Grenoble INP - Ense3

AI and Autonomous systems

Lab 4 - M2-MARS

Linear models : Stochastic gradient descent and ridge (L2) regularization

Author :
Souhaiel BEN SALEM

15 novembre 2021

Lab Objectives: The objective of this lab is to illustrate regression models, in particular stochastic gradient descent and ridge (L2) regularization by experimenting and tuning the parameters of these linear models using tensorflow playground program in a first step and then applying these techniques on different datasets and for different purposes.

Introduction:

The term linear model implies that the model is specified as a linear combination of features. Based on training data, the learning process computes one weight for each feature to form a model that can predict or estimate the target value.
We will be exploring two types of linear models:

- Linear regression, which is used for regression (numerical predictions).
- Logistic regression, which is used for classification (categorical predictions)

1 Notebook 1: Learning Rate and Convergence of SGD algorihtm for a simple linear model

1.1 Task 1:

1.1.1 Exercise:

Observations: A high learning rate means that the learning process would be accelerated. During this learning process, the weights change drastically in each iteration (i.e epoch) which causes the model visualization to change dramatically:

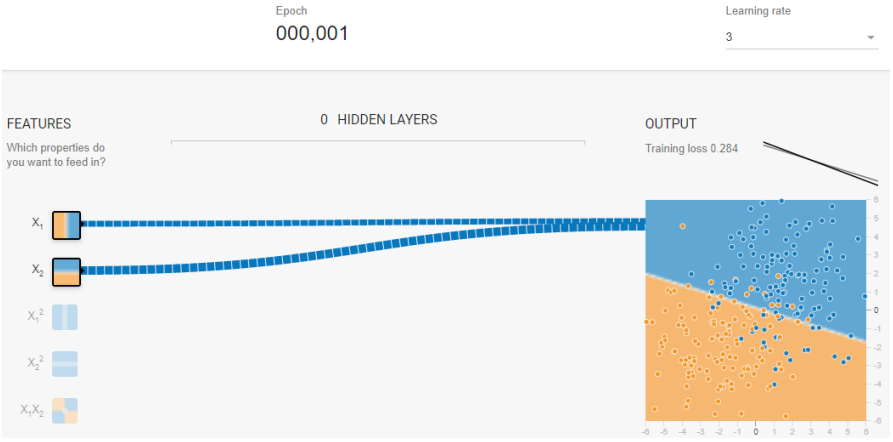


Figure 1: the separation boundary after 1 epoch

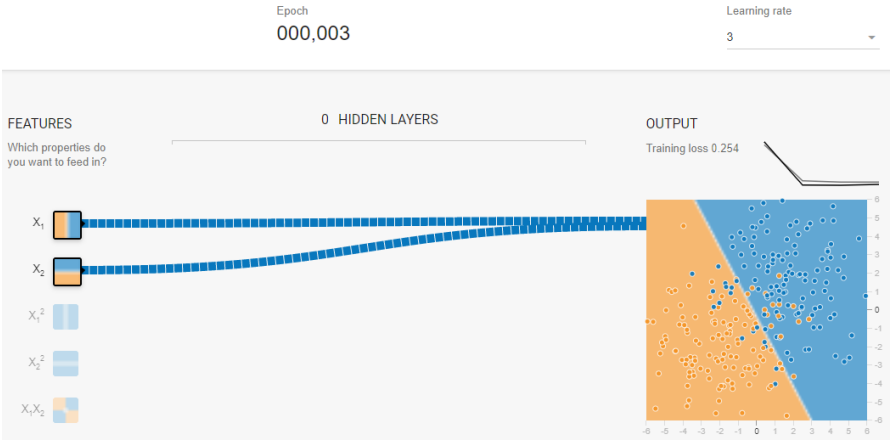


Figure 2: the separation boundary after 3 epochs

We also notice that the model converges quickly due to the high learning rate (the model converges roughly after 10 epochs). This practice can result in **instability** as the gradient can overshoot the minimum. Also, a learning rate that is too large can cause the model to converge too quickly to a **sub-optimal** solution.

