TDA233 / DIT382 Machine Learning: Homework 1
Goal: Start working with Jupyter notebooks, introduction to probability and regression models
Grader: Arman Rahbar
Submitted by: Axel Qvarnström, Maximilian Salén Personal no. 980728, 970105, axelqv@student.chalmers.se, maximilian@live.se

General guidelines:

- All solutions to theoretical and pratical problems must be submitted in this ipynb notebook, and equations wherever required, should be formatted using LaTeX math-mode.
- All discussion regarding practical problems, along with solutions and plots should be specified in this notebook. All plots/results should be visible such that the notebook do not have to be run. But the code in the notebook should reproduce the plots/results if we choose to do so.
- Your name, personal number and email address should be specified above.
- All tables and other additional information should be included in this notebook.
- **Before submitting, make sure that your code can run on another computer. That all plots can show on another computer including all your writing. It is good to check if your code can run here: https://colab.research.google.com**
- **Upload both the .ipynb and the generatable .html file (that can be exported through File > Download As > HTML)**

Jupyter/IPython Notebook is a collaborative Python web-based environment. This will be used in all our Homework Assignments. It is installed in the halls ES61-ES62, E-studio and MT9. You can also use google-colab: https://colab.research.google.com to run these notebooks without having to download, install, or do anything on your own computer other than a browser. Some useful resources:

1. https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/ (Quick-start guide)
2. https://www.kdnuggets.com/2016/04/top-10-ipython-nb-tutorials.html
3. http://data-blog.udacity.com/posts/2016/10/latex-primer/ (latex-primer)
4. http://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%
(markdown)

Theoretical problems

[Bayes Rule, 3 points]

After your yearly checkup, the doctor has bad news and good news. The bad news is that you tested positive for a very serious cancer and that the test is 99.2% accurate i.e. the probability of testing positive given you have the disease is 0.992. The probability of testing negative if you don't have the disease is the same (also 0.992). The good news is that it is a rare condition affecting only 1 in 1,000 people. What is the probability you actually have the disease?

After doing all your calculations you realize that there was a misprint on the test, and the accuracy was actually only 92% (for both testing postive given that you have the disease and for testing negative given that you do not have the disease). How will this change your probability of having the disease?

(Show all calculations and the final result.)

[Setting hyperparameters, 2 points]

Suppose θ is a random variable generated from a beta distribution as: $\theta \sim \mathrm{Beta}(a^2, b)$. Also assume that the expectation of θ is m: $E[\theta] = m$ and the variance of θ is v: $\mathrm{var}(\theta) = v$. Express a and b in term of (only) m and v. For more information about the Beta distribution see https://en.wikipedia.org/wiki/Beta_distribution.

[Correlation and Independence, 2 points]

Let X be a continuous random variable, uniformly distributed in $[-2, +2]$ and let $Y := X^4$. Clearly Y is not independent of X -- in fact it is uniquely determined by X. However, show that the covariance of X and Y is 0: $\mathrm{cov}(X, Y) = 0$. Show and justify every step of the proof. Statements like "it is obvious that, it is trivial ..." will not be accepted.

[Spherical Gaussian estimation, 4 points]

Consider a dataset X consisting of i.i.d. observations generated from a spherical Gaussian distribution $N(\mu, \sigma^2 I)$, where $\mu \in \mathbb{R}^p$, I is the $p \times p$ identity matrix, and σ^2 is a scalar.

Write the mathematical expression for the Maximum Likelihood Estimator (MLE) for μ and σ in above setup.

For more information about the spherical Gaussian distribution, see https://en.wikipedia.org/wiki/Multivariate_normal_distribution. For more information about the identity matrix see: https://en.wikipedia.org/wiki/Identity_matrix

Write Answers here. Double-click (or enter) to edit. Latex mathematical expressions can just be written by starting with "$" and ending with the same character.

