1 Comparing Different Loss Functions [30 Points]

Relevant materials: lecture 3 & 4

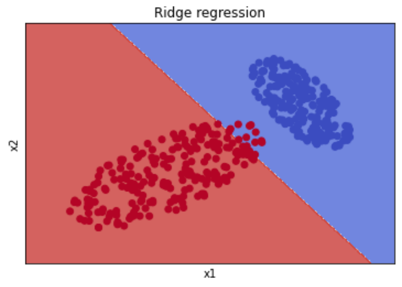
Problem A [3 points]: Squared loss is often a terrible choice of loss function to train on for classification problems. Why?

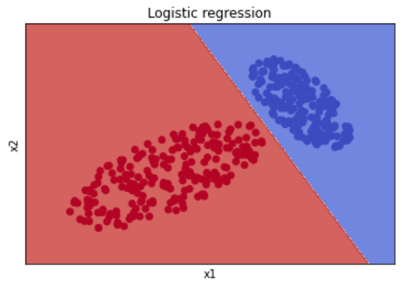
First, because using L2 loss means that we assume that the underlying data has been generated from a normal distribution (a bell-shaped curve) While in reality, a dataset that can be classified into two categories (i.e binary) is not from a normal distribution but a Bernoulli distribution.

Also, for classification the squared error will penalize the outliers excessively, even if the values have the same sign as their respective y.

Problem B [9 points]:

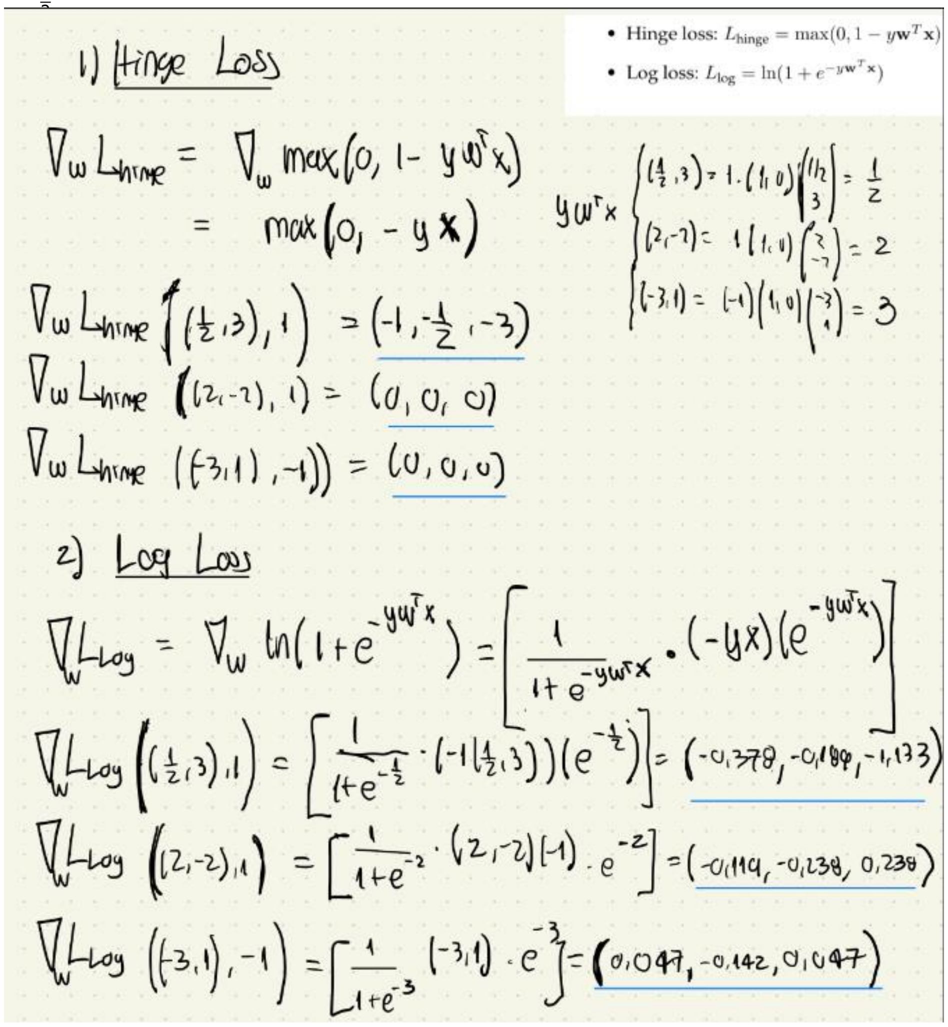
https://colab.research.google.com/drive/1hctfiio9ifuUzmJEL5H5lKNZZ7OTzcub?usp=sharing