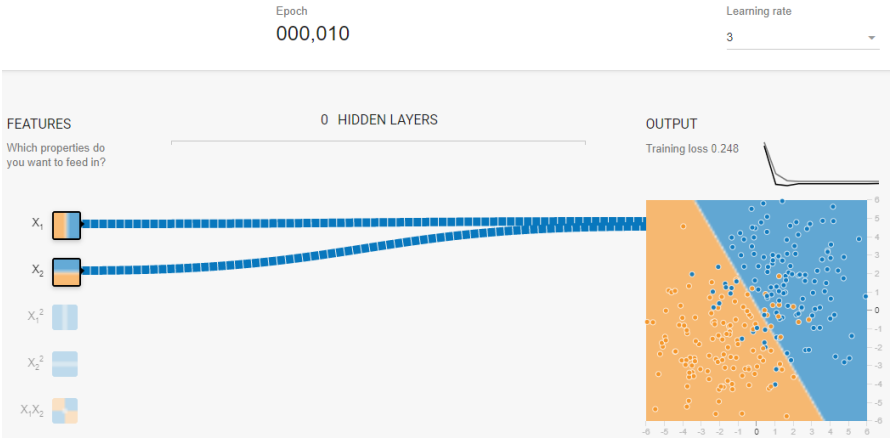


Figure 3: the model converges before the 10th epoch

1.2 Task 2:

1.2.1 Questions:

1. **How did the lower learning rate impact convergence? Examine both the number of steps needed for the model to converge, and also how smoothly and steadily the model converges.** After lowering the learning rate from 3 to 0.3 we can clearly see that the learning process is happening at a much slower rate.

We can also see that the model visualization change is not as dramatic as before as the decision boundary changes **slowly and smoothly** after each epoch. For this particular preset, our model need around 200 epochs to converge.

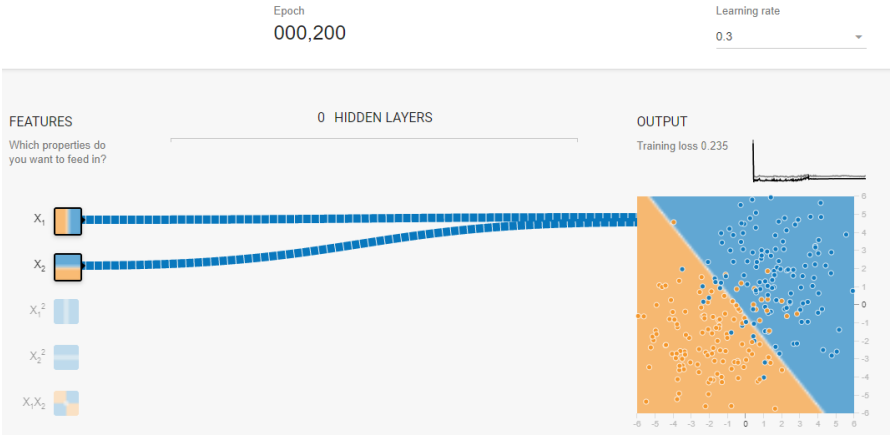


Figure 4: the model converges after 200 epochs

2. **Experiment with even lower values of learning rate. Can you find a learning rate too slow to be useful?**
   If we plug-in a very low learning rate (0.001 for example), the weights of our models would be updated at a very slow rate and the convergence would take a long time.
   Moreover, in practice and for large datasets, a too low learning rate can cause several problems, such as extremely long convergence time as the loss function does not improve with every epoch or getting stuck in a local minimum.

2 Notebook 2: Regularization for Simplicity: L2 (ridge) Regularization

2.1 Task 1:

For this task, we ran the first test with no L2 regularization ($\lambda = 0$) and then we ran another test with a penalization coefficient $\lambda = 0$.
Observations:

1. **Test loss**: We notice that after using L2 regularization the test error decreases.
2. **The delta between Test loss and Training loss**: We notice that the the difference of the test loss and training loss becomes higher (**in absolute vaulue**) after introducing L2. Indeed, $\Delta Loss = test\ loss - training\ loss$ went from 0.012 to 0.034.

This is expected since the L2 regularisation technique introduces the λ penalization coefficient in the loss function to penalize the high order terms which mean that the training error would naturally get higher. This is exactly the point of the regularization, where we increase bias and reduce the variance of the model. All of this means that our model would, hopefully, generalize better and that is why the test error decreases.