Bayes Rules

Before the error is noticed

We start by defining some probabilities:

$P(\text{disease}) = \frac{1}{1000}$

$P(\text{Positive result}|\text{disease}) = 0.992$

$P(\neg\text{disease}|\text{Negative result}) = 0.992$

Bayes rule:

$P(\text{disease}|\text{positive result}) = \frac{P(\text{Positive result}|\text{disease}) \cdot P(\text{disease})}{P(\text{Positive result})}$

$P(\text{Positive result})$ is given by the law of total probability:

$P(\text{Positive result}) = P(\text{Positive result}|\text{disease}) \cdot P(\text{disease}) + P(\text{Positive result}|\neg\text{diseas}$
$= 0.992 \cdot 0.001 + (1 - 0.992) \cdot 0.999 = 0.008984$

This gives in Bayes rule:

$P(\text{disease}|\text{Positive result}) = \frac{0.992 \cdot 0.001}{0.008984} = 0.1104$

After the error is noticed:

$P(\text{Positive result}) = 0.92 \cdot 0.001 + (1 - 0.92) \cdot 0.999 = 0.08084$

$P(\text{disease}|\text{Positive result}) = \frac{0.92 \cdot 0.001}{0.08084} = 0.0113$

The change from 99.2% to 92% accuracy changes the probability of having the disease from 11.04% to 1.13%

Setting hyperparameters

(1)

$$E[\theta] = \frac{a^2}{a^2 + b} = m$$

$$=> b = a^2(\frac{1}{m} - 1)$$

(2)

$$\text{Var}(\theta) = \frac{a^2 b}{(a^2 + b)^2 (a^2 + b + 1)} = v$$

(1) in (2)

$$v = \frac{a^4(\frac{1}{m} - 1)}{(a^2 + a^2(\frac{1}{m} - 1))^2 (a^2 + a^2(\frac{1}{m} - 1) + 1)}$$

Simlified to:

$$v = \frac{m^2(-m + 1)}{a^2 + m}$$

Which then gives:

$$a = \pm\sqrt{\frac{-m(m^2-m+v)}{v}},\ b = \frac{(m-1)(m^2-m+v)}{v}$$

Correlation

X = [-2, 2], Y = X^4 = [0, 16]

$$Cov(X,Y) = E[(X-E(X))(Y-E(Y)] = E(X*Y) - E(X)E(Y)$$
$$\text{where } X = [-2,2] => mean(X) = E(X) = 0$$
$$\text{and } Y = X^4 => E(X*Y) = E(X^5) = 0 \text{ due to odd exponent of the X term}$$
$$\text{Since } E(X*Y) = 0 \text{ and } E(X) = 0 => Cov(X,Y) = 0$$

Spherical Gaussian

We want to determine

$$argmax_{\mu,\sigma} \prod_{i=1}^{N} p(x_i|\mu,\Sigma)$$

Then we apply the log and the log of product becomes the sum of the logs:

$$argmax_{\mu,\sigma} \sum_{i=1}^{N} logp(x_i|\mu,\Sigma)$$

Since $\Sigma = \sigma^2 I$, we can replace Σ with a diagonal vector σ^2:

$$argmax_{\mu,\sigma} \sum_{i=1}^{N} logp(x_i|\mu,\sigma)$$

We have that:

$$p(x_i|\mu,\sigma^2) = \left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x_i-\mu)^2}\right)$$

It implies that:

$$logL = \sum_{i=1}^{N} log\left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x_i-\mu)^2}\right)$$

Applying log rules we get:

$$logL = \sum_{i=1}^{N} \left(-log\left(\sigma\sqrt{2\pi}\right) - \frac{1}{2\sigma^2}(x_i-\mu)^2\right)$$

$$logL = \sum_{i=1}^{N} \left(-log\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2\sigma^2}(x_i-\mu)^2\right)$$

$$logL = \sum_{i=1}^{N}\left(-\frac{1}{2}log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}(x_i-\mu)^2\right)$$

