Notes on Report Writing

In each assignment you are asked to write a report which presents your data/results and discusses them. You're primary aim in these reports is to (i) show clearly and unambiguously that you understand the tools that you are using and (ii) can critically evaluate data and results. It's not enough to just run a program and quote numbers that it outputs. Also, although you will be using software as a tool to perform calculations, visualise data etc, typically you will mostly be combining code snippets given in the lectures and the software is a relatively minor part of the work you do (it's a means to an end rather than the end itself). With that in mind, here are some tips on writing good reports.

- *Numerical results.* When presenting numerical data it's important to:
  - Describe how the values were calculated and what they represent. If you just present an array of numbers, e.g. [0.1, 0.5, -0.6], without further comment then I am left to guess what they mean. Instead you might, for example, say that you "trained a linear model $y = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$ with three features and the trained parameter values are $\theta_1 = 0.1, \theta_2 = 0.5, \theta_3 = -0.6$".
  - Comment upon/discuss the data. E.g. in the example above you might note that "since θ_2 is larger than θ_1 then greater weight is placed by the model on feature x_2 than on feature x_1, and since θ_3 is negative and large then increases in feature x_3 will cause output y to decrease.". Depending on the context, you might then comment briefly as to whether this behaviour is reasonable/expected.
- *Plotting data.* Visualising data well is important for helping to understand it. When plotting data you should:
  - Spend a little while thinking about how to present the data in a clear way - sometimes that involves a bit of trial and error.
  - Make sure lines are thick enough and markers used in scatter plots are large enough to be easily seen. Choose colours and marker shapes to make the plot easy to read.
  - Always label the axes.
  - Make sure all text (including axes labels/ticks, any legend etc) is large enough to be clearly legible
  - In supporting text (i) describe how the values in the plot were calculated and what they represent and (ii) comment upon/discuss the data plotted (similarly to when you present numerical data, see above).

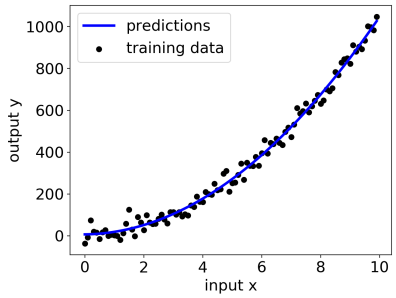


Figure 1: Example plot

  - Figure 1 shows an example of good practice. Observe that both axes are labelled and the text is legible. The black dots mark the training data and the blue line predictions from a machine learning model. By using different colours the data is easier to read, the line and the dots are big enough to be easy to read. Since the training data is noisy and consists of individual data points its better to plot it as a separate points (i.e. as a scatter plot) rather than joining the dots with a line. There's no need to make the plot itself large.
- *Code that carries out calculations.* Its a serious mistake to treat code as a black box that generates mystery numbers. Packages such as sklearn make this v easy to do and so you need to make an active effort to avoid this.

  - When you use an sklearn function make sure that you understand what the output values represent and how they are calculated. For example, some sklearn functions generate "score" values, and if you use one of these then its essential that you understand how the score value is calculated, namely whether its a mean square error, classification accuracy and so on. And you must state this in your report, otherwise I won't know that you understand what the values really are.
  - Sometimes you'll be asked to implement machine learning calculations from scratch, rather than using sklearn. In that case in your report you need to discuss/explain what the code does, e.g. by including relevant short snippets of code in your report write-up along with text explaining what it does. Don't say "see code" and leave it at that, even if the code contains comments.
  - There's no need to explain code used to generate plots or input data, its just the code that does numerical calculations that needs to be explained.
  - Do keep code brief and clean with meaningful variable names etc.

- *Discussion.* Often in assignments you'll be asked to discuss/critically evaluate data and results. It's then necessary to express an opinion, and frequently different opinions can be supported from the same data so there is no "right" answer.

  - The key thing to bear in mind, therefore, when giving an opinion is to present supporting arguments, backed by the data/results, to justify the opinion. Just stating an opinion by itself, e.g. "I think model A is better than model B", without supporting arguments, is of little/no value.
  - Its a good idea to view discussions as a chance to demonstrate your understanding of how interpret the output of the tools/techniques that you are using, as well as a chance to practice using data to support your reasoning.
  - Longer usually isn't better when it comes to discussions. A "brain dump" is a clear symptom of a lack of understanding, as is long and rambling text. Instead, aim for a brief, focussed discussion that clearly states the main points you want to make - bullet point format is fine if you prefer that.

- *Report length.* As a rough guide, an assignment report should be around 5 pages long. If you find yourself going over 10 pages then that's too long - perhaps you can plot multiple curves on the same figure to reduce the number of figures and save space, perhaps your discussions/descriptions are too long and wordy and need to be distilled down, perhaps the text formatting makes excessively poor use of space.

- *Report format.* Please use pdf format for reports and use typed text, not handwritten. Include code in an appendix to the pdf document so its easy for us to check. In addition, please also submit code plus data in a separate zip file that we can run to check the results in the report.

TRINITY COLLEGE DUBLIN

School of Computer Science and Statistics

Mock Weekly Assignment CS7CS4/CSU44061 Machine Learning

Rules of the game:

- **Its ok to discuss with others, but do not show any code you write to others. You must write answers in your own words and write code entirely yourself. All submissions will be checked for plagiarism.**
- **Reports must be typed (no handwritten answers please) and submitted as a separate pdf on Blackboard (not as part of a zip file please).**
- **Important: For each problem, your primary aim is to articulate that you understand what you're doing - not just running a program and quoting numbers it outputs. Long rambling answers and "brain dumps" are not the way to achieve this. If you write code to carry out a calculation you need to discuss/explain what that code does, and if you present numerical results you need to discuss their interpretation. Generally most of the credit is given for the explanation/analysis as opposed to the code/numerical answer. Saying "see code" is not good enough, even if code contains comments. Similarly, standalone numbers or plots without further comment is not good enough.**
- **When your answer includes a plot be sure to (i) label the axes, (ii) make sure all the text (including axes labels/ticks) is large enough to be clearly legible and (iii) explain in text what the plot shows.**
- **Include the source of code written for the assignment as an appendix in your submitted pdf report. Also include a separate zip file containing the executable code and any data files needed. Programs should be running code written in Python, and should load data etc when run so that we can unzip your submission and just directly run it to check that it works. Keep code brief and clean with meaningful variable names etc.**
- **Reports should typically be about 5 pages, with 10 pages the upper limit (excluding appendix with code). If you go over 10 pages then the extra pages will not be marked.**

Downloading Dataset

- Download the assignment dataset from `<url>`. Important: You must fetch your own copy of the dataset, do not use the dataset downloaded by someone else.
- Please cut and paste the first line of the data file (which begins with a #) and include in your submission as it identifies your dataset.
- The data file consists of two columns of data (plus the first header line). The first column is the input feature and the second column is the target value.

.

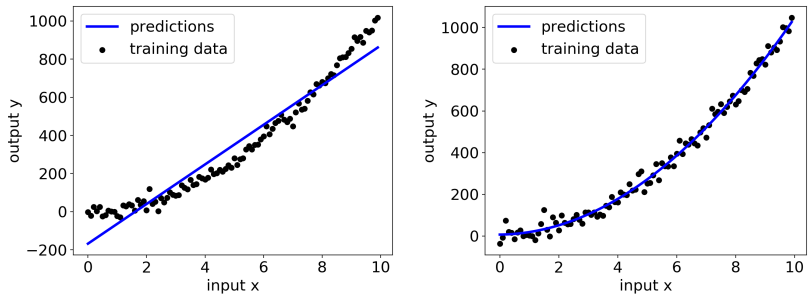
Assignment

(a) (i) Use sklearn's LinearRegression function to train a linear model on the data. Plot the training data and the predictions on the same plot so that they are easy to compare. Also give the parameters of the trained model. Discuss.

(ii) Now augment your model by including a second feature equal to the square of the first feature. Train this new model and discuss its predictions

Model Solution

(a) (i) Figure 2(a) shows the training data and the predictions of a linear model $y = \theta_0 + \theta_1 x$ trained using sklearn's LinearRegression function, where x is the input feature and y is the output prediction. The parameter values of the trained model are $\theta_0 = -158.12$, $\theta_1 = 102.97$ - see discussion in (ii) below. It can be seen that the training data follows a curve. The predictions of the linear model are a best fit straight line through this curved data, but because the predictions fail to capture the curved shape they show consistent errors, especially at the edges of the data and in the centre.



(a) linear model (b) quadratic model

Figure 2: Plot of training data plus predictions from (a) linear model and (b) quadratic model

(ii) A second input feature equal to the square of the first was added using numpy:

```
import numpy as np
Xtrain_poly=np.column_stack((Xtrain,Xtrain**2))
model = LinearRegression().fit(Xtrain_poly, ytrain)
```

Here `Xtrain` is a 1000×1 vector containing the training data input, which is just a single feature so each row of `Xtrain` corresponds to one training data point). Then `column_stack` is used to construct a 1000×2 matrix `Xtrain_poly` where again each row corresponds to one training data point but now there are two input features. Figure 2(b) shows the predictions generated by this extended model $y = \theta_0 + \theta_1 x + \theta_2 x^2$ and it can be seen that owing to addition of the second quadratic feature the model is now able to capture the curve in the data and so the predictions are much more accurate. The parameter values of the trained model are $\theta_0 = 7.00$, $\theta_1 = 1.88$, $\theta_3 = 10.21$. Adding the quadratic feature has resulted in the intercept parameter θ_0 and the weight θ_1 assigned to the original input feature x becoming much lower (previously they were -158.12 and 102.97).

Appendix

```
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(Xtrain, ytrain)
print(model.intercept_, model.coef_)
ypred = model.predict(Xtrain)
import matplotlib.pyplot as plt
plt.rc('font', size=18)
plt.rcParams['figure.constrained_layout.use'] = True
plt.scatter(Xtrain, ytrain, color='black')
plt.plot(Xtrain, ypred, color='blue', linewidth=3)
plt.xlabel("input x"); plt.ylabel("output y")
plt.legend(["predictions","training data"])
plt.show()

import numpy as np
Xtrain_poly=np.column_stack((Xtrain,Xtrain**2))
model = LinearRegression().fit(Xtrain_poly, ytrain)
print(model.intercept_, model.coef_)
```

```
ypred = model.predict(Xtrain_poly)
plt.scatter(Xtrain, ytrain,  color='black')
plt.plot(Xtrain, ypred, color='blue', linewidth=3)
plt.xlabel("input x"); plt.ylabel("output y")
plt.legend(["predictions","training data"])
plt.show()
```