the point is not on the margin line.
- If $\alpha_i > 0$, then $y_i(\beta_0 + x_i^T\beta) - 1 = 0$ and so the point is on the margin line and which will contribute to the values of β that will make up the decision boundary based on this support points on the slab's boundaries.
- Separation will occur according to $\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta})$.
- where $\hat{\beta} = \sum_{i \in \partial slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T\hat{\beta}$ for any $i \in \partial slab$

Maximal Margin Classifier

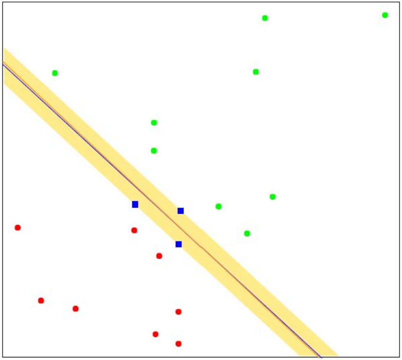
aka Optimal Separating Hyperplanes

- Here, β depends on α through the KKT conditions.
- If optimal $\alpha_i = 0$, then $y_i(\beta_0 + x_i^T\beta) - 1 > 0$ and so the point is not on the margin line.
- If $\alpha_i > 0$, then $y_i(\beta_0 + x_i^T\beta) - 1 = 0$ and so the point is on the margin line and which will contribute to the values of β that will make up the decision boundary based on this support points on the slab's boundaries.
- Separation will occur according to $\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta})$.
- where $\hat{\beta} = \sum_{i \in \partial slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T\hat{\beta}$ for any $i \in \partial slab$

Maximal Margin Classifier

aka Optimal Separating Hyperplanes

Example (Simulated data in $\mathbb{R}^2$)



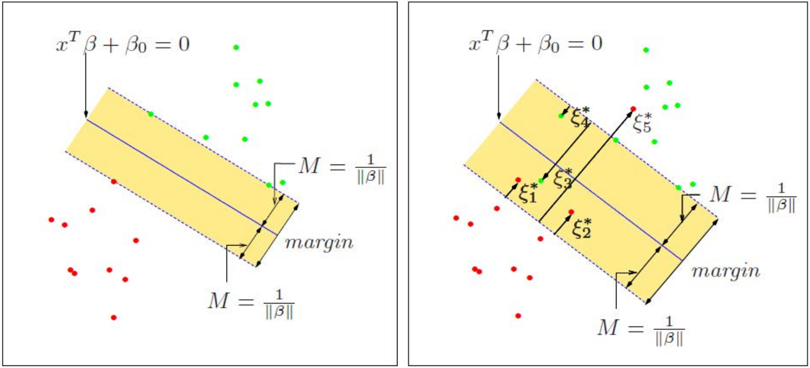
The blue line is the OHS and the red line is due to logistic regression.

Separating Hyperplanes - Support Vector Classifier

Support Vector Classifier

aka Soft Margin Classifier

Example (Simulated data in $\mathbb{R}^2$)

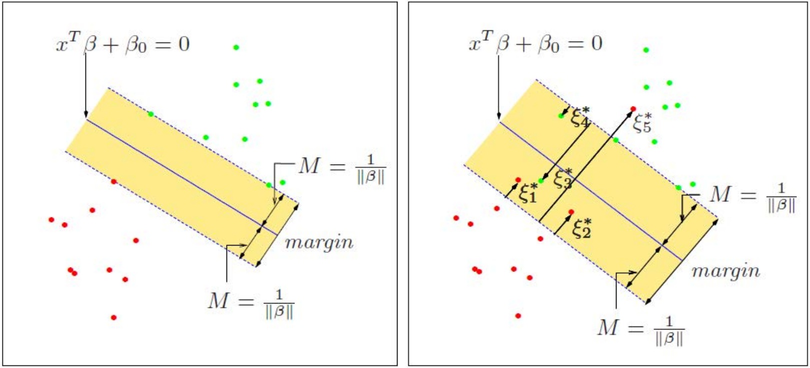


Maximal Margin Classifier works for the left panel with $y_i(x_i^T\beta + \beta_0) \geq 1$ since the points are linearly separable. But it is not the case in the right panel where $y_i(x_i^T\beta + \beta_0) < 1$. Adding *slack* variables $\xi_i \geq 0$ gives $y_i(x_i^T\beta + \beta_0) + \xi_i \geq 1$.

Support Vector Classifier

aka Soft Margin Classifier

Example (Simulated data in $\mathbb{R}^2$)



Maximal Margin Classifier works for the left panel with $y_i(x_i^T\beta + \beta_0) \geq 1$ since the points are linearly separable. But it is not the case in the right panel where $y_i(x_i^T\beta + \beta_0) < 1$. Adding *slack* variables $\xi_i \geq 0$ gives $y_i(x_i^T\beta + \beta_0) + \xi_i \geq 1$.

Support Vector Classifier

aka Soft Margin Classifier

If the vectors are not linearly separable. Let ξ_i is the smallest such that $y_i(x_i^T\beta + \beta_0) + \xi_i = 1$ and

- if $\xi_i = 0$, then $y_i(x_i^T\beta + \beta_0) = 1$ so it is accurately classified point, otherwise

$$y_i(x_i^T\beta + \beta_0) = 1 - \xi_i$$

- if $0 < \xi_i \leq 1$, then $0 \leq y_i(x_i^T\beta + \beta_0) < 1$ so it is "also" accurately classified point. Yet, that point (vector) has violated the margin.
- if $\xi_i > 1$, then $y_i(x_i^T\beta + \beta_0) < 0$, and so it is inaccurately classified point. That point (vector) is on the wrong side of the hyperplane.

Support Vector Classifier

aka Soft Margin Classifier

If the vectors are not linearly separable. Let ξ_i is the smallest such that $y_i(x_i^T\beta + \beta_0) + \xi_i = 1$ and

- if $\xi_i = 0$, then $y_i(x_i^T\beta + \beta_0) = 1$ so it is accurately classified point, otherwise

$$y_i(x_i^T\beta + \beta_0) = 1 - \xi_i$$

- if $0 < \xi_i \leq 1$, then $0 \leq y_i(x_i^T\beta + \beta_0) < 1$ so it is "also" accurately classified point. Yet, that point (vector) has violated the margin.
- if $\xi_i > 1$, then $y_i(x_i^T\beta + \beta_0) < 0$, and so it is inaccurately classified point. That point (vector) is on the wrong side of the hyperplane.

Support Vector Classifier

aka Soft Margin Classifier

If the vectors are not linearly separable. Let ξ_i is the smallest such that $y_i(x_i^T\beta + \beta_0) + \xi_i = 1$ and

- if $\xi_i = 0$, then $y_i(x_i^T\beta + \beta_0) = 1$ so it is accurately classified point, otherwise

$$y_i(x_i^T\beta + \beta_0) = 1 - \xi_i$$

- if $0 < \xi_i \leq 1$, then $0 \leq y_i(x_i^T\beta + \beta_0) < 1$ so it is "also" accurately classified point. Yet, that point (vector) has violated the margin.
- if $\xi_i > 1$, then $y_i(x_i^T\beta + \beta_0) < 0$, and so it is inaccurately classified point. That point (vector) is on the wrong side of the hyperplane.

Support Vector Classifier

aka Soft Margin Classifier

- Thus, the misclassification rate is $\sum_{i=1}^{N} I(\xi_i > 1)$.
- It makes sense to include it in the optimization problem by minimizing $\sum_{i=1}^{N} I(\xi_i > 1)$.
- But, $\sum_{i=1}^{N} I(\xi_i > 1)$ is not differentiable in ξ_i.
- However, since $I(\xi_i > 1) \leq \xi_i$ for all i then it is sufficient to minimize $\sum_{i=1}^{N} \xi_i$.