3. **The learned coefficients of the features and the feature crosses**: we notice that the weights got smaller, which is the result of our penalizing process (penalizing higher values of our model's coefficients). Here we are simplifying our model in order to avoid overfitting.

Figure 5: Result obtained after 500 epoch without using L2 regularization

Figure 6: Result obtained after 500 epoch while using L2 regularization

2.2 Task 2:

1. **How does the Test loss in Task 2 differ from the Test loss in Task 1?**
   The test loss in Task 2 is lower than that of Task 1.In fact, the test loss keeps decreasing when we keep increasing the penalty coefficient.
2. **How does the delta between Test loss and Training loss in Task 2 differ from that of Task 1?**
   The $\Delta Loss$ we get in Task 2 is higher than that we got in Task 1. In fact $\Delta Loss$ keeps getting higher when the penalization coefficient keeps getting higher which is expected since λ affects directly the training error (increase it) and improve our model generalisation capacities (hopefully).
3. **How do the learned ceofficients of each feature and feature cross differ from Task 2 to Task 1?**
   The learned coefficients become more and more smaller when increasing the L2 penalization coefficient.
4. **What do your results say about model complexity?**
   Depending of the value of λ, the model will penalize large values of coefficient estimates more and will reduce the model complexity more.

2.3 Task 3:

Experiment the same model with regularization rate: which value seems optimal?
For my personal experiment, it seems that $\lambda = 0.3$ is the optimal value as it led to more simple and stable model after 500 epochs.

Figure 7: Result obtained after 500 epoch for $\lambda = 0.3$

3 Notebook 3: Image Deconvolution using Ridge regularization

3.1 Exercise:

1. **What happens when there is no regularization ($\lambda = 0$)?**
   When there is no regularisation and we attempt recover the original image by deconvolution, we may face a noise explosion (noise amplification) that can occur for several reasons, the main one being that the Fourrier transform of our kernel (i.e point spread function (PSD)) hF**could be 0 for certain frequencies**, after all it is a low-pass filter so it could be 0 for high frequencies.In this case f is not recoverable and noise explodes because of the division by 0. In fact, the Fourrier transform of the noise is often high (ϵ) and the filter output is close to 0 for some of these frequencies.
   If try to recover the original image by simple deconvolution, we get a "**a division by 0**" error:

```
y_noridge= real(ifft2(yF/hF))
```

```
<ipython-input-70-fa02231ac3e5>:1: RuntimeWarning: divide by zero encountered in true_divide
  y_noridge= real(ifft2(yF/hF))
<ipython-input-70-fa02231ac3e5>:1: RuntimeWarning: invalid value encountered in true_divide
  y_noridge= real(ifft2(yF/hF))
```

Figure 8: Impossible to recover the original image

   Even if we don't get this error, there is a high chance that our recovered image would be completely useless as the noise will be amplified and the useful information will be lost.

2. **Is it the same even when the noise variance sigma is set to zero?**
   When the noise variance is set to 0, it means that the noise is constant (or follows a degenerate distribution in some cases). If the noise is constant (A) then its Fourrier transform would be a centered dirac with the magnitude of $2\pi A$. We would still have the same problem because hF could be 0 or very low (close to 0) for certain frequencies and the noise might explode.
   In fact this problem can occur even if there is no noise as we could have a "$0/0$" division.

