Homework #5: Machine Learning

Kyler Little

April 14, 2018

Problem #1

Consider two finite-dimensional feature transforms Φ_1 and Φ_2 and their corresponding kernels K_1 and K_2.
(a) Define $\Phi(x) = (\Phi_1(x), \Phi_2(x))$. Express the corresponding kernel of Φ in terms of K_1 and K_2.
Firstly, it's important to note that $\Phi(x)$ is also finite-dimensional since both $\Phi_1(x)$ and $\Phi_2(x)$ are finite-dimensional. Next, it's a simple matter of linear algebra to derive the kernel.
$K(x, x') = \Phi(x)^T\Phi(x') = \begin{bmatrix} \Phi_1(x)^T & \Phi_2(x)^T \end{bmatrix} \begin{bmatrix} \Phi_1(x') \\ \Phi_2(x') \end{bmatrix} = \Phi_1(x)^T\Phi_1(x') + \Phi_2(x)^T\Phi_2(x') = K_1(x, x') + K_2(x, x')$.
(b) Consider the matrix $\Phi_1(x)\Phi_2(x)^T$ and let $\Phi(x)$ be the vector representation of the matrix (say, by concatenating all the rows). Express the corresponding kernel of Φ in terms of K_1 and K_2.
First, we need to find exactly what Φ is in this situation (i.e. what does it mean to concatenate rows). When we concatenate the rows, we simply put row 2 behind row 1, then row 3 behind row 2, and so on. We take the transpose of this vector so that it's a column vector. Now, let $u = \Phi_1$ and $v = \Phi_2$ to simplify the notation a bit. Also, assume $\Phi_1(x)$ has n dimensions, and $\Phi_2(x)$ has m dimensions. We have:

$$\begin{aligned} K(x, x') &= \Phi(x)^T\Phi(x') \\ &= \text{concat}(\Phi_1(x)\Phi_2(x)^T)^T\text{concat}(\Phi_1(x')\Phi_2(x')^T) \\ &= \text{concat}(uv^T)^T\text{concat}(u'v'^T) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} u_i v_j u'_i v'_j \\ &= u^T u' v^T v' \\ &= K_1(x, x')K_2(x, x') \end{aligned}$$

In this case, the kernel is simply the product of the other two kernels.
(c) Hence, show that if K_1 and K_2 are kernels, then so are $K_1 + K_2$ and K_1K_2.
Assume otherwise. This means that it's not possible to find kernels $K_1 + K_2$ and K_1K_2 such that K_1 and K_2 are both separately kernels. However, we just did that in parts (a) and (b); thus, we have a contradiction. This means the original statement must be true. $\square$

Problem #2

Exercise 8.16 (e-Chap:8-42) in LFD. Please ignore part (c) and only do (a), (b), (d).
Exercise 8.16 Show that the optimization problem in (8.30) is a QP-problem.

(a) Show that the optimization variable is $u = \begin{bmatrix} b \\ w \\ \xi \end{bmatrix}$, where $\xi = \begin{bmatrix} \xi_1 \\ \vdots \\ \xi_n \end{bmatrix}$.

Although I'm not exactly sure what this question is asking, I'm assuming that it means to demonstrate the minimization problems are identical (i.e. the QP form of the problem is equivalent to the minimization problem of 8.30). If we let the optimization variable be u, then we have:

$$\begin{aligned} \underset{u \in R^{1+d+N}}{\text{minimize}} \quad & \frac{1}{2} u^T \mathrm{Q}_D u + p^T u \\ \text{subject to} \quad & Au \geq C \end{aligned}$$

We need to show this is equivalent to 8.30. Using the Q below, it's easy to see $\frac{1}{2}u^T Q u = \frac{1}{2} w^T w$ since the identity matrix within Q grabs w from within u while the rest of the matrix Q is zeros, removing b and ξ. Using the p below, we see that $p^T u = \sum_{i=1}^{N} C\xi_i = C \sum_{i=1}^{N} \xi_i$, exactly like 8.30.

