Statistical Learning

Support Vector Machines

Spring 2024

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 $f(x) \in \mathbb{R}$
- The classification rule $C(x) = \text{sign}\{f(x)\}$ outputs the label
- The optimal classifier is:

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

Outline

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

Linear SVM in Separable Case

Binary Large-Margin Classifiers

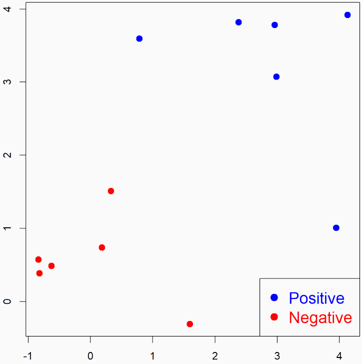
- Since $y_i \in \{-1, 1\}$, our classification rule using $f(x)$ is

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

- 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)

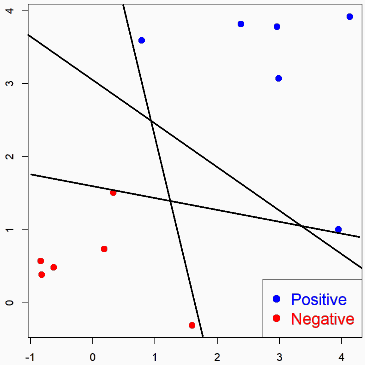
Separating Line

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



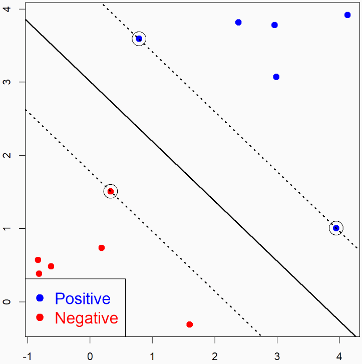
Separating Line

- 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



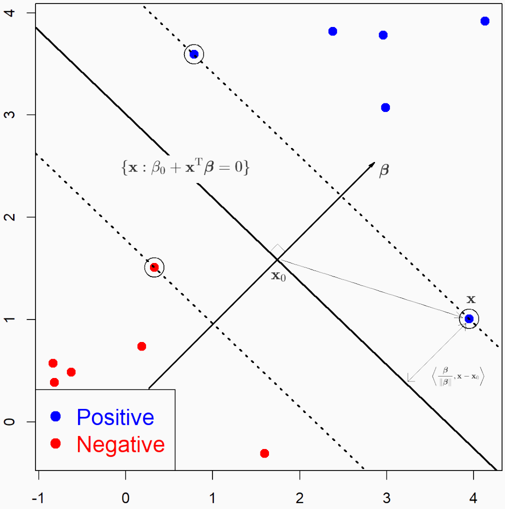
Signed Distance to the Hyperplane

- Define a linear function $f(x) = \beta_0 + x^{\mathsf{T}}\boldsymbol{\beta}$
- We define the (linear) separating hyperplane is

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

- Signed distance of x to the plane is $\langle \frac{\boldsymbol{\beta}}{||\boldsymbol{\beta}||}, x - x_0 \rangle$

Signed Distance to the Hyperplane



Signed Distance to the Hyperplane

- The unit vector (perpendicular) to L is $\boldsymbol{\beta}^* = \boldsymbol{\beta}/\|\boldsymbol{\beta}\|$
- For any point $x_0 \in L$, we have

$$f(x_0) = 0 \;\Leftrightarrow\; x_0^\mathsf{T}\boldsymbol{\beta} = -\beta_0$$

- The signed distance from 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 maximize the distance to the closest points from either class (Vapnik 1996)

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

- Interpretation: All the points are at least a signed distance M from the decision boundary
  - 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

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

- 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

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

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

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.
- 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 $\boldsymbol{\theta}^*$ is a local minimum, then $\nabla g(\boldsymbol{\theta})$ is orthogonal to the surface at $\boldsymbol{\theta}^*$ — 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 $\boldsymbol{\theta}^*$:

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

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

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

- $\mathcal{L}$ has 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, I 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 are sufficient and necessary for a global solution

From Primal to Dual: Formulation

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

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

$$\text{subject to} \quad -\{y_i(x_i^\mathsf{T}\boldsymbol{\beta}+\beta_0)-1\} \leq 0,\ \ i=1,\ldots,n.$$