Support Vector Classifier

aka Soft Margin Classifier

- Thus, the misclassification rate is $\sum_{i=1}^{N} I(\xi_i > 1)$.
- It makes sense to include it in the optimization problem by minimizing $\sum_{i=1}^{N} I(\xi_i > 1)$.
- But, $\sum_{i=1}^{N} I(\xi_i > 1)$ is not differentiable in ξ_i.
- However, since $I(\xi_i > 1) \leq \xi_i$ for all i then it is sufficient to minimize $\sum_{i=1}^{N} \xi_i$.

Support Vector Classifier

aka Soft Margin Classifier

- Thus, the misclassification rate is $\sum_{i=1}^{N} I(\xi_i > 1)$.
- It makes sense to include it in the optimization problem by minimizing $\sum_{i=1}^{N} I(\xi_i > 1)$.
- But, $\sum_{i=1}^{N} I(\xi_i > 1)$ is not differentiable in ξ_i.
- However, since $I(\xi_i > 1) \leq \xi_i$ for all i then it is sufficient to minimize $\sum_{i=1}^{N} \xi_i$.

Support Vector Classifier

aka Soft Margin Classifier

- Thus, the misclassification rate is $\sum_{i=1}^{N} I(\xi_i > 1)$.
- It makes sense to include it in the optimization problem by minimizing $\sum_{i=1}^{N} I(\xi_i > 1)$.
- But, $\sum_{i=1}^{N} I(\xi_i > 1)$ is not differentiable in ξ_i.
- However, since $I(\xi_i > 1) \leq \xi_i$ for all i then it is sufficient to minimize $\sum_{i=1}^{N} \xi_i$.

Support Vector Classifier

aka Soft Margin Classifier

- So the optimization problem becomes

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

subject to

$$y_i(\beta_0 + x_i^T\beta) + \xi_i \geq 1$$

and

$$\xi_i \geq 0$$

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

- Where $C > 0$ is a tuning parameter that is the reciprocal of the cost the problem can afford from misclassification.
- When $C = \infty$, the cost is zero and only solution is the zero solution.

Support Vector Classifier

aka Soft Margin Classifier

- So the optimization problem becomes

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

subject to

$$y_i(\beta_0 + x_i^T\beta) + \xi_i \geq 1$$

and

$$\xi_i \geq 0$$

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

- Where $C > 0$ is a tuning parameter that is the reciprocal of the cost the problem can afford from misclassification.
- When $C = \infty$, the cost is zero and only solution is the zero solution.

Support Vector Classifier

aka Soft Margin Classifier

- So the optimization problem becomes

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

  subject to

$$y_i(\beta_0 + x_i^T\beta) + \xi_i \geq 1$$

  and

$$\xi_i \geq 0$$

  for $i = 1, 2, \ldots, N$.
- Where $C > 0$ is a tuning parameter that is the reciprocal of the cost the problem can afford from misclassification.
- When $C = \infty$, the cost is zero and only solution is the zero solution.

Support Vector Classifier

aka Soft Margin Classifier

- Step 1: is the Lagrange problem to

$$\min_{\beta_0,\beta,\xi} L_p$$

where

$$L_p = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\beta_0 + x_i^T\beta) + \xi_i - 1) - \sum_{i=1}^{N}\mu_i\xi_i$$

$$= \frac{1}{2}\|\beta\|^2 + \sum_{i=1}^{N}(C - \alpha_i - \mu_i)\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\beta_0 + x_i^T\beta) - 1)$$

s.t. the Lagrange multipliers $\alpha_i, \mu_i \geq 0$ and the slack variables $\xi_i \geq 0$.

- Setting derivatives equal to zero leads to

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \text{ and } \sum_{i=1}^{N}\alpha_i y_i x_i = \beta \text{ and } C - \alpha_i - \mu_i = 0 \text{ for all } i$$

Support Vector Classifier

aka Soft Margin Classifier

- Step 1: is the Lagrange problem to

$$\min_{\beta_0,\beta,\xi} L_p$$

where

$$L_p = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\beta_0 + x_i^T\beta) + \xi_i - 1) - \sum_{i=1}^{N}\mu_i\xi_i$$

$$= \frac{1}{2}\|\beta\|^2 + \sum_{i=1}^{N}(C - \alpha_i - \mu_i)\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\beta_0 + x_i^T\beta) - 1)$$

s.t. the Lagrange multipliers $\alpha_i, \mu_i \geq 0$ and the slack variables $\xi_i \geq 0$.

- Setting derivatives equal to zero leads to

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \text{ and } \sum_{i=1}^{N}\alpha_i y_i x_i = \beta \text{ and } C - \alpha_i - \mu_i = 0 \text{ for all } i$$

Support Vector Classifier

aka Soft Margin Classifier

- Step 1: is the Lagrange problem to

$$\min_{\beta_0,\beta,\xi} L_p$$

where

$$L_p = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\beta_0 + x_i^T\beta) + \xi_i - 1) - \sum_{i=1}^{N}\mu_i\xi_i$$

$$= \frac{1}{2}\|\beta\|^2 + \sum_{i=1}^{N}(C - \alpha_i - \mu_i)\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\beta_0 + x_i^T\beta) - 1)$$

s.t. the Lagrange multipliers $\alpha_i, \mu_i \geq 0$ and the slack variables $\xi_i \geq 0$.

- Setting derivatives equal to zero leads to

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \text{ and } \sum_{i=1}^{N}\alpha_i y_i x_i = \beta \text{ and } C - \alpha_i - \mu_i = 0 \text{ for all } i$$

Support Vector Classifier

aka Soft Margin Classifier

- Substituting with those into L_p we get

$$L_p = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- Note that since $C = \alpha_i + \mu_i$, then $0 \le \alpha_i \le C$.

Support Vector Classifier

aka Soft Margin Classifier

- Substituting with those into L_p we get

$$L_p = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- Note that since $C = \alpha_i + \mu_i$, then $0 \leq \alpha_i \leq C$.

Support Vector Classifier

aka Soft Margin Classifier

- Step 2: Using Wolfe dual optimization, the problem becomes

$$\max_{\alpha_i} L_D$$

where

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to (the Karush-Kuhn-Tucker conditions)

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i x_i = \beta \text{ and } C - \alpha_i - \mu_i = 0 \text{ for all } i$$

$$0 \leq \alpha_i \leq C$$

and

$$\alpha_i(y_i(\beta_0 + x_i^T \beta) + \xi_i - 1) = 0 \text{ and } y_i(\beta_0 + x_i^T \beta) + \xi_i - 1 \geq 0$$

and $\mu_i \xi_i = 0$ for $i = 1, 2, \ldots, N$.

Support Vector Classifier

aka Soft Margin Classifier

- Again, β depends on optimal α^* through the KKT conditions.
- If optimal $\alpha_i^* = 0$, then $\mu_i = C$ and $\xi_i = 0$ and so $y_i(\beta_0 + x_i^T\beta) - 1 > 0$. Thus, the point/vector is not on the margin line.
- If optimal $0 < \alpha_i^* < C$, then $\mu_i \neq 0$ and $\xi_i = 0$ and so $y_i(\beta_0 + x_i^T\beta) - 1 = 0$. Thus, the point/vector is on the margin line and α_i^* will contribute to the values of β that will make up the decision boundary. Those points are called margin support vectors.
- ...