Algebraic simplification gives:

$$logL = -\frac{N}{2}log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2$$

Now we are going to determine the MLE of μ

$$\frac{\partial logL}{\partial\mu} = \frac{\partial}{\partial\mu}\left(-\frac{N}{2}log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2\right)$$

$$\frac{\partial logL}{\partial\mu} = \frac{\partial}{\partial\mu}\left(-\frac{N}{2}log\left(2\pi\sigma^2\right)\right) - \frac{\partial}{\partial\mu}\left(\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2\right)$$

$$\frac{\partial logL}{\partial\mu} = -\frac{\partial}{\partial\mu}\left(\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2\right)$$

$$\frac{\partial logL}{\partial\mu} = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}2(x_i-\mu)\cdot -1$$

$$\frac{\partial logL}{\partial\mu} = \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i-\mu)$$

Setting the derivative to zero and solve for μ gives:

$$\frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i-\mu) = 0$$

$$\sum_{i=1}^{N}(x_i-\mu) = 0$$

$$\sum_{i=1}^{N}x_i - \sum_{i=1}^{N}\mu = 0$$

$$N\mu = \sum_{i=1}^{N}x_i$$

$$\mu = \frac{1}{N}\sum_{i=1}^{N}x_i$$

Now doing the same for σ:

$$\frac{\partial logL}{\partial\sigma} = \frac{\partial}{\partial\sigma}\left(-\frac{N}{2}log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2\right)$$

$$\frac{\partial logL}{\partial\sigma} = -\frac{N}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{N}(x_i-\mu)^2 = 0$$

$$-N\sigma^2 + \sum_{i=1}^{N}(x_i - \mu)^2 = 0$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Practical problems

[Linear Regression with regularization 9pt]

You are newly recruited as a Data Scientist at a leading consultancy company in Gothenburg. Your first task at the job is to help the Swedish Public Health Agency (folkhalsomyndigheten) for predicting the diabetes progression of patients. Assume that you are given a dataset D of n patients with $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where $\mathbf{x}_i \in \mathbb{R}^p$ represents numerical features of each patients and $y_i \in \mathbb{R}$ represent the numerical diabetes progression. One can also view the dataset D as a pair of matrices $(\mathbf{X}, \mathbf{y})$ with $\mathbf{X} \in \mathbb{R}^{n\times p}$ and $\mathbf{y} \in \mathbb{R}^{n\times 1}$.

Fresh with the lectures in the machine learning course at Chalmers, you would like to use a linear model to quickly perform the task. In order words, you would like to find a vector $\mathbf{w} \in \mathbb{R}^{p\times 1}$ such that $\mathbf{y} = \mathbf{X}\mathbf{w}$. However, you have just read one of the most popular machine learning book and it argues that standard linear regression (for finding $\mathbf{w}$) can lead to various problems such as non-uniqueness of the solution, overfitting, etc. As a result, you decided to add a penalty term called regularization to control the optimisation problem. More specifically, you want to solve for: $\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$ where $\mathcal{L}(\mathbf{w}) = \left(\sum_{i=1}^{n}(y_i - \mathbf{w}^T\mathbf{x}_i)^2\right) + \left(\alpha \sum_{j=1}^{p} w_j^2\right)$ with $\alpha \in \mathbb{R}$ a small coefficient that you will decide later on.

Please note the slight changes in the notation. Recall that in the lectures we had a dataset $\{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$ where $\mathbf{x}_n \in \mathbb{R}^D$ and $t_n \in \mathbb{R}$. We also appended 1 to the begining of $\mathbf{x}_n$ so both $\mathbf{x}_n$ and $\mathbf{w}$ were in $\mathbb{R}^{D+1}$. Thus, here p is the same thing as $D + 1$. Compare $w_1, w_2, \ldots, w_p$ with $w_0, w_1, \ldots, w_D$.

