STAT 5630, Fall 2019

Support Vector Machines

Xiwei Tang, Ph.D. <xt4yj@virginia.edu>

Department of Statistics, University of Virginia

Overview

- We have training data: $\mathcal{D}_n = \{x_i, y_i\}_{i=1}^n$
  - $x_i \in \mathbb{R}^p$
  - Code $y_i \in \{-1, 1\}$
- Estimate a function $f(x) \in \mathbb{R}$, with classification rule

$$C(x) = \text{sign}\{f(x)\}$$

- Loss function (0/1)

$$L(C(x), y) = \begin{cases} 0 & \text{if} \quad y = C(x) \\ 1 & \text{if} \quad y \neq C(x) \end{cases}$$

- Recall our definition of the Bayes optimal classifier:

$$C^*(x) = \text{sign}\big\{\mathsf{P}(Y = 1 | X = x) - \mathsf{P}(Y = -1 | X = x)\big\}$$

Outline

- Linear SVM in Separable Case (separation margin)
- Linear SVM in non-Separable Case (slack variables)
- Non-linear SVM (Kernel trick)

Linear SVM in Separable Case

Binary Large-Margin Classifiers

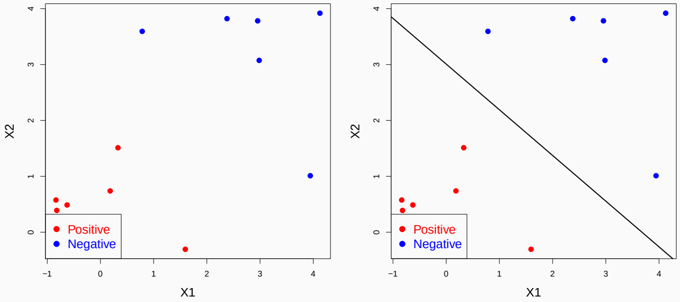
- Since $y_i \in \{-1, 1\}$, our classification rule using $f(x)$ is

$$\hat{y} = +1 \quad \text{if} \quad f(x) > 0$$
$$\hat{y} = -1 \quad \text{if} \quad f(x) < 0$$

- We have a correct classification if $y_i f(x_i) > 0$
- Functional margin $y_i f(x_i)$:
  - positive means good (at the correct side)
  - negative means bad (at the wrong side)

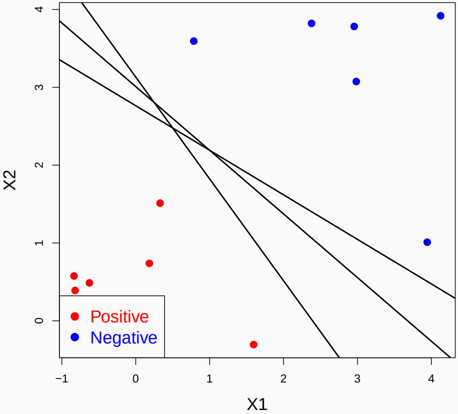
Separating Line

- Linearly separable: find $f(x) = x^\mathsf{T}\boldsymbol{\beta} + \beta_0$ to separate two groups of points



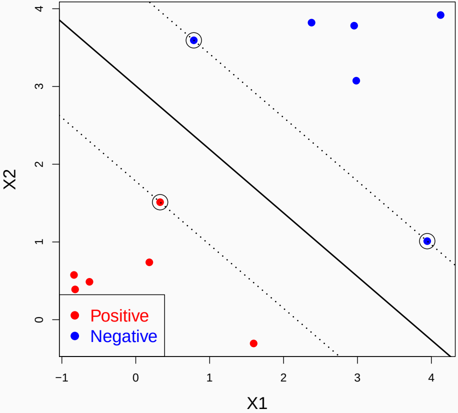
Separating Line

- Which line is the best?
- What would logistic regression do?
- Related to another method called Perceptron



