Deep Learning for Image Analysis

Author: *Mohammed Al-Jaff*
Assignment 1: *Linear Regression & Gradient Decent*
Date: 2021-April-10

Table of Content

- Deep Learning for Image Analysis
  - Mathematical and Conceptual Formalisation and Notation
    - The linear regression model
      - (Exercise 1.2) - Deriving expressions for various partial derivatives of the linear regression model
    - The loss function
      - (Exercise 1.2) - Deriving expressions for various partial derivatives of the loss function
    - The empirical risk function, or more commonly referred as 'Cost' function.
  - (Exercise 1.1) - Deriving expressions for various partial derivatives of the empirical risk J
  - (Exercise 1.4) - Using a linear regression model for the 'Auto.csv' dataset using i) one data feature and ii) all data features.
    - Relationship between learning rate and convergence 'rate'.
    - Comparing learned parameters between standardised and unstandardised features.
    - (Exercise 1.4.2) Converged parameters and reality check.
      - Model 2 expression (using all 7 features for input datapoint).
      - Model 1 expression (using only the 'horsepower' feature):

---

1. Mathematical and Conceptual Formalisation and Notation

In this section of the report, I will give a brief recap of the three main functions for this assignment and their various partial derivates that are relevant for the solution to exercises 1.1 and 1.2 of the assignment. The main outline here is to briefly define I) the linear regression model for a scaler output and vector valued input. II) the loss function, L relating the per-prediction instance 'cost' of a predicted value with the corresponding true value. And III) the 'cost function' J [which should really be referred to as the *empirical risk*, given its common term-usage in statistical learning theory. See: ***Vapnik, Vladimir. "Principles of risk minimization for learning theory." Advances in neural information processing systems. 1992.*** & ***Bousquet O., Boucheron S., Lugosi G. (2004) Introduction to Statistical Learning Theory. In: Bousquet O., von Luxburg U., Rätsch G. (eds) Advanced Lectures on Machine Learning. ML 2003. Lecture Notes in Computer Science, vol 3176. Springer, Berlin, Heidelberg.***] which is a measure of the total 'cost' associated with predicting using a given model with a certain parameterisation on a set of data.

A note on notation: Notation is tricky when there are only so many letters and symbols available with the added difficulty of certain symbols having a historic de facto connotation / meaning. I have chosen the following conventions so as to keep consistent the relationship between the 'iterated variable' and its final value'

"Fixed" values and 'Constants' will always be represented by upper-case letters. For example, i) size of a data set -> Upper case 'N', N. ii) Dimension of a space -> upper-case 'D', eg $\mathbb{R}^D$.

"Iterated" variables and the likes will always be the lower case letter of the value it will be at most or the 'target' value. For example:

Ex1: Dataset of size N of (input, output) tuples:

$$\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$$

Ex2: d is the iterated variable that goes from some starting value to its "largest"/"target" value of D.

$$\mathbf{w}^\top \mathbf{x} = \sum_{d=1}^{D} w_d x_d$$

1.1. The linear regression model

A *linear regression model* is defined in this assignment as a function $z : \mathbb{R}^D \to \mathbb{R}$ *parametrized by* the parameters $\mathbf{w}$ and b, such that:

$$z(\mathbf{x}; \mathbf{w}, b) := b + \mathbf{w}^\top \mathbf{x} = b + \sum_{d=1}^{D} w_d x_d \qquad (1)$$

where the model relationship between input data-points $\mathbf{x_i}$ and output value is through an 'affine transformation', in this specific case an inner product calculation with the *weights vector* $\mathbf{w}$ and the addition of an *offset* b.

