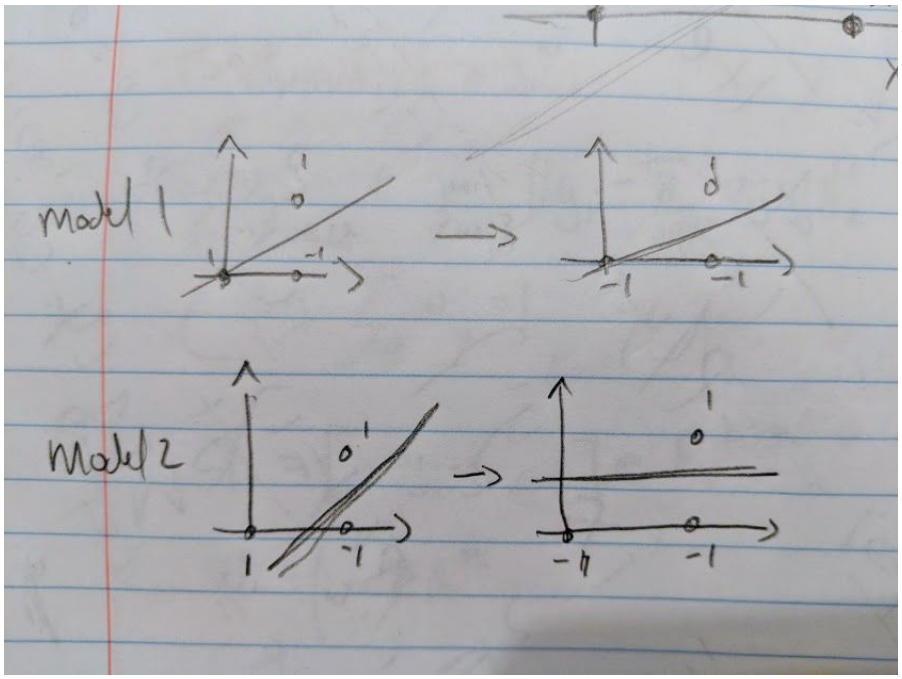
Machine Learning CS 6140
Miles Benjamin
Assignment 2

1. Logistic Regression

A. Since the first model doesn't have a y-intercept term the w values will not likely change much by re-classifying point 3. This is because any decision boundary has to pass through the origin, which is also point3.

Re-classifying point 3 in model 2 will likely change w significantly. See the included figure for more info.

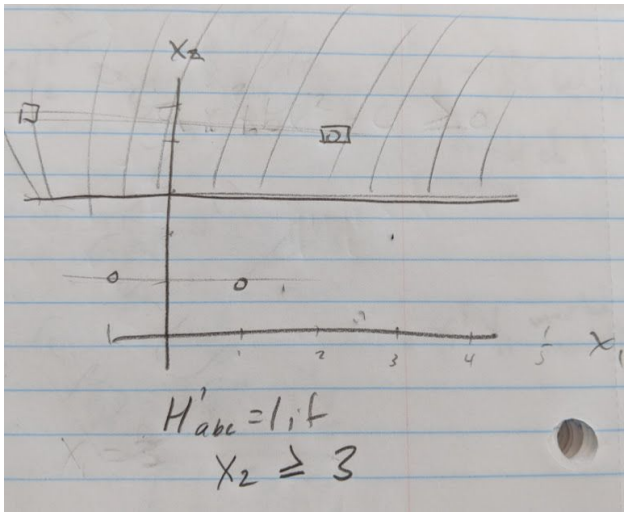


B. Combining our two equations we get that:
$\Sigma \frac{1}{2} y^i w^T x^i - \frac{\lambda}{2} ||w||_2^2$
If we take the derivative with respect to w we can see the MLE equals
$\Sigma \frac{1}{2} y^i x^i - \lambda w$
Which means that as λ increases, w gets smaller.

2. Support Vector Machine

A. There is no way to linearly separate the data in this example. A linear boundary in 1d would be functionally a point. We could place this point between x=1 and x=2, but that would have a classification error. The same is true if we tried to place it between x=-1 and x=-2.

B. Graphing the dataset on the 2D plane below we can see that a half space exists for which



H' = 1 if X2 >=2.5.

C. $K(x,z) = \phi(x)^T \phi(z) = (x, x^2)^T (z, z^2) = (xz + x^2 z^2)$

3. Constructing Kernels

A. For K to be a kernel it must be positive semi-definite and symmetric. As long as the scalars c1 and c2 are non negative. The following statements hold true:
   a. The scaling of a positive semi-definite matrix by a non-negative value is going to be positive semi-definite.
   b. The addition of two positive semi-definite matrices is a positive semidefinite matrix.
   c. The scaling of a symmetric matrix is a symmetric matrix.
   d. The linear combination of two symmetric matrices is a symmetric matrix, this is just a property of symmetric matricies
   e. Therefore K is a kernel if it is the linear combination of two other kernels using only non-negative scalars.