- Lagrangian for our optimization problem is

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

- 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 maximize over α
- This leads to the dual optimization problem:

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

- This is another quadratic programming problem
- 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 for which $\alpha_i > 0$ 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

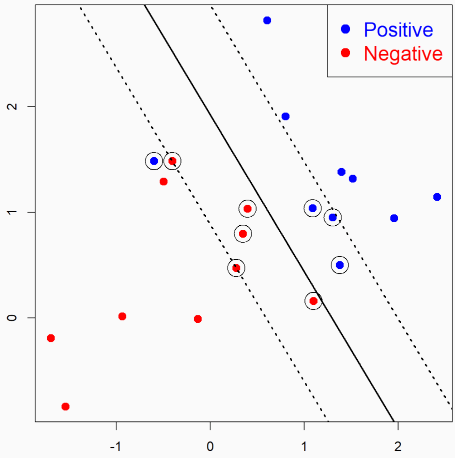
$$\text{sign}\left(x^{\mathsf{T}} \widehat{\boldsymbol{\beta}} + \widehat{\beta}_0\right)$$

Remarks

- If the classes are really Gaussian, then
  - LDA is optimal
  - The separating hyperplane pays a price for focusing on the noisier data at the boundaries
- Optimal separating hyperplane has fewer 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



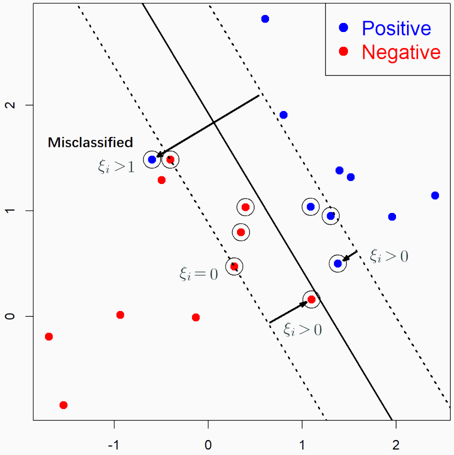
General Case for SVM

- 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

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

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

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

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

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

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

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

- Note that I write $\langle x_i, x_j \rangle$ instead of $x_i^\top x_j$. This will come with more advantages later on.

Support Vectors

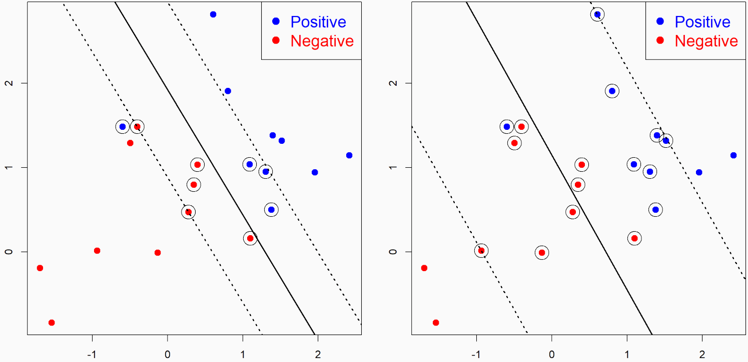
- We can still obtain
$$\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \alpha_i y_i x_i$$
- Based on the KKT condition, we need

$$\begin{aligned}
\alpha_i \big\{ y_i (x_i^\mathsf{T} \boldsymbol{\beta} + \beta_0) - (1 - \xi_i) \big\} &= 0 \\
\gamma_i \xi_i &= 0 \\
y_i (x_i^\mathsf{T} \boldsymbol{\beta} + \beta_0) - (1 - \xi_i) &\geq 0
\end{aligned}$$

Support Vectors

- There are eventually three sets of observations:
  - Useless points: $\alpha_i = 0$ and $\xi_i = 0$
  - Support vectors: $0 < \alpha_i < C$ and $\xi_i = 0$
  - Support vectors: $\alpha_i = C$ and $\xi_i = 1 - y_i(x_i^\mathsf{T}\boldsymbol{\beta} + \beta_0) > 0$
- After obtaining the support vectors, we can extract the ones on the positive side and the negative side.
- We can calculate $y_i(x_i^\mathsf{T}\boldsymbol{\beta})$ for them separately, and the separation line (with $\widehat{\beta}_0$) lies in the middle of them.