Support Vector Classifier

aka Soft Margin Classifier

- Again, β depends on optimal α^* through the KKT conditions.
- If optimal $\alpha_i^* = 0$, then $\mu_i = C$ and $\xi_i = 0$ and so $y_i(\beta_0 + x_i^T\beta) - 1 > 0$. Thus, the point/vector is not on the margin line.
- If optimal $0 < \alpha_i^* < C$, then $\mu_i \neq 0$ and $\xi_i = 0$ and so $y_i(\beta_0 + x_i^T\beta) - 1 = 0$. Thus, the point/vector is on the margin line and α_i^* will contribute to the values of β that will make up the decision boundary. Those points are called margin support vectors.
- ...

Support Vector Classifier

aka Soft Margin Classifier

- Again, β depends on optimal α^* through the KKT conditions.
- If optimal $\alpha_i^* = 0$, then $\mu_i = C$ and $\xi_i = 0$ and so $y_i(\beta_0 + x_i^T\beta) - 1 > 0$. Thus, the point/vector is not on the margin line.
- If optimal $0 < \alpha_i^* < C$, then $\mu_i \neq 0$ and $\xi_i = 0$ and so $y_i(\beta_0 + x_i^T\beta) - 1 = 0$. Thus, the point/vector is on the margin line and α_i^* will contribute to the values of β that will make up the decision boundary. Those points are called margin support vectors.
- ...

Support Vector Classifier

aka Soft Margin Classifier

- Again, β depends on optimal α^* through the KKT conditions.
- If optimal $\alpha_i^* = 0$, then $\mu_i = C$ and $\xi_i = 0$ and so $y_i(\beta_0 + x_i^T\beta) - 1 > 0$. Thus, the point/vector is not on the margin line.
- If optimal $0 < \alpha_i^* < C$, then $\mu_i \neq 0$ and $\xi_i = 0$ and so $y_i(\beta_0 + x_i^T\beta) - 1 = 0$. Thus, the point/vector is on the margin line and α_i^* will contribute to the values of β that will make up the decision boundary. Those points are called margin support vectors.
- ...

Support Vector Classifier

aka Soft Margin Classifier

- If optimal $\alpha_i^* = C$, then $\mu_i = 0$ and $\xi_i \geq 0$ and so $y_i(\beta_0 + x_i^T\beta) + \xi_i - 1 = 0$.
  - If $\xi \leq 1$, the point/vector beyond the margin line but before the hyperplane. Those points are called non-margin support vectors. (A violator but accurately classified.)
  - If $\xi > 1$, the point/vector beyond the hyperplane. (A misclassification.)
- with $\hat{\beta} = \sum_{i \in slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T\hat{\beta}$ for any $i \in slab$
- separation will occur according to

$$\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta}) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i x_i^T x)$$

Support Vector Classifier

aka Soft Margin Classifier

- If optimal $\alpha_i^* = C$, then $\mu_i = 0$ and $\xi_i \geq 0$ and so $y_i(\beta_0 + x_i^T\beta) + \xi_i - 1 = 0$.
  - If $\xi \leq 1$, the point/vector beyond the margin line but before the hyperplane. Those points are called non-margin support vectors. (A violator but accurately classified.)
  - If $\xi > 1$, the point/vector beyond the hyperplane. (A misclassification.)
- with $\hat{\beta} = \sum_{i \in slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T\hat{\beta}$ for any $i \in slab$
- separation will occur according to

$$\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta}) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i x_i^T x)$$

Support Vector Classifier

aka Soft Margin Classifier

- If optimal $\alpha_i^* = C$, then $\mu_i = 0$ and $\xi_i \geq 0$ and so $y_i(\beta_0 + x_i^T\beta) + \xi_i - 1 = 0$.
  - If $\xi \leq 1$, the point/vector beyond the margin line but before the hyperplane. Those points are called non-margin support vectors. (A violator but accurately classified.)
  - If $\xi > 1$, the point/vector beyond the hyperplane. (A misclassification.)
- with $\hat{\beta} = \sum_{i \in slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T\hat{\beta}$ for any $i \in slab$
- separation will occur according to

$$\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta}) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i x_i^T x)$$

Support Vector Classifier

aka Soft Margin Classifier

- If optimal $\alpha_i^* = C$, then $\mu_i = 0$ and $\xi_i \geq 0$ and so $y_i(\beta_0 + x_i^T\beta) + \xi_i - 1 = 0$.
  - If $\xi \leq 1$, the point/vector beyond the margin line but before the hyperplane. Those points are called non-margin support vectors. (A violator but accurately classified.)
  - If $\xi > 1$, the point/vector beyond the hyperplane. (A misclassification.)
- with $\hat{\beta} = \sum_{i \in slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T\hat{\beta}$ for any $i \in slab$
- separation will occur according to

$$\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta}) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i x_i^T x)$$

Support Vector Classifier

aka Soft Margin Classifier

- If optimal $\alpha_i^* = C$, then $\mu_i = 0$ and $\xi_i \geq 0$ and so $y_i(\beta_0 + x_i^T \beta) + \xi_i - 1 = 0$.
  - If $\xi \leq 1$, the point/vector beyond the margin line but before the hyperplane. Those points are called non-margin support vectors. (A violator but accurately classified.)
  - If $\xi > 1$, the point/vector beyond the hyperplane. (A misclassification.)

- with $\hat{\beta} = \sum_{i \in slab} \alpha_i^* y_i x_i$ and $\hat{\beta}_0 = y_i - x_i^T \hat{\beta}$ for any $i \in slab$

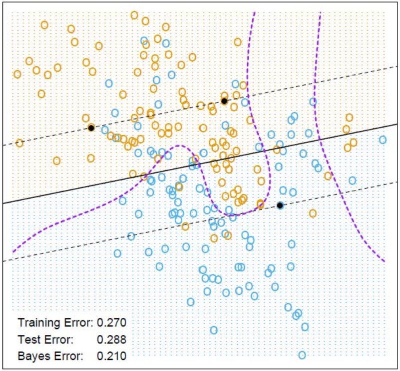
- separation will occur according to

$$\hat{G}(x) = sign(\hat{\beta}_0 + x^T \hat{\beta}) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i x_i^T x)$$

Support Vector Classifier

aka Soft Margin Classifier

Example (Simulated data in $\mathbb{R}^2$)



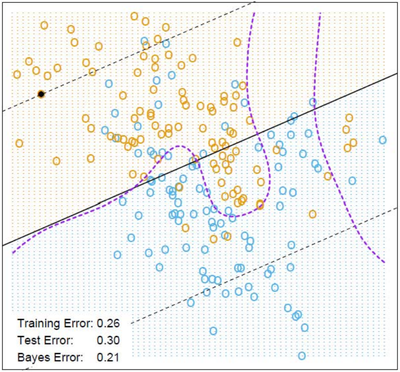
$C = 10000$

The broken purple curve is the Bayes decision boundary. 62% observations are support points.

Support Vector Classifier

aka Soft Margin Classifier

Example (Simulated data in $\mathbb{R}^2$)



$C = 0.01$

The broken purple curve is the Bayes decision boundary. 85% observations are support points.

