Discussion Session Problems 4

02/5/2026

1. Here are a few types of gradient descent updates:

   (a) **Stochastic Gradient Descent:** Uses only single training example to calculate the gradient and update parameters.

   (b) **Batch Gradient Descent:** Calculate the gradients for the whole dataset and perform just one update at each iteration.

   (c) **Mini-batch Gradient Descent:** Mini-batch gradient is a variation of stochastic gradient descent where instead of single training example, mini-batch of samples is used. It's one of the most popular optimization algorithms.

   What are the pros and cons of each one and when should you use them? Walk us through your intuition and give us examples of when you are going to use what update.

   Stochastic gradient descent: noisy/inaccurate, but very cheap, so allows to perform a lot of iterations. It is sometimes better to perform many small inaccurate updates than few accurate gradient updates.
   Batch gradient descent: accurate gradient computation, but expensive. Since each update is expensive, learning rate cannot be very small (otherwise would be too computationally expensive). So one sometimes has to perform fewer updates with a larger learning rate. Therefore, the optimal point of the cost function may be missed.
   Mini-batch gradient descent: medium between stochastic and batch gradient descent.

2. Our internal AI team has been training a model for very long and our biggest problem is that our model does not learn anything. Our cost function is not going down, and our gradients in a layer seem to be the same. We have been debugging our gradient update function but it seems like it is correct. Do you have any tips to help us solve this problem? Walk us through your thought process and show us on the board what is going on.

   This is because the initialization might have been the same. As a result we will not break the symmetry and the neural networks

3. You are asked to build a classification model about meteorites impact with Earth (important project for human civilization). After preliminary analysis, you get 99% accuracy. Should you be happy? Why not? What can you do about it?

   No, because there are rare cases. If you predict no all the time you will get that.
   You can reframe the problem as predicting meteorites which pass below a given distance from Earth. Increasing that distance increases the number of positive cases in your dataset.

4. You are working on a classification problem. For validation purposes, you've randomly sampled the training data set into train and validation. You are confident that your model will work incredibly well on unseen data since your validation accuracy is high. However, you get shocked after getting poor test accuracy. What went wrong?

   Hyperparameters overfit to the validation set (e.g. learning rate). Also make sure that the splits are not random.

5. Why should features be normalized when applying regularization to logistic regression and neural nets? Any further reasons to do it for neural nets even if we are not using regularization?

   Consider a simple example: you try to predict height in meters from leg length x_1 in meters and arm length x_2 in centimeters
   Model: $y = w_1x_1 + w_2x_2 + b$
   Where the true hypothesis is
   $y = 1 * x_1 + (1/100) * x_2$
   If we regularize those weights, w_1 will be heavily regularized compared to w_2 because its value is 100x higher. However this value difference is just due to a difference in units.
   Note that even if variables are measured in the same units, their orders of magnitude can be different, so normalization still must be applied.
   Normalization further helps in neural nets because it ensures that there are both positive and negative values used as inputs for the next layer which makes learning more flexible and that the network's learning regards all input features to a similar extent.
   Remark: in the height example, you should notice that the two variables considered are correlated (leg length and arm length), so in all rigor those two variables should not be both used as input of a linear regression model.

6. Let's explore saliency, a measure of how important a feature is for classification. We define the saliency of the ith input feature for a given example (x, y) to be the absolute value of the partial derivative of the log likelihood of the sample prediction, with respect to that input feature $\left|\frac{\partial LL}{\partial x_i}\right|$. In the images below, we show both input images and the corresponding saliency of the input features (in this case, input features are pixels):

   First consider a trained logistic regression classifier with weights θ. Like the logistic regression classifier that you wrote in your homework it predicts binary class labels. In this question we allow the values of x to be real numbers, which doesn't change the algorithm (neither training nor testing).

   (a) What is the log likelihood of a single training example (x, y)?
   (b) Compute the saliency of feature x_i.
   (c) Show that the ratio of saliency for features i and j equals $\frac{|\theta_i|}{|\theta_j|}$.

