Machine Learning Homework 2

Megi Dervishi

May 20, 2020

Exercise 1

(a) Overfitting is especially a big problem when the training data size is low and the number of features to learn is very large compared to the data. By regularizing you avoid the risk of overfitting and hence have a better accuracy in the test data. So by choosing a good λ regularization significantly reduces the variance of the regression model without a huge increase in the bias of it. In this case the number of features are 784 while the training data has 900 images and the testing data has 250 images.

(b) I used the proximial gradient descent algorithm also known as ISTA for finding the θ_λ for the ℓ_1 norm and applied directly Gradient Descent for the ℓ_2 norm. For both graphs the increase of λ results in the increase of error. This is normal because if λ becomes too high then the bias of the model increases which results in lower accuracy. Also note that for the ℓ_1 norm the error increases much faster than for the ℓ_2 norm.

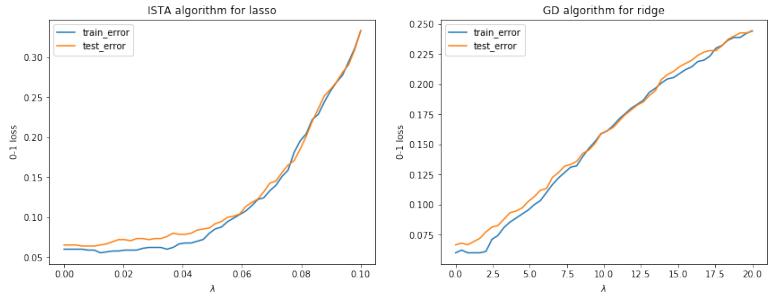

Figure 1: Error for the ISTA algorithm with $\|\cdot\|1$ penalty (left) and GD algorithm with $\|\cdot\|_2^2$ penalty (right).

(c) In order to best tune λ I performed K - fold cross validation which resulted with the following best lambdas: $\lambda_{lasso} = 0.012$ $\lambda_{ridge} = 1.10$

(d) In figure 2 with the increase of λs, we can observe the change of the shape of the estimator for lasso as well as the increase of error. For example in $\lambda = 0.01$ we can distinctly the features of A,C but as λ increases they fade away. On the other hand for ridge it seems as the estimators keeps the same shape however looking at the colorbar we notice that the whole estimator has been scaled down uniformly which results in higher error.

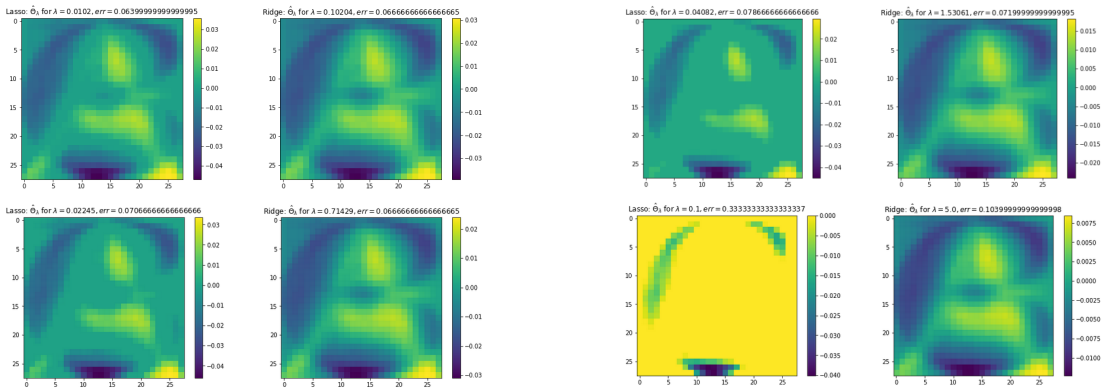

Figure 2: Example θ_λ for the Lasso and Ridge algorithms for different values of λ.