Regression and Kernels

Regression and Kernels

- Let $\{h_m(x), m = 1, 2, \ldots, M\}$ be a set of basis transformations, each of which maps $\mathbb{R}^p$ into $\mathbb{R}$, e.g.
  1. $h_m(x) = X_m$, X_j, X_j^2, X_iX_j, $\log(X_j)$
  2. piece-wise constants $h_m(x) = c_m I(L_m \leq X < U_m)$ with $-\infty = L_1 < U_1 \leq L_2 < U_2 \leq L_3 < \cdots \leq L_M < U_M = \infty$
  3. $h_m(x) = \sum_{i \in A_m} c_{m,i} X_i$ for some set A_m
  4. h_m is a polynomial or spline function
- A regression function

$$f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(x)$$

is a linear function in h_m in the new $M-$ dimensional space

Regression and Kernels

- Let $\{h_m(x), m = 1, 2, \ldots, M\}$ be a set of basis transformations, each of which maps $\mathbb{R}^p$ into $\mathbb{R}$, e.g.

  1. $h_m(x) = X_m,\ X_j,\ X_j^2,\ X_iX_j,\ \log(X_j)$
  2. piece-wise constants $h_m(x) = c_m I(L_m \leq X < U_m)$ with $-\infty = L_1 < U_1 \leq L_2 < U_2 \leq L_3 < \cdots \leq L_M < U_M = \infty$
  3. $h_m(x) = \sum_{i \in A_m} c_{m,i} X_i$ for some set A_m
  4. h_m is a polynomial or spline function

- A regression function

$$f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(x)$$

is a linear function in h_m in the new $M-$ dimensional space

Regression and Kernels

- Let $\{h_m(x), m = 1, 2, \ldots, M\}$ be a set of basis transformations, each of which maps $\mathbb{R}^p$ into $\mathbb{R}$, e.g.

  1. $h_m(x) = X_m$, X_j, X_j^2, X_iX_j, $\log(X_j)$
  2. piece-wise constants $h_m(x) = c_m I(L_m \le X < U_m)$ with $-\infty = L_1 < U_1 \le L_2 < U_2 \le L_3 < \cdots \le L_M < U_M = \infty$
  3. $h_m(x) = \sum_{i \in A_m} c_{m,i} X_i$ for some set A_m
  4. h_m is a polynomial or spline function

- A regression function

$$f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(x)$$

is a linear function in h_m in the new $M-$ dimensional space

Regression and Kernels

- Let $\{h_m(x), m = 1, 2, \ldots, M\}$ be a set of basis transformations, each of which maps $\mathbb{R}^p$ into $\mathbb{R}$, e.g.
  1. $h_m(x) = X_m$, X_j, X_j^2, X_iX_j, $\log(X_j)$
  2. piece-wise constants $h_m(x) = c_m I(L_m \leq X < U_m)$ with $-\infty = L_1 < U_1 \leq L_2 < U_2 \leq L_3 < \cdots \leq L_M < U_M = \infty$
  3. $h_m(x) = \sum_{i \in A_m} c_{m,i} X_i$ for some set A_m
  4. h_m is a polynomial or spline function

- A regression function

$$f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(x)$$

is a linear function in h_m in the new $M-$ dimensional space

Regression and Kernels

- Let $\{h_m(x), m = 1, 2, \ldots, M\}$ be a set of basis transformations, each of which maps $\mathbb{R}^p$ into $\mathbb{R}$, e.g.
  1. $h_m(x) = X_m, X_j, X_j^2, X_iX_j, \log(X_j)$
  2. piece-wise constants $h_m(x) = c_m I(L_m \leq X < U_m)$ with $-\infty = L_1 < U_1 \leq L_2 < U_2 \leq L_3 < \cdots \leq L_M < U_M = \infty$
  3. $h_m(x) = \sum_{i \in A_m} c_{m,i} X_i$ for some set A_m
  4. h_m is a polynomial or spline function

- A regression function

$$f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(x)$$

is a linear function in h_m in the new $M-$ dimensional space

Regression and Kernels

- Let $\{h_m(x), m = 1, 2, \ldots, M\}$ be a set of basis transformations, each of which maps $\mathbb{R}^p$ into $\mathbb{R}$, e.g.

  1. $h_m(x) = X_m,\ X_j,\ X_j^2,\ X_i X_j,\ \log(X_j)$
  2. piece-wise constants $h_m(x) = c_m I(L_m \leq X < U_m)$ with $-\infty = L_1 < U_1 \leq L_2 < U_2 \leq L_3 < \cdots \leq L_M < U_M = \infty$
  3. $h_m(x) = \sum_{i \in A_m} c_{m,i} X_i$ for some set A_m
  4. h_m is a polynomial or spline function

- A regression function

$$f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(x)$$

is a linear function in h_m in the new $M-$ dimensional space

Regression and Kernels

- whose estimate is

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{m=1}^{M} \hat{\beta}_m h_m(x)$$

where parameters β_0 and β are estimated by minimizing the L_2-penalized objective function

$$RSS_\lambda(\beta_0, \beta) = \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \sum_{m=1}^{M} \beta_m^2$$

with $M > N$.

- L could be the squared loss function $L(x, y) = (x - y)^2$
- thus, after estimating β_0 a priori (let us set $\hat{\beta}_0 = 0$ for simplicity) ...

Regression and Kernels

- whose estimate is

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{m=1}^{M} \hat{\beta}_m h_m(x)$$

  where parameters β_0 and β are estimated by minimizing the L_2-penalized objective function

$$RSS_\lambda(\beta_0, \beta) = \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \sum_{m=1}^{M} \beta_m^2$$

  with $M > N$.
- L could be the squared loss function $L(x, y) = (x - y)^2$
- thus, after estimating β_0 a priori (let us set $\hat{\beta}_0 = 0$ for simplicity) ...

Regression and Kernels

- whose estimate is

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{m=1}^{M} \hat{\beta}_m h_m(x)$$

  where parameters β_0 and β are estimated by minimizing the L_2-penalized objective function

$$RSS_\lambda(\beta_0, \beta) = \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \sum_{m=1}^{M} \beta_m^2$$

  with $M > N$.
- L could be the squared loss function $L(x, y) = (x - y)^2$
- thus, after estimating β_0 a priori (let us set $\hat{\beta}_0 = 0$ for simplicity) ...

Regression and Kernels

- the objective function is

$$RSS_\lambda(\beta) = (y - X_h\beta)^T(y - X_h\beta) + \lambda||\beta||^2$$

where the $N \times M$ matrix

$$X_h = \begin{pmatrix} h_1(x_1) & h_2(x_1) & \cdots & h_M(x_1) \\ h_1(x_2) & h_2((x_2) & \cdots & h_M(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(x_N) & h_2(x_N) & \cdots & h_M(x_N) \end{pmatrix}$$

is of the model $y = X_h\beta$

Regression and Kernels

- The penalized least squares solution is determined by differentiation of $RSS_\lambda(\beta)$ and setting the result equal to zero

$$-X_h^T(y - X_h\beta) + \lambda\beta = 0$$

$$-X_h X_h^T(y - X_h\beta) + \lambda X_h\beta = 0$$

and so for $\lambda > 0$

$$\hat{y} = X_h\hat{\beta} = (X_h X_h^T + \lambda I)^{-1} X_h X_h^T y$$

- The $N \times N$ matrix $X_h X_h^T$ has the ij^{th} elements

$$\sum_{m=1}^{M} h_m(x_i)h_m(x_j) = h(x_i)^T h(x_j) = \underbrace{< h(x_i), h(x_j) >}_{\text{inner product}}$$

which requires a total of N^2M calculations.

