ISTA 421 / INFO 521 – Homework 2

Due: Friday, October 7, 5pm

20 points total Undergraduate / 24 points total Graduate

Shuo Yang

Graduate

Instructions

In this assignment you are required to write 3 scripts in python. They will be submitted as 3 separate files, although you are free to copy chunks of code from one script to the next as desired. The names for the script files are specified in problems 1, 4 and 7, below.

In your submission, include a text file called `code.txt` that includes details about your code environment, including python version, and any instructions for executing your code from the command line.

Included in the homework 2 release are two sample scripts

`fitpoly_incomplete.py` and `cv_demo_incomplete.py`

and two data files

`womens100.csv`, and `synthdata2016.csv`

The sample scripts are provided for your convenience – you may use any part of the code in those scripts for your submissions. Note that neither will run just as provided: you must fill in the calculation of $\mathbf{w}$. The data files are provided in `.csv` (comma separated values) format. The script `fitpoly_incomplete.py` contains the function `read_data` which shows how to load the data files in python as a numpy array.

All problems after problem 1 require that you provide some "written" answer (in some cases also figures), so you will also submit a .pdf of your written answers. (You can use LaTeX or any other system (including handwritten; plots, of course, must be program-generated) as long as the final version is in PDF.)

The final submission will include (minimally) the three scripts and a PDF version of your written part of the assignment. You are required to create either a .zip or tarball (.tar.gz / .tgz) archive of all of the files for your submission and submit the archive to the d2l dropbox by the date/time deadline above.

NOTE: Problems 5 and 6 are required for Graduate students only; Undergraduates may complete them for extra credit equal to the point value.

(FCMA refers to the course text: Rogers and Girolami (2012), *A First Course in Machine Learning*. For general notes on using LaTeX to typeset math, see: `http://en.wikibooks.org/wiki/LaTeX/Mathematics`)

1. [2 points] Adapted from **Exercise 1.2** of FCMA p.35:

   Write a Python script that can find the parameters $\mathbf{w}$ for an arbitrary dataset of x_n, t_n pairs. You will use this script to answer problem 2, which only requires fitting a simple line to the data (i.e., you only need to fit parameters w_0 and w_1); however, in problems 4 and 7 you will need to fit higher-order polynomial models (e.g., $t = w_0 + w_1 x + w_2 x^2 + ...$), so you must make your script generalized to handle higher-order polynomials. The script `fitpoly_incomplete.py` is provided to help get you started. `fitpoly_incomplete.py` provides helper functions to read data, plot data, and plot the model (once you've determined the weight vector $\mathbf{w}$), but the function for computing linear least-squares fit, `fitpoly` (starting on line TODO), is *incomplete*. `fitpoly` takes as input the (one-dimensional) data vector $\mathbf{x}$, the target values vector $\mathbf{t}$, and a non-negative integer `model_order`, which represents the highest polynomial order term of the model; `fitpoly` is intended to return the $\mathbf{w}$) vector (as a numpy array).

   Recommended: (If needed) review the Appendix to HW 1, the brief tutorial to numpy arrays!

   Just to state the obvious: the objective of this exercise is for you to implement the linear least squares fit solution (i.e., the normal equation) in their general linear algebra form. **DO NOT** use existing least squares solvers, such as `numpy.linalg.lstsq`, or scikit learn's `sklearn.linear_model.LogisticRegression` as your implemented solution; however, it is certainly fine to use these to help *verify* your implementation's output.

   You will submit your script as a stand-alone file called `fitpoly.py`.

2. [1 point] Adapted from **Exercise 1.6** of FCMA p.35:

   Table 1.3 (p.13) of FCMA lists the women's 100m gold medal Olympic results – this data is provided in the file `womens100.csv` in the data folder. Using your script from problem 1, find the 1st-order polynomial model (i.e., a line with parameters w_0 and w_1) that minimizes the squared loss of this data. Report the model here as a linear equation and also include a figure that plots your model with the data.