The output value for a specific input specific prediction/output value for a specific input $\mathbf{x}_i$ is simply a matter of 'inheriting the index of the input data-point':

$$z_i := z(\mathbf{x}_i; \mathbf{w}, b) := b + \mathbf{w}^\top \mathbf{x}_i = b + \sum_{d=1}^{D} w_d x_{i,d} \qquad (2)$$

1.1.1. (Exercise 1.2) - Deriving expressions for various partial derivatives of the linear regression model

In this section, I'll be deriving the expressions for the partial derivatives of the linear regression model (1)/(2) with respective to its parameters (element-wise) w_j of $\mathbf{w}$ and b:

Expression for $\frac{\partial z}{\partial w_j}$:

$$\frac{\partial z}{\partial w_j} = \frac{\partial}{\partial w_j}\left[b + \sum_{d=1}^{D} w_d x_d\right] = x_j$$

and for a specific z_n, likes: Expression for $\frac{\partial z_n}{\partial w_j}$:

$$\frac{\partial z_n}{\partial w_j} = \frac{\partial}{\partial w_j}\left[b + \sum_{d=1}^{D} w_d x_{n,d}\right] = x_{n,j} \qquad (3).$$

Expression for $\frac{\partial z}{\partial b}$:

$$\frac{\partial z}{\partial b} = \frac{\partial}{\partial b}\left[b + \sum_{d=1}^{D} w_d x_d\right] = 1 \qquad (4)$$

1.2. The loss function

The loss function, L, written as:

$$L(z(\mathbf{x_i}; \mathbf{w}, b), y_i)$$

is a function that aims to capture the notion of a 'cost' or 'loss' for predicting a certain value z_i for a given input $\mathbf{x_i}$ when the actual corresponding true value is y_i. In this assignment, since we are dealing with a regression model, the loss function is defined as the *'squared error'*

$$L(z(\mathbf{x}; \mathbf{w}, b), y) := (y - z(\mathbf{x}; \mathbf{w}, b))^2 \qquad (5).$$

1.2.1. (Exercise 1.2) - Deriving expressions for various partial derivatives of the loss function

The main partial derivative of relevance for later is the expression for $\frac{\partial L}{\partial z}$:

$$\frac{\partial L}{\partial z} = \frac{\partial}{\partial z}(y - z(\mathbf{x}; \mathbf{w}, b))^2 = 2(z(\mathbf{x}; \mathbf{w}, b) - y)$$

or in the "per-instance" case:

$$\frac{\partial L}{\partial z_i} = \frac{\partial}{\partial z_i}(y - z_i)^2 = 2(y_i - z_i)(-1) = 2(z_i - y) \qquad (6)$$

1.3. The empirical risk function, or more commonly referred as 'Cost' function.

The cost function is essentially an estimate of the expected loss associated with a given model. Given that we in most cases do not know the data-generating mechanism that produces our data-set, talking about an 'expectation' of a loss value remains theoretical. But because we have a dataset, we estimate this by the data we have at hand using the data "sample-average" loss as:

$$J(\mathbf{w}, b) := \frac{1}{N}\sum_{n=1}^{N} L(y_n, z(\mathbf{x_n}; \mathbf{w}, b)) = \frac{1}{N}\sum_{n=1}^{N}(y_n - z_n)^2 \qquad (7)$$

where we sum over all data points $\mathbf{x}_n$ in our dataset $\mathcal{D} := \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$

2. (Exercise 1.1) - Deriving expressions for various partial derivatives of the empirical risk J

I) Expresion for $\frac{\partial J}{\partial b}$ in terms of $\frac{\partial L}{\partial z_n}$ and $\frac{\partial z_n}{\partial b}$:

$$\frac{\partial J}{\partial b} = \frac{\partial}{\partial b}\left(\frac{1}{N}\sum_{n=1}^{N} L(y_n, z(\mathbf{x_n}; \mathbf{w}, b))\right) = \frac{1}{N}\sum_{n=1}^{N} \frac{\partial}{\partial b} L(y_n, z(\mathbf{x_n}; \mathbf{w}, b))$$

Because the loss function is implicitly dependent on b through the (per-instance) linear regression model z_n, we need to apply the chain-rule to get the expression for the summand:

$$\frac{\partial}{\partial b} L(y_n, z(\mathbf{x_n}; \mathbf{w}, b)) = \frac{\partial L}{\partial z_n}\frac{\partial z_n}{\partial b} \qquad (8)$$