Regression and Kernels

- The penalized least squares solution is determined by differentiation of $RSS_\lambda(\beta)$ and setting the result equal to zero

$$-X_h^T(y - X_h\beta) + \lambda\beta = 0$$

$$-X_h X_h^T(y - X_h\beta) + \lambda X_h\beta = 0$$

and so for $\lambda > 0$

$$\hat{y} = X_h\hat{\beta} = (X_h X_h^T + \lambda I)^{-1} X_h X_h^T y$$

- The $N \times N$ matrix $X_h X_h^T$ has the ij^{th} elements

$$\sum_{m=1}^{M} h_m(x_i)h_m(x_j) = h(x_i)^T h(x_j) = \underbrace{< h(x_i), h(x_j) >}_{\text{inner product}}$$

which requires a total of $N^2 M$ calculations.

Regression and Kernels

- The penalized least squares solution is determined by differentiation of $RSS_\lambda(\beta)$ and setting the result equal to zero

$$-X_h^T(y - X_h\beta) + \lambda\beta = 0$$

$$-X_h X_h^T(y - X_h\beta) + \lambda X_h\beta = 0$$

and so for $\lambda > 0$

$$\hat{y} = X_h\hat{\beta} = (X_h X_h^T + \lambda I)^{-1} X_h X_h^T y$$

- The $N \times N$ matrix $X_h X_h^T$ has the ij^{th} elements

$$\sum_{m=1}^{M} h_m(x_i)h_m(x_j) = h(x_i)^T h(x_j) = \underbrace{< h(x_i), h(x_j) >}_{\text{inner product}}$$

which requires a total of N^2M calculations.

Regression and Kernels

- The penalized least squares solution is determined by differentiation of $RSS_\lambda(\beta)$ and setting the result equal to zero

$$-X_h^T(y - X_h\beta) + \lambda\beta = 0$$

$$-X_h X_h^T(y - X_h\beta) + \lambda X_h\beta = 0$$

and so for $\lambda > 0$

$$\hat{y} = X_h\hat{\beta} = (X_h X_h^T + \lambda I)^{-1} X_h X_h^T y$$

- The $N \times N$ matrix $X_h X_h^T$ has the ij^{th} elements

$$\sum_{m=1}^{M} h_m(x_i)h_m(x_j) = h(x_i)^T h(x_j) = \underbrace{< h(x_i), h(x_j) >}_{\text{inner product}}$$

which requires a total of $N^2 M$ calculations.

Regression and Kernels

- The penalized least squares solution is determined by differentiation of $RSS_\lambda(\beta)$ and setting the result equal to zero

$$-X_h^T(y - X_h\beta) + \lambda\beta = 0$$

$$-X_h X_h^T(y - X_h\beta) + \lambda X_h\beta = 0$$

and so for $\lambda > 0$

$$\hat{y} = X_h\hat{\beta} = (X_h X_h^T + \lambda I)^{-1} X_h X_h^T y$$

- The $N \times N$ matrix $X_h X_h^T$ has the ij^{th} elements

$$\sum_{m=1}^{M} h_m(x_i)h_m(x_j) = h(x_i)^T h(x_j) = \underbrace{< h(x_i), h(x_j) >}_{\text{inner product}}$$

which requires a total of $N^2 M$ calculations.

Regression and Kernels

- Thus, for a new x_*,

$$\hat{f}(x_*) = h(x_*)^T \hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i h(x_*)^T h(x_i)$$

  where $\hat{\alpha}_i = (X_h X_h^T + \lambda I)^{-1} y_i$

- which could be computationally simplified using a Kernel K and

$$\hat{f}(x_*) = \sum_{i=1}^{N} \hat{\alpha}_i K(x_*, x_i)$$

  since kernel computations requires a total of $N^2/2$ calculations.

Regression and Kernels

- Thus, for a new x_*,

$$\hat{f}(x_*) = h(x_*)^T \hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i h(x_*)^T h(x_i)$$

  where $\hat{\alpha}_i = (X_h X_h^T + \lambda I)^{-1} y_i$

- which could be computationally simplified using a Kernel K and

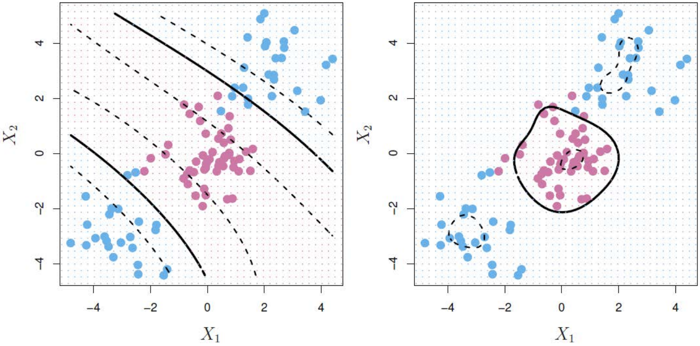
$$\hat{f}(x_*) = \sum_{i=1}^{N} \hat{\alpha}_i K(x_*, x_i)$$

  since kernel computations requires a total of $N^2/2$ calculations.

Non-linear Classification via Support Vector Machine

Support Vector Machine

Example (Simulated data in $\mathbb{R}^2$)



Left panel: using a polynomial of degree 3 kernel and right panel: using radial kernel.

Support Vector Machine

- In the linear classification case, SVC uses

$$\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta}) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{x_i^T x}_{\text{inner product}})$$

- In the non-linear classification case, SVC uses

$$\hat{G}(x) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{h(x_i)^T h(x)}_{\text{inner product}})$$

for some $\{h_m(x), m = 1, 2, \ldots, M\}$ set of basis transformations. Note that $0 < \alpha_i^* < C$ for $i \in slab$.

Support Vector Machine

- In the linear classification case, SVC uses

$$\hat{G}(x) = sign(\hat{\beta}_0 + x^T\hat{\beta}) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{x_i^T x}_{\text{inner product}})$$

- In the non-linear classification case, SVC uses

$$\hat{G}(x) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{h(x_i)^T h(x)}_{\text{inner product}})$$

for some $\{h_m(x), m = 1, 2, \ldots, M\}$ set of basis transformations. Note that $0 < \alpha_i^* < C$ for $i \in slab$.

Support Vector Machine

- In general classification case, SVM is a SVC that uses kernels so that

$$\hat{G}(x) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{K(x_i, x)}_{\text{kernel}})$$

and the previously described optimization problem in SVC is still valid.
- DIY analytically Show that $\hat{G}$ described using the kernel is the decision rule obtained via the optimization problem described in SVC.
- If we don't use penalized objective functions (the regularization parameter $\lambda = 0$), then we need the symmetric kernel K to be positive definite function.

Support Vector Machine

- In general classification case, SVM is a SVC that uses kernels so that

$$\hat{G}(x) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{K(x_i, x)}_{\text{kernel}})$$

  and the previously described optimization problem in SVC is still valid.
- DIY analytically Show that $\hat{G}$ described using the kernel is the decision rule obtained via the optimization problem described in SVC.
- If we don't use penalized objective functions (the regularization parameter $\lambda = 0$), then we need the symmetric kernel K to be positive definite function.

Support Vector Machine

- In general classification case, SVM is a SVC that uses kernels so that

$$\hat{G}(x) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{K(x_i, x)}_{\text{kernel}})$$

  and the previously described optimization problem in SVC is still valid.
- DIY analytically Show that $\hat{G}$ described using the kernel is the decision rule obtained via the optimization problem described in SVC.
- If we don't use penalized objective functions (the regularization parameter $\lambda = 0$), then we need the symmetric kernel K to be positive definite function.

Support Vector Machine

- K is positive definite if there exists a function h such $K(x_i, x_j) =< h(x_i), h(x_j) >$
- Mercer's condition: A symmetric real-valued function $K(x, y)$ is said to satisfy Mercer's condition, if for all $L_2(\mathbb{R}^p)$ real-valued functions g,

$$\int_{\mathbb{R}^p} \int_{\mathbb{R}^p} K(x, y) g(x) g(y) dx dy \geq 0$$

- Theorem: Let $K(x, y)$ be a symmetric real-valued function. There exists a function h such $K(x, y) =< h(x), h(y) >$ if and only if K satisfies Mercer's condition.

