STAT 432: Basics of Statistical Learning

Support Vector Machines

Shiwei Lan, Ph.D. <shiwei@illinois.edu>

http://shiwei.stat.illinois.edu/stat432.html

April 10, 2019

University of Illinois at Urbana-Champaign

Outline

- Linear SVM in Separable Case (separation margin)
- From Primal to Dual
- Non-linear SVM (Kernel trick)
- Linear SVM in non-Separable Case (soft margin and slack variables)

Overview

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

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

Linear SVM in Separable Case

Binary Large-Margin Classifiers

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

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

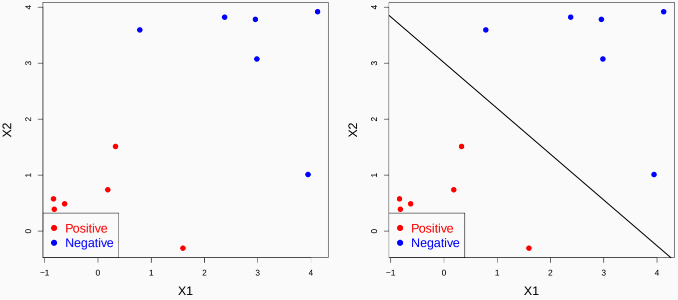
- We have a correct classification if $y_i f(x_i) > 0$
- Functional margin $y_i f(x_i)$:
  - positive means good (at the correct side)
  - negative means bad (at the wrong side)

Separating Line

- Linearly separable: we can find a line

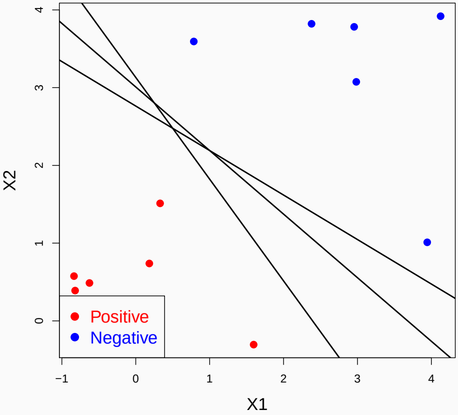
$$\beta_0 + x^\mathsf{T}\boldsymbol{\beta} = 0$$

to separate two groups of points



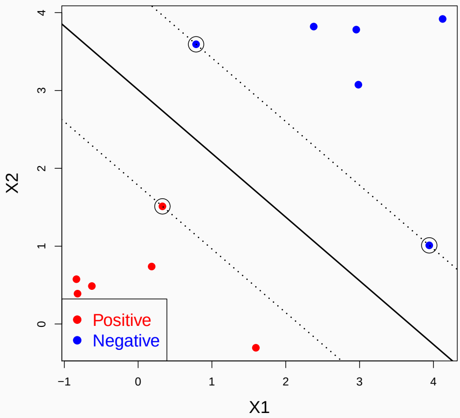
Separating Line

- Which line is the best?



Maximum Separation

- SVM searches for a line by maximizing the margin



Preliminary

- How to calculate the distance to a line?
- Suppose there is vector $\boldsymbol{\beta}$ in a p-dimensional space, then the hyperplane defined as

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

  is orthogonal to $\boldsymbol{\beta}$, for any β_0.
- A normalized vector $\boldsymbol{\beta}^* = \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}$ points to the same direction as $\boldsymbol{\beta}$.
- For any vector $\mathbf{v}$ in the p-dimensional space, its length on this direction is the inner product

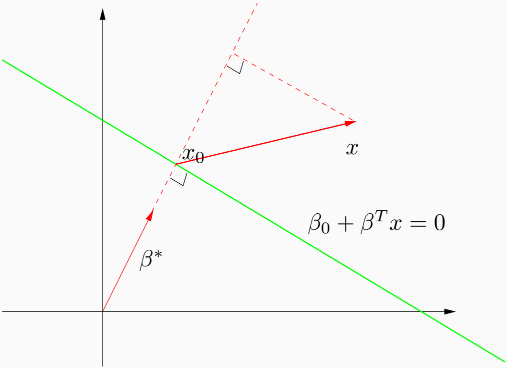
$$\langle \boldsymbol{\beta}^*, \mathbf{v} \rangle = \left\langle \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}, \mathbf{v} \right\rangle$$