Substation of the above expression into the summand expression for $\frac{\partial J}{\partial b}$ gives us the expression in terms of $\frac{\partial L}{\partial z_n}$ and $\frac{\partial z_n}{\partial b}$:

$$\frac{\partial J}{\partial b} = \frac{1}{N}\sum_{n=1}^{N} \frac{\partial L}{\partial z_n}\frac{\partial z_n}{\partial b} \qquad (9)$$

Using (4) and (6) for their respective "lowest level" expressions gives us the final desired expression for the partial derivative of the cost function with respect to the offset parameter of the linear regression model:

$$\frac{\partial J}{\partial b} = \frac{1}{N}\sum_{n=1}^{N} 2(z_n - y_n) \qquad (10)$$

II) Expression for $\frac{\partial J}{\partial w_j}$ in terms of $\frac{\partial L}{\partial z_n}$ and $\frac{\partial z_n}{\partial w_j}$: Like in the above case with the partial derivate w.r.t the offset b, we will take the partial derivative w.r.t an arbitrary weight w_j of the whole cost expression and work our way 'inwards' until we hit a function where we need to apply the chain rule due to implicit dependence:

$$\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j}\left(\frac{1}{N}\sum_{n=1}^{N} L(y_n, z(\mathbf{x_n}; \mathbf{w}, b))\right) = \frac{1}{N}\sum_{n=1}^{N} \frac{\partial}{\partial w_j} L(y_n, z(\mathbf{x_n}; \mathbf{w}, b)) \qquad (11)$$

Same as in the previous situation, the loss is dependent on the model weights implicitly through the linear regression model function, thus requiring us to use the chain rule inside the summand:

$$\frac{\partial}{\partial w_j} L(y_n, z(\mathbf{x_n}; \mathbf{w}, b)) = \frac{\partial L}{\partial z} \frac{\partial z}{\partial w_j} \qquad (12)$$

Replacing the summand in (11) with the above expression gives us the asked for expression for $\frac{\partial J}{\partial w_j}$ in terms of $\frac{\partial L}{\partial z_n}$ and $\frac{\partial z_n}{\partial w_j}$:

$$\frac{\partial J}{\partial w_j} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial L}{\partial z_n} \frac{\partial z_n}{\partial w_j} \qquad (13)$$

Finally, substituting the expressions for $\frac{\partial L}{\partial z_n}$ and $\frac{\partial z_n}{\partial w_j}$ with their actual 'base level' expressions gives us

$$\frac{\partial J}{\partial w_j} = \frac{1}{N} \sum_{n=1}^{N} 2(z_n - y_n) x_{n,j} \qquad (14)$$

---

3. (Exercise 1.4) - Using a linear regression model for the 'Auto.csv' dataset using i) one data feature and ii) all data features.

In this section we describe and interpret our results of applying our linear regression code to model the relationship between a vehicle's horsepower and a number of other properties collected for 392 different vehicles in the 'Auto.cvs' dataset. In one version (referred to as 'Model 1'), we will only model the relationship between 'miles per gallon' (**msg**) and the 'horsepower' feature. In a second experiment (referred to as 'Model 7'), '**msg**' will likewise be treated as the output value but the input data will include all 7 features (except car 'name'). In both cases, we will 'learn' the relationship from the data, via empirical risk minimisation through the gradient decent algorithm that we were required to implement.

