Machine Grading

7. (15 points) Prof. Regu LaRisashun has just joined the 6.036 team, and they are excited to help teach students about machine learning. In particular. Prof. Regu (as they are fondly called) wants to try reducing stress by eliminating the final exam. They believe that nanoquizzes and homeworks should be sufficient to predict exam performance.

   Specifically, Prof. Regu takes the homework and nanoquiz grades (x), and runs a linear regression with hypothesis $\hat{y} = \theta^T x + \theta_0$ to make predictions ($\hat{y}$) for students' midterm grades. They minimize an objective function with just mean square error between the predicted and actual midterm grades (y). Data from 70% of the students are used for training, and the remaining 30% for evaluating the model.

   The initial results do not look so good, but Prof. Regu understands that this often happens with a simple linear model, and it can help a great deal to model and encode features more thoughtfully. Prof. Regu thus writes a problem for the midterm exam, asking students to help make the final exam unnecessary, by exploring five specific ideas.

   (a) Majors

   Prof. Regu notices that some students find the homework questions harder than other students, and believes this could be due to what students have studied in their other classes. Specifically, Prof. Regu notices that EECS majors seem to do better on homeworks than Physics majors. Fortunately, at MIT students' majors are conveniently coded up as a number (e.g. 1 = Civil engineering, 2 = Mech. Eng., 6 = EECS, 8 = Physics, 15 = Management, etc.) so Prof. Regu enters this number for each student as a new feature for the model.

   Is this a good idea? Explain why or why not. If not, what better way might you encode students' majors for the model?

   Solution: This not a good idea.

   Using course numbers (integers) as features implies an ordering between majors (which is incorrect). Using one-hot encoding is a better option.

   (b) Programming experience

   Looking more deeply, Prof. Regu notices that the coding questions seem to be very strong predictors of exam grades, but only if students' prior experience with python programming is taken into consideration. Prof. Regu obtains data from an initial survey students filled out at the start of the semester, where they were asked to check one box on this question:

   What is your level of python programming experience?
   ( ) None   ( ) Beginner   ( ) Experienced   ( ) Expert

   How should data from this question best be encoded for Prof. Regu's model?

   Solution: Prof Regu should use Thermometer encoding to encode programming experience, because the items are ordered, but there is no definitive notion of distance between the items.

An integer or other real number encoding would preserve order, but improperly impose a specific notion of distance.

(c) Name of students

In a fit of exhaustion, after too many days filled with administrative Zoom meetings, Prof. Regu notices that students' exam grades are highly correlated with their names. They decide to one-hot encode each of the names of the ~ 500 students, and use these new features in their model.

How good would you expect the model to perform with this approach? Discuss both the test error and the training error.

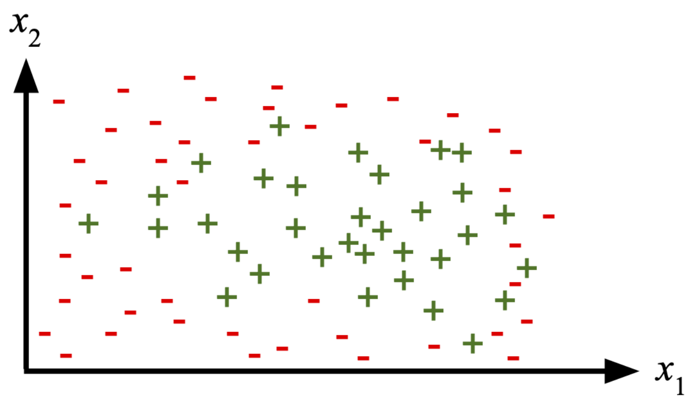
Solution: The resulting model will not perform well. The one-hot encoding will lead to zero or very low training error, because each name will become matched to the corresponding student grade. This is overfitting. However, the test data will have names which were unseen in the training data, and thus predictions for these new names will be poor. Thus, test error will be very high.

(d) Time on task

A kind colleague at the Harvard Graduate School of Education tells Prof. Regu about an interesting experiment: apparently, the Educational Testing Service (which administers major tests like the GRE) is looking at the amount of time students take to answer questions, as a measure for students' understanding of the material. The idea is that a more skilled student should be able to answer questions faster than a less skilled one.

Inspired by this idea, Prof. Regu mines data about how long students are taking to complete 6.036 nanoquizzes (x_1) and homeworks (x_2). Prof. Regu also changes approach: instead of predicting exam grades, Prof. Regu just tries to predict whether the student passes ($y = 1$) or fails ($y = 0$) the midterm exam based on just these x_1 and x_2 data. They employ linear logistic regression, with hypothesis $\hat{y} = \sigma(\theta^T x + \theta_0)$.

However, this model performs poorly! Prof. Regu plots the data to try and understand why, and sees this ($+$ indicates $y = 1$, and $-$ indicates $y = 0$):



Apparently, while it is the case that students who take a long time on nanoquizzes and homeworks indeed tend not to pass the exam, students who take a very short amount of time also tend not to pass! (How are some students able to finish entire homework assignments in just a few minutes?) Prof. Regu decides to try to fix the model to accommodate this peculiar behavior, by employing a feature transform $\phi(x)$, and using the hypothesis $\hat{y} = \sigma(\theta^T \phi(x) + \theta_0)$.

Specify a mathematical function $\phi(x)$ which substantially improves the training error for these data:

Solution:

This is much like the example given in lecture, about predicting heart attacks based on blood pressure and heartbeat rate, where the idea is to allow the model to identify an elliptical region of the form $(ax_1 - b)^2 + (cx_2 - d)^2$. We want the center of this region to be determined by model parameters, and how to do this can be seen by expanding the expression, giving $a^2x_1^2 - 2abx_1 + b^2 + c^2x_2^2 - 2cdx_2 + d^2$. So if our model were of the form $\theta_1 x_1^2 + \theta_2 x_1 + \theta_3 x_2^2 + \theta_4 x_2 + \theta_5$, then it should fit the data well.

Thus, recognizing that the dimension of the output feature space can be larger than the input, let's just include all the polynomial terms needed for such a model:

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1x_2 \end{bmatrix}. \tag{3}$$

Here, we also include an x_1x_2 term, to allow for possible rotation of the axes of the ellipse.