Linearly non-Separable



The support vectors and observations on the wrong side for linearly non-separable case

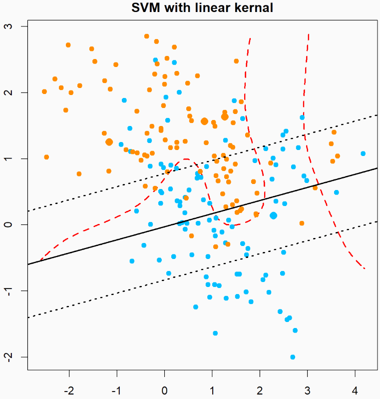
Remark

- 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

Non-linear SVM and Kernel Trick

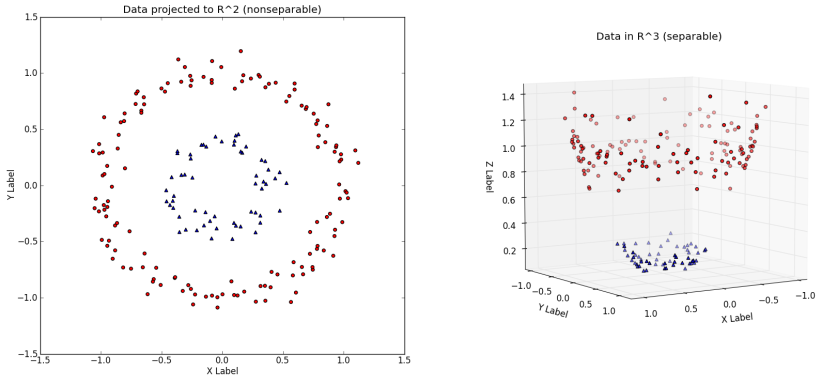
Flexible Classifiers

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

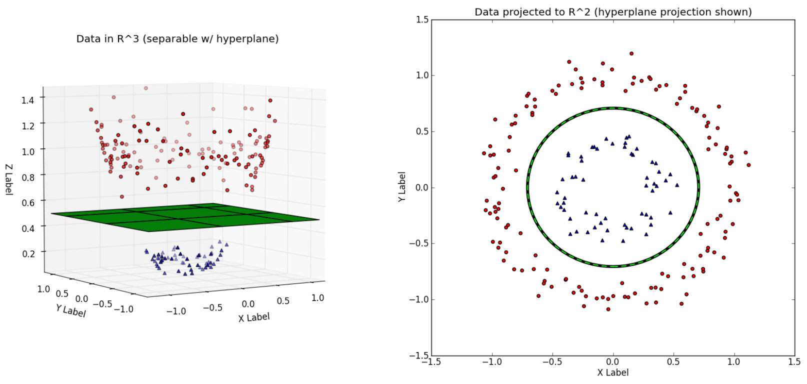


- How do we create nonlinear boundaries?

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

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

- 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

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

- 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
- Consider a kernel function $K(x, z) = (x^\mathsf{T} z)^2$, where both x and z are p dimensional vector.
- Consider alternatively, let $\Phi(x)$ be the basis expansion that consists all $x_k x_l$ for $1 \leq k, l \leq p$
- We can show that the kernel distance is essentially the same as the cross-product of basis expansions

Kernel trick

- 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 $\mathcal{O}(p^2)$
- This saves a lot of computational time, and it is the reason that we went all the way from the primal form to the dual form to solve SVM: the primal form cannot utilize the kernel trick because there is no inner product involved.

Kernel trick

- So, for any given $\Phi(x)$, how do we find the corresponding kernel?
- That is kinda tricky...
- However, for any properly defined kernel function, by Mercer's theorem, we know that it will be correspond 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:

$$K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x}^\mathsf{T}\mathbf{z})^d$$

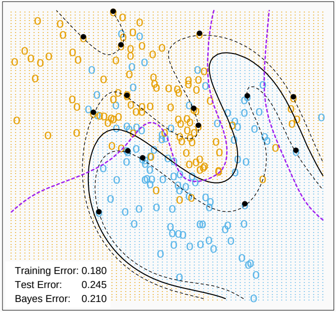
  - Radial basis:

$$K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2 / c)$$

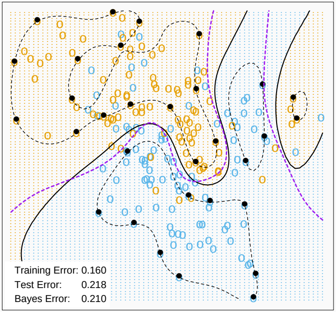
- Be careful that for $\Phi(x)$ to exist, $K(\cdot, \cdot)$ cannot be arbitrary.