   Solution. According to the output of the `fitpoly.py`:

```
Identified model parameters w:
[  4.09241546e+01  -1.50718122e-02]
['40.924155', '-0.015072']
```

   $w_0 = 40.924$ and $w_1 = -0.015$, so the 1st-order polynomial model is:

$$f(x; w) = 40.924 - 0.015 * x$$

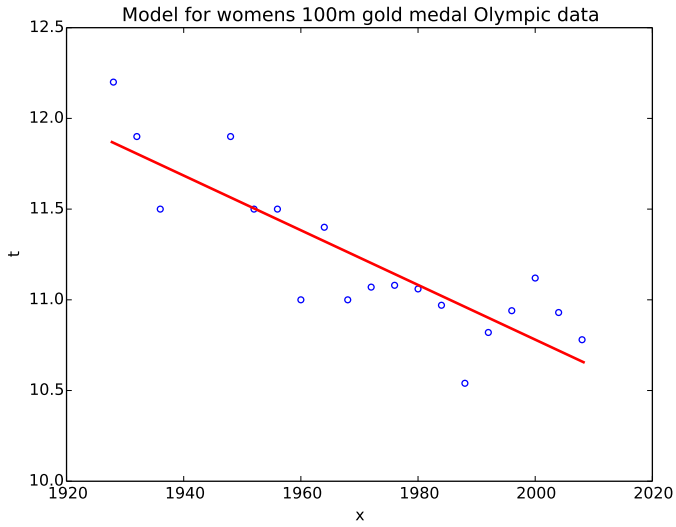

Figure 1: 1st-order polynomial model for women's 100 gold medal Olympic data

3. [1 point] Adapted from **Exercise 1.9** of FCMA p.36:

   Use your python script from problem 1 to load the data stored in the file `synthdata2016.csv` (in the data folder). Fit a 3rd order polynomial function – $f(x; \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$ – to this data (if you extended the fitpoly_incomplete.py script, then `model_order`= 3). Present and describe the parameters you obtain fitting to this data. Also, plot the data and your linear-fit model and include the plot in your answer.

   Solution. According to the output of the `fitpoly.py`:

```
Identified model parameters w:
[  9.02190177   4.54356124  49.43723008   5.00595904]
```

   Therefore the parameters are: $w_0 = 9.02190177$, $w_1 = 4.54356124$, $w_2 = 49.43723008$ and $w_3 = 5.00595904$. The 3rd order polynomial function is:

$$f(x; w) = 9.02190177 + 4.54356124 * x + 49.43723008 * x^2 + 5.00595904 * x^3$$

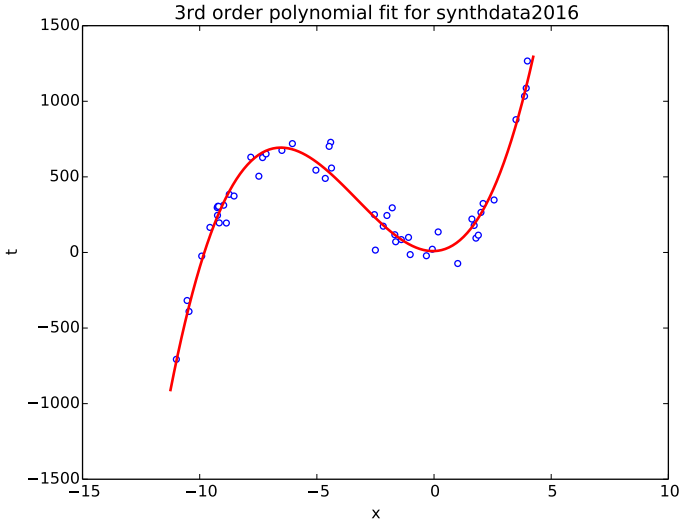

Figure 2: 3rd-order polynomial model for synth data

4. [4 points] Write a script that implements K-fold cross-validation to choose the polynomial order (between orders 0 and 7) with the best predictive error for the `synthdata2016.csv`. The provided script `cv_demo_incomplete.py` implements the synthetic data experiment described in Ch 1 (pp.31-32) of the book; you are welcome to use and adapt any part of this code you like; keep in mind that this script won't successfully execute until you add the general (matrix form) normal equation calculation on line 322. Note also that in the synthetic data experiment in `cv_demo_incomplete.py`, 1000 test data points are generated in addition to the 100 data points used for 10-fold cross-validation; for this problem (problem 4), you won't have this independent test set, only the data from `synthdata2016.csv` on which to perform K-fold cross-validation.