(b) Show that $u^* \leftarrow \mathrm{QP}(Q, p, A, c)$, where

$$Q = \begin{bmatrix} 0 & 0_d^T & 0_N^T \\ 0_d & I_d & 0_{d \times N} \\ 0_N & 0_{N \times d} & 0_{N \times N} \end{bmatrix}, \; p = \begin{bmatrix} 0_{d+1} \\ C \cdot 1_N \end{bmatrix}, \; A = \begin{bmatrix} YX & I_N \\ 0_{N \times d+1} & I_N \end{bmatrix}, \text{ and } c = \begin{bmatrix} 1_N \\ 0_N \end{bmatrix},$$

and YX is the signed data matrix from Exercise 8.4.

Again, the author is not exactly clear in what he wants to be done. From the problem statement, it appears that he wants me to demonstrate that the optimal u is obtained by solving the QP-problem with the given inputs. Thus, I'm assuming this means I need to show the equivalence of conditions (i.e. $Au \geq C \leftrightarrow y_n(w^T x_n + b \geq 1 - \xi_n)$ and $\xi_i \geq 0$).

To do so, I will start by explicitly writing out the $Au \geq C$.

$$\begin{bmatrix} y_1 & \cdots & y_1 x_1^T & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & 0 & \ddots & 0 \\ y_N & \cdots & y_N x_N^T & \cdots & 0 & \cdots & 1 \\ 0 & \cdots & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} b \\ w \\ \xi \end{bmatrix} \geq \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

From here, if we perform the matrix-vector multiplication, we get the following result for the first N rows: $y_i b + y_i x_i^T w + \xi_i, \forall i \in 1, ..., N$. This is equivalently $y_i(w^T x_i + b) \geq 1 - \xi_i, \forall i \in 1, ..., N$. The last N rows result in $\xi_i \geq 0, \forall i \in 1, ..., N$. This exactly equals the problem in 8.30. Now, the problem is in standard QP form. Thus, to solve for the optimal hyperplane u^*, we merely solve the QP problem now.

(d) How do you determine which data points violate the margin, which data points are on the edge of the margin and which data points are correctly separated and outside the margin?

ξ_i captures by how much (x_i, y_i) fails to be separated. Specifically, if we have that $\xi_i = 0$ for

a particular data point, then the margin is not violated and no loss occurs. The data point is correctly classified. This also means that $y_i(w^T x_i + b) > 1$. If we have that $y_i(w^T x_i + b) = 1$, then the data point is on the edge of the margin and again, $\xi_i = 0$. Thus, it is easy to see that ξ_i captures how far into the margin a data point is, since non-zero ξ_i means that the margin was violated (also meaning $y_i(w^T x_i + b) < 1$).

Problem #3

(a) Describe what are hinge loss, logistic regression loss, and 0-1 loss mathematically. Describe their similarities and differences using the unified picture we developed in class.
We start with 0-1 loss.
Machine learning classification problems can be formulated as a "minimizing the loss" problems. In general, suppose we have data points $x_1, x_2, ..., x_N \in R^d$ with corresponding classifications $y_1, y_2, ..., y_N \in \{-1, +1\}$. For the most general case, we classify by the sign of the linear signal. In other words, if $y_i(w^T x_i + b) \geq 0$, then a point is correctly classified. If the linear signal is negative, it's not correctly classified. We can formulate the idea of minimizing the loss by characterizing the problem as:

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & L_{0/1}(y_i(w^T x_i + b)) \\ \text{where} \quad & L_{0/1}(z) = \begin{cases} 1 & \text{if } z < 0 \\ 0 & \text{if } z \geq 0 \end{cases} \end{aligned}$$

In this way, the loss problem is formulated as minimizing the number of misclassified points. Individually, the 0-1 loss function assigns a value of 1 to a misclassified point and a value of 0 to a correctly classified point.
Now what about logistic regression loss and hinge loss? We can use the same logic to define those loss functions. In both SVM and logistic regression, the loss functions are already conveniently defined. For SVM, the hinge loss is defined to be:

$$\begin{aligned} L_{\text{hinge}} &= \max(0, 1 - z) \text{ where} \\ z &= y_i(w^T x_i + b) \end{aligned}$$