1- **1pt** Write down $\mathcal{L}(\mathbf{w})$ in matrix/vector forms using only $\mathbf{X}$, $\mathbf{y}$ and $\mathbf{w}$ and the L2 norm. In other words, you are not allowed to use any components $y_i, \mathbf{w}_j$ or $\mathbf{x}_i$ (For any vector $\mathbf{z}$ use the following notation $|\mathbf{z}|_2$ to mean the L2 norm of $\mathbf{z}$ See http://mathworld.wolfram.com/L2-Norm.html for more information about the L2 norm.)

Solution:

$$\text{Vector form} => L(w) = (y - w^T X)^T (y - w^T X) + \alpha |w|_2^2$$

2- **1pt** Derive and write down in matrix/vector forms the gradient of $\mathcal{L}(\mathbf{w})$ with respect to $\mathbf{w}$. Show all the derivations. (Hint: You can start by computing the gradient of the full expression and then convert it to matrix/vector forms. You can also directly get the gradients from your answer in 1-)

Solution:

$$\frac{\partial L(w)}{\partial w} = \frac{\partial}{\partial w}((y - w^T X)^T (y - w^T X)) + \frac{\partial}{\partial w}\left(\alpha |w|_2^2\right)$$

Where:

$$\frac{\partial}{\partial w}\left(\alpha |w|_2^2\right) = \alpha \sum_{i=1}^{N} \frac{\partial}{\partial w_k} w_i^2 = [0 \text{ for all k} \neq i, \text{else } 2w_k] = 2\alpha I w$$

$$\frac{\partial L(w)}{\partial w} = 2X^T Xw - 2X^T y + 2\alpha I w = 2X^T(Xw - y) + 2\alpha I w$$

3- **2pt** Derive and write down in matrix/vector forms the solution $\mathbf{w}^*$ to the optimization problem $\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$. Show all your derivations. (Hint: $\mathcal{L}(\mathbf{w})$ is convex in $\mathbf{w}$)

Solution:

$$\frac{\partial L(w)}{\partial w} = 0$$

$$2X^T(Xw - y) + 2\alpha I w = 0$$

$$X^T Xw - X^T y + \alpha I w$$

$$w(X^T X + \alpha I) = X^T y$$

$$w^* = (X^T X + \alpha I)^{-1} X^T y$$

4- **2pt** Under which condition on the α is the solution $\mathbf{w}^*$ unique? Prove rigorously your statement. Make no assumptions on $\mathbf{X}$. (Hint: If your solution $\mathbf{w}^*$ requires to invert a matrix, then one necessary condition for uniquess is for the matrix to be invertible. And any positive definitive matrix https://en.wikipedia.org/wiki/Definiteness_of_a_matrix is invertible. You might also want to look at the properties of transposition https://en.wikipedia.org/wiki/Transpose)

Solution:

For α to be a unique solution $(X^T X + \alpha I)$ must be invertible. It is invertible if it is positive definitive. $X^T X$ is positive semi definitive if X only includes real values, since this will be greater or equal to zero. $(X^T X + \alpha I)$ is positive definitive if one of the terms in the sum is positive definitive and the other is positive semi definitive. Identity matrix is positive definitive and a scalar greater than zero multiplied by a positive definitive matrix is still a positive definitive matrix. In other words $(X^T X + \alpha I)$ is invertible if $\alpha > 0$

5- **2pt** Implement in Python a well commented function **fit_linear_with_regularization** that takes as input $\mathbf{X}$, $\mathbf{y}$ and α and return $\mathbf{w}^*$ as computed in question 3. You are not allowed to use any loops (for-loop, while-loop ...) to do the implementation. Instead use and abuse as much as possible numpy vectorization techniques. A skeleton of the function is shown in the code cell below.