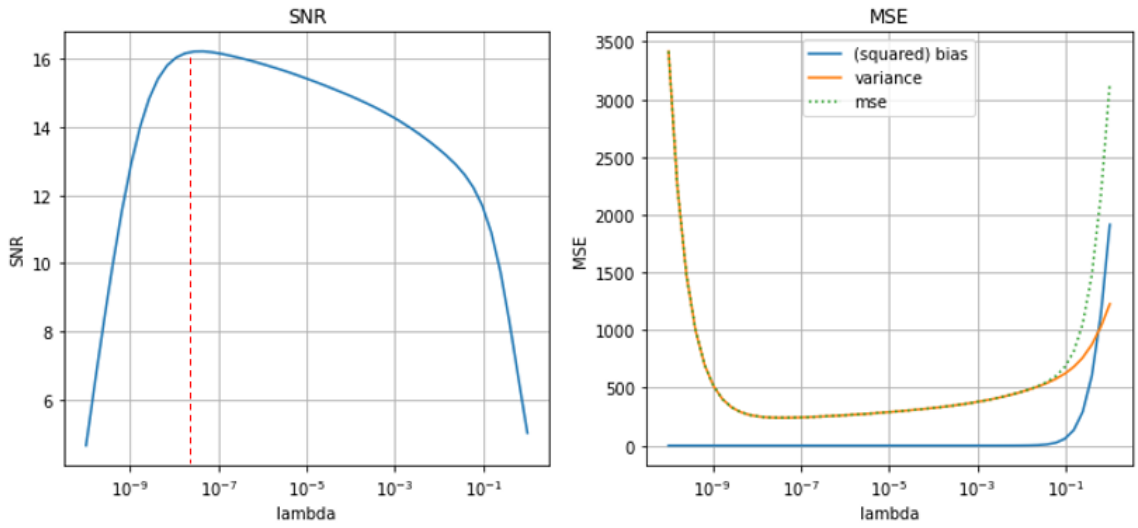
3.2 Exercise:

1. **Comment the shape of the SNR and MSE curves: what are the problems on each tail? Does it worth to introduce a small bias?**
   the problems on each tail:

   - **left tail of SNR graph (**$\lambda istoosmall$**)**: We get a low SNR which mean that the recovered signal (i.e image) is unusable, which is expected given the low value of λ that cause the noise to explode.
   - **left tail of MSE graph**: the bias is low, however the variance is very high between the original and the recovered images which means that the recovered image is not representative of the original one.

- **right tail of SNR graph (*$\lambda istoolarge$*)**: Once again, we notice that the SNR is way too low which means there is a lot of noise in the recovered image. Here the penalty on the the prior energy is too high.
- **right tail of MSE graph**: Both the variance and bias become large (**i.e MSE gets larger**, which is the worst case scenario.

2. **Find/Display/Comment the optimal solution**. The optimal solution would correspond to the highest SNR. So, to find the optimal solution (i.e optimal λ), we can just loop over the "snrs" array to find the highest SNR, take its index and determine the optimal regularization coefficient from the "lambdas" array.
   This can be achieved by these lines of code:




```
i = max(snrs)
n=where(snrs == i)
lambda_opt = lambdas[n]
y_optimal =real(ifft2(yF * hF / (abs(hF)**2 + lambda_opt)))
print (lambda_opt)
```

```
[4.49843267e-08]
```

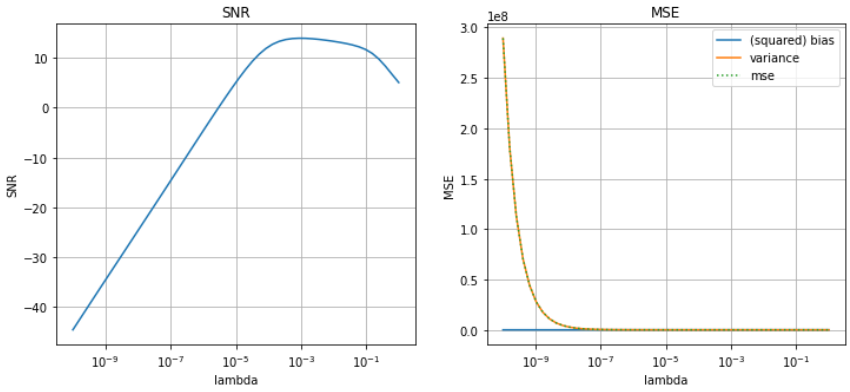
Figure 9: Determining the optimal value of λ

After determining the optimal value of λ, we can plot the optimal recovered image along with its SNR:

```
fig, ax = plt.subplots(1,2, figsize=(15,5))
ax[0].imshow(y_obs, interpolation='nearest', cmap='gray')
ax[0].axis('off')
ax[0].set_title('Noisy observed image, SNR={:.2f}'.format(snr(y_sharp, y_obs)));
ax[1].imshow(y_optimal, interpolation='nearest', cmap='gray')
ax[1].axis('off')
ax[1].set_title('Ridge filtered image, SNR={:.2f}'.format(snr(y_sharp, y_optimal)));
```

