Intro Deep Learning: Homework 1

Daniel Novikov

February 10th, 2022

Problem 1

What is the difference between Root Mean Squared Error (RMSE) and Mean Squared Error(MSE). Describe the context in which they will be most useful? Which of these losses penalizes large differences between the predicted and expected results and why?

Answer: The MSE loss function is the average squared difference between the predictions of your model and the expected results. This function is most useful for fitting curves to data, because the squaring exaggerates large errors. However, the square also means that the units of the MSE loss are not the same as the units of the data. This is where RMSE, which is the square root of the MSE, is useful. The square root undoes the squaring of units; thus ensuring the units of your loss match those of the data. This allows for a better intuition of the error when reasoning about the performance of your model. Finally, since RMSE is the square root of MSE; MSE will always penalize the model more than RMSE for all errors larger than 1.

Problem 2

Which of the following statements is true about the relationship between features and dataset sizes? Explain the chosen answer.

Answer: **A**. When training a model, as you add more features to the dataset, you often need to increase the dataset's size to ensure the model learns reliably.

As we add more features, our model becomes more flexible; i.e., the degrees of freedom in our model increase. If our dataset size doesn't grow, we are more prone to overfitting the random fluctuations in our training subset, and fail to generalize to the true data distribution.

Problem 3

You are building a deep learning model and after X epochs you see that the training loss is decreasing, while your validation loss is either constant or increasing. What would be the most likely cause of this and suggestion to rectify it.

Answer: **C**. Insufficient training data size.

Our model is overfitting our training data. We do not have enough training data for the number of features we have. We have these options: We can reduce the training duration (number of epochs), we can get more data, or we can reduce the number of features. Reducing the number of features is called dimensionality reduction, and there are many algorithms for this problem, including principle component analysis and linear embedding.

Problem 4

Perceptron

Answer:

A) (**TRUE**): Is a Linear Classifier

B) (**FALSE**) and cannot be trained on linearly unseparable data. *If we allow some error, we can still train a perceptron on *roughly* linearly seperable data. Furthermore, we can train it on anything if we don't care about performance. So the statement is not completely true.*

Problem 5

Logistic Regression

Answer:

A) (**FALSE**): Is a Linear Classifier

B) (**FALSE**): and always has a unique solution

Problem 6

Which of the following is true, given the optimal learning rate?

Answer: **E**. For convex loss functions (i.e. with a bowl shape), both stochastic gradient descent and batch gradient descent will eventually converge to the global optimum.

This is because convex functions have a single minimum, and both batch and stochastic gradient descent are guaranteed to converge to a local minimum.

Problem 7

Fit a line through origin that minimizes MSE on points $(-1,-2),(1,-1),(2,3)$

Answer:

We want to find w that minimizes MSE. To do that, we will set the derivative of MSE w.r.t w to 0.

$$MSE(w) = \frac{1}{n}\sum_{i}^{n}(wx-y)^2$$

$$\frac{dMSE}{dw} = \frac{1}{n}\sum_{i}^{n}\frac{d}{dw}((wx-y)^2)$$

$$= \frac{1}{n}\sum_{i}^{n}\frac{d}{dw}((wx-y)(wx-y))$$

$$= \frac{1}{n}\sum_{i}^{n}\frac{d}{dw}(w^2x^2-2wxy+y^2)$$

$$= \frac{1}{n}\sum_{i}^{n}(2x^2w-2xy)$$

$$= \frac{1}{n}\sum_{i}^{n}(2x(wx-y))$$

Since the $2x$ term does not depend on w, $\frac{dMSE}{dw} = 0$ when $(wx-y)=0$

$$wx - y = 0$$

$$wx = y$$

$$w = y/x$$

$$\frac{dMSE}{dw} = 0 => w = \frac{1}{n}\sum_{i}^{n}(y_i/x_i)$$

$$w = \frac{\frac{-2}{-1}+\frac{-1}{1}+\frac{3}{2}}{3} = 0.8333$$

Equation of line through origin minimizing MSE on given dataset:

$$y = 0.833x$$

Problem 8

Solution provided on Github:

Problem 9

Is this a good idea to initialize parameters with zeros when training a logistic regression model? Explain your answer.

Answer: If all weights are 0, then each neuron will output the same activation value $\sigma(0) = 0.5$, and therefore during backpropagation, all neurons will receive the same update. This subverts the whole point of using nonlinear activation functions, which is to distribute the loss gradient unevenly during the backwards pass, to represent nonlinear contributions to loss of different features. That's why its not a good idea to initialize weights to 0's for logistic regression.

Problem 10

Consider the logistic regression likelihood function.

$$cost(f(x), y) = -y \log(f(x)) - (1 - y) \log(1 - f(x))$$

$$y \in 0, 1$$

What will happen when y = 1 and f(x) = 1? Will this work in actual implementations?

Answer: When y=1, only the first term contributes to the equation.

$$cost(f(x), y = 1) = -y \log f(x)$$

$$cost(f(x), 1) = -1 * \log f(x)$$

$$cost(1, 1) = -1 * 0 = 0$$

So when $f(x) = y = 1$, the cost returned is 0, which is what we expect from a cost function. However, in actual implementations, the second term will be computed anyway. This term includes $\log(1 - f(x))$, which when $f(x) = 1$ leads to the undefined operation $\log(0)$. Therefore this wouldn't work in actual implementations.

Problem 11 (EC)

In the three examples, find the perceptron weights w0, w1, w2 for each of them.

Answer:

1.) $w = [1, 0]$

2.) $w = [-2, -1.4]$

3.) $w = [[-1, 0], [1, 0]]$ I understand there are 2 linear boundaries in this example, but I'm not sure how to capture them both in a single w vector.

Problem 12 (EC)

Find partial derivatives w.r.t a, b, d of

$$E = g(x, y, z) = \sigma(c(ax + by) + dz)$$

$$= \sigma(cax + cby + dz)$$

$$E = \frac{1}{1 + e^{-cax-cby-dz}}$$

By quotient rule, the partial derivative of E w.r.t a variable x

$$\frac{\partial E}{\partial x} = \frac{0 - \frac{\partial}{\partial x}(1 + e^{-cax-cby-dz})}{(1 + e^{-}cax - cby - dz)^2}$$

Also,

$$\frac{\partial}{\partial x}(e^{-cax-cby-dz}) = \frac{\partial}{\partial x}(e^{-cax}e^{-cby}e^{-dz}) = e^{-cby}e^{-dz}\frac{\partial}{\partial x}(e^{-cax})$$

Since the terms e^{-cby} and e^{-dz} are constant w.r.t x.

Therefore,

$$\frac{\partial}{\partial x}(e^{-cax-cby-dz}) = e^{-cby}e^{-dz} * -cae^{-cax} = -ca * e^{-cax-cby-dz}$$

This same principle can be used to remove exponent terms that don't depend on the current variable of derivation.

$$\frac{\partial E}{\partial a} = \frac{cx * e^{-cax-cby-dz}}{(1 + e^{-cax-cby-dz})^2}$$

$$\frac{\partial E}{\partial b} = \frac{cy * e^{-cax-cby-dz}}{(1 + e^{-cax-cby-dz})^2}$$

$$\frac{\partial E}{\partial d} = \frac{z * e^{-cax-cby-dz}}{(1 + e^{-cax-cby-dz})^2}$$