6- **3pt** Implement in Python a well commented function **predict** that takes as input a dataset $\mathbf{X}_{\mathbf{test}}$ in the same dimensions as $\mathbf{X}$ and return the predictions. Write down the mean squared error (https://en.wikipedia.org/wiki/Mean_squared_error) of your predictions. Then on the same plot with legends, show:

a) A scatter plot of the first feature of $\mathbf{X}_{\mathbf{test}}$ (x-axis) and the diabetes progression $\mathbf{y}_{\mathbf{test}}$

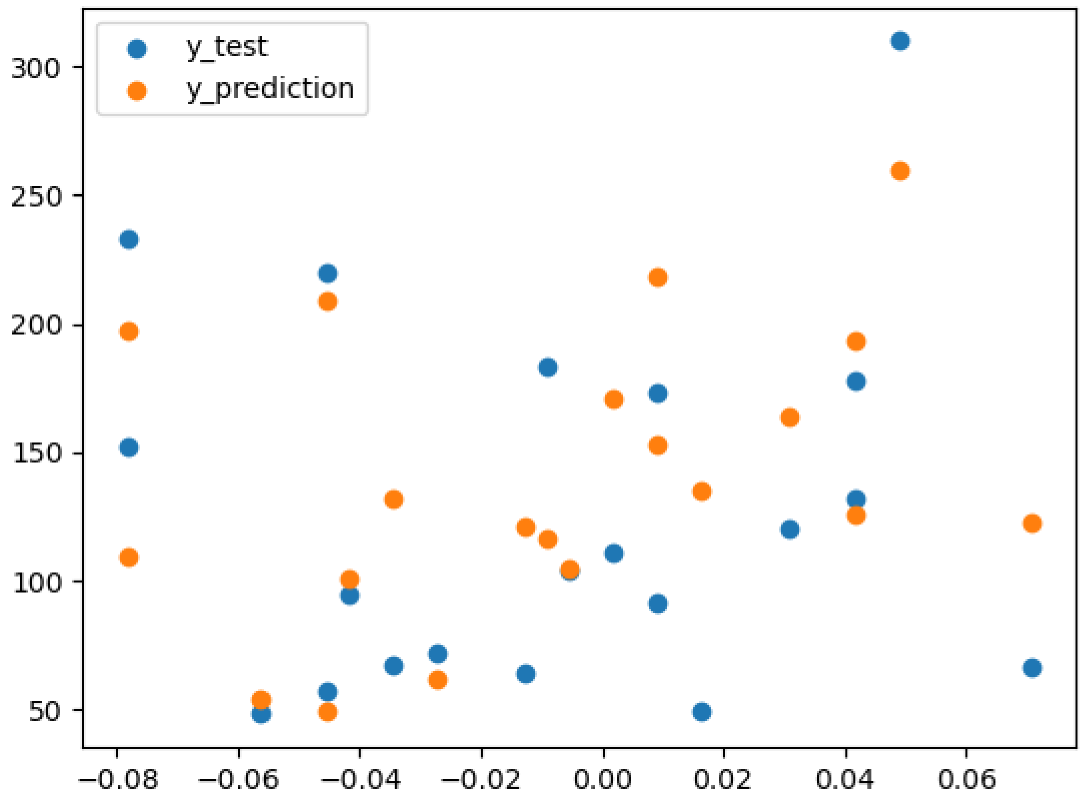
b) A plot of your prediction for $\mathbf{X}_{\mathbf{test}}$

The skeleton code in the cell below already implements most of data loading and you should only have to fill in the *TODO* part. Again here no loops are allowed (for-loop, while loop in the implementation of the plots and the **predict**)