Let $p = \sigma(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$.
(a):

$$LL(x, y) = y \log p + (1 - y) \log(1 - p)$$

(b):

$$\frac{\partial LL}{\partial x_i} = \frac{\partial LL}{\partial p} \cdot \frac{\partial p}{\partial(\theta^T x)} \cdot \frac{\partial(\theta^T x)}{\partial x_i}$$

$$= (y-p)\cdot p(1-p)\cdot\theta_i \div [p(1-p)] = (y-p)\theta_i$$

So saliency is

$$\left|\frac{\partial LL}{\partial x_i}\right| = |y-p|\,|\theta_i|$$

(c):

$$\frac{\left|\frac{\partial LL}{\partial x_i}\right|}{\left|\frac{\partial LL}{\partial x_j}\right|} = \frac{|y-p|\,|\theta_i|}{|y-p|\,|\theta_j|} = \frac{|\theta_i|}{|\theta_j|}$$

7. Suppose a hidden layer uses ReLU activation:

$$A = \max(0, Z)$$

and during backprop you receive $\frac{\partial L}{\partial A} = dA$. Write the vectorized expression for $\frac{\partial L}{\partial Z}$.
The derivative of ReLU is 1 where $Z > 0$ and 0 otherwise.

$$dZ = dA \odot \mathbf{1}(Z > 0)$$

where $\odot$ is element-wise multiplication.

8. Let:

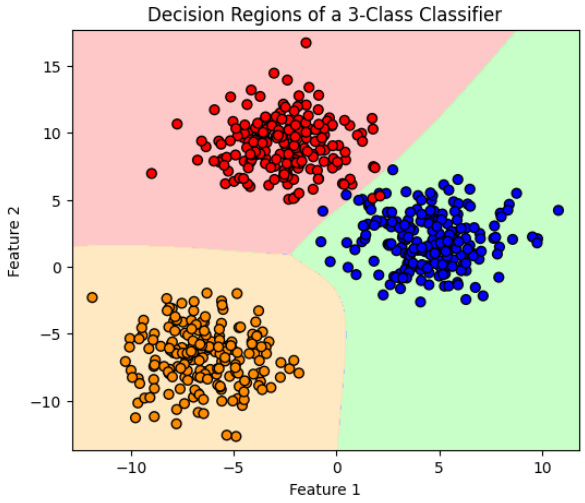
$$X : (m \times n), \quad W_1 : (n \times h), \quad W_2 : (h \times K)$$

Give the shapes of:

(a) $Z_1 = XW_1$
(b) $A_1 = \text{ReLU}(Z_1)$
(c) $Z_2 = A_1 W_2$
(d) $\hat{Y} = \text{softmax}(Z_2)$
(e) $\frac{\partial L}{\partial W_2}$

$$Z_1 : (m \times h)$$
$$A_1 : (m \times h)$$
$$Z_2 : (m \times K)$$
$$\hat{Y} : (m \times K)$$
$$\frac{\partial L}{\partial W_2} = A_1^T(\hat{Y} - Y) \Rightarrow (h \times m)(m \times K) = (h \times K)$$

9. The figure below shows decision regions of a 3-class classifier in 2D. Each color represents a predicted class.



Answer the following:

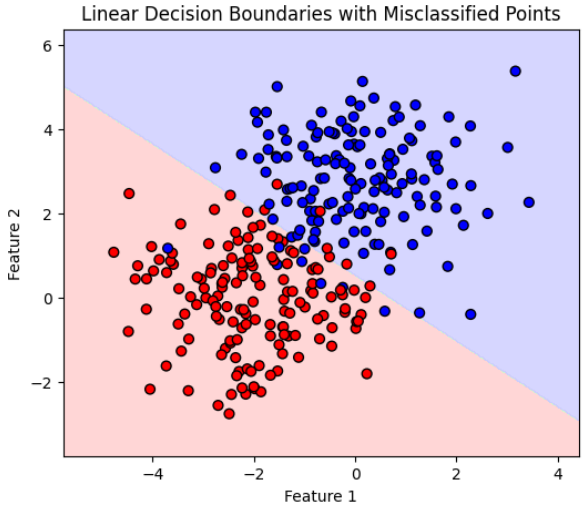
(a) Is this classifier linear or non-linear? Explain.

(b) Could this be produced by a single linear layer? Why or why not?

(c) What change to a model could produce curved boundaries like these?

The classifier is non-linear because the decision boundaries are curved rather than straight lines.
A single linear layer can only produce linear decision boundaries, which are straight lines (or hyperplanes in higher dimensions). Therefore, this cannot come from a single linear layer.
Curved boundaries could be produced by adding hidden layers with nonlinear activations (such as ReLU or tanh), which allow the model to represent nonlinear decision surfaces.

10. In the figure below, three linear decision boundaries divide the plane into regions for classes A and B. Training points from both class lie deep inside the region predicted as the opposite class.



(a) What does this tell you about the model's capacity?

(b) Would adding more training time fix this issue?

(c) Name one model change that could help.

This suggests the model underfits and does not have enough capacity to correctly separate the classes. Simply training longer will not help if the decision boundaries are fundamentally too simple.
Adding nonlinear features, hidden layers, or using a nonparametric method would increase model capacity and could allow correct classification.