3.1. Relationship between learning rate and convergence 'rate'.

As mentioned above, gradient decent was used to find weights and offset values that minimised the empirical risk/cost function (7). 'Learning from the data' with gradient decent here involves updating the model linear regression parameters (the weights $\mathbf{w}$ and offset b) iteratively in the below fashion

$$\mathbf{w}_{new} \leftarrow \mathbf{w}_{old} - \alpha \nabla_{\mathbf{w}} J \qquad (15.1)$$

$$b_{new} \leftarrow b_{old} - \alpha \nabla_b J \qquad (15.2)$$

either a fixed number of iterations, or until some criteria of 'convergence' occurs. Normally, 'convergence' happens once we reach a stage where iterative parameter updates result in the cost difference being below some 'epsilon' value. Note however that there is an important crucial hyperparameter at play here in the update formula, namely the learning rate α which controls the amount by which we update our parameters. In general, relatively large learning rates have the benefit of making the 'training/learning' phase faster towards converge. On the other hand, a large learning comes with the added risk of either oscillatory update behaviour or even worse, non-convergence away from a stable minima of the cost function. A small learning rate counteracts these risks, but comes with the downsides of longer learning/training times (ie more iterations required) and possibly ending up getting stuck in a local minima of the "loss-surface" of expression (7). In figures 1 and 2 below we depict the training phase for both models 1 and 2 for various learning rates and how they influence the number of iterations until convergence.

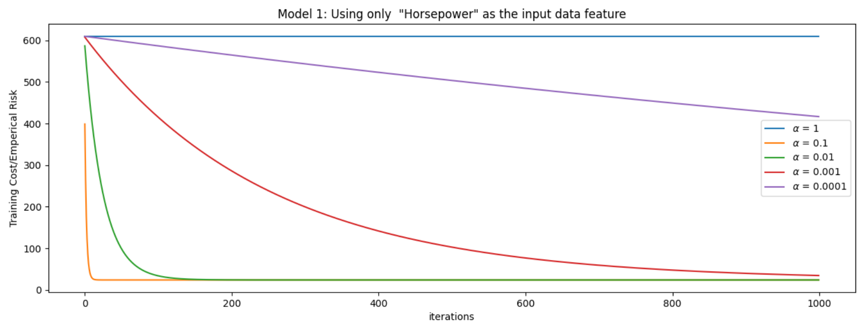

Figure 1: Model 1 case - *Learning rates* $\alpha = 1, 0.1, 0.01, 0.001, 0.0001$

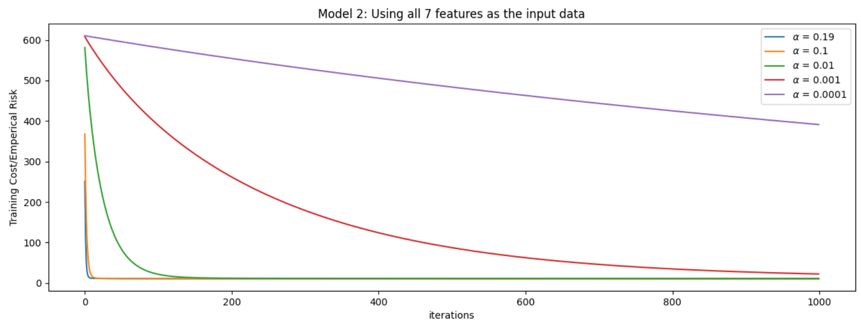

Figure 2: Model 2 case - *Learning rates* $\alpha = 0.19, 0.1, 0.01, 0.001, 0.0001$ for model 2.

In both model cases, we see the general trend of smaller learning rates resulting in slower convergence times, ie the smaller we make the learning rate α the more iterations required until we reach a stable minimum cost. A key interesting thing that we can observe is that for the highest learning rate of $\alpha = 1.0$ in model 1, we don't even converge to the global minima but stay at constant cost of around 600. This is very strange since we should either diverge towards infinity or converge towards the global minima since we are dealing with a quadratic loss with a linear model and no non-linearities. My understanding so far is that this should give rise to a single global optima since the loss is quadratic and therefore a convex function. If I had to guess how and why we end up at a constant loss at around 600 it could be that we might be bouncing around a set up updates that result in the same loss and that update 'to each other' because the negative gradient 'sends' one to the other' so to say(?).

An additional thing to notice is that we have a maximum learning rate or 0.19 in the modal 2 case (all 7 features as input data). This is because gradient decent 'blows up' and diverges towards infinity for any larger learning rate value above 0.2.