Signed Distance to the Hyperplane

- Let's first pick any point x_0 on the hyperplane, hence,

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

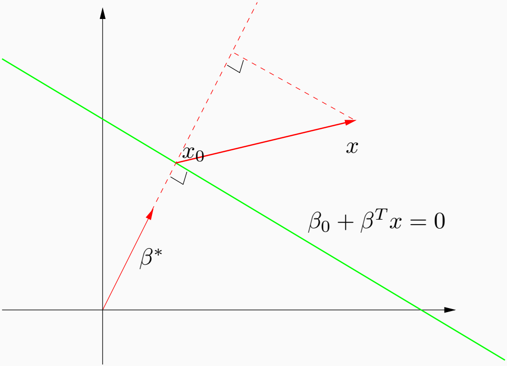
- The projection of $x - x_0$ onto the direction of $\boldsymbol{\beta}^*$ is the distance from x to the separating hyperplane.



Signed Distance to the Hyperplane

- This is the signed distance of x to the hyperplane, defined as

$$\left\langle \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}, x - x_0 \right\rangle$$



Signed Distance to the Hyperplane

- In SVM, we define a linear function $f(x) = \beta_0 + x^\mathsf{T}\boldsymbol{\beta}$ that will be used for classification
- The affine space (hyperplane) L is defined as

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

- The points on one side of this hyperplane will be labeled as $+1$ and the points on the other side will be labeled as -1.
- To calculate which side a given point is on, we first get the normalized vector (perpendicular) to L

$$\boldsymbol{\beta}^* = \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}$$

Signed Distance to the Hyperplane

- Then, 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

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

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)

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

- 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 $f(x_i) \leq -M$.
- Interpretation: All the points are at least a signed distance M from the decision boundary
- Maximize the minimum distance M (margin)

Maximum Margin Classifier

- The problem requires the constraint $\|\boldsymbol{\beta}\| = 1$, otherwise we can artificially increase the margin. However, equality constrained optimizations can be difficult.
- To get rid of this, we replace the conditions of the margin with

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

- Since the scale of $\boldsymbol{\beta}$ does not affect the optimization (classification rule), we can arbitrarily set $\|\boldsymbol{\beta}\| = 1/M$.
- Hence, we can change the original problem into

$$\max_{\boldsymbol{\beta}, \beta_0, \|\boldsymbol{\beta}\| = 1/M} \quad M$$

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

Maximum Margin Classifier

- Then, maximizing M is the same as minimizing $\|\boldsymbol{\beta}\|$.
- Hence, we solve

Linear separable SVM primal problem

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

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

- Recall our previous derivation of the signed distance, this is requiring that all points are at least $1/\|\boldsymbol{\beta}\|$ away from the hyperplane.
- $1/2$ is added for convenience.

From Primal to Dual

Optimization Problem for SVM

- Solve for parameters $\boldsymbol{\beta}$ and β_0 in the primal form

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

  is still a difficult task.
- For any given dataset $\{x_i, y_i\}_{i=1}^n$, can you directly give a workable solution that at least satisfies the constrains?

Equality Constrained Optimization Problem

- Consider an equality constrained optimization problem:

$$\begin{aligned} \text{minimize}_{\boldsymbol{\theta}} \quad & g(\boldsymbol{\theta}) \\ \text{subject to} \quad & h(\boldsymbol{\theta}) = 0 \end{aligned}$$

- $g(\boldsymbol{\theta})$: objective function
- $h(\boldsymbol{\theta})$: equality constrain(s)
- $\mathcal{S} = \{\boldsymbol{\theta} : h(\boldsymbol{\theta}) = 0\}$: feasible set
- feasible point: a point in the feasible set

