CS412 Final Exam

Kimmy Liu

1.

    1. No, we can't train a hard margin linear SVM on the given dataset, because the 2-class classification dataset is not linearly separable. A hard margin linear SVM requires the dataset to be linearly separable by class.
    2. Yes, we can train such a soft margin linear SVM on the given dataset. Soft Margin Formulation allow SVM to make a certain number of mistakes and keep margin as wide as possible so that other points can still be classified correctly. The new constraint permits a functional margin that is less than 1, and contains a penalty of cost $C\xi i$ for any data point that falls within the margin on the correct side of the separating hyperplane (i.e., when $0 < \xi i \leq 1$), or on the wrong side of the separating hyperplane (i.e., when $\xi i > 1$). We thus state a preference for margins that classify the training data correctly, but soften the constraints to allow for non-separable data with a penalty proportional to the amount by which the example is misclassified.

       It is not possible that all slack variables will be zero. Because the sum total of the penalties ξi over all i is an upper bound on the training classification errors, and since the dataset is not linearly separable, we can't acchive 0 error.
    3. Yes, I agree with Professor Kernel’s claim. As this dataset is linear non-separable in 2D. We can map these linearly non-separable data to a higher dimensional space to achive linearly separable. In this case, the 6-dimensional space will project data in 3D space, where the points with a higher value of x1i^2+x2i^2 will be projected as blue data, and with a lower value of x1i^2+x2i^2 will be projected as orange data. So we can find a hyperplane in this 3D place to separate these two classes.

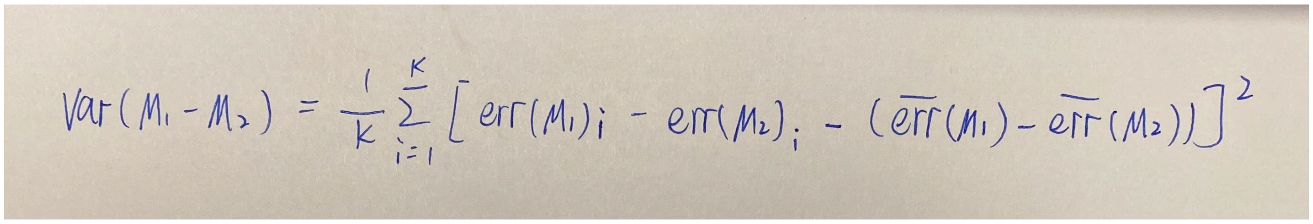
2.

    1. The degrees of freedom for the test will be k-1=10-1=9. The individual error rates in the cross-validations can be considered as different, independent samples from a probability distribution. A and B samples look like follow a t-distribution, and t-distribution have a degrees of k-1 freedom. When (k−1) samples are known, the other one can be calculated, which does not need a single sample.
    2.

       We use the following formula to calculate t-statistics

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}},$$

And we use the following formula to calculate variance:



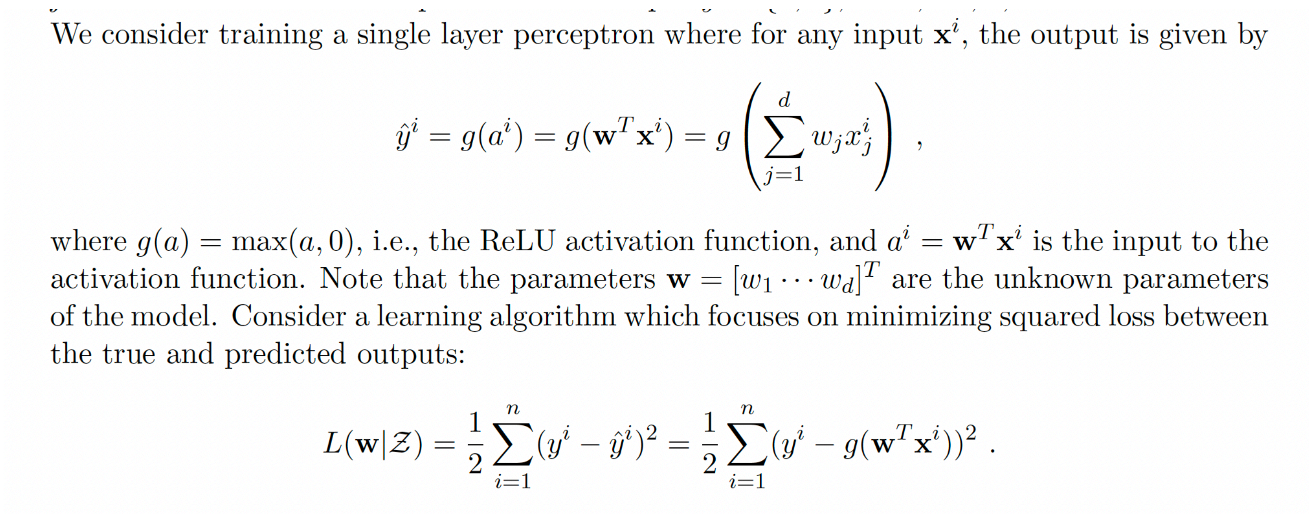
$$Var(M_1 - M_2) = \frac{1}{K}\sum_{i=1}^{K}\left[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))\right]^2$$