In [2]:
```
# Make it possible to show plots in the notebooks.
%matplotlib inline
import numpy as np
from sklearn import datasets
import matplotlib
import matplotlib.pyplot as plt



def fit_linear_with_regularization(X, y, alpha):
    # split the equation into two terms, the first correspond to the inverted term
    first_term = np.linalg.inv(np.matmul(np.transpose(X), X) + alpha * np.eye(np.sl
    second_term = np.matmul(np.transpose(X),y)
    return np.matmul(first_term, second_term)  #returns w*


def predict(X_test, w):
    # make a prediction using the weigth matrix with the formula y = w^T * X
    return np.matmul(X_test, w) # TODO change me



def plot_prediction(X_test, y_test, y_pred):

    # Scatterplot the first feature of X_test (x-axis) and y_test (y-axis)
    plt.scatter(X_test[:,1],y_test, label='y_test')

    # Scatterplot y_pred using the first feature of X_test as x-axis
    plt.scatter(X_test[:,1],y_pred, label='y_prediction') # scatter the first feat

    # Compute the mean squared error
    mean_squared_error = 1/len(y_test)*np.sum((y_test - y_pred)**2)
    return mean_squared_error


# Load the diabetes dataset
X, y = datasets.load_diabetes(return_X_y=True)
X = np.array(X)

X = np.c_[np.ones(len(X)), X]

# Split the dataset into training and test set
num_test_elements = 20
```

```
X_train = X[:-num_test_elements]
X_test = X[-num_test_elements:]

y_train = y[:-num_test_elements]
y_test = y[-num_test_elements:]





# Set alpha
alpha = 0.01

# Train using linear regression with regularization and find optimal model
w = fit_linear_with_regularization(X_train, y_train, alpha)


# Make predictions using the testing set X_test
y_pred = predict(X_test, w)



# Plots and mean squared error
error = plot_prediction(X_test, y_test, y_pred)



print('Mean Squared error is ', error)


# Show the plot
plt.legend()
plt.show();
```

```
Mean Squared error is  2003.79096500609
```



Bonus question

Answering this question, will not give you any additional points. Not answering this question will not prevent you from getting full points (if all other questions are answered correctly). However, if you answer this question, we will pick exactly one question where you didn't receive full points in this assignment and give you **at most** 4 more points there. In particular, between the questions for which you have reasonably attempted a solution, we will pick the one where the difference between the full point and the point you received is the maximum.

Bayesian Linear Regression 4pt

Proud of finishing the task using a linear model with regularization, you show your results to a representative of the Swedish Public Health Agency. You barely finish explaining your solution when the face of the representative turns red and you could distinctly hear: "Bayesian is the only way: How come didn't you use any probabilities?".

You quickly head back to your desk and now assume a Gaussian prior on the solution $\mathbf{w}$, that is $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$ where $\lambda \in \mathbb{R}$ is a constant real number, I is the $p \times p$ identity matrix and $\mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$ is used to mean the multivariate gaussian distribution with mean $\mathbf{0} \in \mathbb{R}^p$, a vector of zeros of dimension p and covariance matrix $\lambda^{-1}\mathbf{I}$. Then, you use the following likelihood:

$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{n} \mathcal{N}(\mathbf{w}^T\mathbf{x}_i, \gamma^{-1})$ where here $\gamma \in \mathbb{R}$ is a constant real number and $\mathcal{N}(\mathbf{w}^T\mathbf{x}_i, \gamma^{-1})$ is the gaussian distribution with mean $\mathbf{w}^T\mathbf{x}_i$ and variance γ^{-1}.

7- **2pt** Derive and express in vector/matrix form as a function of $\mathbf{X}, \mathbf{y}, \mathbf{w}$ the log posterior $\ln p(\mathbf{w}|\mathbf{y}, \mathbf{X})$. Show all the derivations. You can ignore normalizing constants.

8- **2pt** Show that maximizing the posterior in 7- is similar to minimizing the function $\mathcal{L}(\mathbf{w})$ seen in the previous section. Show your derivations. (Note: You should show this without computing the maximum of the posterior. Instead, you should express the log posterior in term of $\mathcal{L}(\mathbf{w})$, ignoring constants if necessary. Then find the α of $\mathcal{L}(\mathbf{w})$ in term of λ and γ).