Lagrange Multiplier

- Define the Lagrangian

$$\mathcal{L} = g(\boldsymbol{\theta}) + \alpha h(\boldsymbol{\theta})$$

  where α is called the Lagrange multiplier.
- There must exist a $\alpha = \alpha_0$ corresponds to a stationary point $(\boldsymbol{\theta}_0, \alpha_0)$ of $\mathcal{L}(\boldsymbol{\theta}, \alpha)$
- Intuition:
  - For every $\boldsymbol{\theta}$ such that $h(\boldsymbol{\theta}) = 0$, $\nabla h(\boldsymbol{\theta})$ is orthogonal to the surface defined by the feasible set;
  - If $\boldsymbol{\theta}_0$ is a local minimum, then $\nabla g(\boldsymbol{\theta}_0)$ must also be orthogonal to the surface of $h(\boldsymbol{\theta})$ at $\boldsymbol{\theta}_0$ — otherwise we would move along that surface and reach a smaller value
- This leads to the conclusion that the gradients $\nabla h(\boldsymbol{\theta})$ and $\nabla g(\boldsymbol{\theta})$ have to be parallel at $\boldsymbol{\theta}_0$:

$$\nabla g(\boldsymbol{\theta}_0) = -\alpha \nabla h(\boldsymbol{\theta}_0)$$

Inequality Constrained Optimization Problem

- Consider an inequality constrained optimization problem:

$$\begin{aligned} \text{minimize}_{\boldsymbol{\theta}} \quad & g(\boldsymbol{\theta}) \\ \text{subject to} \quad & h_i(\boldsymbol{\theta}) \leq 0, \ \text{for all } i = 1, \ldots n \end{aligned}$$

- Consider a generalized version of Lagrangian

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\alpha}) = g(\boldsymbol{\theta}) + \sum_{i=1}^{n} \alpha_i h_i(\boldsymbol{\theta})$$

- Find the stationary point (minimum) of $\mathcal{L}$, with two arguments $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$

Primal to Dual Problem

- Lets look at this problem from two different ways:
- If we maximize α_i's first (for a fixed $\boldsymbol{\theta}$):

$$\max_{\boldsymbol{\alpha} \succeq 0} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\alpha})$$

- In this case, if $\boldsymbol{\theta}$ violates any of the constraints, i.e., $h_i(\boldsymbol{\theta}) > 0$ for some i, we can choose an extremely large α_i such that the above quantity is ∞.
- Hence, we can consider the primal problem

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha} \succeq 0} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\alpha})$$

- The solution of this has to satisfy all the constraints, and $g(\boldsymbol{\theta})$ is minimized

Primal to Dual Problem

- If we minimize θ first, then maximize for α, we would get the dual problem

$$\max_{\boldsymbol{\alpha} \succeq 0} \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\alpha})$$

- The two are generally not the same

$$\underbrace{\max_{\boldsymbol{\alpha} \succeq 0} \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\alpha})}_{\text{dual}} \leq \underbrace{\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha} \succeq 0} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\alpha})}_{\text{primal}}$$

- However, they are the same if (sufficient)
  - both g and h_i's are convex
  - and the constraints h_i's are feasible
- A convex optimization problem.
- Further reading: The Karush-Kuhn-Tucker (KKT) conditions (this is also used in theory of Lasso solution)

From Primal to Dual: Formulation

- Now we are finally in a position to solve the dual problem, the original primal can be written as

$$\begin{aligned} \underset{\boldsymbol{\beta}, \beta_0}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2 \\ \text{subject to} \quad & -\{y_i(x_i^\mathsf{T}\boldsymbol{\beta} + \beta_0) - 1\} \leq 0, \ \ i = 1, \ldots, n. \end{aligned}$$

- Lagrangian for our optimization problem is

$$\mathcal{L}(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \sum_{i=1}^n \alpha_i \{y_i(x_i^\mathsf{T}\boldsymbol{\beta} + \beta_0) - 1\}$$

- Instead of solving this using the primal, we solve for the dual, which first minimize $\mathcal{L}(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha})$ with respect to $\boldsymbol{\beta}$ and β_0, then maximize over $\boldsymbol{\alpha}$.