Exercise 2

(a) By Baysian Law we get the result (1). By law of total probability we find $\mathbb{P}[X = x]$and substitute it back into (1) which gives the final result (2):

$$\mathbb{P}[Y = 1|X] = \frac{\mathbb{P}[X = x|Y = 1]\,\mathbb{P}[Y = 1]}{\mathbb{P}[X = x]} = \frac{\mathcal{N}(\mu_1, \Sigma)(x) \cdot \pi}{\mathbb{P}[X = x]} \tag{1}$$

$$\begin{aligned}\mathbb{P}[X = x] &= \mathbb{P}[X = x|Y = 1]\,\mathbb{P}[y = 1] + \mathbb{P}[X = x|Y = -1]\,\mathbb{P}[y = -1] \\ &= \mathcal{N}(\mu_1, \Sigma)(x)\pi + \mathcal{N}(\mu_{-1}, \Sigma)(x)(1 - \pi)\end{aligned}$$

$$\frac{\mathcal{N}(\mu_1, \Sigma)(x) \cdot \pi}{\mathbb{P}[X = x]} = \frac{1}{1 + \frac{\mathcal{N}(\mu_{-1}, \Sigma)(x) \cdot (1-\pi)}{\mathcal{N}(\mu_1, \Sigma)(x) \cdot \pi}} \tag{2}$$

By substituting the normal density function and simplifying we have (3). Now we want to transform (3) into something of the form $\exp(-\beta_0 - \langle \beta, x \rangle) = \exp(-\beta_0 - tr(\beta^T x))$:

$$\frac{\mathcal{N}(\mu_{-1}, \Sigma)(x) \cdot (1 - \pi)}{\mathcal{N}(\mu_1, \Sigma)(x) \cdot \pi} = \exp(-\frac{1}{2}(x - \mu_{-1})^T \Sigma^{-1}(x - \mu_{-1}) + \frac{1}{2}(x - \mu_1)^T \Sigma^{-1}(x - \mu_1) + \log(\frac{1 - \pi}{\pi})) \tag{3}$$

$$\begin{aligned}&= \exp(\frac{1}{2}(\mu_{-1}^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu_{-1}) + \frac{1}{2}(-\mu_1^T \Sigma^{-1} x - x^T \Sigma^{-1} \mu_1) + \frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1 - \mu_{-1}^T \Sigma^{-1} \mu_{-1}) + \log(\frac{1 - \pi}{\pi})) \\ &= \exp(\frac{1}{2}(\mu_{-1}^T \Sigma^{-1} x + \mu_{-1}^T \Sigma^{-1} x) + \frac{1}{2}(-\mu_1^T \Sigma^{-1} x - \mu_1^T \Sigma^{-1} x) - \underbrace{(\frac{1}{2}(\mu_{-1}^T \Sigma^{-1} \mu_{-1} + \mu_1^T \Sigma^{-1} \mu_1) - \log(\frac{1 - \pi}{\pi}))}_{\beta_0}) \\ &= \exp((\mu - 1^T \Sigma^{-1} + \mu_1^T \Sigma^{-1}) x - \beta_0) \\ &= \exp(-(\mu_1^T \Sigma^{-1} - \mu_{-1}^T \Sigma^{-1}) x - \beta_0) \\ &= \exp(-\beta^T x - \beta_0)\end{aligned}$$

Since $-\beta^T x$ is a value then it is equal to $tr(-\beta^T x)$. We finally conclude that indeed:

$$\mathbb{P}[Y = 1|X] = \frac{1}{1 + \exp(-\beta_0 - \langle \beta, X \rangle)}$$

(b) We have that:

$$\mathcal{L}(\beta_0, \beta) = \prod_{j=1}^{n} \mathbb{P}[Y = y_j | X = x_j] = \prod_{j=1}^{n} \frac{1}{1 + \exp(-y_j \beta_0 - y_j \langle \beta, x_j \rangle)} \tag{4}$$

The reason behind the second equality is because we notice that $\mathbb{P}[Y = -1|X] = \frac{1}{1+\exp(\beta_0 + \langle \beta, X \rangle)}$ and by (a) we can generalize $\mathbb{P}[Y = j|X = x_j]$ to the above expression. We have to maximize (4) in other words find $\arg \max_{\beta_0, \beta \in \mathbb{R}, \mathbb{R}^d} (\mathcal{L}(\beta_0, \beta))$. Since log is an increasing function we can take the log of the likelihood. Hence:

$$\begin{aligned}\arg \max_{\beta_0, \beta \in \mathbb{R}, \mathbb{R}^d} \log(\mathcal{L}(\beta_0, \beta)) &= \arg \max_{\beta_0, \beta \in \mathbb{R}, \mathbb{R}^d} \sum_{j=1}^{n} \log(\frac{1}{1 + \exp(-y_j(\beta_0 + \langle \beta, x_j \rangle))}) \\ &= \arg \max_{\beta_0, \beta \in \mathbb{R}, \mathbb{R}^d} -\sum_{j=1}^{n} \log(1 + \exp(-y_j(\beta_0 + \langle \beta, x_j \rangle)) \\ &= \arg \min_{\beta_0, \beta \in \mathbb{R}, \mathbb{R}^d} \frac{1}{n} \sum_{j=1}^{n} \log(1 + \exp(-y_j(\beta_0 + \langle \beta, x_j \rangle)))\end{aligned}$$

If we let $\beta_0 = 0$ then this corresponds to the solution of logistic regression with $\lambda = 0$. Note(since $\frac{1}{n}$ is a constant adding it to arg min does not change).