B. K is a kernel because the dot product of two positive semi-definite matrices is also a positive semi-definite matrix. Furthermore the dot product of two symmetric matrices is also a symmetric matrix. This is a property of symmetric matricies.

   Additionally we can think of the dot product of two matrices as the linear combination of the vectors that comprise them. As these vectors are also positive semi-definite and as all the scalars must be non-negative as they come from a positive semi-definite matrix, the result must also be positive semi-definite.

C. We can think of this question as a linear combination of many kernels using non-negative scalars. As we showed in part A, the linear combination of any number of Kernels with non-negative scalars will also be a kernel.

D. The function $K(x,z) = exp(K_1(x,z))$ can be re-written as the following limit:
   $\lim_{n\to\infty}(1 + \frac{K(x,z)}{n})^n$
   Since this is really just an infinite linear combination of Kernels using non-negative scalars, K(x,z) is a Kernel matrix by the properties we've shown in A, B, and C.

E. The singular value decomposition of a Kernel can be represented as $x^T Az$ so provided that A is a positive semi-definite matrix, this expression is a valid Kernel representation.

F. $K = exp(-||x||_2^2 - ||z||_2^2 + 2x^T z$
$= (exp(-||x||_2^2)exp(-||z||_2^2))exp(2x^T z)$
$= f(x)f(z)exp(2x^T z)$

Since we know that the exp(xtz) is a kernel as we showed in part D, and we know that two kernels multiplied together are a Kernel (part B), then all we have to show is that f(x)f(z) is a kernel.

f(x)f(z) forms a grammian matrix, which is always positive semidefinite and symmetric.

Therefore the whole expression is a valid Kernel.

4. Support Vectors

A. The upper bounds on the number of misclassified instances is simply the number of times that the error is greater than 1. We can quantify this with the following equation:
$\sum_{i=1}^{N} 1(\xi > 1)$

B. C is the scaling of the error term in our primal function. Roughly speaking it determines how much we are penalizing misclassified instances and correctly classified instances that are within our margin.

If C = 0 then we are not counting any error in our cost function, so this would allow for points that are very far on the wrong side of the margin (very large)

If C = ∞ then we have a very large penalty for misclassification, however in a dataset where the data is not linearly separable this could cause our classification function to never converge.

C. Using a Kernel can allow you to skip having to calculate ϕ(x) explicitly. A kernel function is defined as:
$k(x, z) = \varphi(x)^T \varphi(z)$
$\varphi(x)^T \varphi(z) = \Sigma\Sigma x_i x_j z_i z_j = (\Sigma x_i z_i) * (\Sigma x_j z_j) = (x^T z)^2$
Or more generally:
$k(x, z) = (x^T z)^n$
This allows you to directly calculate the kernel without ever actually calculating ϕ(x).

Given a new x you can use the Kernel trick again to calculate a new kernel without explicitly computing ϕ(x), this is simply adding a row and column to the Kernel, the rest of it stays the same!

5. Generalized Lagrangian Function

Generally speaking we can say that for a given f(x):

$$min\, f(x) \leq f(x) \leq max\, f(x)$$

For functions with multiple max and min, the largest minimum (max min) is still going to be less than or equal to the function. Similarly the smallest maximum (min max) is still going to be greater than or equal to the function.

So given the Lagrangian Function, it holds that

$$min\ L(w, \alpha, \beta) \leq L(w, \alpha, \beta) \leq max\ L(w, \alpha, \beta)$$

And thus:

$$max\ min\ L(w, \alpha, \beta) \leq L(w, \alpha, \beta) \leq min\ max\ L(w, \alpha, \beta)$$

Or simply:

$$max\ min\ L(w, \alpha, \beta) \leq min\ max\ L(w, \alpha, \beta)$$

6. Dual Optimization

A. $$L(\alpha, x) = \tfrac{1}{2}x^T P_0 x + q_0^T x + r_0 + \sum_{i=1}^{n} \alpha_i (\tfrac{1}{2}x^T P_i x + q_i^T x + r_i)$$

B. If we take the derivative of this function with respect to alpha we get

$$dL/d\alpha = \sum_{i=1}^{n} (\tfrac{1}{2}x^T P_i x + q_i^T x + r_i) = 0$$

$$dL/dx = P_0 x + q_0 + \sum_{i=1}^{n} P_i x + q_i = 0$$

I'm not sure how to solve it from here, but I want to substitute them back into the original function to find an expression that does not depend on x. Using this we can get the final dual optimization for part C.

7 & 8. See attached Jupyter Notebook files.