   Run your script with 10-fold cross-validation and Leave-One-Out-CV (LOOCV) multiple times, each while randomizing order of the data (see the `randomize_data` option of the `run_cv` function). Which model order do the two cross-validation methods predict as the best order for predictive accuracy? Do the two different cross-validation runs always agree?

   Report the best-fit model parameters for the best model order according to LOOCV, and plot this model with the data. Include a plot of the CV-loss and training loss for the 8 different (0..7) polynomial model orders for **one example each** of 10-fold cross-validation and LOOCV (i.e., you will include four plots: (1) 10-fold CV and (2) related training loss, (3) LOOCV, and (4) related training loss).

   You will submit your script as a stand-alone file called `cv.py`

   Solution. The two cross-validation methods predict the 4th-order model as the best order. The two methods don't always agree. 10-fold CV sometimes predicts that the 3rd-order model as the best order. But most time, the two agree that 4th-order model as the best one.

   The 4th-order model parameters according to LOOCV are: $w_0 = -5.37190559$, $w_1 = -13.37341483$, $w_2 = 51.20015423$, $w_3 = 6.24772985$ and $w_4 = 0.09055497$. The 4th order polynomial function is:

$$f(x; w) = -5.37190559 - 13.37341483 * x + 51.20015423 * x^2 + 6.24772985 * x^3 + 0.09055497 * x^4$$

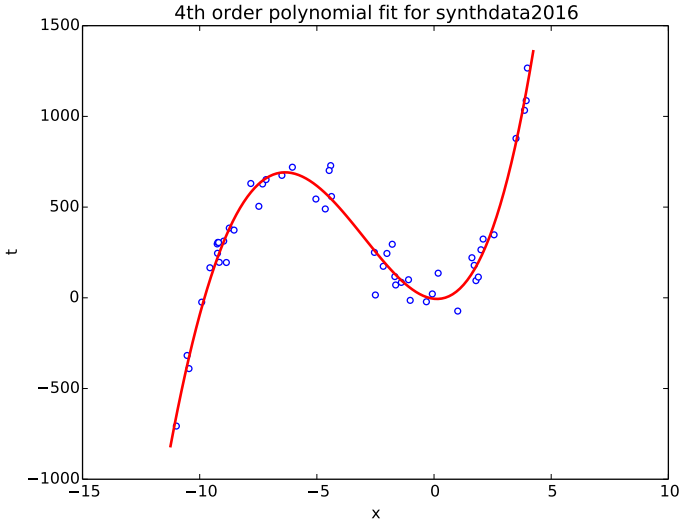

Figure 3: 4th-order polynomial model for synth data

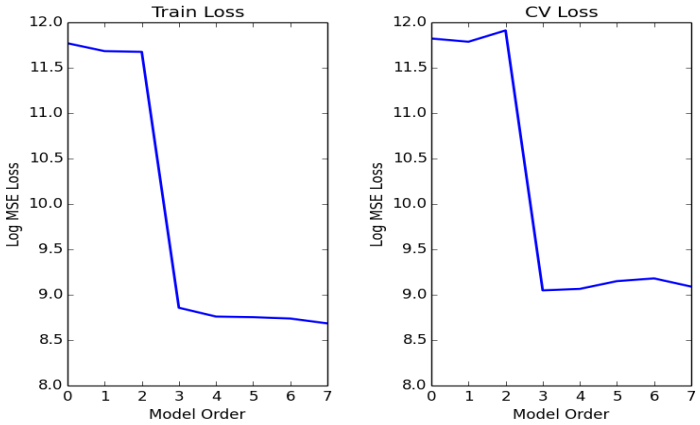

Figure 4: 10-fold CV: CV-loss and training loss for the 8 different (0..7) polynomial model orders