From (detailed step on the next page), we get t_stat = -19.528984873617613, and p-value = 1.120231307716324e-08. Since p-value is much smaller than significance level 0.05, we reject the null hypothesis and we know that one is significantly different than the other.

3. With same formula as 2(b), we get t_stat = 8.532204687249003 and p-value = 1.3184220974293837e-05. Since p-value is much smaller than significance level 0.05, we reject the null hypothesis and we know that one is significantly better than the other. (Detailed step on the next page)

3.

1.



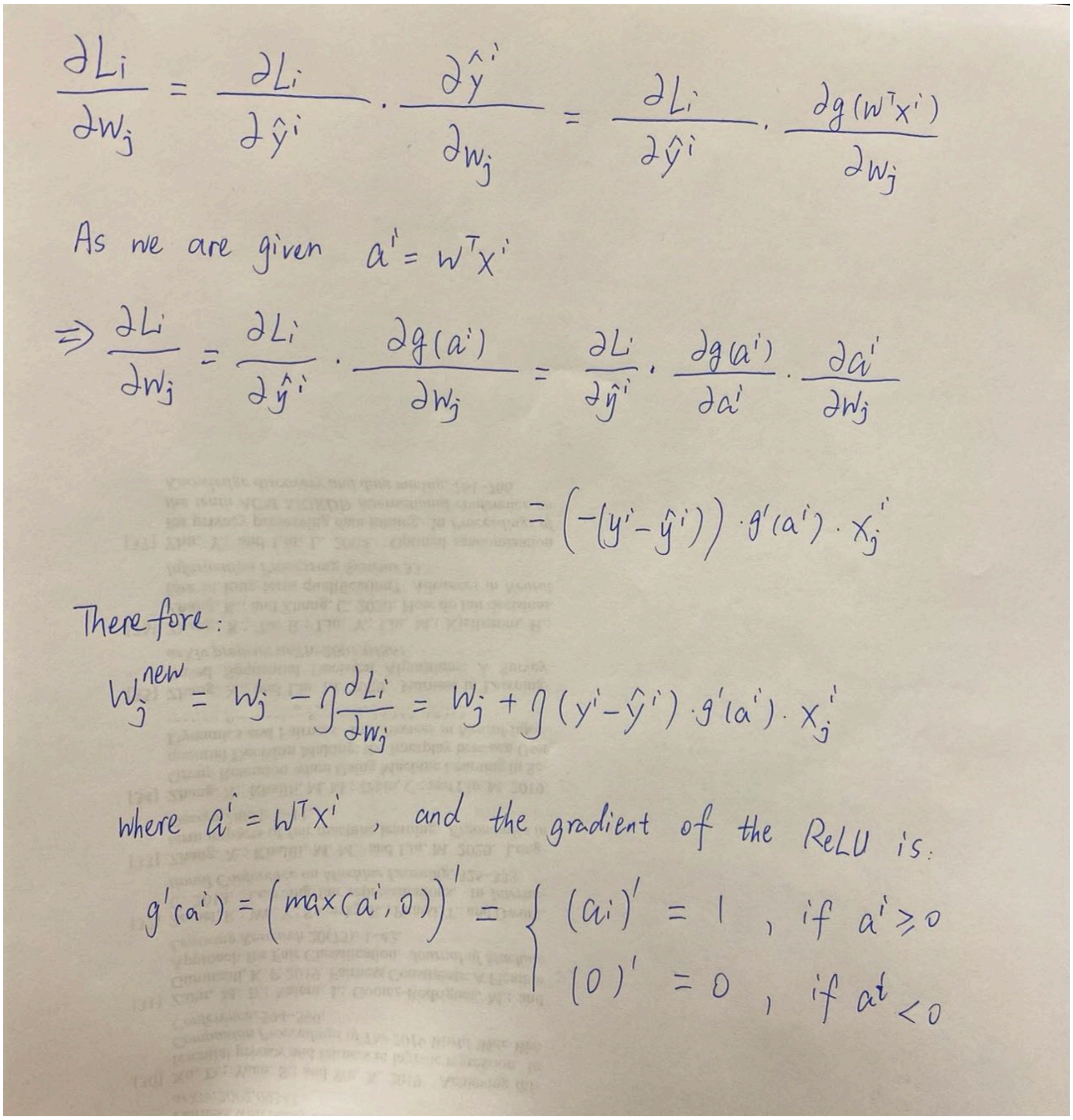
We consider training a single layer perceptron where for any input $\mathbf{x}^i$, the output is given by