The decision boundary for logistics regression classified the points very well, separating all the points correctly, but the decision boundary for the ridge regression performed a little bad, misclassifying some red points. This can be explained considering the squared loss function (used for ridge regression) penalizes correctly points that are far away from the decision boundary, causing that the decision boundary shift towards them in order to decrease the loss. The difference with the log loss case, is that this error function penalizes just considering misclassified values according to their magnitudes and thus, produces lower loss for points that are far from the boundary and were correctly classified

Problem C [9 points]:



- Hinge loss: $L_{\text{hinge}} = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$
- Log loss: $L_{\log} = \ln(1 + e^{-y\mathbf{w}^T\mathbf{x}})$

1) Hinge Loss

$$\nabla_w L_{hinge} = \nabla_w \max(0, 1 - yw^Tx)$$
$$= \max(0, -y\mathbf{x})$$

$$yw^Tx \begin{cases} (\tfrac{1}{2}, 3) = 1 \cdot (1, 0)\begin{pmatrix} 1/2 \\ 3 \end{pmatrix} = \tfrac{1}{2} \\ (2, -2) = 1(1, 0)\begin{pmatrix} 2 \\ -2 \end{pmatrix} = 2 \\ (-3, 1) = (-1)(1, 0)\begin{pmatrix} -3 \\ 1 \end{pmatrix} = 3 \end{cases}$$

$$\nabla_w L_{hinge}\left((\tfrac{1}{2}, 3), 1\right) = \underline{(-1, -\tfrac{1}{2}, -3)}$$
$$\nabla_w L_{hinge}\left((2, -2), 1\right) = \underline{(0, 0, 0)}$$
$$\nabla_w L_{hinge}\left((-3, 1), -1\right) = \underline{(0, 0, 0)}$$

2) Log Loss

$$\nabla_w L_{Log} = \nabla_w \ln(1 + e^{-yw^Tx}) = \left[\frac{1}{1 + e^{-yw^Tx}} \cdot (-yx)(e^{-yw^Tx})\right]$$

$$\nabla_w L_{Log}\left((\tfrac{1}{2}, 3), 1\right) = \left[\frac{1}{1 + e^{-\frac{1}{2}}} \cdot (-1(\tfrac{1}{2}, 3))(e^{-\frac{1}{2}})\right] = \underline{(-0{,}378, -0{,}189, -1{,}133)}$$

$$\nabla_w L_{Log}\left((2, -2), 1\right) = \left[\frac{1}{1 + e^{-2}} \cdot (2, -2)(-1) \cdot e^{-2}\right] = \underline{(-0{,}119, -0{,}238, 0{,}238)}$$

$$\nabla_w L_{Log}\left((-3, 1), -1\right) = \left[\frac{1}{1 + e^{-3}} (-3, 1) \cdot e^{-3}\right] = \underline{(0{,}047, -0{,}142, 0{,}047)}$$

Problem D [4 points]: Compare the gradients resulting from log loss to those resulting from hinge loss. When (if ever) will these gradients converge to 0? For a linearly separable dataset, is there any way to reduce or altogether eliminate training error without changing the decision boundary?

For the points when the error for hinge loss is zero, the error is small but not zero in the Log loss function

The gradients for the hinge loss will converge to zero when y* wTx is greater than or equal to 1 (this is when the point is outside of the margin in the correct side), meanwhile in the case of the log loss function, the gradient converge when y* wTx goes to infinite (as far from the margin as possible in the correct side). It is possible to reduce or even eliminate the training error by scaling the w, since it will increase the value of y* wTx. For the case of the Hinge loss, it will be completely eliminated if y* wTx is greater than or equal to 1, meanwhile the log loss will decrease as y* wTx increase.

Problem E [5 points]:

Because minimizing just the L_{hinge} function can be done by scaling the w, without changing the decision boundary, which is what we want to control in order to classify the data correctly maximizing the margin of the points to the boundary. The role of the penalty term *is to fix that constraining the magnitude of* w

2 Effects of Regularization [40 Points]

Relevant materials: Lecture 3 & 4

Problem A [4 points]:

No, it won't decrease the training error. Adding this term will increase the training error because the training error was already minimized by the unregularized loss function.
Not always. If there is already overfitting, adding a regularization term will decrease the training error for small λ, *but for large λ, the model can go from overfitting to underfitting easily, which can increase the validation error.*
Adding a penalty term will decrease the out-of-sample error only where there exists overfitting.

