Introduction to Machine Learning
Problem Set: Linear Regression and Gradient Descent

Summer 2021

1. **Residuals**

   Is the following statement *True* or *False*? Explain your answer.

   > The residual of a regression model, $y - \hat{y}$, is random error with no pattern or systematic trend. There is no machine learning model that can "explain" any part of the residual.

   Solution: This statement is False. The residual may include stochastic noise, but it can *also* include many systematic sources of error - for example, if there is a non-linear association between the feature(s) and the target variable. A different machine learning model may be capable of "explaining" the part of the residual that is not stochastic noise.

2. **Regression metrics**. For the following questions, your answers can be in the form of a range, like "$-\infty$ to ∞" or "≤ 0", along with a brief explanation.

   Suppose you have training data and test data sampled from the data-generating process

   $$y = A + Bx + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$

   where A and B are real valued scalar constants.

   You fit an ordinary least squares linear regression model to the training data. Finally, you compute R2 for the fitted model.

   (a) What is the range of values you could get for R2 on your training data?

   Solution: $0 \leq R2 \leq 1$.

   The upper bound of R2 is never greater than 1; from the expression for R2 we see that when the MSE is 0, R2 is 1.

   To convince yourself that the lower bound is 0, consider the following: the linear regression finds weights that minimize the error on the training data. The linear regression will only have negative R2 if its error is *greater* than "prediction by mean". But "prediction by mean" is in the assumed model class (use $w_0 = \bar{y}$ and all other $w_i = 0$). The linear regression can "find" those weights if they do have the smallest possible error, in which case it would have exactly the same error as prediction by mean and R2 of 0.

(b) What is the range of values you could get for R2 on your test data?

Solution: $R2 \leq 1$.

The upper bound of R2 is the same as in the previous question.

The lower bound is different, however. The least squares linear regression finds the weights for which error on the *training set* is minimized. However, we know that due to the random draw of training and test sets, the weights that minimize error on the training set may not be the weights that minimize error on the test set! It is possible for the linear regression (fitting on the training set) to have greater error on the test set than prediction by mean on the test set, so R2 will be negative.

Answer the same questions, but assuming $\sigma = 0$:

(a) What is the range of values you could get for R2 on your training data?

Solution: $R2 = 1$. Here, we have $\epsilon_i = 0, \quad \forall i$, i.e. there is no stochastic error in the data. In this case, there will be no error in the model output:

- The irreducible error term is 0 because $\sigma = 0$.
- There is no under-modeling: the true function $y = A + Bx$ is in the assumed model class for a least squares linear regression.
- There is no error in the parameter estimates that are fitted by the model. Since there is no noise in the training set, the model will learn the true parameters, $\hat{w} = w_t$, i.e. $\hat{w}_0 = A, \hat{w}_1 = B$.

(b) What is the range of values you could get for R2 on your test data?

Solution: $R2 = 1$, for the same reasons as the training data. Unlike the previous case, where the optimal parameters on the training set do not minimize the error on the test set, when there is no noise in the data, there is 0 error on both training and test sets when $\hat{w}_0 = A, \hat{w}_1 = B$.

Answer the same questions again, but if the *test data* was actually sampled from

$$y = C + Dx + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$

where C and D are real valued scalar constants. Also assume $\sigma = 0$.

(a) What is the range of values you could get for R2 on your training data?

Solution: $R2 = 1$. This is the same as the previous part.

(b) What is the range of values you could get for R2 on your test data?

Solution: $R2 < 1$. In the best case, if $A = C$ and $B = D$, then this is the same as the previous question and $R2 = 1$. However, if $A \neq C$ and $B \neq D$ then the model will be substantially worse on the test data. In particular, note that if B and D have opposite signs (i.e. the training data has an upward slope, the test data has a downward slope), the model fitted to the training data will be worse than prediction by mean on the test data, and the R2 will be negative.

3. **Linear basis function regression**.

You have labeled data $(x_i, y_i), i = 1, \ldots, n$ and you want to fit an exponential model of the form,

$$\hat{y}_i = \sum_{j=0}^{d} w_j e^{-jx_i}$$

where x_i and y_i are scalars. You will use a linear basis function regression.

(a) Given training data $((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5), (x_6, y_6))$, write out the entries of the design matrix Φ you would use to fit the model above, for $d = 2$.

Solution: For the given training data,

$$\Phi(x) = \begin{bmatrix} 1 & e^{-x_1} & e^{-2x_1} \\ 1 & e^{-x_2} & e^{-2x_2} \\ 1 & e^{-x_3} & e^{-2x_3} \\ 1 & e^{-x_4} & e^{-2x_4} \\ 1 & e^{-x_5} & e^{-2x_5} \\ 1 & e^{-x_6} & e^{-2x_6} \end{bmatrix}$$

(b) Suppose you have training data loaded into a `numpy` array `x` in Python, and you also have a variable with the value of `d` (not necessarily 2 for this part). Write Python code to create the design matrix and save it in another variable, `X`.

Try to minimize your use of **`for`** loops - in fact, you can try to write an answer that is a single line of code!

Solution: You could use

```
X = np.exp(-x[:,None]*np.arange(0,d+1))
```

or equivalent.

4. **Gradient descent**.

For this question you will change some parameters in the "Gradient descent in depth" notebook, re-run the notebook with the new parameters,and answer questions about the results. You do not have to write any new code, and you should not submit any code.

(Copy the relevant output images to a regular document, answer the questions there, and submit that document - don't submit the Colab notebook with all the gradient descent code.)

(a) Re-run the "Descent path" section with three different learning rates: `lr = 0.0002`, `lr = 0.002`, and `lr = 0.02` (and leave other parameters at their default settings). For each learning rate,

i. Show the plot of coefficient value vs. iteration, and the plot of the descent path on the MSE contour.

ii. What is the estimate of w after 50 iterations?

iii. Describe whether the gradient descent diverges, converges within 50 iterations, or starts to converge but does not get very close to the optimum value within 50 iterations.

(b) Re-run the "Stochastic gradient descent" section with `lr=0.1` and `n=1`, then with `lr=0.01` and `n=10`, and finally with `lr = 0.001` and `n = 100` (and leave the other parameters at their default settings).

(Note: in this question, we are primarily observing the effect of changing the mini-batch size, n. We are varying the learning rate only so that the learning rate per sample is constant as we increase the mini-batch size.)

For each,

i. Show the plot of coefficient value vs. iteration, and the plot of the descent path on the MSE contour.

ii. Comment on the descent path. Does it go smoothly toward the optimal solution?

Solution: Refer to the gradient descent solution document.

5. **Linear regression on the Advertising data**.

Please refer to the homework notebook posted on the class site.

Solution: Refer to the solution notebook.