Support Vector Machine

- K is positive definite if there exists a function h such $K(x_i, x_j) =< h(x_i), h(x_j) >$
- Mercer's condition: A symmetric real-valued function $K(x, y)$ is said to satisfy Mercer's condition, if for all $L_2(\mathbb{R}^p)$ real-valued functions g,

$$\int_{\mathbb{R}^p}\int_{\mathbb{R}^p} K(x, y)g(x)g(y)dxdy \geq 0$$

- Theorem: Let $K(x, y)$ be a symmetric real-valued function. There exists a function h such $K(x, y) =< h(x), h(y) >$ if and only if K satisfies Mercer's condition.

Support Vector Machine

- K is positive definite if there exists a function h such $K(x_i, x_j) =< h(x_i), h(x_j) >$
- Mercer's condition: A symmetric real-valued function $K(x, y)$ is said to satisfy Mercer's condition, if for all $L_2(\mathbb{R}^p)$ real-valued functions g,

$$\int_{\mathbb{R}^p} \int_{\mathbb{R}^p} K(x, y)g(x)g(y)dxdy \geq 0$$

- Theorem: Let $K(x, y)$ be a symmetric real-valued function. There exists a function h such $K(x, y) =< h(x), h(y) >$ if and only if K satisfies Mercer's condition.

Support Vector Machine

- Mercer-Hilbert Schmidt Theorem: Let $K(x, y)$ be a symmetric real-valued function that satisfies Mercer's condition, then there exists a set of orthonormal eigenfunctions $\{v_i(x)\}_{i=1}^{\infty}$ such that

$$\int_{\mathbb{R}^p} K(x, y) v_i(y) dy = \lambda_i v_i(x)$$

for $i = 1, 2, \ldots, \infty$, and

$$K(x, y) = \sum_{j=1}^{\infty} \lambda_j v_j(x) v_j(y)$$

.

- Thus, define $h_j(x) = \sqrt{\lambda_j} v_j(x)$ so that

$$K(x, y) = < h(x), h(y) >$$

Support Vector Machine

- Mercer-Hilbert Schmidt Theorem: Let $K(x, y)$ be a symmetric real-valued function that satisfies Mercer's condition, then there exists a set of orthonormal eigenfunctions $\{v_i(x)\}_{i=1}^{\infty}$ such that

$$\int_{\mathbb{R}^p} K(x, y) v_i(y) dy = \lambda_i v_i(x)$$

for $i = 1, 2, \ldots, \infty$, and

$$K(x, y) = \sum_{j=1}^{\infty} \lambda_j v_j(x) v_j(y)$$

.

- Thus, define $h_j(x) = \sqrt{\lambda_j} v_j(x)$ so that

$$K(x, y) = < h(x), h(y) >$$

Support Vector Machine

Popular Kernels in SVM are:

- Polynomial kernel of degree d

$$K(x, y) = (1 + \sum_{i=1}^{p} x_i y_i)^d$$

- (Gaussian) Radial kernel (strong local support)

$$K(x, y) = \exp(-\gamma \sum_{i=1}^{p} (x_i - y_i)^2)$$

- (Laplace) Radial kernel (weak local support)

$$K(x, y) = \exp(-\gamma \sum_{i=1}^{p} |x_i - y_i|)$$

Support Vector Machine

Popular Kernels in SVM are:

- Polynomial kernel of degree d

$$K(x,y) = (1 + \sum_{i=1}^{p} x_i y_i)^d$$

- (Gaussian) Radial kernel (strong local support)

$$K(x,y) = \exp(-\gamma \sum_{i=1}^{p} (x_i - y_i)^2)$$

- (Laplace) Radial kernel (weak local support)

$$K(x,y) = \exp(-\gamma \sum_{i=1}^{p} |x_i - y_i|)$$

Support Vector Machine

Popular Kernels in SVM are:

- Polynomial kernel of degree d

$$K(x,y) = (1 + \sum_{i=1}^{p} x_i y_i)^d$$

- (Gaussian) Radial kernel (strong local support)

$$K(x,y) = \exp(-\gamma \sum_{i=1}^{p} (x_i - y_i)^2)$$

- (Laplace) Radial kernel (weak local support)

$$K(x,y) = \exp(-\gamma \sum_{i=1}^{p} |x_i - y_i|)$$

Support Vector Machine

Popular Kernels in SVM are:

- Cauchy kernel

$$K(x, y) = \gamma \frac{1}{1 + \sum_{i=1}^{p} (x_i - y_i)^2}$$

- Neural kernel

$$K(x, y) = \tanh(\kappa_1 \sum_{i=1}^{p} x_i y_i + \kappa_2)$$

Support Vector Machine

Popular Kernels in SVM are:

- Cauchy kernel

$$K(x, y) = \gamma \frac{1}{1 + \sum_{i=1}^{p} (x_i - y_i)^2}$$

- Neural kernel

$$K(x, y) = \tanh(\kappa_1 \sum_{i=1}^{p} x_i y_i + \kappa_2)$$

Support Vector Machine

Example

Consider the polynomial kernel of degree 2, for $x, y \in \mathbb{R}^2$

$$\begin{aligned} K(x, y) &= (1 + x_1 y_1 + x_2 y_2)^2 \\ &= 1 + x_1^2 y_1^2 + 2x_1 y_1 + 2x_1 y_1 x_2 y_2 + 2x_2 y_2 + x_2^2 y_2^2 \\ &= h(x)^T h(y) \end{aligned}$$

such that $h(x) = (1, x_1^2, \sqrt{2}x_1, \sqrt{2}x_1 x_2, \sqrt{2}x_2, x_2^2) \in \mathbb{R}^6$

Support Vector Machine

- SVM uses

$$\hat{G}(x) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{K(x_i, x)}_{\text{kernel}})$$

  such that $0 < \alpha_i^* < C$ for $i \in slab$.
- If C is large then most of $\xi_i = 0$ and that leads to wiggly boundary in the input space (an overfitting situation).
- If C is small then most of α_i are small and so is $\hat{\beta} = \sum_{i \in slab} \alpha_i^* y_i h(x_i)$ and that leads to smooth boundary.

Support Vector Machine

- SVM uses

$$\hat{G}(x) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{K(x_i, x)}_{\text{kernel}})$$

  such that $0 < \alpha_i^* < C$ for $i \in slab$.
- If C is large then most of $\xi_i = 0$ and that leads to wiggly boundary in the input space (an overfitting situation).
- If C is small then most of α_i are small and so is $\hat{\beta} = \sum_{i \in slab} \alpha_i^* y_i h(x_i)$ and that leads to smooth boundary.

Support Vector Machine

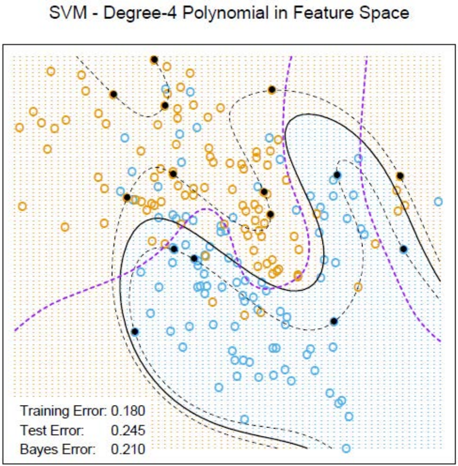
- SVM uses

$$\hat{G}(x) = sign(\hat{\beta}_0 + \sum_{i \in slab} \alpha_i^* y_i \underbrace{K(x_i, x)}_{\text{kernel}})$$

  such that $0 < \alpha_i^* < C$ for $i \in slab$.
- If C is large then most of $\xi_i = 0$ and that leads to wiggly boundary in the input space (an overfitting situation).
- If C is small then most of α_i are small and so is $\hat{\beta} = \sum_{i \in slab} \alpha_i^* y_i h(x_i)$ and that leads to smooth boundary.

