CS344 Exercise 3

Task 1: Gradient descent

Answer the following TRUE/FALSE or multiple choice questions.

Given an input value of 0, the sigmoid function returns 0.5

TRUE FALSE

The cost function used for logistic regression has multiple local minima

TRUE FALSE

The goal of gradient descent is to find the best values for the parameter ***w*** and *b*

TRUE FALSE

Gradient descent is executed on testing data

TRUE FALSE

The reason that gradients (i.e., derivatives) are calculated as part of gradient descent is to know how big a step to take each iteration

TRUE FALSE

Back propagation involves calculating information in order to update the parameters ***w*** and *b*

TRUE FALSE

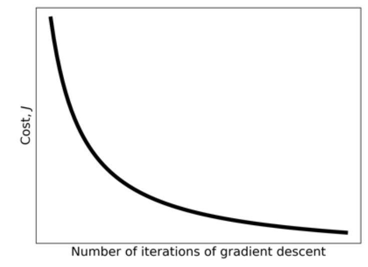
Vectorization parallelizes calculations to improve runtime performance

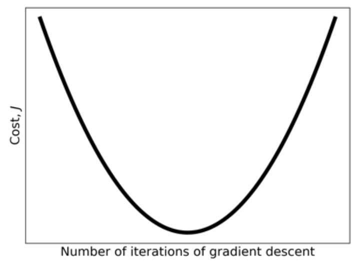
TRUE FALSE

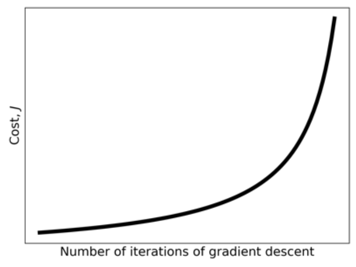
For gradient descent, one advantage of a large learning rate (alpha) is faster descent. One disadvantage is lack of convergence.

TRUE FALSE

Assuming gradient descent is working as expected, as gradient descent executes, the relationship between the number of iterations of gradient descent and the cost function is best represented by which figure?







A model's performance on training data is a better predictor of how it will perform on new unseen data than a model's performance on testing data

TRUE FALSE

Overfitting occurs when a model's accuracy is too high

TRUE FALSE

One way to combat overfitting is:

A. Initialize ***w*** and *b* to be 0 at the beginning of gradient descent
B. Use a smaller learning rate (alpha)
C. Run gradient descent for a larger number of iterations
D. Use regularization
E. Use vectorization

The regularization parameter λ controls how much relative weight should be put on having smaller parameter values vs. fitting the training data well

TRUE FALSE

Task 2: Validation data

Thus far, we've mostly talked about splitting data into two parts: *training* and *testing*. Even better, data should be split into three parts: *training*, *validation*, and *testing*. *Training data* are used to determine the model parameters ***w*** and *b*. *Validation data* are used to determine values of *hyperparameters*. Once all decisions about the model have been finalized (parameters and hyperparameters), then *testing data* may be used to evaluate how the model performs on new previously unseen data.

As an example, the learning rate α and the regularization parameter λ are hyperparameters. How do we know what values we should set them to? Split the data into three parts: training, validation, and testing. Choose one set of values for α and λ, fit the model using the training data to learn ***w*** and *b*, and evaluate the accuracy of the model using the *validation* data. Now try a different set of values for α and λ, fit the model using the training data to learn ***w*** and *b*, and evaluate the accuracy of the model using the *validation* data. Keep repeating this process with different values for the hyperparameters α and λ, each time evaluating the model's accuracy with the validation data. Whichever values for α and λ yield the best accuracy on the validation data are the values we'll ultimately choose. This is called *hyperparameter tuning*. It's a bit like trial-and-error... trying different hyperparameter values and seeing which lead to the best results on the validation data. Once we've chosen all hyperparameter values, we can use the testing data to assess the model's performance on new data.