Solving the Dual Problem

- To solve for $\boldsymbol{\beta}$ and β_0, we take derivatives with respect to them:

$$\boldsymbol{\beta} - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \quad (\nabla_{\boldsymbol{\beta}} \mathcal{L} = 0)$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad (\nabla_{\beta_0} \mathcal{L} = 0)$$

- Take the solutions of $\boldsymbol{\beta}$ and β_0 and plug back into the Lagrangian, we have

$$\mathcal{L}(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^{\mathsf{T}} x_j$$

Solving the Dual Problem

- We need to then maximizing over α
- This leads to the dual optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^\intercal x_j \\ \text{subject to} \quad & \alpha_i \geq 0, \ i = 1, \ldots, n. \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

- This is another quadratic programming problem.
- Now, can you think of a workable solution that at least satisfies the constrain?
- There are additional advantages (kernel trick coming soon)

Linear SVM algorithm (dual form)

- The SVM problem for separable case can be carried out as follows:
  - Solve dual for α_i's (those points with $\alpha_i > 0$ are called "support vectors")
  - Obtain $\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \alpha_i y_i x_i$
  - Obtain β_0 by calculating the midpoint of two "closest" support vectors to the separating hyperplane

$$\widehat{\beta}_0 = -\ \frac{\max_{i:y_i=-1} x_i^\mathsf{T} \widehat{\boldsymbol{\beta}} + \min_{i:y_i=1} x_i^\mathsf{T} \widehat{\boldsymbol{\beta}}}{2}$$

  - For any new observation x, the prediction is

$$\text{sign}\left(x^\mathsf{T} \widehat{\boldsymbol{\beta}} + \widehat{\beta}_0\right)$$

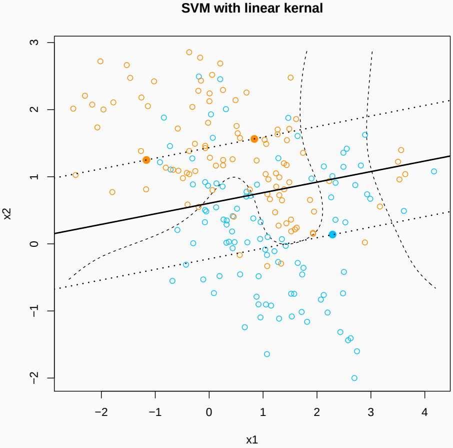
Remarks

- If the classes are really Gaussian, then
  - LDA is optimal
  - The LDA separating hyperplane pays a price for being influenced by noisier data
- SVM optimal separating hyperplane has less assumptions, thus more robust to model mis-specification
  - The logistic regression solution can be similar to the operating hyperplane
  - However, for perfectly separable case, the likelihood is infinity, hence logistic regression does not work

Non-linear SVM and Kernel Trick

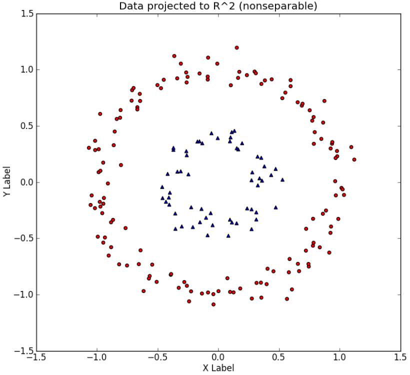
Flexible Classifiers

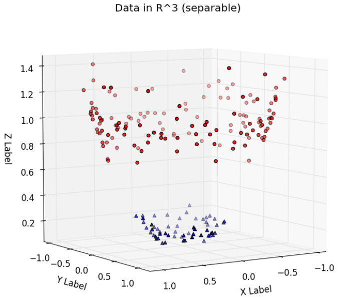
- In many cases, linear classifier is not flexible enough
- An example from the HTF text book:



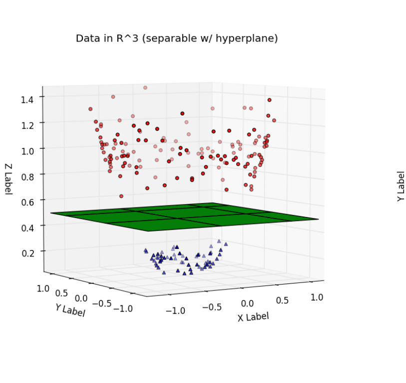
- How do we create nonlinear boundaries?