Maximum Separation

- SVM searches for a line by maximizing the separation margin

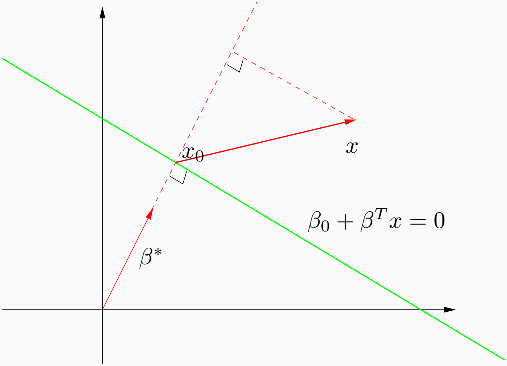


Signed Distance to the Hyperplane

- We define the (linear) separating hyperplane as

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

- For any point x_0 on the hyperplane $x_0^{\mathsf{T}}\boldsymbol{\beta} = -\beta_0$
- Signed distance of x to the plane is $\langle \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}, x - x_0 \rangle$



Signed Distance to the Hyperplane

- Define the linear function $f(x) = \beta_0 + x^{\mathsf{T}}\boldsymbol{\beta}$
- The affine hyperplane L: $\{x : f(x) = \beta_0 + x^{\mathsf{T}}\boldsymbol{\beta} = 0\}$
- The normal vector (perpendicular) to L is $\boldsymbol{\beta}^* = \boldsymbol{\beta}/\|\boldsymbol{\beta}\|$
- For any point $x_0 \in L$, we have

$$x_0^{\mathsf{T}}\boldsymbol{\beta} = -\beta_0$$

- The signed distance of any point x to L is

$$\begin{aligned}(x - x_0)^{\mathsf{T}}\boldsymbol{\beta}^* &= \frac{1}{\|\boldsymbol{\beta}\|}(x^{\mathsf{T}}\boldsymbol{\beta} + \beta_0) \\ &= \frac{f(x)}{\|\boldsymbol{\beta}\|}\end{aligned}$$

Thus $f(x)$ is proportional to the signed distance from x to L.

Maximum Margin Classifier

- Goal: Separate two classes and maximizes the distance to the closest points from either class (Vapnik 1996)

$$\max_{\boldsymbol{\beta}, \beta_0, \|\boldsymbol{\beta}\|=1} M$$
$$\text{subject to} \quad y_i(x_i^\mathsf{T}\boldsymbol{\beta} + \beta_0) \geq M, \ i = 1, \dots, n.$$

- Interpretation: All the points are at least a signed distance M from the decision boundary
- Recall that $f(x_i) = (x_i^\mathsf{T}\boldsymbol{\beta} + \beta_0)/\|\boldsymbol{\beta}\|$ is the signed distance.
  - If y_i is $+1$, we require $f(x_i) \geq M$;
  - If y_i is -1, we require $f(x_i) \leq -M$.
- Maximize the minimum distance (margin)

Maximum Margin Classifier

- This problem requires the constraint $\|\boldsymbol{\beta}\| = 1$
- To get rid of this, we replace the conditions with

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

- Since the scale of $\boldsymbol{\beta}$ does not play a role in this inequality, we can arbitrarily set $\|\boldsymbol{\beta}\| = 1/M$. Hence the original problem is equivalent to

$$\begin{aligned} &\min_{\boldsymbol{\beta},\beta_0} \|\boldsymbol{\beta}\|^2 \\ \text{subject to}\quad & y_i(x_i^\mathsf{T}\boldsymbol{\beta} + \beta_0) \geq 1,\ i = 1, \ldots, n. \end{aligned}$$

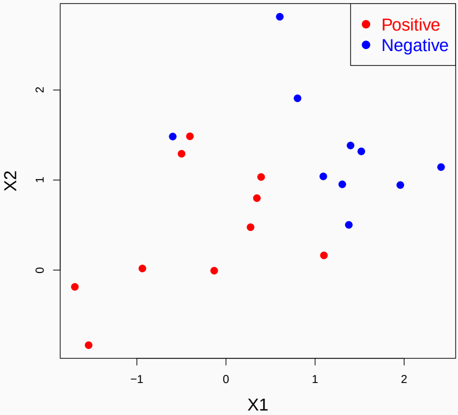
- Recall our previous derivation of the signed distance, this is requiring that all points are at least $1/\|\boldsymbol{\beta}\|$ away from the separating plane

