1 Introduction

In the previous lectures, we have discussed prediction of data with the knowledge of a linearity assumption. But typically we might have a much smaller sample of data in a much higher-dimensional space, so we cannot blindly choose a linear model and assume it will be accurate.

If the model is too highly-parameterized, it will overfit the data, and we will fail to learn the underlying patterns. However, models with too few parameters may not even describe the training data, and will provide similarly bad performance. Hence we now look to capture nonlinear patterns in the data. These non-linear patterns can be thought of as,

- Nonlinear Regression: Relationship where the input - output may not be linear.
- Nonlinear Classification: Classes may not be separable by a linear boundary.

Consider the following example for genotype-phenotype (Haploid SNPs - Discrete Phenotype) relationship.

<table>
<thead>
<tr>
<th>SNP_A</th>
<th>SNP_B</th>
<th>Phenotype</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The above example is an XOR relationship between the genotype and phenotype hence it violates the linear assumption given by,

\[ y_i = \beta_0 + \beta_1 x_i + \epsilon_i \]

The non-linear relation is often referred to as epistasis which accounts for the unexplained heritability that is calculated in GWAS. One way to counter the violation of the linear assumption is to consider a model with higher dimensions for richer representation of the model,

\[ y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon_i \]

In general, in such situations we will have to resort to more complex (non-linear) classifiers. We shall consider two concepts which will provide us with a flexible, computationally feasible method for implementing this scheme.

1. Kernel Machines
2. Bayesian Non - Parametrics : Gaussian Process
2 Kernel Machines

2.1 Kernel Definition

Kernels are a way to represent data flexibly so that it can be used to compare the samples in a complex space. A kernel is basically an arbitrary function that maps data to a higher dimensional space enabling linearity and easier comparisons of complex features. By doing this mapping kernels enables us to utilize a linear model in the new input space, hence overcoming all issues listed in Section 1.

2.2 Kernel Function

Given some abstract space $\chi$, a kernel function is given as,

$$\kappa : \chi \times \chi \mapsto \mathbb{R}$$  \hspace{1cm} (1)

Kernel functions are used to quantify similarity between a pair of objects $x$ and $x'$ in $\chi$. A kernel function typically satisfies two properties,

1. **Symmetric**: $\forall x, x' \in \chi, \kappa(x, x') = \kappa(x', x)$
2. **Non-negative**: $\forall x, x' \in \chi, \kappa(x, x') \geq 0$

**Mercer Kernels**: Consider $X = \{x_1, \ldots, x_n\}$ be a finite set of $n$ samples from $\chi$. The Gram matrix of $X$ is given as $K_{ij} = \kappa(x_i, x_j)$. If the matrix $K$ is positive definite, then $\kappa$ is called a Mercer Kernel. A mercer kernel is symmetric by definition ($K = K^T$)

2.3 Feature Mapping

Another interpretation of kernels can be seen using feature space representation. So if $\kappa_1(x, x')$ and $\kappa_2(x, x')$ are some kernels on $\mathbb{R}^N \times \mathbb{R}^N$. To each kernel $\kappa_1$ there corresponds atleast one feature map (kernelized feature vector) given by $\Phi$, where

$$\Phi = \text{Mapping of } X \text{ to another space : } x \rightarrow \Phi(x)$$

$$\Phi(x) = [\kappa(x, \mu_1), \kappa(x, \mu_2), \ldots, \kappa(x, \mu_D)]$$

where $\mu$ – set of D centroids, corresponding to each dimension

$$\kappa(x, x') = \Phi^T(x)\Phi(x')$$

2.4 Examples of Kernel Functions

2.4.1 Quadratic Polynomial $\kappa_2$

The quadratic polynomial $\kappa_2$ is given as,

$$\kappa_2(x, x') = (1 + x^T x')^2$$  \hspace{1cm} (2)

We can show that there are functions $\Phi$ such that $\kappa(x, x') = \Phi^T(x)\Phi(x')$. We can compute the kernel explicitly,

$$\kappa_2(x, x') = (1 + x^T x')^2 = (1 + x_1'x_1 + x_2'x_2)^2$$

$$= 1 + (x_1')^2 + (x_2')^2 + 2x_1x_2'x_1x_2' + 2x_1' + 2x_2'$$

$$\Phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2) \quad \text{.. Feature Vector}$$

$$\kappa_2(x, x') = \Phi^T(x)\Phi(x')$$
2.4.2 Radial Bias Function (RBF)/Gaussian Kernel

The RBF kernel is given as,

$$
\kappa(x, x') = e^{-\frac{||x - x'||^2}{2\sigma^2}}
$$

(3)

The feature vector of a RBF kernel is of infinite size due to the exponential series representation.