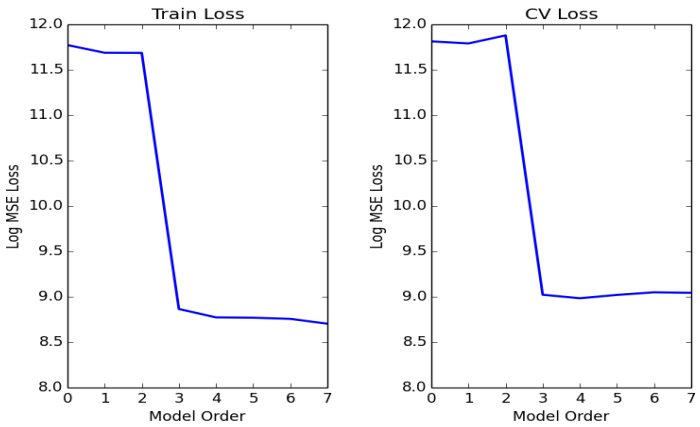

Figure 5: LOOCV: CV-loss and training loss for the 8 different (0..7) polynomial model orders

5. [2 points – **Required only for Graduates**] **Exercise 1.10** from FCMA p.36

   Derive the optimal least squares parameter value, $\hat{\mathbf{w}}$, for the total training loss:

$$\mathcal{L} = \sum_{n=1}^{N} \left(t_n - \mathbf{w}^\top \mathbf{x}_n\right)^2$$

   How does the expression compare with that derived from the average (mean) loss? (Hint: Express this loss in the **full** matrix form and derive the normal equation.)

   Solution. First, we express the loss function in full matrix form. We have:

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \mathbf{x_n} = \begin{bmatrix} 1 \\ x_n \end{bmatrix}, \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

   and the design matrix $\mathbf{X}$ as:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix}$$

   Now we express the operations involving all of the data:

$$\mathbf{t} - \mathbf{X}\mathbf{w} = \begin{bmatrix} t_1 - w_0 - w_1 x_1 \\ t_2 - w_0 - w_1 x_2 \\ \vdots \\ t_N - w_0 - w_1 x_N \end{bmatrix}$$

Further,

$$(\mathbf{t}-\mathbf{Xw})^T(\mathbf{t}-\mathbf{Xw}) = (t_1 - \mathbf{w^T x_1})^2 + \cdots + (t_N - \mathbf{w^T x_N})^2$$
$$= \sum_{n=1}^{N}(t_n - \mathbf{w^T x_n})^2$$

Therefore, we have the matrix version of the loss function:

$$\begin{aligned}\mathcal{L} &= (\mathbf{t}-\mathbf{Xw})^T(\mathbf{t}-\mathbf{Xw}) \\ &= (\mathbf{Xw}-\mathbf{t})^T(\mathbf{Xw}-\mathbf{t}) \\ &= ((\mathbf{Xw})^\mathbf{T} - \mathbf{t^T})(\mathbf{Xw}-\mathbf{t}) \\ &= (\mathbf{Xw})^\mathbf{T}\mathbf{Xw} - \mathbf{t^T}\mathbf{Xw} - (\mathbf{Xw})^\mathbf{T}\mathbf{t} + \mathbf{t^T}\mathbf{t} \\ &= (\mathbf{Xw})^\mathbf{T}\mathbf{Xw} - 2\mathbf{w^T}\mathbf{X^T}\mathbf{t} + \mathbf{t^T}\mathbf{t}\end{aligned}$$

Note that $(\mathbf{Xw})^\mathbf{T}\mathbf{t}$ and $\mathbf{t^T}\mathbf{Xw}$ are the transpose of one another and both products come out to be scalars.