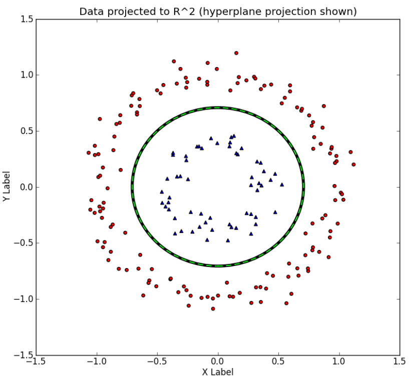
Flexible Classifiers





Flexible Classifiers





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

- Naive approach: If we know $\Phi(x)$, we could calculate it for all x_i's, treat them as the new features, and optimize

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle \Phi(x_i), \Phi(x_j) \rangle \\ \text{subject to} \quad & \alpha_i \geq 0, \ i = 1, \ldots, n. \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

- However, this is not necessary.
- Kernel trick saves computation time!

Kernel trick

- An example: suppose we want to include all (just) second order terms of all variables:

$$x_k x_l \text{ for all } k, l = 1, \ldots, p$$

- We define $\Phi(x)$ as a vector consists of all $(x_k x_l)$'s
- To calculate the inner product for two observations x and z, we need

$$\langle \Phi(x), \Phi(z) \rangle = \sum_{k,l=1}^{p} (x_k x_l)(z_k z_l)$$

- The complexity for calculating this is $\mathcal{O}(p^2)$.

Kernel trick

- Consider a kernel function $K(x, z) = (x^{\mathsf{T}} z)^2$
- Its easy to see that

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

- The complexity is $\mathcal{O}(p)$.

Kernel trick

- Calculating $\langle \Phi(x_i), \Phi(x_j) \rangle$ directly for subject pair (i, j) would require p^2 multiplications for both $\Phi(x_i)$ and $\Phi(x_j)$ (because this is a large vector), then again calculating the inner project. The computation time is $\mathcal{O}(p^2)$.
- Calculating the kernel distance requires doing p products in $x^{\mathsf{T}}z$ and square the sum. So the computation time is $\mathcal{O}(p)$.
- This saves a lot of computational time, and it is one of the reasons that we use dual instead of primal.

Kernel trick

Separable SVM with Kernel Trick

$$
\begin{aligned}
\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\
\text{subject to} \quad & \alpha_i \geq 0, \; i = 1, \ldots, n. \\
& \sum_{i=1}^{n} \alpha_i y_i = 0
\end{aligned}
$$

Kernel trick

- So, for any given $\Phi(x)$, how do we find the corresponding kernel?
- That is difficult...
- However, for any properly defined kernel function, by Mercer's theorem, we know that it corresponds to some feature mapping construction $\Phi(x)$.
- This requires $K(\cdot, \cdot)$ to be symmetric, and the corresponding kernel matrix ($n \times n$ matrix for all pairwise distance of n samples) is positive semi-definite
- There are numerous articles about Mercer's theorem and related concept, the Reproducing Kernel Hilbert Space

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:

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

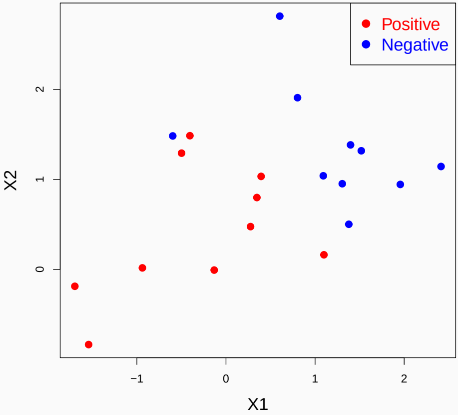
  - Radial basis:

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

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

Linear SVM in non-Separable Case

Linearly non-Separable



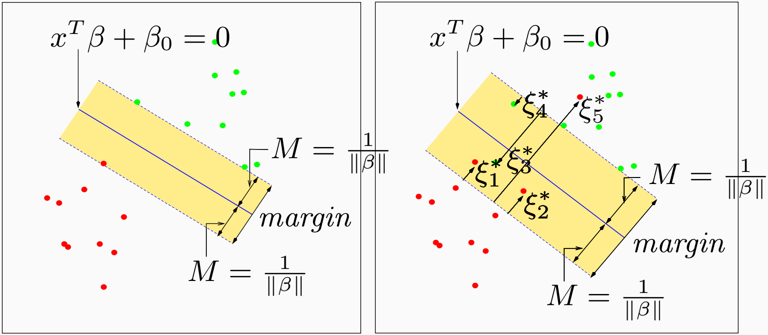
General Case for SVM

- 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

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

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

Solving SVM with Slack Variables

- The new optimization problem does nothing but putting more constraints

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

- We can again write the Lagrangian primal $\mathcal{L}(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}, \boldsymbol{\xi})$ is

$$\frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\big\{y_i(x_i^\mathsf{T}\boldsymbol{\beta} + \beta_0) - (1 - \xi_i)\big\} - \sum_{i=1}^{n}\gamma_i\xi_i$$

where α_i, $\gamma_i \geq 0$.

Solving SVM with Slack Variables

- It is trivial now to get the derivatives:

$$
\begin{aligned}
\boldsymbol{\beta} - \sum_{i=1}^{n} \alpha_i y_i x_i &= 0 \quad (\nabla_{\boldsymbol{\beta}} \mathcal{L} = 0) \\
\sum_{i=1}^{n} \alpha_i y_i &= 0 \quad (\nabla_{\beta_0} \mathcal{L} = 0) \\
C - \alpha_i - \gamma_i &= 0 \quad (\nabla_{\xi_i} \mathcal{L} = 0)
\end{aligned}
$$

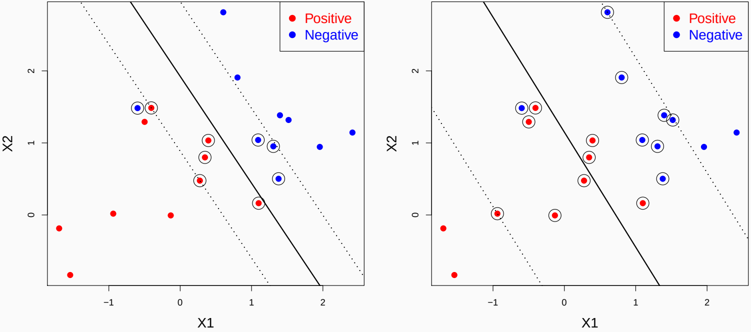
Solving SVM with Slack Variables

- Substituting them back into the Lagrangian, we have the dual form

$$
\begin{aligned}
\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \\
\text{subject to} \quad & 0 \leq \alpha_i \leq C, \; i = 1, \ldots, n. \\
& \sum_{i=1}^{n} \alpha_i y_i = 0
\end{aligned}
$$

- Note that we write $\langle x_i, x_j \rangle$ instead of $x_i^\mathsf{T} x_j$, because we can again use the kernel trick.

Linearly non-Separable



The support vectors for linearly non-separable case

Remark

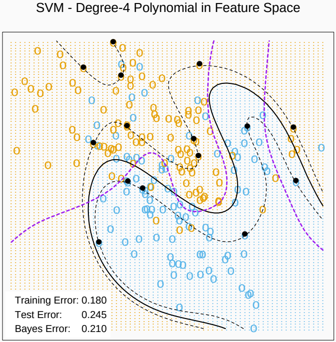
- Large C puts more weight on misclassification rate than margin width
- Small C puts more attention on data further away from the boundary
- Cross-validation to select C