2.5 Properties of Kernels

Kernels have properties that make it easier to construct complex kernels using simpler kernels. Some of these properties are given below. Consider any space $X$ of samples and kernels $\kappa_1(\ldots)$ and $\kappa_2(\ldots)$ over $X$. Then $\kappa(\ldots)$ is also a kernel when,

1. $\kappa(x, x') = \kappa_1(x, x') + \kappa_2(x, x')$
2. $\kappa(x, x') = a \kappa_1(x, x')$ where $a > 0$
3. $\kappa(x, x') = f(x).f(x')$ for any function $f$ on $x$
4. $\kappa(x, x') = \frac{\kappa_1(x, x')}{\sqrt{\kappa_1(x, x)} \sqrt{\kappa_1(x', x')}}$

2.6 Kernelized Ridge Regression

In the previous lectures, we developed a linear form of ridge regression. We now seek to obtain a kernelized form. We already know the ridge estimator is given as,

$$
\hat{\beta}_{ridge} = X^T (XX^T + \lambda I_n)^{-1} y
$$

(4)

Suppose a feature mapping is applied to data $X$ using the feature function $\Phi(x)$ then the estimate becomes,

$$
\hat{\beta}_{ridge} = \Phi^T (\Phi\Phi^T + \lambda I_n)^{-1} y
$$

(5)

We can now define a Gram Matrix (Kernelized Inner Product - Kernel Trick) as follow,

$$
K = (\Phi\Phi^T)_{ij} = \sum_{k=1}^{d} \Phi_{ik}\Phi_{jk}
$$

(6)

$$
K_{ij} = \Phi(x_i)^T \Phi(x_j)
$$

(7)

Using the gram matrix we have can have a kernelized ridge regression represented as,

$$
\hat{\beta}_{ridge} = \sum_{i=1}^{n} \alpha_i \Phi_i
$$

(8)

$$
\alpha_i = (K + \lambda I_n)^{-1} y
$$

(9)

Using the above definition we can predict a new data point as follows,

$$
y_* = \hat{\beta}_{ridge}^T \Phi(x_*)
$$

$$
= \sum_{i=1}^{n} \alpha_i \Phi_i^T \Phi_*)
$$

$$
= \sum_{i=1}^{n} \alpha_i \kappa(x_*, x_i)
$$
Kernels in Bioinformatics

Kernels can be used in construction Linear mixed models in Association Studies of Rare Variants or Gene Set. The model would look like,

\[ y \sim N(0, \sigma^2 \mathbf{I} + \sigma^2 \kappa) \]  

(10)

Kernels can also be used in the setting of Support Vector Machines as discussed below.

3 Support Vector Machine (SVM)

3.1 SVM Problem

Support Vector Machines (SVMs) are a popular learning tool for designing classifier function. SVMs can be defined in two different settings,

- Classification - Binary valued outcome
- Regression - Real valued outcome

For the purposes of this class, we will only be considering the classification setting (binary outcome). In order to set up the SVM problem we consider \( n \) samples \((x_1, y_1) \ldots (x_n, y_n), \) where \( x_i \in \mathbb{R}^d \) and \( y_i \in \{-1, 1\} \) \forall\( i. \)

The objective function of the classification problem is given as follows,

\[ \mathbb{J}(\beta) = \min_{\beta} \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \lambda \|\beta\|^2 \]  

(11)

where,

\[ L(y_i, \hat{y}_i) = \text{Loss function} \]

\[ \hat{y}_i = \text{Classification outcome} \in \{-1, 1\} \]

\[ = \hat{\beta} \Phi(x_i) + \beta_0 \]

3.2 Choosing the Correct Loss Function

In the case where the classification outcome \( \hat{y}_i \) and the training point, \( y_i, \) have the same sign, we have a correct classification and want the loss function \( L \) to be small or zero. When there is an incorrect classification we want the loss function \( L \) to be larger.

We might consider a step function for \( L \) but the step function is not ideal as it’s derivative is undefined at \( x = 0 \) making it not differentiable and also the step function is not convex. Non-convex functions make the task of minimization harder. In order to address this deficiencies we consider the hinge loss function.

\[ L(y_i, \hat{y}_i) = \max (0, 1 - (y_i \hat{y}_i)) \]  

(12)

Using the Hinge Loss function, the loss is linear for \( y_\hat{y} < 1 \) and zero elsewhere. Given the hinge loss function, the object function becomes,

\[ \mathbb{J}(\beta) = \min_{\beta, \beta_0} \frac{1}{n} \sum_{i=1}^{n} (1 - y_i \hat{y}_i) + \lambda \|\beta\|^2 \]  

(13)
3.3 Slack Variables

The hinge loss is not differential due to the kink in $\mathcal{L}$ at $y_i \tilde{y}_i = 1$ as shown in the figure above. Hence we introduce slack variables $\xi_i$ to make the objective function differentiable and easier to optimize. Hence the new problem statement is,

$$J(\beta) = \min_{\beta, \beta_0, \xi} \frac{1}{n} \sum_{i=1}^{n} \xi_i + \lambda \|\beta\|^2_2$$  \hspace{1cm} (14)

subject to:

$$\xi_i \geq 0 \hspace{1cm} i = 1, \ldots, n$$ \hspace{1cm} (15)

(affine-inequalities): $\xi_i \geq 1 - y_i \tilde{y}_i \hspace{1cm} i = 1, \ldots, n$  \hspace{1cm} (16)

3.4 Primal SVM

In order to make the problem appear as it does in SVM literature, one modification is made to the problem statement. Use parameter $C$ instead of $\lambda$. Hence we have $C = \frac{1}{2\lambda n}$. Hence we have Primal SVM as,

$$\min_{\beta, \beta_0, \xi} C \sum_{i=1}^{n} \xi_i + \frac{1}{2} \|\beta\|^2_2$$ \hspace{1cm} (17)

subject to:

$$\xi_i \geq 0 \hspace{1cm} i = 1, \ldots, n$$ \hspace{1cm} (18)

$$\xi_i \geq 1 - (\beta_0 + \beta^T \Phi(x_i)) y_i \hspace{1cm} i = 1, \ldots, n$$ \hspace{1cm} (19)

3.5 Dual & Lagrange’s Multiplier

(17) is a convex optimization problem and can be solved using the following approach,

- **Lagrangian**: Formulate from the primal as in Lagrange multipliers.
- **Dual**: Associate one dual variable to each primal constraint in the Lagrangian. Solve the dual problem.
We associate one slack variable for each constraint (18) and (19). The Lagrangian for primal SVM is thus given as,

\[
L(\beta, \beta_0, \xi, \alpha, \varsigma) = C \sum_{i=1}^{N} \xi_i + \frac{1}{2} \|\beta\|^2 - \sum_{i=1}^{N} \varsigma_i \xi_i - \sum_{i=1}^{N} \alpha_i (\xi_i - 1 + y_i (\beta_0 + \beta^T \Phi(x_i)))
\]  

(20)

The primal problem hence is to find the minimum, with respect to the dual variables \((\alpha, \varsigma)\), of the infimax value of the Lagrangian with respect to the primal variables. Formally this can be written as,

\[
\min_{\beta, \beta_0, \xi} \max_{\alpha > 0, \varsigma > 0} L(\beta, \beta_0, \xi, \alpha, \varsigma)
\]  

(21)

For some classes of problem the \(\min \max\) ordering is interchangeable and in these cases we say that the problem has strong duality. In general for weak duality: \(\min \max L \geq \max \min L\). SVM has the property of strong duality and hence we have the dual problem for SVM,

\[
\max_{\alpha > 0, \varsigma > 0} \min_{\beta, \beta_0, \xi} L(\beta, \beta_0, \xi, \alpha, \varsigma)
\]  

(22)

Considering the dual representation of SVM in (22) we have,

\[
g(\alpha, \varsigma) = \min_{\beta, \beta_0, \xi} L
\]  

(23)

where

\[
\hat{\beta} = \sum_{i=1}^{n} \Phi(x_i) y_i \alpha_i
\]  

(24)

\[
0 \leq \alpha_i \leq C
\]  

(25)

Eq. (24) shows \(\alpha_i\) to be vector of values where the non-zero elements are called as the support vectors. Eq. (25) induces sparsity into the problem where decrease in \(C\) leads to fewer support vectors.

### 3.6 Predicting New Data Point

A new data point in SVM is predicted as follows,

\[
\hat{y}(x_*) = \hat{\beta}_0 + \hat{\beta}^T \Phi(x_n)
\]

\[
= \hat{\beta}_0 + \sum_{i=1}^{n} y_i \alpha_i \kappa(x_i, x_*)
\]

### 4 Bayesian Non-Parametrics

#### 4.1 Definition

Up until now we have considered \textbf{parametric modeling} which assumes that the data being represented by the models have a fixed finite number of parameters. Now we consider a \textbf{nonparametric modeling} approach where-in the number of parameters is not fixed, and often grows with the sample size. In Bayesian non-parametrics, the number of parameters is itself considered to be a random variable.

Bayesian non-parametric methods provide a Bayesian framework for model selection and adaptation using nonparametric models which isn’t provided in the models given above or in previous lectures.
4.2 Gaussian Process

A Gaussian process defines a distribution over functions, \( f \), where \( f \) is a function mapping some input space \( X \) to \( \mathbb{R} : f : X \to \mathbb{R} \). Here \( f \) is an infinite-dimensional quantity. Given an arbitrary set of points \( x_1, \ldots, x_n \) we define \( f \) to be an \( n \)-dimensional vector of function values evaluated at \( n \) points of \( X \) and is given by,

\[
\mathbf{f} = (f(x_1), f(x_2), \ldots, f(x_n))
\]  

Consider the distribution of the function \( f \) be given as \( P(f) \). \( P(f) \) is a Gaussian process if for any finite subset \( \{x_1, \ldots, x_n\} \subset X \) the marginal distribution over that finite subset \( P(f) \) has a multivariate Gaussian distribution.

Gaussian processes (GPs) are parameterized by a mean function, \( m(x) \), and a covariance function, \( \kappa(x, x') \).

\[
P(f(X)) \sim \text{GP}(m(x), \kappa(x, x'))
\]  

\[
(f(x_1), f(x_2), \ldots, f(x_n)) \sim \mathcal{N}((m(x_1), \ldots, m(x_n)), \mathbf{K})
\]

\[
K_{ij} = \kappa(x_i, x_j)
\]

4.3 Gaussian Process Regression

Gaussian Process can be used to model non-linear regression.

Imagine observing a dataset \( D = \{(x_i, y_i)\}_{i=1}^n \) = (\( \mathbf{x}, \mathbf{y} \)). The model is given as,

\[
y_i = f(x_i) + \epsilon_i
\]

\[
f \sim \text{GP}(., m(x), \kappa(x, x'))
\]

\[
\epsilon_i \sim \mathcal{N}(., 0, \sigma^2)
\]

The statistical parameters of the model are given as,

\[
E[y_i|x_i] = m(x_i)
\]

\[
\text{Cov}(y_1, y_2) = \kappa(x_1, x_2)
\]

\[
\text{Var}(y_i) = \kappa(x_i, x_i) + \sigma^2
\]

\[
\text{Cov}(y|X) = \mathbf{K} + \sigma^2 \frac{I}{n}
\]

**Predicting New Data Point** : Consider the case where \( m(x) = 0 \) we have

\[
\begin{pmatrix} y \\ f(x_*) \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} 0 \\ K_y \\ K_s^T \\ K_s \end{pmatrix} \right)
\]

\[
E(f_*|x_*, \text{Data}) = K_s^T K_y^{-1} \mathbf{y} = \sum_{i=1}^n \kappa(x_i, x_*) \alpha_i
\]

5 Conclusion

The lecture showed two concepts for non-linear classifiers namely, **Kernelized Machines and Gaussian Process**. Both the Primal and Dual form of Support Vector machines was derived using a kernelized approach. Bayesian non-parametrics concept was introduced along with representation of the Gaussian Process in regression.