Next, we take the derivative to get the matrix normal equation:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 2\mathbf{X^T}\mathbf{Xw} - 2\mathbf{X^T}\mathbf{t} = 0$$

Therefore, we have:

$$\mathbf{X^T}\mathbf{Xw} = \mathbf{X^T}\mathbf{t}$$

Multiplying the both sides with $(\mathbf{X^T}\mathbf{X})^{-1}$, we get:

$$(\mathbf{X^T}\mathbf{X})^{-1}\mathbf{X^T}\mathbf{Xw} = (\mathbf{X^T}\mathbf{X})^{-1}\mathbf{X^T}\mathbf{t}$$

Because $(\mathbf{X^T}\mathbf{X})^{-1}(\mathbf{X^T}\mathbf{X}) = I$, we get:

$$\hat{\mathbf{w}} = (\mathbf{X^T}\mathbf{X})^{-1}\mathbf{X^T}\mathbf{t}$$

This experssion is the same as the one that derived from the average (mean) loss because the constant N doesn't matter.

6. [2 points – **Required only for Graduates**] **Exercise 1.11** from FCMA p.36

   The following expression is known as the *weighted* average loss:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\alpha_n\left(t_n - \mathbf{w}^\top\mathbf{x}_n\right)^2$$

where the influence of each data point is controlled by its associated parameter. Assuming that each α_n is fixed, derive the optimal least squares parameter value $\hat{\mathbf{w}}$. (Hint: When expressing in the full matrix form, the *alpha*'s become a matrix...)

Solution. To express the loss function in full matrix form, we define the matrix $\mathbf{A}$ as a diagonal matrix:

$$\mathbf{A} = \begin{bmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha_n \end{bmatrix}$$

Based on the matrix experssion derived from the previous problem, we can now experess the loss funciton as:

$$\begin{aligned} \mathcal{L} &= \frac{1}{N}(\mathbf{Xw} - \mathbf{t})^T \mathbf{A}(\mathbf{Xw} - \mathbf{t}) \\ &= \frac{1}{N}((\mathbf{Xw})^\mathbf{T}\mathbf{A} - \mathbf{t}^\mathbf{T}\mathbf{A})(\mathbf{Xw} - \mathbf{t}) \\ &= \frac{1}{N}((\mathbf{Xw})^\mathbf{T}\mathbf{AXw} - \mathbf{t}^\mathbf{T}\mathbf{AXw} - (\mathbf{Xw})^\mathbf{T}\mathbf{At} + \mathbf{t}^\mathbf{T}\mathbf{At}) \\ &= \frac{1}{N}(\mathbf{w}^\mathbf{T}\mathbf{X}^\mathbf{T}\mathbf{AXw} - 2\mathbf{w}^\mathbf{T}\mathbf{X}^\mathbf{t}\mathbf{At} + \mathbf{t}^\mathbf{T}\mathbf{At}) \end{aligned}$$

The 3rd step to 4th step holds because $((\mathbf{Xw})^\mathbf{T}\mathbf{At})^\mathbf{T} = \mathbf{t}^\mathbf{T}\mathbf{A}^\mathbf{T}\mathbf{Xw} = \mathbf{t}^\mathbf{T}\mathbf{AXw}$ since $\mathbf{A}$ is a diagonal matrix, and both $(\mathbf{Xw})^\mathbf{T}\mathbf{At}$ and $\mathbf{t}^\mathbf{T}\mathbf{AXw}$ are scalars.