Remarks

- If the classes are really Gaussian, then
  - LDA is optimal
  - The separating hyperplane pays a price for focusing on the noisier data at the boundaries
- Optimal separating hyperplane has less assumptions, thus more robust to model misspecification
  - The logistic regression solution can be similar to the operating hyperplane
  - For perfectly separable case, the likelihood solution can be infinity

Linear SVM in non-Separable Case

Linearly non-Separable



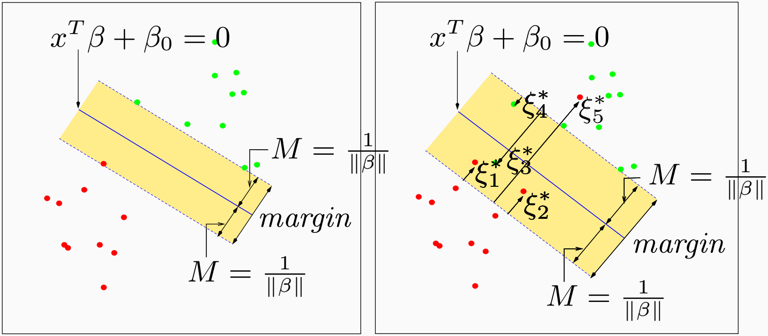
General Case for SVM

- Non-separable means that the "zero"-error is not attainable
- We introduce "slack variables" $\{\xi_i\}_{i=1}^n$ that accounts for these errors
- Change the original optimization problem to

$$\begin{aligned} & \text{minimize } \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^n \xi_i \\ \text{subject to} \quad & y_i(x_i^\mathsf{T}\boldsymbol{\beta} + \beta_0) \geq (1 - \xi_i),\ i = 1, \dots, n, \\ & \xi_i \geq 0,\ i = 1, \dots, n, \end{aligned}$$

where $C > 0$ is a tuning parameter for "cost"

Linearly non-Separable

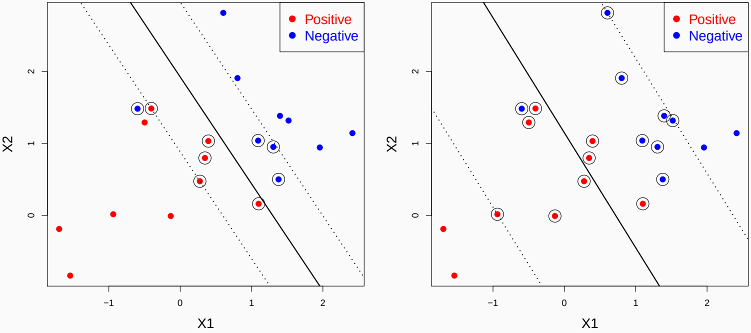


Slack variables in linearly non-separable case

Interpretation

- The objective function consists of two parts
  - For observations that cannot be classified correctly, $\xi_i > 1$. So $\sum_i \xi_i$ is an upper bound on the number of training errors
  - Minimize the inverse margin $\frac{1}{2}\|\boldsymbol{\beta}\|^2$
- The tuning parameter C
  - Balances the error and margin width
  - For separable case, $C = \infty$
- Inequality constraints
  - Soft classification to allow some errors