Figure 10: the reconstructed image corresponding to the optimal λ

3. **What happens for larger noise variance σ^2 ?**
   When the variance becomes large, the optimal regularization coefficient becomes larger and the image becomes more noisier and harder to reconstruct using the process deconvolution:




```
i = max(snrs)
n=where(snrs == i)
lambda_opt = lambdas[n]
y_optimal =real(ifft2(yF * hF / (abs(hF)**2 + lambda_opt)))
print (lambda_opt)
```

```
[0.00086851]
```

Figure 11: the optimal value of λ for a variance σ^2=3

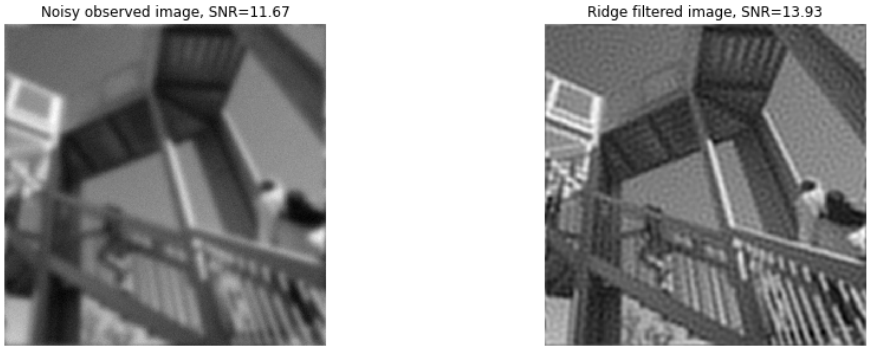


Figure 12: the reconstructed image for the optimal λ

4. **How can we proceed to efficiently compute the regularized solution when the Fourier approach is no longer suitable (for example when the blurring operation is not spatially invariant, or when the penalty is not quadratic as for the total variation prior energy)?**
   In such cases, the mathematical model is practically impossible to know a priori. Therefore, it is most of the time necessary to estimate PSF and restore the image at the same time (blind deconvolution).

4 Notebook 4: Ridge regression on the NIR biscuits dataset

4.1 Exercise:

1. **Q: What do you think about the smoothness of these spectra and the possible correlation between the input variables for the considered wavelength discretization**
   The spectra is is not smooth as it presents many fluctuations across the observed wave-lengths. This makes any correlation deduction, if it exists, hard to draw.

2. **Q: By visual inspection, is there a relationship between NIR spectra (input variables) and sugar content (target)?**
   The visual plot show that response has some fluctuations (quick changes in monotony) but in general the response keeps gets higher for higher wave-lengths independently of the sugar percentage %Y.

3. **Q:How can we explain that the residuals are all zero on the training set?**
   The 0-residuals obtained on the training set are the result of our model overfitting. This happened because we have way more features than samples so our model tend to identify single data points by single features and build a special case just for a single data point which causes overfitting.

4. **Q: Do you think that linear regression fit with ordinary least squares estimation is suitable here (explain why)?**
   Linear regression fit with ordinary least squares estimation (OLS) are not suitable in our case

because they lead to complex model that overfit the training data and has a large RSS because of its sensitivity to outliers. In fact, here we notice that our linear model has some large coefficients and large weights mean that if our input has a small error, when multiplied with large weight it become large error in output / prediction, so the model is unstable, so it has high variance and once again, this is a proof of overfitting and poor generalization.

5. **Q: What strategies can be investigated to improve the performances?**
   The prediction score $R^2 = 0.4702$ proves that the performance of our linear model is poor, as it only explains 47% of the data variance along the x axis.
   To improve the performance of our model, we need to reduce the dimensionality of our problem. To do so, we can **PCA** to remove the multicollinearity of the input variables.
   We can also use **Ridge regression** to remove the multicollinearity and lower the weights of the model and thus lower the variance and improve the performance.

6. **Q: Why PCR procedure may allows one to mitigate overfitting?**
   PCR uses only a subset of all the principal components for regression. In other words, the dimensions of the input variables is reduced by eliminating features with high correlation (**feature selection process**). Since PCA reduces the complexity of the model, we can expect that overfitting would be mitigated.