Next, we take the derivative to get the matrix normal equation:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{1}{N}(2\mathbf{X}^\mathbf{T}\mathbf{AXw} - 2\mathbf{X}^\mathbf{T}\mathbf{At}) = 0$$

Therefore, we have:

$$\mathbf{X}^\mathbf{T}\mathbf{AXw} = \mathbf{X}^\mathbf{T}\mathbf{At}$$

Multiplying the both sides with $(\mathbf{X}^\mathbf{T}\mathbf{AX})^{-1}$, we get:

$$(\mathbf{X}^\mathbf{T}\mathbf{AX})^{-1}\mathbf{X}^\mathbf{T}\mathbf{AXw} = (\mathbf{X}^\mathbf{T}\mathbf{AX})^{-1}\mathbf{X}^\mathbf{T}\mathbf{At}$$

Because $(\mathbf{X}^\mathbf{T}\mathbf{AX})^{-1}(\mathbf{X}^\mathbf{T}\mathbf{AX}) = I$, we get:

$$\hat{\mathbf{w}} = (\mathbf{X}^\mathbf{T}\mathbf{AX})^{-1}\mathbf{X}^\mathbf{T}\mathbf{At}$$

7. [4 points] Variant of **Exercise 1.12** from FCMA p.36

   Write a new Python script that uses K-fold cross-validation to find the value of λ that gives the (approximate) best predictive performance on the synthetic data (`synthdata.csv`) using **regularized**

least squares with a **7th order polynomial** model. Recall from lecture that when we don't have a more informed way to search, we can use "grid search", trying (usually equally-spaced) values from a range of possible values. In this case, we can evaluate the performance of different values of λ using CV, over a range of values. Report here (1) the λ you find that provides the best predictive performance, (2) the best fit linear model using that lambda (i.e., the inferred weights, expressed as a linear equation), (3) a plot of the log MSE loss as a function of λ in order to show the loss curve with a minimum, and (4) a plot of the best-fit model with the data.

You will submit the script as a stand-alone file called `regularize.py`. (Feel free to reuse (copy) any code from your other solutions.)

Solution. According to the output of the `regularize.py`:

```
best lambda found: 0.363636363636
identified model parameters:
['9.130581', '3.373820', '42.852487', '0.681856', '0.022730',
 '0.270414', '0.039131', '0.001595']
```

(1) the λ I found that provides the best predictive performance is: 0.363636363636.

(2) the best fit linear model using that lambda is:

$$f(x;w) = 9.130581 + 3.373820 * x + 42.852487 * x^2 + 0.681856 * x^3$$
$$+0.02273 * x^4 + 0.270414 * x^5 + 0.039131 * x^6 + 0.001595 * x^7$$

(3) the plot of the log MSE loss as a function of λ.

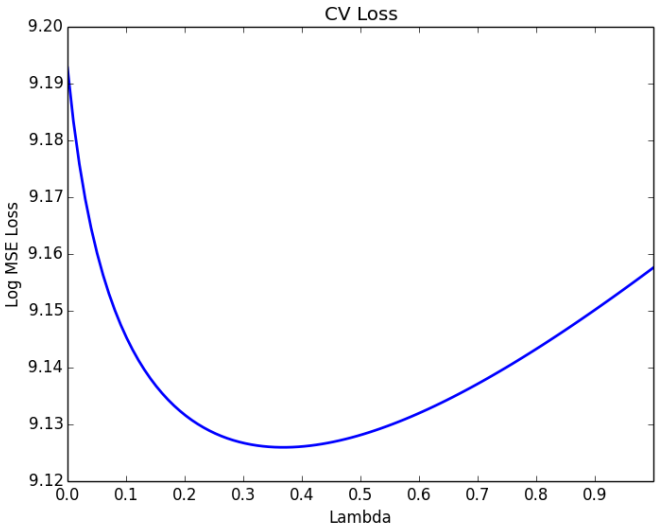

Figure 6: log MSE loss as a function of λ

(4) the plot of the best-fit model.