3.2. Comparing learned parameters between standardised and unstandardised features.

In this section we compare the learned parameters when we model the relationship between the output 'msg' and the input 'horsepower' when i) we standardise the input feature and ii) when we forgo standardisation and just use the raw input values for each data-point. I chose to have the same learning rate for both model to be able to compare training times between the two situations. The learning rate I chose was 0.0001 because any larger value resulted in the 'non-standardised' case to diverge to infinity.
In figure 3 below we can see the comparison between the two training phases in terms of run times.

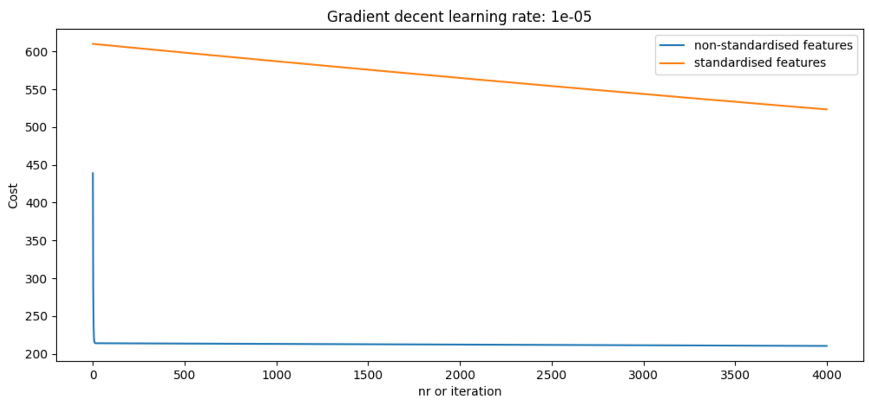

Figure 3: *The training phase of linear regression models with gradient decent for two models with same data but different input feature standardisation strategies : (blue) non-standardised features. (orange) standardised features.*

One observation is the relative speeds of convergence between the two cases: For non-standardised input features (and with a good enough tuned learning rate), gradient decent manages to converge much faster than when we standardise the input data, *given the same learning rate*. Although the take-home message here shouldn't be that we should not standardise features but notice and realise that good choices of learning rates is standardisation depended.

Things turn out worse for the non-standardised features case when we look at the minimum cost it converges to and the resulting best-fit line. In figure 4 below we plot the true and model predicted 'miles per gallon' output values for each respective non-standardised 'horsepower' features. In the same figure we have also plotted the 'best-fit' line corresponding to the parameters the gradient decent algorithm converged to (ca ~ 230). Here we see that we get a positive reaction between 'horsepower' and 'miles per gallon' which is both counterintuitive (and rightly so) and "wrong" when we also see that we actually have more or less negative correlation between the two features. This indicates that despite converging to some parameters, the resulting model is far from accurate. Contrast this with the same type of figure but for the resulting model where we had standardised our input feature (figure 5).

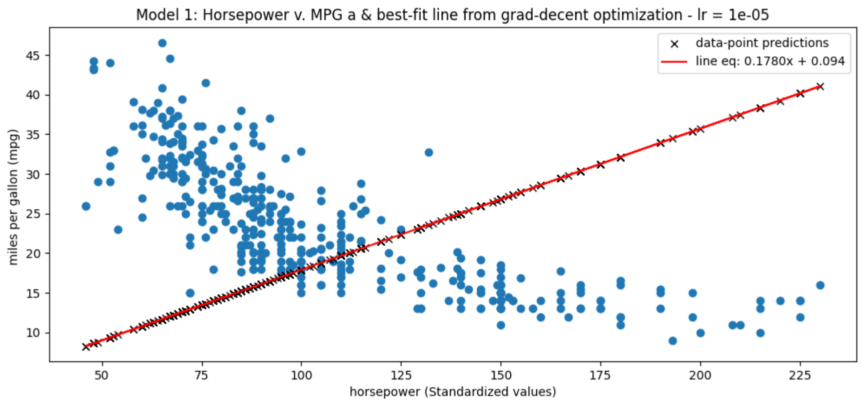

Figure 4: *Plot of true and predicted data-points between miles per gallon' and the non-standardised 'horsepower' feature.* The red line represents the best-fit line of the linear regression model when trained with non-standardised data. Black crosses are the predicted output according to the model for each input. Blue dots are the true data-points.

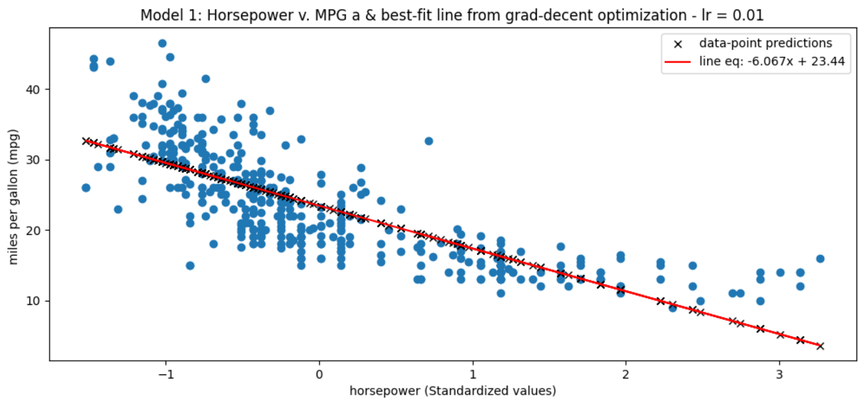

Figure 5: *Plot of true and predicted data-points between miles per gallon' and standardised 'horsepower' feature.* The red line represents the best-fit line of the linear regression model when trained with standardised data. Black crosses are the predicted output according to the model for each input. Blue dots are the true data-points.

Here we see a much more better fit to the data, at least in a linear approximation way.

3.3. (Exercise 1.4.2) Converged parameters and reality check.

In this section I'll present the final trained models and write down their mathematical expressions with the learned weights and offset. I'll also try to check if the sign of the weights for each feature 'make sense' w.r.p it effect on the output value 'milles per gallon'.

3.3.1. Model 2 expression (using all 7 features for input datapoint).

The final weights and offset for model 2 are below expressed in the linear regression term:

$$'miles - per - gallon' = -0.37cylinders + 0.50displacement - 0.95horsepower - 4.32weight - 0.09acceleration + 2.68year + 1.08origin + 23.44$$

- The weights for $cylinders, horsepower$ and $acceleration$ are all negative and therefore have an 'opposite' influence on 'miles-per-gallon'. This makes a lot of sense given that engines with higher values of these properties are often ones that are designed with performance in mind rather than on fuel efficiency and therefor are more 'fuel thirsty'.
- We also find a negative weight associated with 'vehicle weight'. This also happens to be the largest weight in absolute terms. Here to, this makes perfect sense because the heavier the vehicle, the more work the engine needs to put in there-by reducing the amount it can travel for a unit amount of fuel.
- It's interesting to see a positive weight associated with the 'year' feature because it could be rationalised as 'new model cars tend to be more fuel efficient'.
- It appears that 'origin' contributes positively to the output value but here I don't think it makes much sense to interpret too much about this weight because the underlying feature consist of non-ordinal and categorical values/types which we can assign numerical values arbitrarily and is less suited for linear regression models and more suited for eg decision tree's based models.
- I think the offset can be more or less whatever it should be so as to minimise the cost in relation to the other weights and take care of the 'all zero case'. If I'm not misstaken, I wouldn't put too much interpretation on the value of the offset apart from it being sufficiently positive to ensure we end with a reasonable non-negative valued 'miles-per-gallon'.

3.3.2. Model 1 expression (using only the 'horsepower' feature):

The final weights and offset for model 1 are below expressed in the linear regression term:

$$'miles - per - gallon' = -6.06horsepower + 23.44$$

Here, the weight associated 'horsepower' makes sense as in the case for similar reasons.