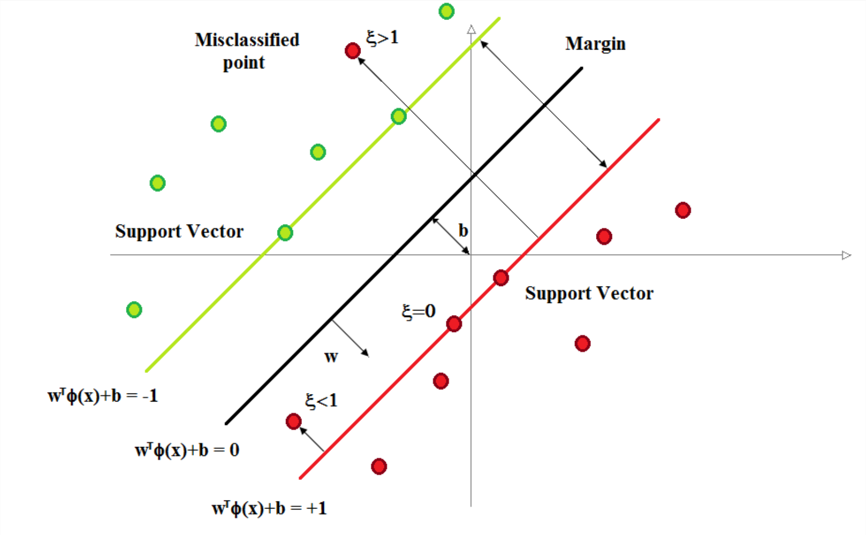
Linearly non-Separable



The support vectors for linearly non-separable case

Remark

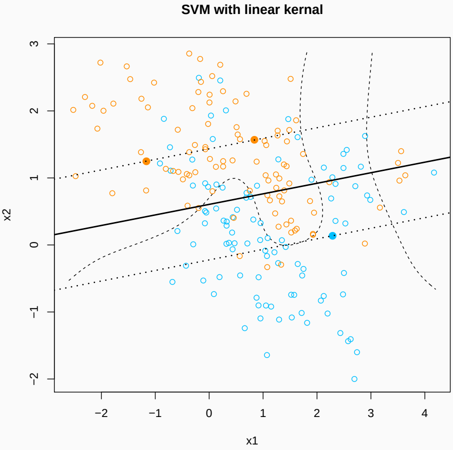
- Large C puts more weight on misclassification rate than margin width
- Small C puts more attention on data further away from the boundary
- Cross-validation to select C


Misclassified point
ξ>1
Margin
Support Vector
b
Support Vector
ξ=0
w
$w^T\phi(x)+b = -1$
ξ<1
$w^T\phi(x)+b = 0$
$w^T\phi(x)+b = +1$

Non-linear SVM and Kernel Trick

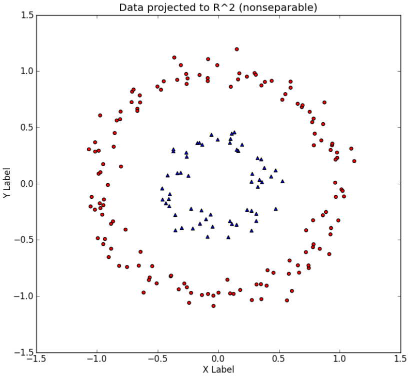
Flexible Classifiers

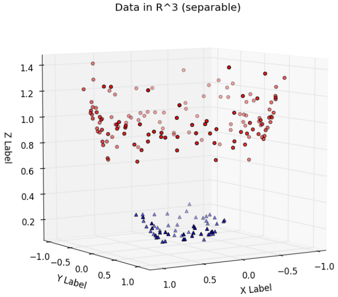
- In many cases, linear classifier is not flexible enough
- An example from the HTF text book:



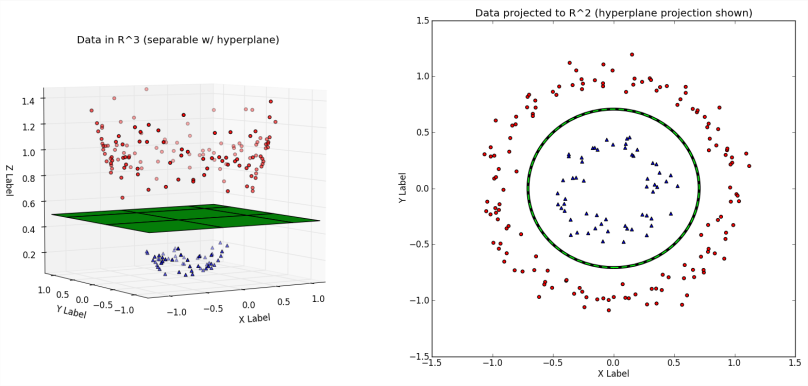
- How do we create nonlinear boundaries?