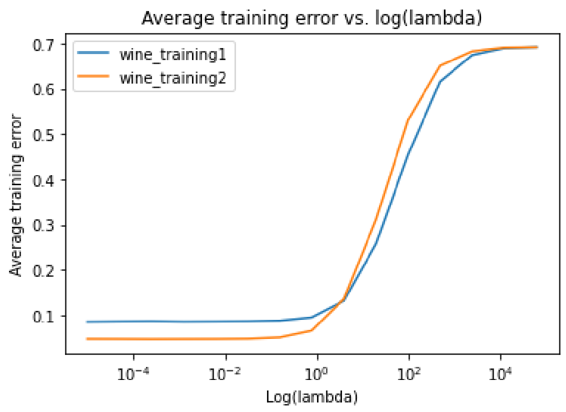
Problem B [4 points]:

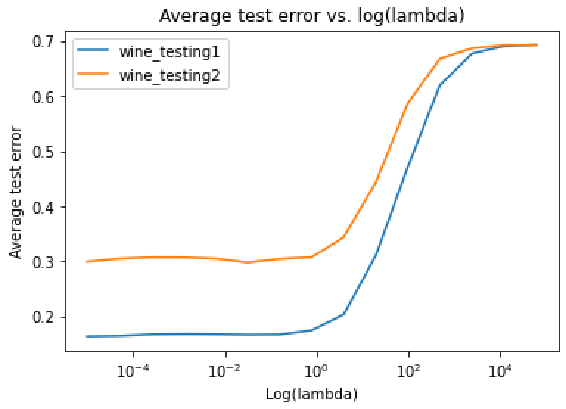
That is because $\ell 1$ and $\ell 0$ are discontinuous and non-convex functions. So numerical methods such as gradient descent cannot be used.

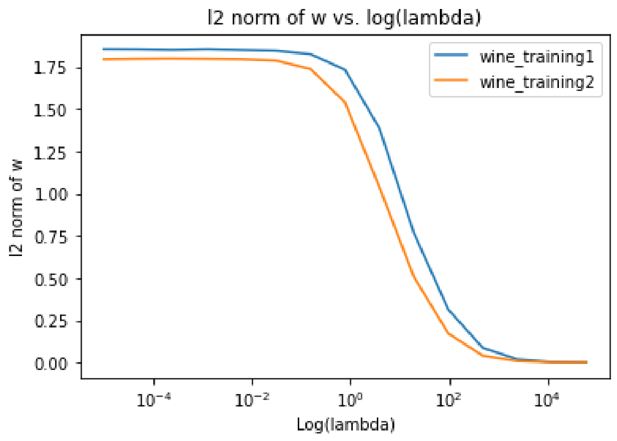
Implementation of ℓ_2 regularization:

Problem C [16 points]:

https://colab.research.google.com/drive/1LMELRCbBhyEFKj6mGkDEUHAsC0YzqnE6?usp=sharing


Average training error vs. log(lambda)
wine_training1
wine_training2
Average training error
Log(lambda)
0.1
0.2
0.3
0.4
0.5
0.6
0.7
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}


Average test error vs. log(lambda)
wine_testing1
wine_testing2
Average test error
Log(lambda)
0.2
0.3
0.4
0.5
0.6
0.7
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}


l2 norm of w vs. log(lambda)
wine_training1
wine_training2
l2 norm of w
Log(lambda)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}

Problem D [4 points]:

Given that second training set is a subset of the first training set and thereby contains fewer training points, we expect and observe the testing error resulting from training set 1 to be lower than that from training set 2. The test error for wine_training2 is much higher with low lambda, because its small sample size causes it to overfit.