$$\hat{y}^i = g(a^i) = g(\mathbf{w}^T\mathbf{x}^i) = g\left(\sum_{j=1}^{d} w_j x_j^i\right) ,$$

where $g(a) = \max(a, 0)$, i.e., the ReLU activation function, and $a^i = \mathbf{w}^T\mathbf{x}^i$ is the input to the activation function. Note that the parameters $\mathbf{w} = [w_1 \cdots w_d]^T$ are the unknown parameters of the model. Consider a learning algorithm which focuses on minimizing squared loss between the true and predicted outputs:

$$L(\mathbf{w}|\mathcal{Z}) = \frac{1}{2}\sum_{i=1}^{n}(y^i - \hat{y}^i)^2 = \frac{1}{2}\sum_{i=1}^{n}(y^i - g(\mathbf{w}^T\mathbf{x}^i))^2 .$$

The stochastic gradient descent (SGD) algorithm updates the parameters based on a random chosen point (xi; yi) in each step. The SGD update for parameter wj with step size n is of the form:



$$\frac{\partial L_i}{\partial w_j} = \frac{\partial L_i}{\partial \hat{y}^i} \cdot \frac{\partial \hat{y}^i}{\partial w_j} = \frac{\partial L_i}{\partial \hat{y}^i} \cdot \frac{\partial g(w^T x^i)}{\partial w_j}$$

As we are given $a^i = w^T x^i$

$$\Rightarrow \frac{\partial L_i}{\partial w_j} = \frac{\partial L_i}{\partial \hat{y}^i} \cdot \frac{\partial g(a^i)}{\partial w_j} = \frac{\partial L_i}{\partial \hat{y}^i} \cdot \frac{\partial g(a^i)}{\partial a^i} \cdot \frac{\partial a^i}{\partial w_j}$$

$$= \left(-(y^i - \hat{y}^i)\right) \cdot g'(a^i) \cdot x_j^i$$

There fore :

$$w_j^{new} = w_j - \eta \frac{\partial L_i}{\partial w_j} = w_j + \eta (y^i - \hat{y}^i) \cdot g'(a^i) \cdot x_j^i$$

where $a^i = w^T x^i$, and the gradient of the ReLU is:

$$g'(a^i) = \left(\max(a^i, 0)\right)' = \begin{cases} (a_i)' = 1 , & \text{if } a^i \geq 0 \\ (0)' = 0 , & \text{if } a^i < 0 \end{cases}$$

2.

i. linear activation, g(ai) = ai

In this case, we need to update both (1) and (2).