In this loss function, a correctly classified point must satisfy the condition: $y_i(w^T x_i + b) \geq 1$ due to how the SVM problem is formulated. This loss function assigns a value of $1 - y_i(w^T x_i + b)$ or $1 - z$ to a misclassified point and a value of 0 to a correctly classified point.
Lastly, we look at the logistic regression loss function. Again, it is already conveniently defined by the cross-entropy term. It is defined to be:

$$\begin{aligned} L_{\text{logistic}} &= \log_2(1 + \exp^{-z}) \text{ where} \\ z &= y_i(w^T x_i + b) \end{aligned}$$

In this case, classified and misclassified points are actually treated the same since the loss function is not piecewise. Both are assigned values: $\log_2(1 + \exp^{-y_i(w^T x_i + b)})$. This means that correctly classified points result in a small loss while misclassified points result in a large

loss. We can visualize all three loss functions with the graph below. Note: I borrowed this from the notes online, so it includes an extra loss function not discussed in this homework (the modified huber loss).

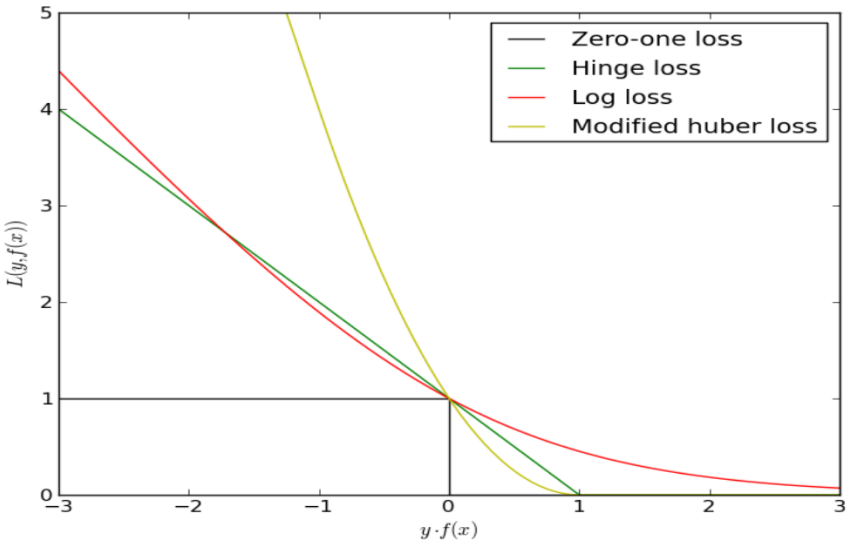

Figure 1: An overview of different loss functions

(b) By relying on the result in the above question, consider a point that is correctly classified and distant from the decision boundary. Why would SVMs decision boundary be unaffected by this point, but the one learned by logistic regression be affected?
The SVM's decision boundary is unaffected because the SVM only cares about support vectors (points that are on the decision boundary). In fact, non-support vectors are not even used in the calculation of the margin in SVM. Thus, SVM's decision boundary is completely unaffected by that point. On the other hand, logistic regression is still affected. This is because the goal of logistic regression is to maximize the likelihood that all points are correctly classified. Thus, it needs to take into consideration all points in order to do so.

Problem #4

Summarize your observations from the coding portion into a short report. In your report, please report the accuracy result and total support vector number of each model. A brief analysis based on the results is also needed.

Cost Parameter	Kernel	Support Vectors	Test Accuracy
0.01	Default	1112	88.68%
0.1	Default	424	95.99%
1	Default	163	95.99%
2	Default	131	95.99%
3	Default	117	96.23%
5	Default	104	96.23%
Default	Linear	162	95.99%
Default	Polynomial	543	95.52%
Default	Radial Basis	163	95.99%

Varying Cost: As the cost increases, we care more about violating the margin. In this case, we see that the number of support vectors decreases as the cost increases. On the other hand, the test accuracies increase as the cost increases. This intuitively makes sense because, as the cost increases, the soft-margin SVM increasingly resembles the hard-margin SVM. The hard-margin SVM should certainly be more accurate with linearly separable data because we emphasize or more highly value correctly classifying data points.
Varying Kernels: As for varying the kernels, all three of them achieve roughly equivalent test accuracies. However, the polynomial kernel requires 3-4x as many support vectors, which is computationally more expensive.