Flexible Classifiers





Flexible Classifiers



Kernel Trick

- Recall the KNN method: $\hat{y}^* = \sum_{i=1}^{n} y_i k(x_i, x^*)$
    - $k(x_i, x^*)$ referes to a “distance”/“similarity” measure
    - In regular KNN, $k(x_i, x^*) = \mathbf{1}\{i \in N_k(x^*)\}$
    - Can be extended to a weighted version: $\hat{y}^* = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x^*)$
- Instead of estimating exact model parameters, it’s also equivalent to estimate those α_i’s.
- This $k(\cdot, \cdot)$ refers to a kernel function.

Dual form of the SVM

- For the SVM, it is equivalent to solve the following dual problem:

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \ i = 1, \dots, n. \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

- Obtain $\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \alpha_i y_i x_i$

- Those points for which $\alpha_i > 0$ are called "support vectors"

- Note that I write $\langle x_i, x_j \rangle$ instead of $x_i^\mathsf{T} x_j$. This will come with more advantage later on.

Flexible Classifiers

- Enlarge the feature space via basis expansions: map into the feature space

$$\Phi : \mathcal{X} \to \mathcal{F}, \;\; \Phi(x) = (\phi_1(x), \phi_2(x), \ldots)$$

  where $\mathcal{F}$ has finite or infinite dimensions.
- The decision function becomes

$$f(x) = \langle \Phi(x), \boldsymbol{\beta} \rangle$$

- Kernel trick: only the inner product matters

$$K(x, z) = \langle \Phi(x), \Phi(z) \rangle$$

  we do not need to explicitly calculate the mapping Φ.

Kernel trick

- An example: suppose we want to include all (just) second order terms of all variables
- Consider a kernel function $K(x,z) = (x^{\mathsf{T}}z)^2$, where both x and z are p dimensional vector.
- Its easy to see that

$$
\begin{aligned}
K(x,z) = & \left(\sum_{k=1}^{p} x_k z_k\right)\left(\sum_{l=1}^{p} x_l z_l\right) \\
= & \sum_{k=1}^{p}\sum_{l=1}^{p} x_k z_k x_l z_l \\
= & \sum_{k,l=1}^{p} (x_k x_l)(z_k z_l) \\
= & \langle \Phi(x), \Phi(z) \rangle
\end{aligned}
$$

- For the last line, we define $\Phi(x)$ as a vector consists of all $(x_k x_l)$ for $1 \leq k, l \leq p$, which are just the second order terms of all variables

Kernel trick

- What is the advantage here?
- Calculating this kernel distance requires doing p products and square the sum, if the length of x is p. So the computation time is $\mathcal{O}(p)$
- However, calculating $\langle \Phi(x_i), \Phi(x_j) \rangle$ directly for subject pair (i, j) would require p^2 for either $\Phi(x_i)$ or $\Phi(x_j)$ (because this is a large vector), then again calculating the inner project. The computation time is $\mathcal{O}(p^2)$

Kernel trick

- All its left for us is to find a proper kernel function, and use that in the SVM
- Popular choices of Kernels:
  - dth degree polynomial:

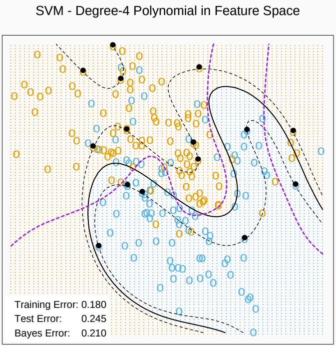
$$K(x_1, x_2) = (1 + x_1^\intercal x_2)^d$$