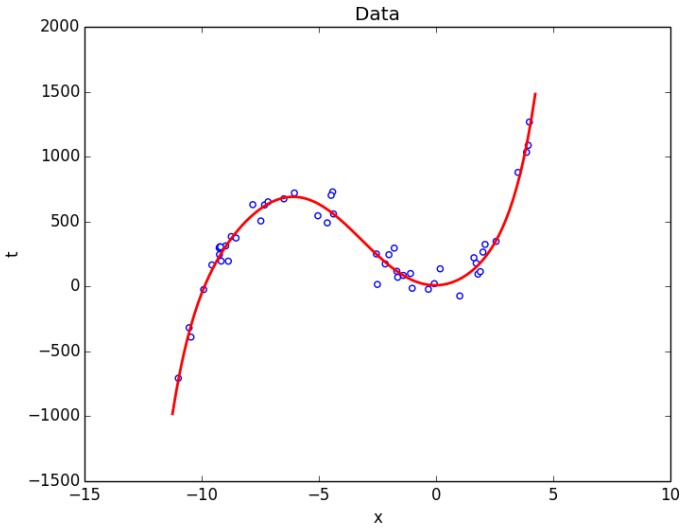

Figure 7: best-fit model with $\lambda = 0.363636363636$, model order: 7

8. [2 points] Adapted from **Exercise 2.3** of FCMA p.90:

   Let Y be a random variable that can take any non-negative integer value. The likelihood of these outcomes is given by the Poisson pmf (probability mass function):

$$p(y) = \frac{\lambda^y}{y!} e^{-\lambda} \tag{1}$$

   By using the fact that for a discrete random variable the pmf gives the probabilities of the individual events occurring and the probabilities are additive...

   (a) Compute the probability that $Y \leq 6$ for $\lambda = 8$, i.e., $P(Y \leq 6)$. Write a (very!) short python script to compute this value, and include a listing of the code in your solution.

   (b) Using the result of (a) and the fact that one outcome has to happen, compute the probability that $Y > 6$.

   Solution.

Code Listing 1: `exercise8.py`

```
import math

y = 6
lambd = 8

prob = 0.0
for i in range(y+1):
    prob += (math.pow(lambd, i) * math.exp(-lambd)) / math.factorial(i)

print 'The probability that Y<=6 for lambda=8: ', prob
```

(a) according to the output of `exercise8.py`:

```
dhcp-10-134-245-12:hw-2-shuo-yang shuoyang$ python exercise8.py
The probability that Y<=6 for lambda=8:  0.313374277536
```

So the probability that $Y \leq 6$ for $\lambda = 8$ is 0.313374277536.

(b) $P(Y > 6) = 1 - P(Y \leq 6) = 1 - 0.313374277536 = 0.686625722464$

9. [3 points] Adapted from **Exercise 2.5** of FCMA p.91:

Assume that $p(\mathbf{w})$ is the Gaussian pdf for a D-dimensional vector $\mathbf{w}$ given in

$$p(\mathbf{w}) = \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{w}-\mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{w}-\mu)\right\}.$$

By expanding the vector notation and re-arranging, show that using $\mathbf{\Sigma} = \sigma^2\mathbf{I}$ as the covariance matrix assumes independence of the D elements of $\mathbf{w}$. You will need to be aware that the determinant of a matrix that only has entries on the diagonal ($|\sigma^2\mathbf{I}|$) is the product of the diagonal values and that the inverse of the same matrix is constructed by simply inverting each element on the diagonal. (Hint, a product of exponentials can be expressed as an exponential of a sum. Also, just a reminder that $\exp\{x\}$ is e^x.)

Solution. Given the covariance matrix $\mathbf{\Sigma} = \sigma^2\mathbf{I}$, we have:

$$|\mathbf{\Sigma}| = \prod_{d=1}^{D} \sigma^2$$
$$|\mathbf{\Sigma}|^{1/2} = (\prod_{d=1}^{D} \sigma^2)^{1/2}$$
$$\mathbf{\Sigma}^{-1} = \frac{1}{\sigma^2}\mathbf{I}$$

