CS 6362 Machine Learning, Fall 2017: Homework 2

Yunhua Zhao

September 18, 2017

Question 1:

(a) $\lim_{d\to\infty} \frac{V_s}{V_c} = \lim_{d\to\infty} \frac{r^d \pi^{d/2}/\Gamma(d/2+1)}{(2r)^d} = \lim_{d\to\infty} \frac{\pi^{d/2}}{4^{d/2}\Gamma(d/2+1)} = \lim_{d\to\infty} \frac{(\frac{\pi}{4})^{d/2}}{\Gamma(d/2+1)}$

At this step, we can see that the numerator approaches zero as d approaches infinity. Thus, it is sufficient to show that the gamma function in the denominator does not also approach zero. We know that:

$\lim_{z\to\infty} \frac{\Gamma(z+1)}{\sqrt{2\pi z}e^{-z}z^z} = \lim_{d\to\infty} \frac{\Gamma(d/2+1)}{\sqrt{\pi d}e^{-d/2}(d/2)^{d/2}} = \lim_{d\to\infty} \frac{\Gamma(d/2+1)}{\sqrt{\pi d}(\frac{d}{2e})^{d/2}} = 1$

We can see that the the denominator is guaranteed to diverge to infinity, as every term monotonically increases after $d > 2e$. Thus, in order for the numerator to be asymptotically equal to the denominator, it must also diverge, while our $(\frac{\pi}{4})^{d/2}$ term, and thus the entire limit, converges to 0. More formally, we can show this using the reciprocal of this limit to cancel the gamma function from our initial limit:

$\lim_{d\to\infty} \frac{(\frac{\pi}{4})^{d/2}}{\Gamma(d/2+1)} \frac{\Gamma(d/2+1)}{\sqrt{\pi d}(\frac{d}{2e})^{d/2}} = \lim_{d\to\infty} \frac{(\frac{\pi}{4})^{d/2}}{\sqrt{\pi d}(\frac{d}{2e})^{d/2}} = \lim_{d\to\infty} (\frac{2\pi e}{4d})^{d/2} \frac{1}{\sqrt{\pi d}} = 0$

(b) k-NN functions by finding samples within a certain radius of a common sample in the feature space. Unfortunately, the higher the dimension of the feature space, the more insignificant the volume of the hypersphere region containing the neighbors is compared to the volume of the rest of the hyperspace. That is, it becomes far more likely that samples are distributed in the feature space such that clusters are poorly defined or non-existent.

Question 2: Since both sets of parameters have equal cross validation errors, as far as we know, they generalize roughly equally well. With just the available training data, there is no way to tell which one will actually have better performance on new data. As such, we would select (c_1, γ_1), as having fewer support vectors makes prediction computationally easier.

Question 3:

(a) Since we would like to show that $H(a) = \max(1-a, 0)$ is convex, we will consider the functions in the max function independently.

$f(a) = 1 - a \implies f'(a) = -1 \implies f''(a) = 0 \geq 0 \ \forall a$

Since the second derivative of $f(a)$ is nonnegative everywhere, it must be convex.

$g(a) = 0 \implies g'(a) = g''(a) = 0 \ \forall a$

Likewise, the second derivative of $g(a)$ is nonnegative everywhere, so it must also be convex.

We know that a function $f(x)$ is convex if and only if the region above it is also convex. That is, if $(x, y) \in S \ \forall x, y$ where $y \geq f(x)$. So the region above $H(a)$ is the set of all points that are above both $f(a)$ and $g(a)$, or, in other words, the intersection of the sets of points above two convex functions. We know that the intersection of two convex sets is convex, so the set of points above $H(a)$, and thus $H(a)$ itself, must also be convex.