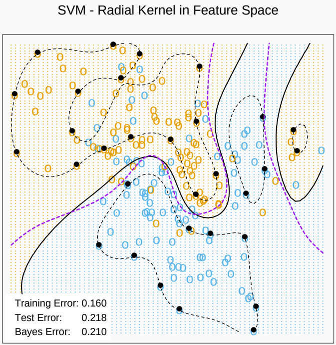
  - Radial basis:

$$K(x_1, x_2) = \exp(-\|x_1 - x_2\|^2 / c)$$

- Be careful that for $\Phi(x)$ to exist, $K(\cdot, \cdot)$ cannot be arbitrary.

Polynomial and Radial Kernels





SVM as a Penalization Method

Loss + Penalty

- Recall that SVM with soft margin is trying to solve

$$\begin{aligned} & \text{minimize} \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(x^\mathsf{T}\boldsymbol{\beta} + \beta_0) \geq (1 - \xi_i),\ i = 1, \ldots, n, \\ & \xi_i \geq 0,\ i = 1, \ldots, n, \end{aligned}$$

- We can consider letting $f(x) = x^\mathsf{T}\boldsymbol{\beta} + \beta_0$, and treat $1 - y_i(x^\mathsf{T}\boldsymbol{\beta} + \beta_0)$ as a certain loss, we reach to a penalized loss framework:

$$\text{minimize} \sum_{i=1}^{n} \big[1 - y_i f(x_i)\big]_+ + \lambda\|\boldsymbol{\beta}\|^2$$

- "Loss L + Penalty $P(\beta)$", the regularization parameter $\lambda = 1/C$.
- No constrains, same solution as the SVM

Loss + Penalty

- The loss function that we are using is not the squared loss, its called the Hinge loss
- Hinge Loss

$$L(y, f(x)) = [1 - yf(x)]_+ = \max(0, 1 - yf(x))$$

- However, this Hinge loss is not differentiable. There are some other loss functions for classification purpose:
- Logistic loss:

$$L(y, f(x)) = \log(1 + e^{-yf(x)})$$

- Modified Huber Loss:

$$L(y, f(x)) = \begin{cases} \max(0, 1 - yf(x))^2 & \text{for} \quad yf(x) \geq -1 \\ -4yf(x) & \text{otherwise} \end{cases}$$

Loss + Penalty

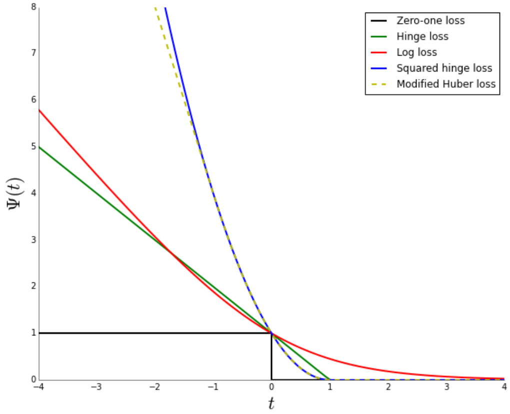
- Some other losses that we have seen before:
- Squared error loss

$$L(y, f(x)) = (1 - yf(x))^2$$

- 0/1 loss

$$L(y, f(x)) = \mathbf{1}\{yf(x) \geq 0\}$$

Comparing loss functions



Comparing loss functions

- Since Hinge Loss is not differentiable, we cannot use gradient methods, but a sub-gradiant exist
- Logistic loss, Modified Huber Loss and Squared error loss can be solved using gradient decent
- These methods will be faster and maybe preferred when solving a large system
- 0/1 loss is hard to implement since it is not continuous

R packages and functions

- R packages:
  - e1071 : function svm
  - kernlab : function ksvm
  - svmpath : compute the entire regularized solution path
  - quadprog : solving quadratic programming problems (primal or dual)
- Machine learning R packages overview: `cran.r-project.org/web/views/MachineLearning.html`