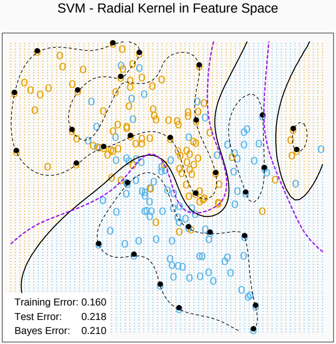
Kernel trick

Soft-Margin SVM with Kernel Trick

$$\begin{aligned}
\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\
\text{subject to} \quad & 0 \le \alpha_i \le C,\ \ i = 1, \ldots, n. \\
& \sum_{i=1}^{n} \alpha_i y_i = 0
\end{aligned}$$

Polynomial and Radial Kernels





Convexity of SVM

- Is SVM a convex (taking out the negative sign) optimization problem? (especially after the Kernel trick)

$$\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{1}$$
$$= \boldsymbol{\alpha}^\mathsf{T} \,\mathrm{diag}(\mathbf{y})\, \mathbf{K} \,\mathrm{diag}(\mathbf{y})\, \boldsymbol{\alpha}$$

- Convexity will be guaranteed if the Kernel matrix $\mathbf{K}$ is positive semidefinite.
- **Mercer's theorem:** The kernel matrix $\mathbf{K}$ is positive semidefinite iff the function $K(x_i, x_j)$ is equivalent to some inner product $\langle \Phi(x_i), \Phi(x_j) \rangle$.

SVM as a Penalization Method

Loss + Penalty

- Recall that SVM with soft margin is trying to solve

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

- 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 a penalized loss framework:

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

- "Loss + Penalty", the regularization parameter $\lambda = \frac{1}{2C}$.
- 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

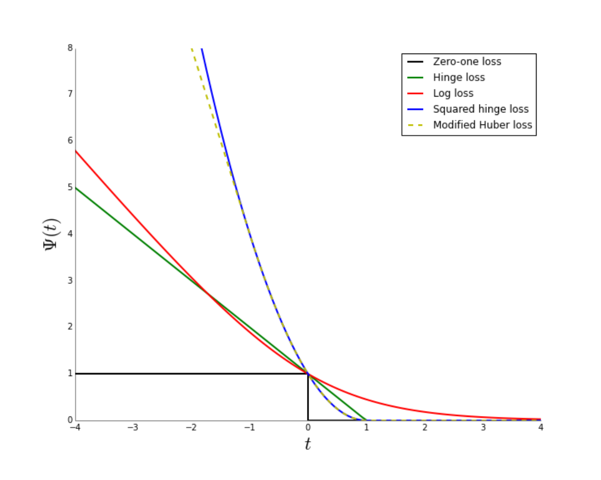
- Some other losses that we have seen before:
- Squared error loss

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

- 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

Nonlinear SVM

- Again, we might want to consider nonlinear decision functions. A nonlinear SVM (with hinge loss) solves

$$\min_f \sum_{i=1}^{n} [1 - y_i f(x_i)]_+ + \lambda \|f\|^2_{\mathcal{H}_\mathcal{K}}$$

  where f (nonlinear) belongs to a reproducing kernel Hilbert space $\mathcal{H}_\mathcal{K}$, which is determined by the kernel function K, and $\|f\|^2_{\mathcal{H}_\mathcal{K}}$ denotes the corresponding norm.
- This space can be very large, however, the solution to this can be simple (Representer Theorem: Kimeldorf and Wahba, 1970), and takes the following form

$$\begin{aligned}&\arg\min_{f \in \mathcal{H}_\mathcal{K}} \sum_{i=1}^{n} [1 - y_i f(x_i)]_+ + \lambda \|f\|^2_{\mathcal{H}_\mathcal{K}} \\ &= \beta_1 K(x, x_i) + \cdots + \beta_n K(x, x_n)\end{aligned}$$

Representer Theorem

- Hence the optimization becomes

$$\sum_{i=1}^{n} L(y_i, \mathbf{K}_i^{\mathsf{T}} \boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^{\mathsf{T}} \mathbf{K} \boldsymbol{\beta},$$

  where k is the kernel matrix with $\mathbf{K}_{ij} = K(x_i, x_j)$, and $\mathbf{K}_i$ is the i the column of $\mathbf{K}$
- An unconstrained optimization problem
- Can use gradient decent if L is differentiable
- So this is a ridge penalty? and we won't get sparse solution?

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