Support Vector Machine

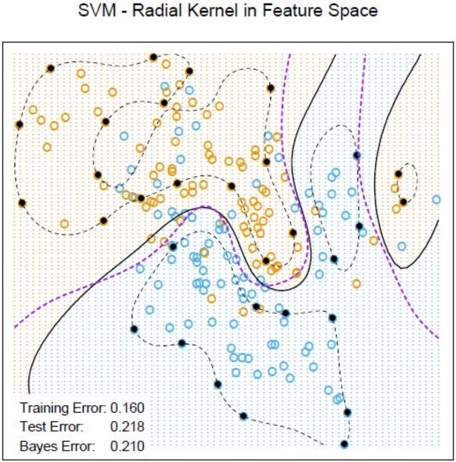
Example (Simulated data in $\mathbb{R}^2$)



$C = 1$. The broken purple curve is the Bayes decision boundary.

Support Vector Machine

Example (Simulated data in $\mathbb{R}^2$)



$C = 1$. It closer to Bayes optimal. The broken purple curve is the Bayes decision boundary.

SVM for more than two classes

Support Vector Machine

- One-versus-one classification:
  In a C_2^K round-robin, two classes k, k' play against each other coded $y_k + 1$ and $y_{k'} = -1$ and tally the classifications of x_* and assign it at the end of to the class with the highest number of classifications/votes.

- One-versus-all classification:
  Here, one class k (with $y_k = +1$) plays against the rest $K - 1$ classes (with $y_{[1,K]-\{k\}} = -1$) and estimate $f_k(x)$. Then use $\hat{G}(x_*) = argmax_k f_k(x_*)$.

Notation: $[1, K] = \{1, 2, \ldots, K\}$

Support Vector Machine

- One-versus-one classification:
  In a C_2^K round-robin, two classes k, k' play against each other coded $y_k + 1$ and $y_{k'} = -1$ and tally the classifications of x_* and assign it at the end of to the class with the highest number of classifications/votes.

- One-versus-all classification:
  Here, one class k (with $y_k = +1$) plays against the rest $K - 1$ classes (with $y_{[1,K]-\{k\}} = -1$) and estimate $f_k(x)$. Then use $\hat{G}(x_*) = argmax_k f_k(x_*)$.

Notation: $[1, K] = \{1, 2, \ldots, K\}$

Support Vector Machine

- One-versus-one classification:
  In a C_2^K round-robin, two classes k, k' play against each other coded $y_k + 1$ and $y_{k'} = -1$ and tally the classifications of x_* and assign it at the end of to the class with the highest number of classifications/votes.

- One-versus-all classification:
  Here, one class k (with $y_k = +1$) plays against the rest $K - 1$ classes (with $y_{[1,K]-\{k\}} = -1$) and estimate $f_k(x)$. Then use $\hat{G}(x_*) = argmax_k f_k(x_*)$.

Notation: $[1, K] = \{1, 2, \ldots, K\}$

Support Vector Machine

- One-versus-one classification:
  In a C_2^K round-robin, two classes k, k' play against each other coded $y_k + 1$ and $y_{k'} = -1$ and tally the classifications of x_* and assign it at the end of to the class with the highest number of classifications/votes.

- One-versus-all classification:
  Here, one class k (with $y_k = +1$) plays against the rest $K - 1$ classes (with $y_{[1,K]-\{k\}} = -1$) and estimate $f_k(x)$. Then use $\hat{G}(x_*) = argmax_k f_k(x_*)$.

Notation: $[1, K] = \{1, 2, \ldots, K\}$

SVM's relationship to Hinge Loss function and others

Support Vector Machine

In case of,
linear ($f(x) = \beta_0 + x^T\beta$) or nonlinear ($f(x) = \beta_0 + h(x)^T\beta$) classification problems

- The optimization problem is

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

  subject to

$$y_i(\beta_0 + x_i^T\beta) + \xi_i \geq 1$$

  and

$$\xi_i \geq 0$$

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

- The constraints are equivalent to the constraint

$$\xi_i \geq [1 - y_i f(x_i)]_+$$

  giving are the smallest value attainable by ξ_i for all i.

Support Vector Machine

In case of,
linear ($f(x) = \beta_0 + x^T\beta$) or nonlinear ($f(x) = \beta_0 + h(x)^T\beta$) classification problems

- The optimization problem is

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

  subject to

$$y_i(\beta_0 + x_i^T\beta) + \xi_i \geq 1$$

  and

$$\xi_i \geq 0$$

  for $i = 1, 2, \ldots, N$.
- The constraints are equivalent to the constraint

$$\xi_i \geq [1 - y_i f(x_i)]_+$$

  giving are the smallest value attainable by ξ_i for all i.

Support Vector Machine

- The minimization problem becomes

$$\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}[1 - y_i f(x_i)]_+$$

where $C > 0$.

- Setting $C = \frac{1}{2\lambda}$, it further becomes

$$\min_{\beta_0,\beta} \sum_{i=1}^{N}[1 - y_i f(x_i)]_+ + \lambda\|\beta\|^2$$

Support Vector Machine

- The minimization problem becomes

$$\min_{\beta_0,\beta} \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{N} [1 - y_i f(x_i)]_+$$

  where $C > 0$.
- Setting $C = \frac{1}{2\lambda}$, it further becomes

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \lambda \|\beta\|^2$$

Support Vector Machine

- The minimization problem becomes

$$\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}[1 - y_i f(x_i)]_+$$

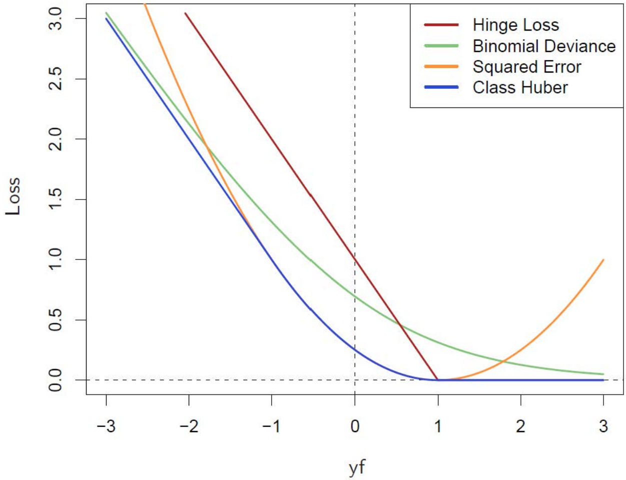
where $C > 0$.

- Setting $C = \frac{1}{2\lambda}$, it further becomes

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \underbrace{[1 - y_i f(x_i)]_+}_{\text{hinge loss function}} + \lambda\|\beta\|^2$$

which is an L_2 regularized problem with a new loss function (hinge loss function) that allows no contribution from the non-support vectors.

Support Vector Machine



Note: $y_i f(x_i) > 1$ is for accurately classified points, and using hinge loss function, makes their contribution nil.