We first update (2):

$$g(a^i) = a^i \Rightarrow g'(a^i) = 1, \ \forall a^i$$

And we use g'(ai) = 1 in formula (1) to update it to:

$$W_j^{new} = W_j - \eta \frac{\partial L_i}{\partial W_j} = W_j + \eta (y^i - \hat{y}^i) \cdot g'(a^i) \cdot X_j^i$$

$$= W_j + \eta (y^i - \hat{y}^i) \cdot X_j^i$$

ii. sigmoid activation

In this case, we need to update both (1) and (2).

We first update (2):

$$g(a^i) = \frac{1}{1 + e^{-a^i}} \Rightarrow g'(a^i) = \frac{e^{-a^i}}{(1+e^{-a^i})^2} = g(a^i)(1 - g(a^i)), \ \forall a^i$$

And we use g'(ai) = exp(-ai)/(1+exp(-ai))^2 in formula (1) to update it to:

$$W_j^{new} = W_j - \eta \frac{\partial L_i}{\partial W_j} = W_j + \eta (y^i - \hat{y}^i) \cdot g'(a^i) \cdot X_j^i$$

$$= W_j + \eta (y^i - \hat{y}^i) \cdot \frac{\exp(-a^i)}{(1 + \exp(-a^i))^2} \cdot X_j^i$$

4.

1. As shown on the course slides, the k-mean algorithm is sensitive to outliers since an object with an extremely large value may substantially distort the distribution of the data. Instead of taking the mean value of the object in a cluster as a reference point, the k-medoids method takes the most centrally located object in a cluster.

Pseudocode and brief description:

```
Step1:  randomly select k points from the n data points as the initial representative objects, which called initial k medoids.

Step2: Associate each data point to the closest medoid by using any common distance metric methods.

Step3: repeat while the cost decreases:
  1. For each medoid m, and for each non-medoid data point o of the n data points:
      1. Consider the swap of m and o, and compute the cost change
      2. If the cost change is the current best, remember this m and o combination
      3. Else, undo the swap
  2. Perform the best swap of m_best and o_best if it decreases the cost function. Otherwise, the algorithm terminates.
```

2. The computational complexity of the Partitioning Around Medoids (PAM) k-medoids clustering algorithm is O(k(n-k)^2).

   Assume that the first sets of medoids are the worst medoids, so the cost function calculated through these medoids is the maximum of all the possible sets of medoids and we will always randomly choose one that decreases the cost function. Again, assume we always choose the next worst one in all the possibilities, therefore we will exhaust all the remaining medoids (n-k) to find the set of medoids that has the minimum cost function. So the outmost loop in step3 would be k that loop through all the medoids. Then it will be n-k in the inner loop, to loop through all the non-medoid data points. Then n-k again for choosing the random medoid. Big O notation denotes the upper bound of an algorithm, so we need to always consider the worst case when calculate computational complexity. As mentioned on slides, some improvements on Partitioning Around Medoids (PAM) k-medoids clustering algorithm have a less complexity such as CLARA, which has O(ks^2+k(n-k)).

3. The motivation behind using the k-medians algorithm instead of the k-means algorithm in certain situations is that medians are less sensitive to outliers than means, for example, in a large company, mean salary will be largely affected and higher than median salary when we add in a few top executives.

   In k-medians algorithm, instead of taking the mean value of the object in a cluster as a reference point, medians are used and L1-norm is used for distance measure. This change has the impact of minimizing the error over all cluster with the respect to the 1-norm distance metric. It is important to note that k-means in the other has the squared 2-norm distance metric, which. is Euclidean distance. Furthermore, we can say that k-means minimizes the within cluster variance which is equivalent to squared Euclidean distances while k-medians minimizes the absolute deviations which is equivalent to Manhattan distance.

   Now moving on to explain the demand for computation. K-medians is far more computationally expensive than k-means. This is because of the way their algorithms are set up. For k-median algorithm, the coordinates of cluster data need to be sorted which can be more costly than simply calculating the mean. These differences become more significant when the size of data increases.