Regarding the training error, set 1 has a slightly higher error at lower λ *but is slightly lower for high* λ*, compared to set 2. This can be explained considering that at lower values of* λ*, complex models are able to fit the dataset containing less points (set 2) and thus, explaining the reason why training set 1 has higher errors than for small* λ*. But, for large* λ *(and thus simpler models), points with larger error from set 2 (subset of set 1) will negatively affect the error of the smaller dataset more than the error of the larger dataset.*

Problem E [4 points]:

As λ *increases, it is more notorious there is underfitting considering both training and testing increase, reflecting that we are regularizing too much and the constrained model is too simple to capture the dataset accurately. The test error for the set 1 is very low and has a minimum for a* λ *of around 0.1 showing that there are not very much differences in test and training error for smaller* λ*, meaning there is not lot of overfitting and that the model generalizes well with a* λ *of 0,1. When* λ *increases more, the model tends to underfit.*

Problem F [4 points]:

We can visualize that the norm of w decreases when λ *increases (equivalent to norm-constrained training using lagrangian). This makes sense because with greater* λ*s, the weight of the regularization term (in the loss function) is higher and thus w should be less (the regularization term encourages model coefficients to be small as possible).*

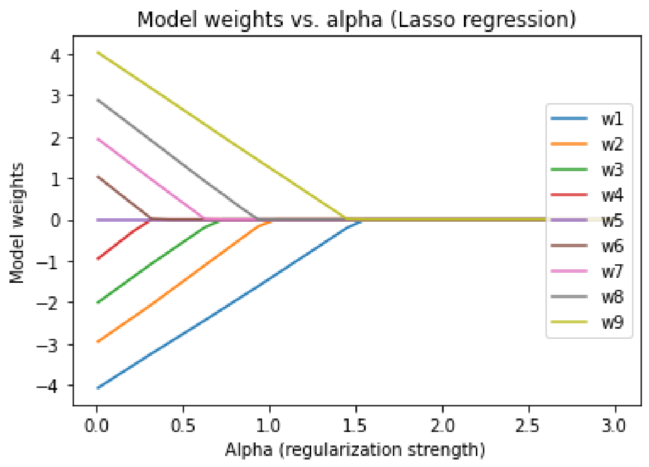
Problem G [4 points]: If the model were trained with wine training2.txt, which λ would you choose to train your final model? Why?

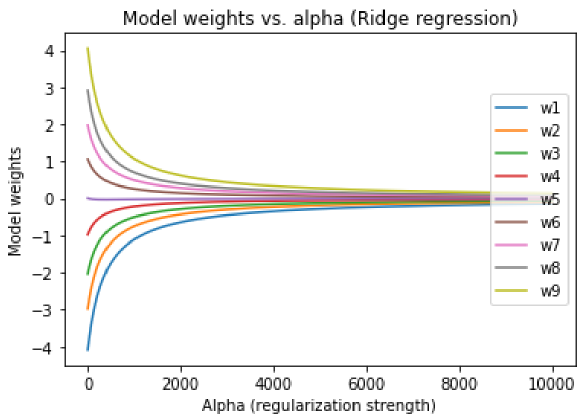
By visualizing the plots, we can see that the λ*s that produces the lower* E_{out} *are between* 10^{-4} and 1 (are very stable on this range) with a optimal value around 10^-1 (after this range the error in sample increases dramatically)

3 Lasso (ℓ_1) vs. Ridge (ℓ_2) Regularization [25 Points]

Problem A [11 points]:

https://colab.research.google.com/drive/1UBgNjTFlDcU8-0ccJK7ziDqUmpIgPJd5?usp=sharing



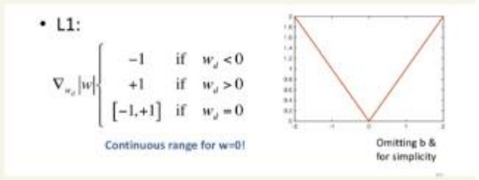


For lasso Regression, as alpha increases al the weighs become exactly zero, which doesn't occur for ridge regression where the model weights seem so converge to 0 when alpha tends to infinite (asymptotically).

Problem B [7 points]:

First, we need to differenciate the function with respect to w in order to minimize it (setting it equal to 0). Considering $\lambda|w|$ is a non-differentiable function, we use the sub-gradient, which is:

$$\begin{cases} -\lambda & \text{if } w_d<0 \\ \lambda & \text{if } w_d>0 \\ [-\lambda,\lambda] & \text{if } w_d=0 \end{cases}$$



• L1:

$$\nabla_{w_d}|w| \begin{cases} -1 & \text{if } w_d<0 \\ +1 & \text{if } w_d>0 \\ [-1,+1] & \text{if } w_d=0 \end{cases}$$

Continuous range for w=0!