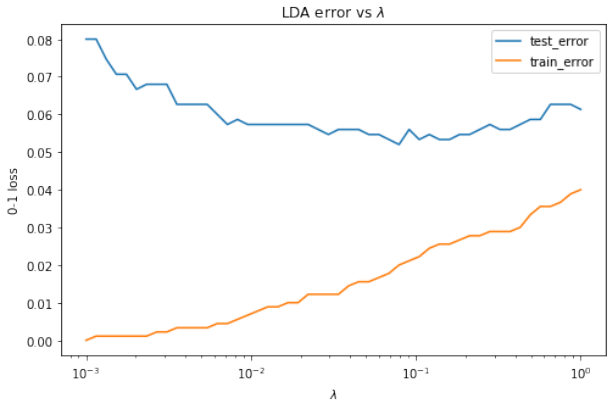

Figure 3: Evolution of the error for the LDA-classifier as a function of λ.

(c) For the algorithm that I implemented I based myself on the following: https://eigenfoo.xyz/lda/. To improve the results I add $\lambda_0 = 0.08$. In figure 3 we have computed the error for multiple values of λ and we see that λ does indeed act like a regularization since it reduces the overfitting with a small bias change. However it simply describes an added isotropic variance, which slightly expands in all directions the gaussian 'bells' of the two distributions.

(d) False positive means that you predict positive when the correct answer is false and False negative means that you predict negative when the correct answer is true. A classification error simply describes the ratio between the number of wrongly predicted values and the total number of values. The confusion matrix shows the ways in which your model was confused. In other words it gives more insights on which label was wrongly predicted i.e. when do you have false positive or false negative. By having these extra information you could pin-point better where the error comes from and perhaps improve it. In figure 4 we have computed the confusion matrices for our model which have a very good performance. There may be some slight overfitting however overall it is good.

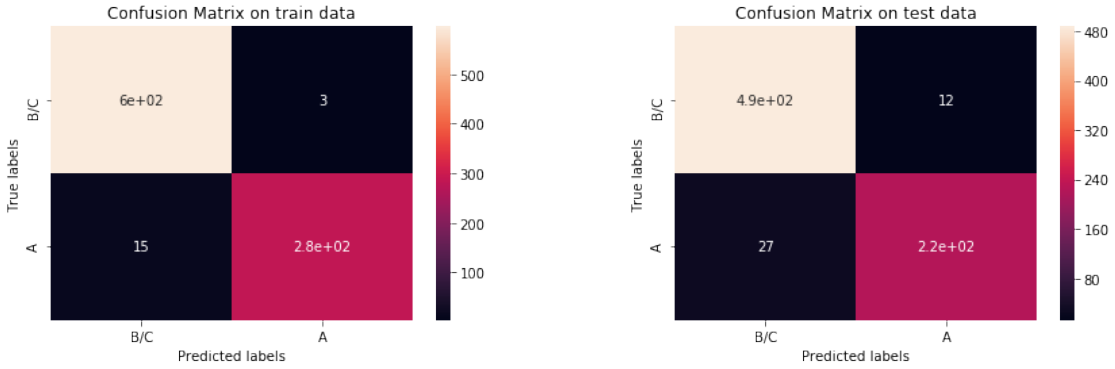

Figure 4: Confusion matrix from the LDA-classification with a $\lambda = 0.08$, a train accuracy of 98% and test accuracy of 94.8%.

Exercise 3

(a) Since we are doing unsupervised learning I have reloaded the data into one and reshuffled it. The dataset is composed by 1650 of 28 by 28 pixels images which are normalized. Hence the input space is $\mathcal{X} \in [0,1]^{784}$ and $\mathcal{Y} \in \{0,1,2\}$ i.e. the image will have a label $Y_i = 0$ if it shows "A", 1 if it shows "B" and 2 if it shows "C". In figure 5 we have the PCA representation of the best result out of 10 iterations of our k-means algorithm. The clusters are overlapping in the centered PCA however when we compare this with the `sklearn` PCA we obtain the same result.