(b) Let $\min_{w,b} \sum_{i=1}^{n} H(y_i(w^T x_i)) + \lambda ||w||_2^2 = \sum_{i=1}^{n} H(y_i(w^{*T} x_i)) + \lambda ||w^*||_2^2$

and

$\min_{w,b} \sum_{i=1}^{n} H'(y_i(w^T x_i)) + \lambda' ||w||_2^2 = \sum_{i=1}^{n} H'(y_i(w'^{*T} x_i)) + \lambda' ||w'^*||_2^2$

By the definition of $H(a)$ and $H'(a)$, we can see that $H'(a) = \frac{1}{2} H(2a)$. Substituting this in, we have:

$\sum_{i=1}^{n} H'(y_i(w'^{*T} x_i)) + \lambda' ||w'^*||_2^2 = \sum_{i=1}^{n} \frac{1}{2} H(2y_i(w'^{*T} x_i)) + \lambda' ||w'^*||_2^2$

$= \frac{1}{2} \sum_{i=1}^{n} H(y_i(2w'^{*T} x_i)) + 2\lambda' ||w'^*||_2^2$

We can let $w^* = 2w'^*$, and substitute in:

$= \frac{1}{2} \sum_{i=1}^{n} H(y_i(w^{*T} x_i)) + 2\lambda' ||\frac{w^*}{2}||_2^2 = \frac{1}{2} \sum_{i=1}^{n} H(y_i(w^{*T} x_i)) + \frac{2\lambda'}{4} ||w^*||_2^2$

So we have $\lambda = \lambda'/2$.

Question 4:

(a) Increasing d makes overfitting more likely. A larger value for d disproportionally rewards adhering closely to the training data. This allows the regression to be more flexible to fitting the training samples more precisely and less able to generalize, making the SVM more prone to overfitting.

(b) Increasing σ makes overfitting less likely. The larger the value for σ, the larger the distance between x and x' (represented by the numerator in the kernel function) can be while still being significant. Thus, the boundaries are more generalized. Conversely, smaller values for σ tighten these bounds.

(c) Since K_1 and K_2 are kernel functions, we can say the following:

$K_1(x_i, x_i') = \langle \phi_1(x_i), \phi_1(x_i') \rangle$ where $\phi_1 : \mathbb{R}^m \to \mathbb{R}^{m_1}$

$K_2(x_i, x_i') = \langle \phi_2(x_i), \phi_2(x_i') \rangle$ where $\phi_2 : \mathbb{R}^m \to \mathbb{R}^{m_2}$

Then, given that $K(x_i, x_i') = K_1(x_i, x_i') + K_2(x_i, x_i')$, we must define a transformation ϕ to show that K is a kernel. ϕ_1 maps feature vector x_i from m dimensions to m_1 dimensions, and ϕ_2 maps it from m dimensions to m_2 dimensions. So we can define ϕ to map it from m dimensions to $m_1 + m_2$ dimensions. If we simply append the elements from $\phi_2(x)$ to the end of $\phi_1(x)$ to create every $\phi(x)$, we have show that:

$\phi : \mathbb{R}^m \to \mathbb{R}^{m_1+m_2}$

Furthermore, we have $K(x_i, x_i') = K_1(x_i, x_i') + K_2(x_i, x_i') = \langle \phi_1(x_i), \phi_1(x_i') \rangle + \langle \phi_2(x_i), \phi_2(x_i') \rangle$. Inner products are scalar sums of the element-wise multiplication of the individual components of the vectors. Thus, our method of concatenating ϕ_1 and ϕ_2 to form ϕ allows its inner product to equal the sum of the two separate inner products, as follows:

$K(x_i, x_i') = \langle \phi_1(x_i), \phi_1(x_i') \rangle + \langle \phi_2(x_i), \phi_2(x_i') \rangle = \langle \phi(x_i), \phi(x_i') \rangle$

Thus, we have shown that K is a kernel function.

Question 5:

(a) Assuming that "examples" refers to the data on which the SVM is making predictions, the computational complexity of prediction of a linear SVM is $O(nm)$, where n is the number of examples and m is the number of features, as the calculation is simply a dot product. The complexity of prediction on a single sample is just $O(m)$.

(b) The computational complexity of prediction of a non-linear SVM is $O(nms)$, again assuming that n is the number of examples that the SVM is classifying, m is the number of features, and s is the number of support vectors. For an individual sample, the complexity is $O(ms)$.