Support Vector Machine

- Minimizing the L_2-penalized objective function

$$RSS_\lambda(\beta_0, \beta) = \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \sum_{m=1}^{M} \beta_m^2$$

  with $M > N$.
- L could take other forms for other functional forms of f

Support Vector Machine

- Minimizing the L_2-penalized objective function

$$RSS_\lambda(\beta_0, \beta) = \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda \sum_{m=1}^{M} \beta_m^2$$

  with $M > N$.
- L could take other forms for other functional forms of f

Support Vector Machine

Loss Function	$L[y, f(x)]$	Minimizing Function
Binomial Deviance	$\log[1 + e^{-yf(x)}]$	$f(x) = \log \frac{\Pr(Y=+1\|x)}{\Pr(Y=\text{-}1\|x)}$
SVM Hinge Loss	$[1 - yf(x)]_+$	$f(x) = \text{sign}[\Pr(Y=+1\|x) - \frac{1}{2}]$
Squared Error	$[y - f(x)]^2 = [1 - yf(x)]^2$	$f(x) = 2\Pr(Y=+1\|x) - 1$
"Huberised" Square Hinge Loss	$-4yf(x)$, $yf(x) < \text{-}1$; $[1 - yf(x)]_+^2$ otherwise	$f(x) = 2\Pr(Y=+1\|x) - 1$

Also, ...

L_2-Regularized Logistic Regression

- Recall, the L_2-Regularized Logistic Regression for any $f(x)$ is due to the maximization problem

$$max_\beta \sum_{i=1}^{N} \left[\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)})\right] - \lambda \sum_{j=1}^{p} \beta_j^2$$

for $\tilde{y}_i = 0, 1$

- which is easily transformed into ± 1 using $\tilde{y}_i = \frac{y_i+1}{2}$ for $y_i = -1, +1$
- $\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)}) = -\log(1 + e^{y_i f(x_i)})$
- So the optimization problem becomes, for $y_i = \pm 1$

$$min_\beta \sum_{i=1}^{N} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Also, ...

L_2-Regularized Logistic Regression

- Recall, the L_2-Regularized Logistic Regression for any $f(x)$ is due to the maximization problem

$$max_\beta \sum_{i=1}^{N} \left[\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)})\right] - \lambda \sum_{j=1}^{p} \beta_j^2$$

for $\tilde{y}_i = 0, 1$

- which is easily transformed into ± 1 using $\tilde{y}_i = \frac{y_i + 1}{2}$ for $y_i = -1, +1$
- $\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)}) = -\log(1 + e^{y_i f(x_i)})$
- So the optimization problem becomes, for $y_i = \pm 1$

$$min_\beta \sum_{i=1}^{N} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Also, ...

L_2-Regularized Logistic Regression

- Recall, the L_2-Regularized Logistic Regression for any $f(x)$ is due to the maximization problem

$$max_\beta \sum_{i=1}^{N} \left[\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)})\right] - \lambda \sum_{j=1}^{p} \beta_j^2$$

for $\tilde{y}_i = 0, 1$

- which is easily transformed into ± 1 using $\tilde{y}_i = \frac{y_i + 1}{2}$ for $y_i = -1, +1$
- $\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)}) = -\log(1 + e^{y_i f(x_i)})$
- So the optimization problem becomes, for $y_i = \pm 1$

$$min_\beta \sum_{i=1}^{N} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Also, ...

L_2-Regularized Logistic Regression

- Recall, the L_2-Regularized Logistic Regression for any $f(x)$ is due to the maximization problem

$$max_\beta \sum_{i=1}^{N} \left[\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)})\right] - \lambda \sum_{j=1}^{p} \beta_j^2$$

for $\tilde{y}_i = 0, 1$

- which is easily transformed into ± 1 using $\tilde{y}_i = \frac{y_i+1}{2}$ for $y_i = -1, +1$
- $\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)}) = -\log(1 + e^{y_i f(x_i)})$
- So the optimization problem becomes, for $y_i = \pm 1$

$$min_\beta \sum_{i=1}^{N} \log(1 + e^{y_i f(x_i)}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

Also, ...

L_2-Regularized Logistic Regression

- Recall, the L_2-Regularized Logistic Regression for any $f(x)$ is due to the maximization problem

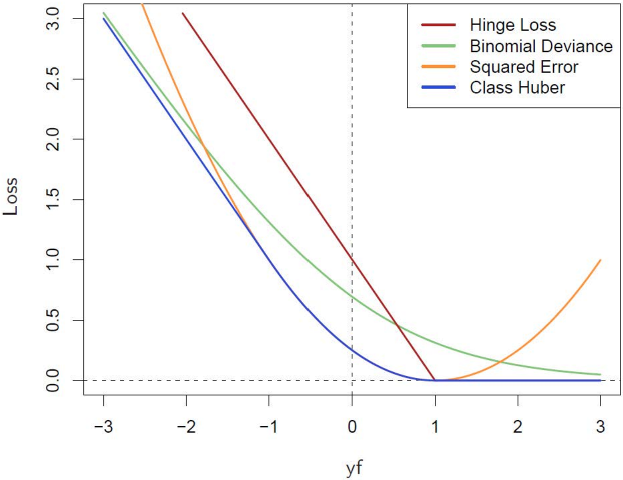
$$max_\beta \sum_{i=1}^{N} \left[\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)})\right] - \lambda \sum_{j=1}^{p} \beta_j^2$$

  for $\tilde{y}_i = 0, 1$

- which is easily transformed into ± 1 using $\tilde{y}_i = \frac{y_i + 1}{2}$ for $y_i = -1, +1$
- $\tilde{y}_i f(x_i) - \log(1 + e^{f(x_i)}) = -\log(1 + e^{y_i f(x_i)})$
- So the optimization problem becomes, for $y_i = \pm 1$

$$min_\beta \sum_{i=1}^{N} \underbrace{\log(1 + e^{y_i f(x_i)})}_{\text{binomial deviance}} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Support Vector Machine



Note: $y_i f(x_i) > 1$ is for accurately classified points, and using binomial deviance (in a logistic regression approach), makes their contribution positive but very small.

Support Vector Machine

Example (SA Heart Disease)

DIY in R

1. Carry out SVC classification using *e1071*. It is svm with Kernel="linear"
2. Carry out SVM classification using *e1071*. Use kernel = "polynomial" and kernel = "radial"
3. To evaluate the performance of the classifiers: use the receiver operating characteristic (ROC) curve using *ROCR*.

Support Vector Machine

Example (SA Heart Disease)

DIY in R

1. Carry out SVC classification using *e1071*. It is svm with Kernel="linear"
2. Carry out SVM classification using *e1071*. Use kernel = "polynomial" and kernel = "radial"
3. To evaluate the performance of the classifiers: use the receiver operating characteristic (ROC) curve using *ROCR*.

Support Vector Machine

Example (SA Heart Disease)

DIY in R

1. Carry out SVC classification using *e1071*. It is svm with Kernel="linear"
2. Carry out SVM classification using *e1071*. Use kernel = "polynomial" and kernel = "radial"
3. To evaluate the performance of the classifiers: use the receiver operating characteristic (ROC) curve using *ROCR*.

Support Vector Machine

Example (SA Heart Disease)

DIY in R

1. Carry out SVC classification using *e1071*. It is svm with Kernel="linear"
2. Carry out SVM classification using *e1071*. Use kernel = "polynomial" and kernel = "radial"
3. To evaluate the performance of the classifiers: use the receiver operating characteristic (ROC) curve using *ROCR*.

Please study the different methods in the ISL book. See also SVM with multiple classes.

End of Set 5