Omitting b & for simplicity

Thus, we have the following

$$\frac{d}{dw}\left(\|y-Xw\|^2+\lambda\|w\|_1\right)=\begin{cases} -2X^T(y-Xw)+\lambda & \text{if } w>0 \\ -2X^T(y-Xw)-\lambda & \text{if } w<0 \end{cases}$$

$$w=\begin{cases} -2X^Ty+2X^TXw+\lambda=0 \rightarrow w=(2X^TX)^{-1}\left[2X^Ty-\lambda\right] & \text{if } w>0 \\ -2X^Ty+2X^TXw-\lambda=0 \rightarrow w=(2X^TX)^{-1}\left[\lambda+2X^Ty\right] & \text{if } w<0 \end{cases}$$

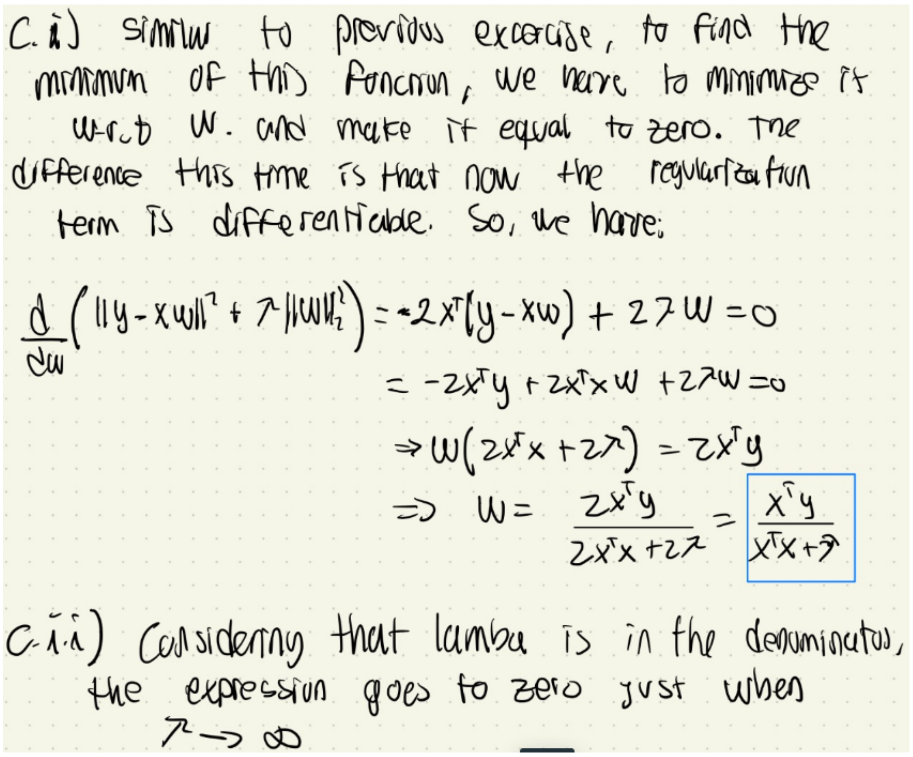
$$\begin{cases} w>0 \text{ only if } 2X^Ty>\lambda \text{ given } \lambda\geq 0 \\ w<0 \text{ only if } 2X^Ty>-\lambda \text{ given } \lambda\geq 0 \\ w=0 \text{ only if } -\lambda\leq 2X^Ty\leq\lambda \end{cases}$$

Thus

$$w=\begin{cases} (2X^Ty-\lambda)(2X^TX)^{-1} & \text{if } 2X^Ty>\lambda \\ (2X^Ty+\lambda)(2X^TX)^{-1} & \text{if } 2X^Ty<-\lambda \\ 0 & \text{if } -\lambda\leq 2X^Ty\leq\lambda \end{cases}$$

ii) Yes. As shown in the previous item, $w=0$ if $-\lambda\leq 2X^Ty\leq\lambda$. So the smallest value for λ is $2X^Ty$

Problem C [7 points]:



C.i) similar to previous excercise, to find the minimum of this function, we have to minimize it w.r.t W. and make it equal to zero. The difference this time is that now the regularization term is differentiable. So, we have:

$$\frac{d}{dw}\left(\|y - Xw\|^2 + \lambda\|w\|_2^2\right) = -2X^T(y - Xw) + 2\lambda w = 0$$

$$= -2X^T y + 2X^T X w + 2\lambda w = 0$$

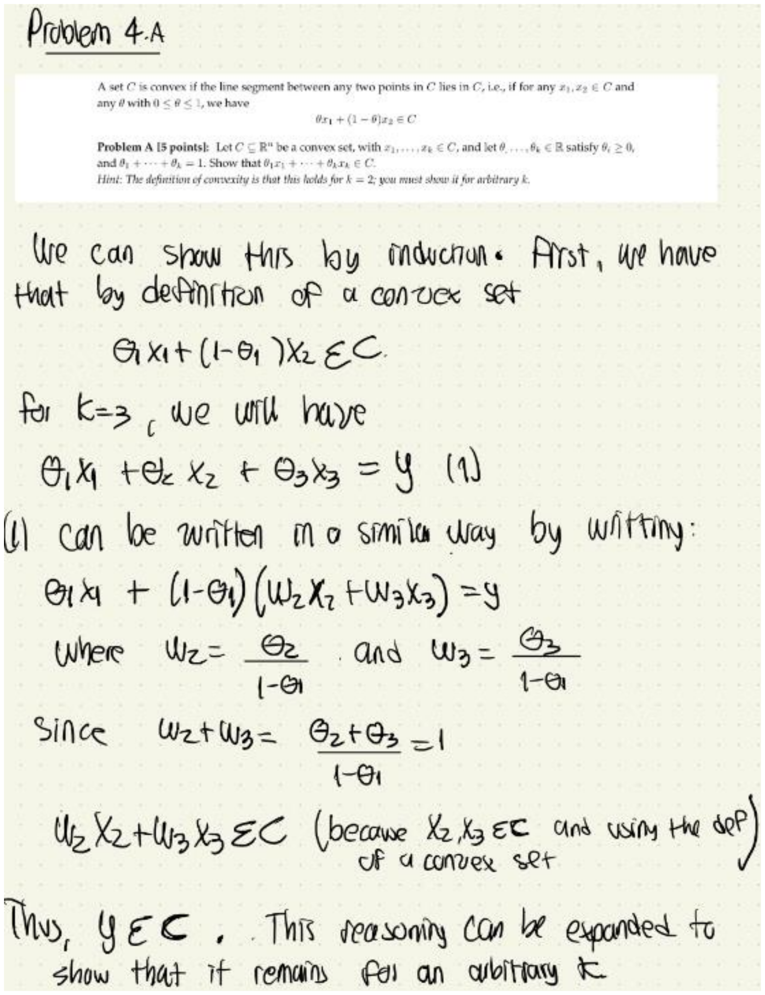
$$\Rightarrow w(2X^T X + 2\lambda) = 2X^T y$$

$$\Rightarrow w = \frac{2X^T y}{2X^T X + 2\lambda} = \boxed{\frac{X^T y}{X^T X + \lambda}}$$

C.ii) Considering that lamba is in the denominator, the expression goes to zero just when $\lambda \to \infty$

4 Convexity and Lipschitz Continuity [10 Points]

Problem A [5 points]:



Problem 4.A

A set C is convex if the line segment between any two points in C lies in C, i.e., if for any $x_1, x_2 \in C$ and any θ with $0 \leq \theta \leq 1$, we have

$$\theta x_1 + (1-\theta)x_2 \in C$$

Problem A [5 points]: Let $C \subseteq \mathbb{R}^n$ be a convex set, with $x_1, \dots, x_k \in C$, and let $\theta, \dots, \theta_k \in \mathbb{R}$ satisfy $\theta_i \geq 0$, and $\theta_1 + \cdots + \theta_k = 1$. Show that $\theta_1 x_1 + \cdots + \theta_k x_k \in C$.
Hint: The definition of convexity is that this holds for $k = 2$; you must show it for arbitrary k.

We can show this by induction. First, we have that by definition of a convex set

$\theta_1 x_1 + (1-\theta_1) x_2 \in C$.

for $k=3$, we will have

$\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 = y$ (1)

(1) can be written in a similar way by writting:

$\theta_1 x_1 + (1-\theta_1)(w_2 x_2 + w_3 x_3) = y$

where $w_2 = \frac{\theta_2}{1-\theta_1}$ and $w_3 = \frac{\theta_3}{1-\theta_1}$

since $w_2 + w_3 = \frac{\theta_2 + \theta_3}{1-\theta_1} = 1$

$w_2 x_2 + w_3 x_3 \in C$ (because $x_2, x_3 \in C$ and using the def of a convex set)

Thus, $y \in C$. This reasoning can be expanded to show that it remains for an arbitrary k

Problem B [5 Extra Credit points]: