Kernels & Support Vector Machines Cont.

November 21, 2019

1 EXAM #1 - MARCH 2ND, IN CLASS

- Covers all topics up through last class
- Review all assignments, reading, lecture notes, everything we have covered.
- We will start next class with an exam review. The exam review will consist of me answering any questions you have about the material that is covered on the exam.
- I have placed example questions (from exams that I have previously given) in the lecture notes repository (under Lecture 13). These questions do not cover all topics but give examples of the types of questions that will be asked.

2 REVIEW KERNELS

- Last class we introduced kernels. Let us quickly review.
- **Kernel:** Let X be a non-empty set. A function $k : X \times X \to \mathbb{R}$ is called a kernel if there exists an $\mathbb{R}$-Hilbert space and a map $\phi : X \to H$ such that $\forall x, x' \in X$,

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \tag{1}$$

- We showed that the radial basis function kernel is an inner product in an *infinite dimensional space*:

$$k(\mathbf{x}, \mathbf{x}') \quad = \quad \exp(-\gamma \left\| \mathbf{x} - \mathbf{x}' \right\|^2) \tag{2}$$

- One main idea behind using kernels is to go into a higher dimensional space where it might be easier to segment or analyze the structure of the data.
- You can construct kernels from other kernels (e.g. sum of two kernels is a kernel, product of two kernels is a kernel)
- Kernel examples:
  - Consider: $\phi(x) = [x, x^2]$

- Consider: $\phi(\mathbf{x}) = \left[x_1^2, \sqrt{2}x_1x_2, x_2^2\right]$ where $\mathbf{x} = [x_1, x_2]$
- Easiest example: linear kernel, $\phi(\mathbf{x}) = \mathbf{x}$ so, $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$
- RBF Kernel
- Polynomial Kernel

- If you are interested in further reading, these are good notes on hilbert spaces and kernels: `http://www.mit.edu/~9.520/scribe-notes/class03_gdurett.pdf`

3 THE KERNEL TRICK

- Recall: the motivation is to have a non-linear mapping into a feature space (usually, a higher dimensional space) where the data is hopefully easier to classify and analyze
- When your method can make use of the *kernel trick*, you do not need to explicitly deal with the feature space mapping. You only need to deal with kernels in the original space.
- *What is the kernel trick?* Only operate on kernel functions, i.e., the inner products and skip the feature space representation directly
- Let us step more carefully through some of the math we did last class for deriving the dual representation of the following regression objective:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \tag{3}$$

- We want to minimize $J(\mathbf{w})$ with respect to $\mathbf{w}$:

$$\frac{\partial J}{\partial \mathbf{w}} = \sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)\phi(\mathbf{x}_n) + \lambda\mathbf{w} = 0 \tag{4}$$

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)\phi(\mathbf{x}_n) \tag{5}$$

  - Lets carefully work through the derivative. Recall that we can write the objective com-

pactly in matrix/vector form:

$$J(\mathbf{w}) = \frac{1}{2}\left([w_0, w_1, \ldots, w_D]\begin{bmatrix} \phi(x_1)_1 & \phi(x_2)_1 & \ldots & \phi(x_N)_1 \\ \phi(x_1)_2 & \phi(x_2)_2 & \ldots & \phi(x_N)_2 \\ \phi(x_1)_3 & \phi(x_2)_3 & \ldots & \phi(x_N)_3 \\ \vdots & \vdots & \ddots & \vdots \\ \phi(x_1)_D & \phi(x_2)_D & \ldots & \phi(x_N)_D \end{bmatrix} - [t_1, t_2, \ldots, t_N]\right)$$
$$\left([w_0, w_1, \ldots, w_D]\begin{bmatrix} \phi(x_1)_1 & \phi(x_2)_1 & \ldots & \phi(x_N)_1 \\ \phi(x_1)_2 & \phi(x_2)_2 & \ldots & \phi(x_N)_2 \\ \phi(x_1)_3 & \phi(x_2)_3 & \ldots & \phi(x_N)_3 \\ \vdots & \vdots & \ddots & \vdots \\ \phi(x_1)_D & \phi(x_2)_D & \ldots & \phi(x_N)_D \end{bmatrix} - [t_1, t_2, \ldots, t_N]\right)^T$$

– Ok, lets look closer at the minimization we discussed last class:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0 = \frac{1}{2} 2\mathbf{\Phi}^T \left(\mathbf{w}^T\mathbf{\Phi}^T - \mathbf{t}^T\right)^T \tag{6}$$
$$0 = \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{w} - \mathbf{\Phi}^T\mathbf{t} \tag{7}$$
$$\mathbf{\Phi}^T\mathbf{t} = \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{w} \tag{8}$$
$$\mathbf{w} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{t} \tag{9}$$

where $\mathbf{\Phi}^T = \begin{bmatrix} \phi(x_1)_1 & \phi(x_2)_1 & \ldots & \phi(x_N)_1 \\ \phi(x_1)_2 & \phi(x_2)_2 & \ldots & \phi(x_N)_2 \\ \phi(x_1)_3 & \phi(x_2)_3 & \ldots & \phi(x_N)_3 \\ \vdots & \vdots & \ddots & \vdots \\ \phi(x_1)_D & \phi(x_2)_D & \ldots & \phi(x_N)_D \end{bmatrix}$.

– Suppose N = 2 and D = 3:

$$
\begin{aligned}
J(\mathbf{w}) &= \frac{1}{2}\left([w_0, w_1, w_2]\begin{bmatrix}\phi(x_1)_1 & \phi(x_2)_1\\ \phi(x_1)_2 & \phi(x_2)_2\\ \phi(x_1)_3 & \phi(x_2)_3\end{bmatrix} - [t_1, t_2]\right) \\
&\quad \left([w_0, w_1, w_2]\begin{bmatrix}\phi(x_1)_1 & \phi(x_2)_1\\ \phi(x_1)_2 & \phi(x_2)_2\\ \phi(x_1)_3 & \phi(x_2)_3\end{bmatrix} - [t_1, t_2]\right)^T \\
&= \frac{1}{2}\left([w_0\phi(x_1)_1 + w_1\phi(x_1)_2 + w_2\phi(x_1)_3, w_0\phi(x_2)_1 + w_1\phi(x_2)_2 + w_2\phi(x_2)_3] - [t_1, t_2]\right) \\
&\quad \left([w_0\phi(x_1)_1 + w_1\phi(x_1)_2 + w_2\phi(x_1)_3, w_0\phi(x_2)_1 + w_1\phi(x_2)_2 + w_2\phi(x_2)_3] - [t_1, t_2]\right)^T \\
&= \frac{1}{2}\left([w_0\phi(x_1)_1 + w_1\phi(x_1)_2 + w_2\phi(x_1)_3 - t_1, w_0\phi(x_2)_1 + w_1\phi(x_2)_2 + w_2\phi(x_2)_3 - t_2]\right) \\
&\quad \left([w_0\phi(x_1)_1 + w_1\phi(x_1)_2 + w_2\phi(x_1)_3 - t_1, w_0\phi(x_2)_1 + w_1\phi(x_2)_2 + w_2\phi(x_2)_3 - t_2]\right)^T \\
&= \frac{1}{2}\left((w_0\phi(x_1)_1 + w_1\phi(x_1)_2 + w_2\phi(x_1)_3 - t_1)^2\right. \\
&\quad \left. + (w_0\phi(x_2)_1 + w_1\phi(x_2)_2 + w_2\phi(x_2)_3 - t_2)^2\right)
\end{aligned} \tag{10}
$$

– So, let us work out the derivative with respect to the vector $\mathbf{w}$ of the first term:

$$
\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \left[\frac{\partial J(\mathbf{w})}{\partial w_0}, \frac{\partial J(\mathbf{w})}{\partial w_1}, \frac{\partial J(\mathbf{w})}{\partial w_2}\right]^T \tag{11}
$$

– Now, work out the partials for the second regularization term.

- Going back to our derivation from last class:
- Lets call the following a_n: $-\frac{1}{\lambda}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right) = a_n$
- So,

$$
\mathbf{w} = \sum_{n=1}^{N} a_n\phi(\mathbf{x}_n) = \Phi^T\mathbf{a} \tag{12}
$$

  where Φ is the *design matrix* whose n^{th} row is given by $\phi(\mathbf{x}_n)$

- So, we now can use $\mathbf{w} = \Phi^T\mathbf{a}$ to rewrite $J(\mathbf{w})$:

$$
J(\mathbf{w}) = \frac{1}{2}\left(\mathbf{w}^T\Phi^T - \mathbf{t}\right)\left(\mathbf{w}^T\Phi^T - \mathbf{t}\right)^T - \frac{\lambda}{2}\mathbf{w}^t\mathbf{w} \tag{13}
$$

$$
= \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{w}^T\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \tag{14}
$$

- Plug in for $\mathbf{w} = \Phi^T \mathbf{a}$

$$= \frac{1}{2}\left(\Phi^T\mathbf{a}\right)^T \Phi^T\Phi\Phi^T\mathbf{a} - \left(\Phi^T\mathbf{a}\right)^T \Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\left(\Phi^T\mathbf{a}\right)^T \Phi^T\mathbf{a} \quad (15)$$

$$= \frac{1}{2}\mathbf{a}\Phi\Phi^T\Phi\Phi^T\mathbf{a} - \mathbf{a}\Phi\Phi^T\mathbf{t} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{a}\Phi\Phi^T\mathbf{a} \quad (16)$$

- Let $\mathbf{K} = \Phi\Phi^T$ be the *Gram Matrix*. Note that the Gram Matrix is symmetric
- $K_{nm} = \phi(\mathbf{x}_n)^T\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{aKKa} - \mathbf{aKt} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{aKa} \quad (17)$$

$$\frac{\partial J(\mathbf{a})}{\partial \mathbf{a}} = \mathbf{KKa} - \mathbf{Kt} + \lambda\mathbf{Ka} \quad (18)$$

$$\mathbf{a} = \left(\mathbf{K} + \lambda\mathbf{I}\right)^{-1}\mathbf{t} \quad (19)$$

- Since $\mathbf{w}^T = \mathbf{a}^T\Phi$,

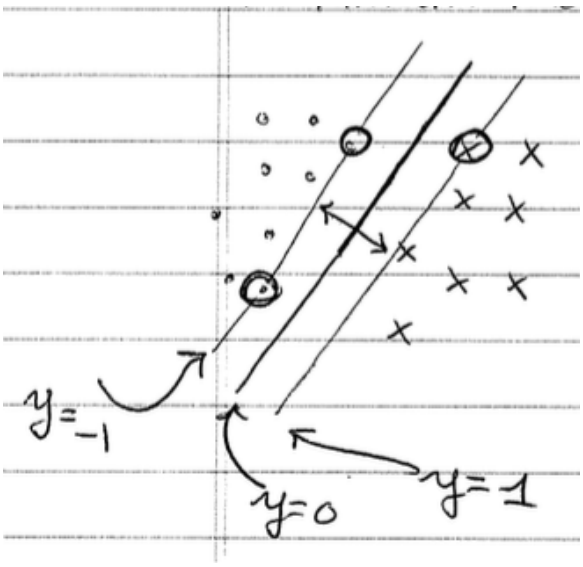
$$y(\mathbf{x}) = \mathbf{a}^T\Phi\phi(\mathbf{x}) \quad (20)$$

$$= \mathbf{K}(\mathbf{x})^T\left(\mathbf{K} + \lambda\mathbf{I}\right)^{-1}\mathbf{t} \quad (21)$$

  where $\mathbf{K}(\mathbf{x}) = \Phi\phi(\mathbf{x})$
- The Dual formulation shows you do not need to deal with the feature space mapping at all
- *Mercer's Theorem* - 1980 - said if you have $\mathbf{K}(\mathbf{x}, \mathbf{y})$ and it is a positive definite matrix, then, there is an equivalent $\phi(\mathbf{x})^T\phi(\mathbf{y})^T$ in some Hilbert space (i.e., a vector space where an inner product is defined - for our purposes)
- *What's the big deal?* Well, the feature space mapping can be infinite dimensional (e.g., RBF kernel). So, you can do analysis in an infinite dimensional feature space while only needing to compute kernel functions.