Hyperparmaters correspond to any decision we make about the model other than values for parameters ***w*** and *b*. Example hyperparameters include:

- the learning rate α
- whether to use batch or stochastic or mini-batch gradient descent
- the batch size (if using mini-batch gradient descent)
- the maximum number of iterations of gradient descent, *max_iter*
- whether to use regularization
- the regularization parameter λ (if using regularization)

Every time the value of a hyperparameter is changed, the model needs to be fit with the training data in order to learn values for the parameters ***w*** and *b*

TRUE FALSE

Changing the value of a hyperparameter after evaluating the model's performance on testing data would be an example of *data leakage*

TRUE FALSE

Task 3: Multiclass classification and softmax

We've talked about logistic regression as a ***binary*** classifier. Because it is. However, it can be used for ***multiclass*** classification. Suppose we have images of cats, dogs, and rabbits, and we want a model to identify which of the three classes (cat or dog or rabbit) a picture belongs to. How can we use logistic regression, since it is fundamentally designed to distinguish 2 classes rather than 3? If we have 3 classes, rather than use 1 logistic regression model, we use 3 different logistic regression models. One model we train on images of cats and non-cats (dogs and rabbits). One model we train on images of dogs and non-dogs (cats and rabbits). And one model we train on images of rabbits and non-rabbits (cats and dogs). Thus, we have three models: one for identifying cat pictures, one for identifying dog pictures, and one for identifying rabbit pictures. Given a new picture for which we want to make a prediction, we run it through all three models and see which one yields the highest predicted value.

More generally, if we have data labeled with *k* > 2 classes, we construct *k* different logistic regression models. When making a prediction for a new data point, we run it through all *k* models and whichever of the *k* returns the highest value is the class we predict for the new data point.

One issue is that the values returned by the *k* logistic regression models aren't probabilities, i.e., they don't necessarily sum to 1.0. And we like probabilities :). Thus, a ***softmax*** function is used so that the *k* resulting values can be interpretted as probabilities (they sum to 1.0). The ***softmax*** formula is:

$$softmax(z_j) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

In the cat/dog/rabbit example, suppose we run a new picture through the 3 logistic regression models and the cat/non-cat model yields z_{cat} = 0.4, the dog/non-dog model yields z_{dog} = 0.1, and the rabbit/non-rabbit model yields z_{rabbit} = 0.9. The highest value of the 3 is 0.9 so we'd predict the picture is a rabbit. But 0.4, 0.1, and 0.9 are not probabilities (they don't sum to 1.0). Applying the ***softmax*** function gives:

$$\text{softmax}(z_{cat}) = \frac{e^{0.4}}{e^{0.4}+e^{0.1}+e^{0.9}} = 0.295 \qquad \text{softmax}(z_{dog}) = \frac{e^{0.1}}{e^{0.4}+e^{0.1}+e^{0.9}} = 0.219 \qquad \text{softmax}(z_{rabbit}) = \frac{e^{0.9}}{e^{0.4}+e^{0.1}+e^{0.9}} = 0.486$$

We'd predict the picture is a rabbit with probability 48.6%.

Suppose we run a different picture through the mulitclass logistic regression models and they compute z_{cat} = 0.7, z_{dog} = 0.8, and z_{rabbit} = 0.1. What are the probabilities, after applying the ***softmax*** function, that the picture is a cat, is a dog, is a rabbit?

Task 4: Hyperparameter tuning

Download the Jupyter Notebook for Exercise 3 from the course website. Open the Notebook in your web browser and work through it. As you work through the Notebook, answer the following questions.

How many milliseconds (wall time) did the computation take that used a loop? How many milliseconds (wall time) did the vectorized computation take?

What value for the regularization parameter yielded the best performance on the ***validation*** data?

What is the accuracy of the tuned model on the ***testing*** data?

For the thyroid cancer data, what is the accuracy of the tuned model on the ***testing*** data?

For the Internet ad data, what is the accuracy of the tuned model on the ***testing*** data?

For the job attrition data, what is the accuracy of the tuned model on the ***testing*** data?

What is the performance of your tuned model on the testing data (hopefully better than that of a naïve model, 17%)?

CS344 Exercise 3 Final Page

In the *TIME* column, please estimate the time you spent on this exercise. Please try to be as accurate as possible; this information will help us to design future exercises.

PART	**TIME**
Exercise	