Polynomial and Radial Kernels

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



Convexity of SVM

- Is SVM a convex optimization problem?

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

- 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 $\langle \Phi(x_i), \Phi(x_j) \rangle$.

Example: Gaussian Kernel

- For example, the Gaussian kernel is associated with an infinite dimensional feature map:

$$
\begin{aligned}
e^{-\gamma\|\mathbf{x}-\mathbf{z}\|^2} &= e^{-\gamma\|\mathbf{x}\|^2+2\gamma\mathbf{x}^\mathsf{T}\mathbf{z}-\gamma\|\mathbf{z}\|^2} \\
=& e^{-\gamma\|\mathbf{x}\|^2-\gamma\|\mathbf{z}\|^2}\left[1+\frac{2\gamma\mathbf{x}^\mathsf{T}\mathbf{z}}{1!}+\frac{(2\gamma\mathbf{x}^\mathsf{T}\mathbf{z})^2}{2!}+\frac{(2\gamma\mathbf{x}^\mathsf{T}\mathbf{z})^3}{3!}+\cdots\right]
\end{aligned}
$$

- $\mathbf{x}^\mathsf{T}\mathbf{z}$ is the inner product of all first order feature maps. We also showed previously $(\mathbf{x}^\mathsf{T}\mathbf{z})^2$ is equivalent to the inner product of all second order feature maps ($\Phi_2(\mathbf{x})$), and $(\mathbf{x}^\mathsf{T}\mathbf{z})^3$ would be equivalent to the third order version ($\Phi_3(\mathbf{x})$), etc.
- Hence, the feature map of Gaussian kernel is

$$
e^{-\gamma\|\mathbf{x}\|^2}\left[1,\sqrt{\frac{2\gamma}{1!}}\mathbf{x}^\mathsf{T},\sqrt{\frac{(2\gamma)^2}{2!}}\Phi_2^\mathsf{T}(\mathbf{x}),\sqrt{\frac{(2\gamma)^3}{3!}}\Phi_3^\mathsf{T}(\mathbf{x}),\cdots\right]
$$

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 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 $\lambda = 1/2C$.
- No constraints, same solution as the SVM

Loss + Penalty

- The loss function that we are using is not a squared loss or 0/1 loss, it is called the Hinge loss:

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

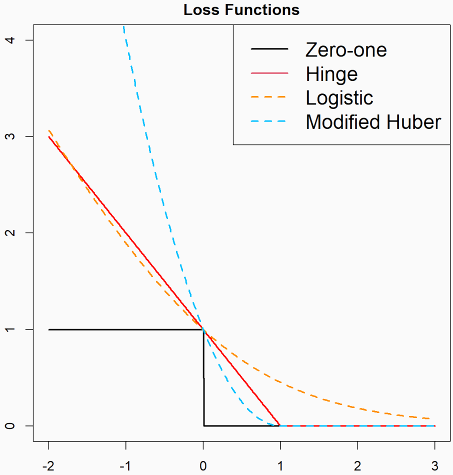
- However, Hinge loss is not differentiable. There are some other loss functions for classification purpose:
- Logistic loss:

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

- Modified Huber Loss:

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

Comparing loss functions



Comparing loss functions

- Since Hinge Loss is not differentiable, we cannot use gradient methods, but a sub-gradient exist
- Logistic loss, Modified Huber Loss and Squared error loss can be solved using gradient descent
- 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

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

  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

$$\widehat{f}(x) = \alpha_0 + \alpha_1 K(x, x_1) + \cdots + \alpha_n K(x, x_n)$$

Representer Theorem

- Hence the optimization becomes

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

  where $\mathbf{K}$ 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
- We can use gradient descent if L is differentiable

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`