(b) We get an error of 20% which may seem as a lot. However when looking at the confusion matrix we observe that the model mostly confuses the "B" and the "C" and is always good at separating those from the "A". The main source of error comes from the false positives of 'C' which turn out to be 'B'. This probably comes from the fact B and C have similar features which we can notice from the images of the θ in Figure 2 where A is distinctly recognizable whereas B and C can be merged together easily.

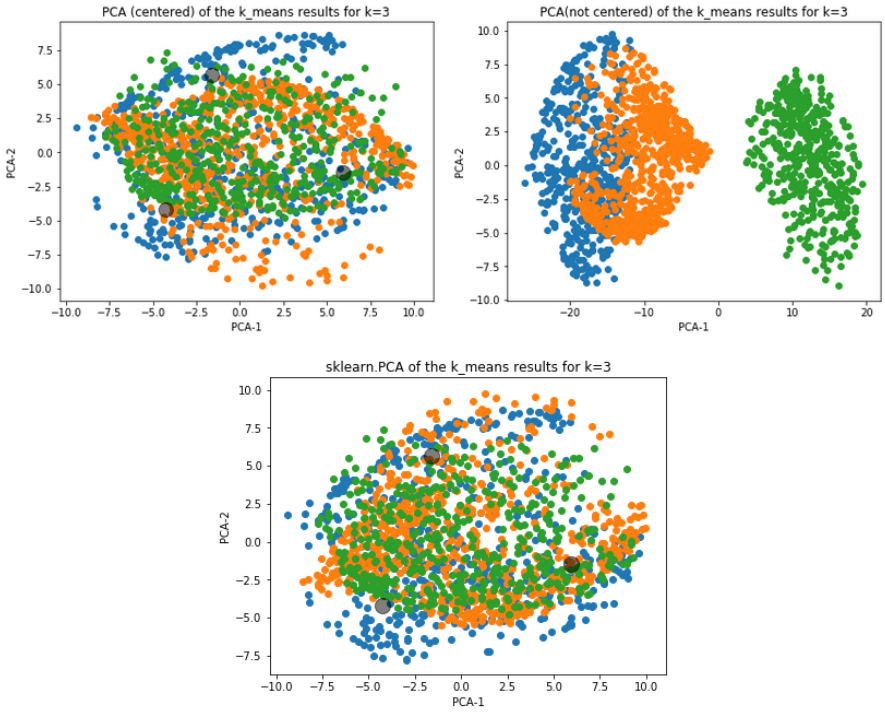

Figure 5: PCA results of the k-means algorithm with k = 3.

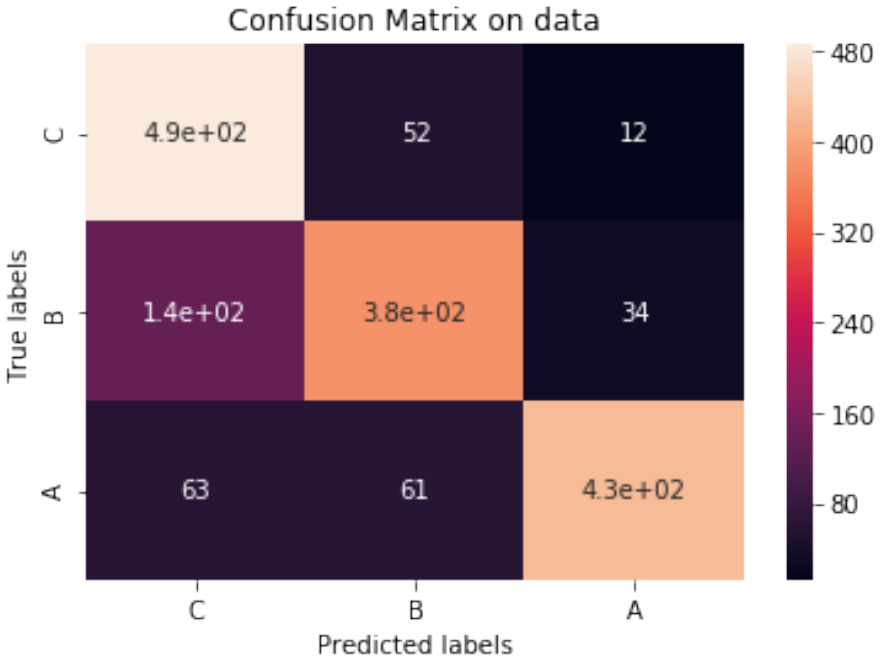

Figure 6: Confusion Matrix of k-means