4 Introduction to Support Vector Machines

- SVMs are Maximum Margin Classifiers
- Two class classification problems: $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$



- We cannot directly compute $\phi(\mathbf{x})$ for all mappings, we want to use a kernel trick. So, we have to write up a dual representation of the problem in terms of K matrices
- We will start with the case where the $\phi(x)$ are linearly separable in the kernel space
- Note: the SVM finds a linear decision boundary in the feature space. But since we can do non-linear transformations to get to the feature space, the descision boundary can be non-linear in the feature space.
- We want to find $\mathbf{w}$ and b so that $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$ is $y(\mathbf{x}_n) > 0$ for $t_n = 1$ and $y(\mathbf{x}_n) < 0$ for $t_n = -1$. Or, in other words, we want:

$$t_n y(\mathbf{x}_n) > 0 \forall n \tag{22}$$

- The SVM finds the particular $\mathbf{w}$ and b that maximizes the margin (the distance between the closest point and the decision boundary for each class)
- Note: The SVM is inherently a classification algorithm. It is based on margin - separating two classes.
- We want to maximize the smallest distance between points from both classes. So we need the form for the distance and we can then plug that into our equation for the linear model

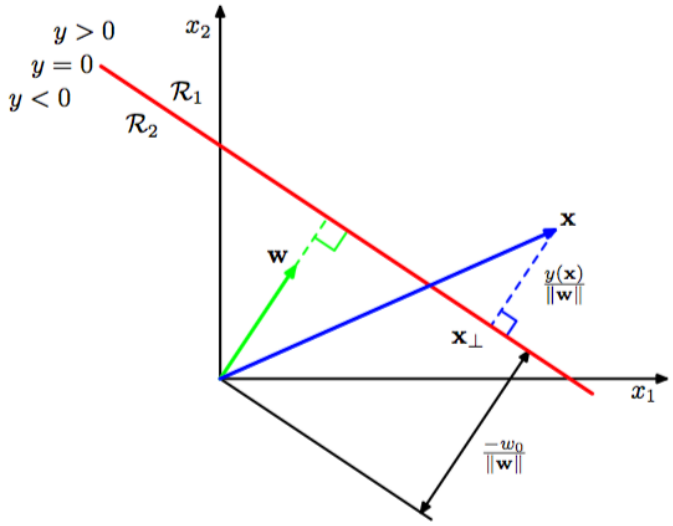


Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.

- Let $\mathbf{z}$ be $\phi(\mathbf{x})$

$$y(\mathbf{z}) = \mathbf{w}^T\mathbf{z} + b \tag{23}$$

$$y(\mathbf{z}) = \mathbf{w}^T\left(\mathbf{z}_p + r\frac{\mathbf{w}}{\|w\|}\right) + b \tag{24}$$

$$y(\mathbf{z}) = \left(\mathbf{w}^T\mathbf{z}_p + b\right) + r\frac{\mathbf{w}^T\mathbf{w}}{\|w\|} \tag{25}$$

$$y(\mathbf{z}) = 0 + r\frac{\mathbf{w}^T\mathbf{w}}{\|w\|} = r\frac{\|w\|^2}{\|w\|} \tag{26}$$

$$y(\mathbf{z}) = r\,\|w\| \tag{27}$$

$$\frac{y(\mathbf{z})}{\|w\|} = r \tag{28}$$

- So, the distance r is $\frac{y(\mathbf{z})}{\|w\|}$

- We want $t_n y(\mathbf{x}_n) > 0 \forall n$:

$$\frac{t_n y(\mathbf{x}_n)}{\|w\|} = \frac{t_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) + b\right)}{\|w\|} > 0 \tag{29}$$

- So, we can define the following objective function:

$$\arg\max_{w,b} \left\{ \frac{1}{\|w\|} \min_n \left[t_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) + b\right) \right] \right\} \tag{30}$$