We also have:

$$(\mathbf{w}-\mu)^T = \begin{bmatrix} w_1-\mu_1 & w_2-\mu_2 & \dots & w_D-\mu_D \end{bmatrix}$$
$$(\mathbf{w}-\mu)^T\mathbf{\Sigma}^{-1} = \begin{bmatrix} w_1-\mu_1 & w_2-\mu_2 & \dots & w_D-\mu_D \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma^2} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\sigma^2} \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{\sigma^2}(w_1-\mu_1) & \frac{1}{\sigma^2}(w_2-\mu_2) & \dots & \frac{1}{\sigma^2}(w_D-\mu_D) \end{bmatrix}$$
$$(\mathbf{w}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{w}-\mu) = \begin{bmatrix} \frac{1}{\sigma^2}(w_1-\mu_1) & \frac{1}{\sigma^2}(w_2-\mu_2) & \dots & \frac{1}{\sigma^2}(w_D-\mu_D) \end{bmatrix} \begin{bmatrix} w_1-\mu_1 \\ w_2-\mu_2 \\ \dots \\ w_D-\mu_D \end{bmatrix}$$
$$= \frac{1}{\sigma^2}\sum_{d=1}^{D}(w_d-\mu_d)^2$$

Because a product of exponentials can be expressed as an exponential of a sum, we have:

$$\exp\left\{-\frac{1}{2}(\mathbf{w}-\mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{w}-\mu)\right\} = \exp\left\{-\frac{1}{2}\frac{1}{\sigma^2}\sum_{d=1}^{D}(w_d-\mu_d)^2\right\}$$
$$= \prod_{d=1}^{D}\exp\left\{-\frac{1}{2\sigma^2}(w_d-\mu_d)^2\right\}$$

Combining the above equations, we get:

$$\begin{aligned} p(\mathbf{w}) &= \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{w}-\mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{w}-\mu)\right\} \\ &= \frac{1}{(\prod_{d=1}^{D}(2\pi))^{1/2}(\prod_{d=1}^{D}(\sigma^2))^{1/2}} \prod_{d=1}^{D} \exp\left\{-\frac{1}{2\sigma^2}(w_d-\mu_d)^2\right\} \\ &= \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(w_d-\mu_d)^2\right\} \end{aligned}$$

Therefore when using $\mathbf{\Sigma} = \sigma^2\mathbf{I}$ as the covariance matrix, we are assuming the independence of the D elements of $\mathbf{w}$.

10. [3 points] Adapted from **Exercise 2.9** of FCMA p.91:

    Assume that a dataset of N binary values, $x_1, ..., x_n$, was sampled from a Bernoulli distribution, and each sample x_i is independent of any other sample. Explain why this is *not* a Binomial distribution. Derive the maximum likelihood estimate for the Bernoulli parameter.

    Solution. Binomial distribution is used to describe the probability distribution of a certain number of successes in a sequence of success/fail experiments, while Bernoulli distribution describes the probability distribution of one such experiment. Since the dataset is sampled from a Bernoulli distribution, it cannot give the probablity for a certain number of experiments, thus is not a Binomial distribution.

    Assume that the success probability is p. We have the likelihood function as:

$$\begin{aligned} \mathbf{L}(p|\mathbf{x}) &= \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} \\ &= p^{(x_1+x_2+\cdots+x_n)}(1-p)^{n-(x_1+x_2+\cdots+x_n)} \end{aligned}$$

    Then apply the natural logarithmic function on both side:

$$\begin{aligned} \ln \mathbf{L}(p|\mathbf{x}) &= (\sum_{i=1}^{n} x_i)\ln p + (n - \sum_{i=1}^{n} x_i)\ln(1-p) \\ &= n\bar{x}\ln p + n(1-\bar{x})\ln(1-p) \end{aligned}$$

    Take the derivative and set to zero to get the maximum likelihood estimate for the Bernoulli parameter:

$$\frac{\partial \ln \mathbf{L}(p|\mathbf{x})}{\partial p} = \frac{n\bar{x}}{p} + \frac{n(1-\bar{x})}{1-p}$$

    The above equation is zero when $p = \bar{x}$ which gives us the maximum. Thus the maximum likelihood estimation is:

$$\hat{p}(\mathbf{x}) = \bar{x}$$