7. **Q: What are the hyperparameters to be optimized for PCR?**
   PCR, being a 2-step process (PCA + Linear regression), the hyperparameters to be optimized are the direction that maximize the variance (**number of components**) and the $\widehat{\beta_k}$ coefficients of the regression model.

8. **Q: what is the dimension reduction ratio with respect to (full) linear regression?**
   The full linear regression used all the 700 dimensions whereas PCR used only 7 principal components. So the problem's dimension was reduced a **100 times** $(700/7)$.

9. **Q: why the number of principal components to be considered can not exceed here 36 for the $K = 10$ fold cross-validated PCR?**
   In a data set, the maximum number of principal component loadings is a minimum of (n-1, p) where n denotes the samples and p the features. In our case we used a $K = 10$ fold cross-validated PCR on a dataset of 40 samples which means we are dealing with **36 training samples** for each iteration (case) where we take 36 samples for training and the remaining 4 for test. So, the actual maximum of components is 35 components.

10. **Q: Does it worth to use here PCR rather than linear regression?**
    Yes, using the PCR here improves the performance of our model as the variance is reduced and so we avoid overfitting. The performance improvement is shown by the fact that the prediction score is close to 1 ($R^2 = 0.9311$ which means that the variance is low).

11. **Q: Why ridge regression is referred to as a shrinkage procedure?**
    Ridge regression is considered as a shrinkage method because it shrinks the coefficients of the predictor β by enforcing a quadratic penalty to the candidate slope coefficients.

12. **Q: Based on all these results, is there valuable information in NIR spectra to predict sucrose content? What is the standard error magnitude for predicting the percentage of sucrose?**
Based on the results obtained, especially by using the **PCR** and **ridge regression**, we can see that the NIR spectra explains more than 90% of the variance of the data (according to the R^2 prediction score) which means that the predictions of sugar concentration we got on the test data are accurate. To get the MSE of our model we can either use the Sci-kit learn's mean_squared_error metric or use the fact that $R^2 = 1 - MSE/Var(y)$. We get MSE=0.6104.

13. **Q: How confident are you in the estimation of the R^2 score by cross-validation?**
Since K-fold cross validation evaluates the model on $K-1$ folds of the data every loop, we can be sure that all f the data points are used for test once and the performance measure reported at the end is the average of the values computed in the loop. This leads to a more robust model and lower bias. So, we can be confident about the prediction score obtained R^2.

14. **Q: How confident are you in the estimation of the best regularization coefficient alpha by cross-validation (hint: randomize the CV folds by setting shuffle=True and changing the random$_state$)?**
Shuffling in our context means that the data is first randomly shuffled before splitting into test/train. The random_state will allow the way in which the data is shuffled to be repeatable. After experimenting with these parameters, we can see that the regularization coefficient and prediction score depend on how the data is split before applying the k-fold cross validation. However, in this case we notice that the highest score when "shuffle" parameter is set to "False".

15. **Q: What are the best regularization coefficient the estimated R^2 for LOOCV on this dataset?**
If we apply LOO cross validation we get a 0 MSE on the training set and a 0 regularization coefficient α which mean that our model is overfitting.

```
# Use LOOCV (default setting)
alphas = np.logspace(-5, 0.5, 100)
ridge = RidgeCV(alphas=alphas, normalize=True)
ridge.fit(Xtrain,Ytrain)
Ytrain_hat =  ridge.predict(Xtrain)
Ytest_hat = ridge.predict(Xtest)
# plot the residuals
fig, ax = plt.subplots(figsize=(8,5))
plot_residuals(ax, 'Ridge regression')
# display best cross-validated hyperparameters
print(mean_squared_error(Ytest,Ytest_hat))

print('CV estimated ridge regularization coeff alpha: {:6.4f}'.format(ridge.alpha_))
```

```
9.789225187027776
CV estimated ridge regularization coeff alpha: 0.0000
```

Figure 13: the regression residuals obtained after LOOCV

Conclusion:

During this lab, we experimented with the SGD algorithm to optimise the parameters of our regression model. We have also experimented with $L2$ ridge regularization technique and its applications for different problems like deconvolution. We also examined the methods used to estimate the optimal regularization rate through the study of NIR biscuit dataset.